## About error and warning messages

Messages are displayed with the message class first, followed by the source file name and line number where the error was detected, and finally with the text of the message itself.

The following categories of messages can occur:

| Category | Indicates |
|---|---|
| Informational | Progress such as build status. |
| Error | A problem that should be fixed such as a missing declaration or a type mismatch. |
| Warning | A problem that can be overlooked. |
| Fatal | A problem of critical nature that prevents execution from continuing. |

Be aware that the compiler generates messages as they are detected. Because C and C++ don't force any restrictions on placing statements on a line of text, the true cause of the error might occur one or more lines before or after the line number specified in the error message.

Many of the messages appear in the Message view. For those messages, context-sensitive help is available. Point to the message and press F1 to display the message description.

If you are working from the command line or want to look up information on an error message, refer to the alphabetical list of Error and warning messages. Find the message you're interested in and click on it to display its description.

### Symbols

Some messages include a symbol (such as a variable, file name, or module) that is taken from your program. In the following example, 'filename' will be replaced by the file causing the problem:

```
Error opening 'filename' for output
```

Here's what the symbols in error and warning messages stand for.

| Symbol | Meaning |
|---|---|
| address | A hexadecimal number indicating the address where the error occurred |
| argument | An argument |
| class | A class name |
| filename | A file name (with or without extension) |
| function | A function name |
| group | A group name |
| identifier | An identifier (variable name or other) |
| language | The name of a programming language |
| member | The name of a data member or member function |
| message | A message string |
| module | A module name |
| name | Any type of name |
| num | An actual number |
| number | An actual number |
| option | An option |
| parameter | A parameter name |
| path | A path name |
| reason | Reason given in message |
| segment | A segment name |
| size | An actual number |
| specifier | A type specifier |

| | |
|---|---|
| symbol | A symbol name |
| type | A type name |
| variable | A program variable |

Some messages begin with a symbol name such as the following:

```
'filename' not found
```

These messages are listed alphabetically using the name of the symbol. The above message would be filed under f.

**Compiler errors and warnings**

Compile-time error messages indicate errors in program syntax, command-line errors, or errors in accessing a disk or memory. When most compile-time errors occurs, the compiler completes the current phase (preprocessing, parsing, optimizing, and code-generating) of the compilation and stops. But when fatal compile-time errors happen, compilation stops completely. If a fatal error occurs, fix the error and recompile.

Runtime errors occur after the program has successfully compiled and is running. Runtime errors are usually caused by logic errors in your program code. If you receive a runtime error, you must fix the error in your source code and recompile the program for the fix to take effect.

Warnings indicate that conditions which are suspicious but legitimate exist or that machine-dependent constructs exist in your source files. Warnings do not stop compilation.

Warnings are issued as a result of a variety of conditions, such as:

| | |
|---|---|
| ANSI violations | Warn you of code that is acceptable to C++Builder (because of C++ code or C++Builder extensions), but is not in the ANSI definition of C. |
| Frequent warnings | Alert you to common programming mistakes. These warning messages point out conditions that are not in violation of the C++Builder language but can yield the wrong result. |
| Less frequent warnings | Alert you to less common programming mistakes. These warning messages point out conditions that are not in violation of the C++Builder language but can yield the wrong result. |
| Portability warnings | Alert you to possible problems with porting your code to other compilers. These usually apply to C++Builder extensions. |
| C++ warnings | Warn you of errors you've made in your C++ code. They might be due to obsolete items or incorrect syntax. |

**Runtime errors and warnings**

Runtime errors occur after the program has successfully compiled and is running.

**Linker errors and warnings**

As a rule, linker errors do not stop the linker or cause .EXE or .MAP files to be deleted. When such errors happens, don't try to execute the .EXE file. Fix the error and relink.

A fatal link error, however, stops the linker immediately. In such a case, the .EXE file is deleted. All Linker errors are treated as fatal errors if you are compiling from the Integrated Development Environment (IDE).

Linker warnings point out conditions that you should fix. When warnings occur, .EXE and .MAP files are still created.

**Librarian errors and warnings**

Librarian errors and warnings occur when there is a problem with files or extended dictionaries, when memory runs low, or when there are problems as libraries are accessed.

# Error and warning messages

Overview

{button Symbols,JI(`',`errorsxref_symbols')} {button a,JI(`',`errorsxref_A')} {button b,JI(`',`errorsxref_B')} {button c,JI(`',`errorsxref_C')} {button d,JI(`',`errorsxref_D')} {button e,JI(`',`errorsxref_E')} {button f,JI(`',`errorsxref_F')} {button g,JI(`',`errorsxref_G')} {button h,JI(`',`errorsxref_H')} {button i,JI(`',`errorsxref_I')} {button l,JI(`',`errorsxref_L')} {button m,JI(`',`errorsxref_M')} {button n,JI(`',`errorsxref_N')} {button o,JI(`',`errorsxref_O')} {button p,JI(`',`errorsxref_P')} {button q,JI(`',`errorsxref_Q')} {button r,JI(`',`errorsxref_R')} {button s,JI(`',`errorsxref_S')} {button t,JI(`',`errorsxref_T')} {button u,JI(`',`errorsxref_U')} {button v,JI(`',`errorsxref_V')} {button w,JI(`',`errorsxref_W')}

## Symbols

#undef directive ignored—Compiler warning

( expected—Compiler error

) expected—Compiler error

, expected—Compiler error

: expected after private—Compiler error

: expected after protected—Compiler error

: expected after public—Compiler error

< expected—Compiler error

> expected—Compiler error

@ seen, expected a response-files name—Librarian error

{ expected—Compiler error

} expected—Compiler error

16-bit segments not supported in module 'module'—Linker error

## A

Abnormal program termination—Runtime error

Access can only be changed to public or protected—Compiler error

Access specifier of property identifier must be a member of function—Compiler error

Access violation. Program terminated—Incremental Linker error

Added file 'filename' does not begin correctly, ignored—Librarian warning

Address of overloaded function 'function' doesn't match 'type'—Compiler error

Ambiguity between 'function1' and 'function2'—Compiler error

Ambiguous member name 'name'—Compiler error

Ambiguous operators need parentheses—Compiler warning

Ambiguous override of virtual base member 'function1': 'function2'—Compiler error

Application is running—Runtime error

Array allocated using 'new' may not have an initializer—Compiler error

Array bounds missing ]—Compiler error

Array must have at least one element—Compiler error

Array of references is not allowed—Compiler error

Array size for 'delete' ignored —Compiler warning or error

Array size too large—Compiler error

Array variable 'identifier' is near—Compiler warning

Assembler stack overflow—Compiler error

Assembler statement too long—Compiler error

Assertion failed: module at \address\, line number—Incremental Linker error

Assigning 'type' to 'enumeration'—Compiler warning

Assignment to 'this' not allowed, use X::operator new instead—Compiler error

Attempt to export non-public symbol 'symbol'—Linker error

Attempt to free NULL pointer in module, 'line number'—Incremental Linker error

Attempt to grant or reduce access to 'identifier'—Compiler error

Attempting to return a reference to a local object—Compiler error

Attempting to return a reference to local variable 'identifier'—Compiler error

## B

Bad call of intrinsic function—Compiler error
Bad character in parameters -> 'char'—Linker error
Bad 'directive' directive syntax—Compiler error
Bad field list in debug information in module 'module'—Linker error
Bad file name 'filename'—Linker error
Bad file name format in include directive—Compiler error
Bad filename format in include statement—MAKE error
Bad file name format in line directive—Compiler error
Bad GRPDEF type encountered, extended dictionary aborted—Librarian warning
Bad header in input LIB—Librarian error
Bad LF_POINTER in module 'module'—Linker error
Bad loc for fixupp in module 'module' near file offset 'offset'—Linker error
Bad macro output translator—MAKE error
Bad object file 'filename' near file offset 'offset'—Linker error
Bad OMF record type 'type' encountered in module 'module'—Librarian error
Bad secondary target for fixup in module 'module'—Linker error
Bad syntax for pure function definition—Compiler error
Bad 'type' debug info in module 'module'—Linker error
Bad undef statement syntax—MAKE error
Base class 'class1' is also a base class of 'class2'—Compiler warning
Base class 'class' contains dynamically dispatchable functions—Compiler error
Base class 'class' is inaccessible because also in 'class'—Compiler warning
Base class 'class' is included more than once—Compiler error
Base class 'class' is initialized more than once—Compiler error
Base initialization without a class name is now obsolete—Compiler warning
'base' is an indirect virtual base class of 'class'—Compiler error
Bit field cannot be static—Compiler error
Bit field too large—Compiler error
Bit fields must be signed or unsigned int—Compiler error
Bit fields must be signed or unsigned int—Compiler warning
Bit fields must contain at least one bit—Compiler error
Bit fields must have integral type—Compiler error
Body has already been defined for function 'function'—Compiler error
Both return and return with a value used—Compiler warning

## C

Call of nonfunction—Compiler error
Call to function 'function' with no prototype—Compiler warning
Call to function with no prototype—Compiler warning
Call to undefined function 'function'—Compiler error
Can't grow LE/LIDATA record buffer—Librarian error
Can't inherit non-RTTI class from RTTI base 'class'—Compiler error
Can't inherit RTTI class from non-RTTI base 'class'—Compiler error
Cannot add or subtract relocatable symbols—Compiler error
Cannot allocate a reference—Compiler error
Cannot call 'main' from within the program—Compiler error
Cannot call near class member function with a pointer of type 'type'—Compiler error
Cannot cast from 'type1' to 'type2'—Compiler error
Cannot convert 'type1' to 'type2'—Compiler error
Cannot create instance of abstract class 'class'—Compiler error
Cannot create precompiled header: 'reason'—Compiler error
Cannot declare or define 'identifier' here—Compiler error
Cannot define a pointer or reference to a reference—Compiler error

Cannot define 'identifier' using a namespace alias—Compiler error
Cannot find 'class'::operator=('class'&) to copy a vector—Compiler error
Cannot find class::class (class &) to copy a vector—Compiler error
Cannot find default constructor to initialize array element of type 'class'—Compiler error
Cannot find default constructor to initialize base class 'class'—Compiler error
Cannot find default constructor to initialize member 'identifier'—Compiler error
Cannot find MAKE.EXE—MAKE error
Cannot find tasm program: tasm.exe—IDE error
Cannot generate 'function' from template function 'template'—Compiler error
Cannot have a non-inline function in a local class—Compiler error
Cannot have a static data in a local class—Compiler error
Cannot have multiple paths for implicit rule—MAKE error
Cannot have path list for target—MAKE error
Cannot inherit non-RTTI class from RTTI base—Compiler error
Cannot initialize 'type1' with 'type2'—Compiler error
Cannot initialize a class member here—Compiler error
Cannot locate file I/O hook functions for resource compiler—Resource compiler error
Cannot Load Linker: linker—IDE error
Cannot modify a const object—Compiler error
Cannot overload 'main'—Compiler error
Cannot take address of 'main'—Compiler error
Cannot throw 'type' -- ambiguous base class 'base'—Compiler error
Cannot use local type 'identifier' as template argument—Compiler error
Cannot use tiny or huge memory model with Windows—Compiler error
Cannot write a string option—MAKE error
Cannot write GRPDEF list, extended dictionary aborted—Librarian warning
Cannot write to disk—Incremental Linker error
Case bypasses initialization of a local variable—Compiler error
Case outside of switch—Compiler error
Case statement missing :—Compiler error
'catch' expected —Compiler error
Character constant must be one or two characters long—Compiler error
Character constant too long—MAKE error
Circular dependency exists in makefile—MAKE error
Circular property definition :—Compiler error
Class 'class' may not contain pure functions—Compiler error
Class 'classname' is abstract because of 'member = 0'—Compiler error
Classid requires definition of ‘type’ as a pointer type—Compiler error
Class member 'member' declared outside its class—Compiler error
Code has no effect—Compiler warning
CodeGuarded programs must use the large memory model and be targeted for Windows—Compiler error
Colon expected—MAKE error
Command arguments too long—MAKE error
Command syntax error—MAKE error
Comparing signed and unsigned values—Compiler warning
Compiler could not generate copy constructor for class 'class'—Compiler error
Compiler could not generate default constructor for class 'class'—Compiler error
Compiler could not generate default destructor for class 'class'—Compiler error
Compiler could not generate operator = for class 'class'—Compiler error
Compiler stack overflow—Compiler error
Compiler table limit exceeded—Compiler error
Compound statement missing }—Compiler error
Condition is always false—Compiler warning

## D

.DEF file heap reserve size < 64K; 1MB default will be used—Linker warning
.DEF file stack reserve size < 64K; 1MB default will be used—Linker warning
Default argument value redeclared—Compiler error
Default argument value redeclared for parameter 'parameter'—Compiler error
Default expression may not use local variables—Compiler error
Default outside of switch—Compiler error
Default value missing—Compiler error
Default value missing following parameter 'parameter'—Compiler error
Define directive needs an identifier—Compiler error
Delete array size missing ]—Compiler error
Destructor cannot be declared 'const' or 'volatile'—Compiler error
Destructor cannot have a return type specification—Compiler error
Destructor for 'class' required in conditional expression—Compiler error
Destructor for class is not accessible—Compiler error
Destructor name must match the class name—Compiler error
Dispid ‘number’ already used by ‘identifier’ Divide error—Compiler error
Dispid only allowed in __automated sections—Compiler error
Divide error—Runtime error
Division by zero—Compiler error
Division by zero—Compiler warning
Division by zero—MAKE error
do statement must have while—Compiler error
do-while statement missing (—Compiler error
do-while statement missing )—Compiler error
do-while statement missing ;—Compiler error
DPMI programs must use the large memory model—Compiler error
Duplicate case—Compiler error
Duplicate file ' filename'  in list, not added!—Librarian error
Duplicate handler for 'type1', already had 'type2'—Compiler error
Duplicate ordinal for exports: 'string' ('ordval1') and 'string' ('ordval2')—Linker error

## E

Earlier declaration of 'identifier'—Compiler error
End of system input buffer encountered—Linker error
Empty LEDATA record in module 'module'—Linker warning
Enum syntax error—Compiler error
Error changing file buffer size—Librarian error
Error directive: 'message'—Compiler error
Error directive: 'message'—MAKE error
Error opening 'filename'—Librarian error
Error opening 'filename' for output—Librarian error
Error processing module 'module'—Incremental Linker error
Error renaming 'filename' to 'filename'—Librarian error
Error writing output file—Compiler error
'__except' or '__finally' Expected Following '__try'—Compiler error
Exception handling not enabled—Compiler error
Exception handling variable may not be used here—Compiler error
Exception specification not allowed here—Compiler error
Explicit stacks are ignored for PE images—Linker warning
Export 'symbol' has multiple ordinal values: 'value1' and 'value2'—Linker warning
Export 'symbol' is duplicated—Linker warning
Expression expected—Compiler error
Expression of scalar type expected—Compiler error
Expression syntax—Compiler error

Expression syntax error in !if statement—MAKE error
Extern 'symbol' was not qualified with __import in module 'module'—Linker warning
Extern variable cannot be initialized—Compiler error
Extra argument in template class name 'template'—Compiler error
Extra parameter in call—Compiler error
Extra parameter in call to function—Compiler error

## F

Failed to create map file 'filename' (error code 'number')—Incremental Linker error
Failed read from 'filename'—Linker error
Failed write to 'filename'—Linker error
Far COMDEFs are not supported—Linker error
FATAL ERROR: Cannot Load Linker: 'linker'—IDE error
FATAL ERROR: Linker CREATE Failed—IDE error
FATAL ERROR: Linker missing CREATE Entry Point—IDE error
FATAL ERROR: Linker missing DESTROY Entry Point—IDE error
FATAL ERROR: GP FAULT—MAKE error
File must contain at least one external declaration—Compiler error
File name too long—Compiler error
Filename too long—MAKE error
'filename' couldn't be created, original won't be changed—Librarian warning
'filename' does not exist—don't know how to make it—MAKE error
'filename' file not found—Librarian error
'filename' file not found—Librarian warning
'filename' is not a valid library—Linker warning
'filename' not a MAKE—MAKE error
Fixup to zero length segment in module 'module'—Linker error
Fixupps found for an LIDATA record—Linker error
Floating point error: Divide by 0—Runtime error
Floating point error: Domain—Runtime error
Floating point error: Overflow—Runtime error
Floating point error: Partial loss of precision—Runtime error
Floating point error: Stack fault—Runtime error
Floating point error: Underflow—Runtime error
For statement missing (—Compiler error
For statement missing )—Compiler error
For statement missing ;—Compiler error
Friends must be functions or classes—Compiler error
Function body ignored—Compiler warning
Function call missing )—Compiler error
Function call terminated by unhandled exception 'value' at address 'addr'—Compiler error
'function' cannot return a value—Compiler error
Function defined inline after use as extern—Compiler error
Function definition cannot be a typedef'ed declaration—Compiler error
Function 'function' cannot be static—Compiler error
'function' is obsolete—Compiler warning
'function' must be declared with no parameters—Compiler error
'function' must be declared with one parameter—Compiler error
'function' must be declared with two parameters—Compiler error
Function should return a value—Compiler error
Function should return a value—Compiler warning
'function' was previously declared with the language 'language'—Compiler error
'function1' cannot be distinguished from 'function2'—Compiler error
'function1' hides virtual function 'function2'—Compiler warning or error

## G



## H



## I

Illegal component to GRPDEF in module 'module'—Linker error
Illegal initialization—Compiler error
Illegal number suffix—Compiler error
Illegal octal digit—Compiler error
Illegal octal digit—MAKE error
Illegal parameter to __emit__—Compiler error
Illegal pointer subtraction—Compiler error
Illegal structure operation—Compiler error
Illegal to take address of bit field—Compiler error
Illegal type ‘type’ in automated section—Compiler error
Illegal type of entry point—Linker error
Illegal 'type' fixup index in module 'module'—Incremental Linker error
Illegal use of floating point—Compiler error
Illegal use of member pointer—Compiler error
Illegal use of pointer—Compiler error
Illegal/invalid option in CMDSWITCHES directive 'option'—MAKE error
Image base address must be a multiple of 0x10000—Linker error
Image linked as EXE, but with DLL extension—Linker warning
Images fixed at specific addresses typically will not run under Win32s—Linker warning
Implicit conversion of 'type1' to 'type2' not allowed—Compiler error
Import by ordinal not supported by ILINK32—Incremental Linker error
Import record does not match previous definition—Linker warning
Import 'symbol' in module 'module' clashes with prior module—Librarian error
Improper use of typedef 'identifier'—Compiler error
Include files nested too deep—Compiler error
Incompatible type conversion—Compiler error
Incompatible version of RLINK32.DLL—Incremental Linker error
Incorrect command line argument: —MAKE error
Incorrect number format—Compiler error
Incorrect option—Compiler error
Incorrect use of default—Compiler error
Incorrect version of RLINK32.DLL—Linker error
Initialization is only partially bracketed—Compiler warning
Initializer for object 'x' ignored—Compiler warning
Initializing 'identifier with 'identifier'—Compiler error
Initializing enumeration with type—Compiler warning
Initializing 'type' with 'type'—Compiler warning
Inline assembly not allowed—Compiler error
Inline assembly not allowed in inline and template function—Compiler error
Int and string types compared—MAKE error
Internal code generator error—Compiler error
Internal compiler error—Compiler error
Internal failure -- Retrying link...—Incremental Linker error
Invalid combination of opcode and operands—Compiler error
Invalid exe filename: 'filename'—Linker error
Invalid file/object alignment value 'value'—Linker error
Invalid indirection—Compiler error
Invalid macro argument separator—Compiler error
Invalid map filename: 'filename'—Linker error
Invalid overlay switch specification—Linker error
Invalid page size value ignored—Librarian warning
Invalid pointer addition—Compiler error
Invalid register combination (e.g. [BP+BX])—Compiler error

Invalid size specified for segment alignment—Linker error
Invalid size specified for segment packing—Linker error
Invalid stack reserve/commit size 'size'—Linker error
Invalid target /T 'target'—Linker error
Invalid template argument list—Compiler error
Invalid template member definition—Compiler error
Invalid template qualified name 'template::name'—Compiler error
Invalid use of dot—Compiler error
Invalid use of namespace 'identifier'—Compiler error
Invalid use of template 'template'—Compiler error
Irreducible expression tree—Compiler error

## L

Last parameter of 'operator' must have type 'int'—Compiler error
Library contains COMDEF records—extended dictionary not created—Librarian warning
Library too large, please restart with /P 'size'—Librarian error
Library too large, restart with library page size 'size'—Librarian error
Linkage specification not allowed—Compiler error
Linker CREATE Failed—IDE error
Linker missing CREATE Entry Point—IDE error
Linker missing DESTROY Entry Point—IDE error
Link terminated by user—IDE error
Local data exceeds segment size limit—Compiler error
Lvalue required—Compiler error

## M

Macro argument syntax error—Compiler error
Macro definition ignored—Compiler warning
Macro expansion too long—Compiler error
Macro expansion too long—MAKE error
Macro replace text 'string' is too long—MAKE error
Macro substitute text 'string' is too long—MAKE error
'macroname'—')' missing in macro invocation—MAKE error
Main must have a return type of int—Compiler error
Malformed command-line—Linker error
Malloc of number bytes failed in module, line number—Incremental Linker error
Matching base class function 'function' has different dispatch number—Compiler error
Matching base class function 'function' is not dynamic—Compiler error
Maximum precision used for member pointer type type—Compiler error
Member function must be called or its address taken—Compiler error
Member identifier expected—Compiler error
Member is ambiguous: 'member1' and 'member2'—Compiler error
'member' is not accessible—Compiler error
'member' is not a valid template type member—Compiler error
Member 'member' cannot be used without an object—Compiler error
Member 'member' has the same name as its class—Compiler error
Member 'member' is initialized more than once—Compiler error
Member pointer required on right side of .* or ->*—Compiler error
Memory full listing truncated!—Librarian warning
Memory reference expected—Compiler error
Misplaced break—Compiler error
Misplaced continue—Compiler error
Misplaced decimal point—Compiler error
Misplaced elif directive—Compiler error

Misplaced elif statement—MAKE error
Misplaced else—Compiler error
Misplaced else statement—MAKE error
Misplaced else directive—Compiler error
Misplaced endif directive—Compiler error
Misplaced endif statement—MAKE error
'module' already in LIB, not changed!—Librarian warning
'module' contains invalid OMF record, type 0xHH—Incremental Linker error
'module' ILINK32 does not support segmentation - use TLINK32—Incremental Linker error
'module' not found in library—Librarian warning
Mixed common types in module 'module'. Cannot mix COMDEFs and VIRDEFs—Linker error
Mixing pointers to different 'char' types—Compiler error
Mixing pointers to signed and unsigned char—Compiler warning
Multiple base classes not supported for VCL classes—Compiler error
Multiple base classes require explicit class names—Compiler error
Multiple declaration for 'identifier'—Compiler error
Multiple entry points defined—Linker error
Multiple public definitions for symbol 'symbol' in module 'module;' link case sensitively—Linker error
Multiple stack segments found. The most recent one will be used.—Linker warning
Must take address of a memory location—Compiler error

## N

Namespace member 'identifier' declared outside its namespace—Compiler error
Namespace name expected—Compiler error
Need an identifer to declare—Compiler error
Negating unsigned value—Compiler warning
No : following the ?—Compiler error
No base class to initialize—Compiler error
No closing quote—MAKE error
No declaration for function 'function'—Compiler error or warning
No DEF file—Linker warning
No file name ending—Compiler error
No file names given—Compiler error
No filename ending—MAKE error
No internal name for IMPORT in .DEF file—Linker error
No macro before =—MAKE error
No match found for wildcard 'expression'—MAKE error
No output file specified—Linker error
No program entry point—Linker warning
No terminator specified for in-line file operator—MAKE error
No type OBJ file present. Disabling external types option—Compiler warning
Non-ANSI Keyword Used: 'keyword'—Compiler error
Non-const function function called for const object—Compiler error
Non-constant function 'function' called for constant object—Compiler warning
Non-existent segment 'segment' in SEGMENTS section of .DEF file—Linker warning
Non-virtual function 'function' declared pure—Compiler error
Non-volatile function 'function' called for volatile object—Compiler error or warning
Nonportable pointer comparison—Compiler error or warning
Nonportable pointer conversion—Compiler error
Nonportable pointer conversion—Compiler warning
Nontype template argument must be of scalar type—Compiler error
Not an allowed type—Compiler error
Not enough memory—MAKE error
Not enough memory for command-line buffer—Librarian error

Null pointer assignment—Runtime error
Numeric constant too large—Compiler error

## O

Object module 'filename' is invalid—Librarian error
Objects of type 'type' cannot be initialized with { }—Compiler error
Old debug information in module 'module' will be ignored—Linker warning
Only __fastcall functions allowed in __automated section—Compiler error
Only member functions may be 'const' or 'volatile'—Compiler error
Only one of a set of overloaded functions can be functions can be "C"—Compiler error
Only <<KEEP or <<NOKEEP—MAKE error
Only read or write clause allowed in property declaration in __automated section—Compiler error
Operand of 'delete' must be non-const pointer—Compiler error
operator -> must return a pointer or a class—Compiler error
operator [] missing ]—Compiler error
operator delete must return void—Compiler error
Operator delete[] must return void—Compiler error
Operator must be declared as function—Compiler error
'operator' must be declared with one or no parameters—Compiler error
'operator' must be declared with one or two parameters—Compiler error
Operator new must have an initial parameter of type size_t—Compiler error
Operator new must return an object of type void *—Compiler error
Operator new[] must have an initial parameter of type size_t—Compiler error
Operator new[] must return an object of type void—Compiler error
Operators may not have default argument values—Compiler error
Out of disk space—Incremental Linker error
Out of memory—Compiler error
Out of memory—Librarian error
Out of memory—Incremental Linker error
Out of memory creating extended dictionary—Librarian error
Out of memory in block 'address'—Incremental Linker error
Out of memory reading LE/LIDATA record from object module—Librarian error
Out of space allocating per module debug struct—Librarian error
Output device is full—Librarian error
Overlays only supported in medium, large, and huge memory models—Compiler error
'overload' is now unnecessary and obsolete—Compiler warning
Overloadable operator expected—Compiler error
Overloaded 'function name' ambiguous in this context—Compiler error
Overloaded prefix 'operator' used as a postfix operator—Compiler error and warning

## P

Parameter mismatch in 'specifier' access specifier of property 'property'—Compiler error
Parameter names are used only with a function body—Compiler error
Parameter 'number' missing name—Compiler error
Parameter 'parameter' is never used—Compiler error and warning
'path'—path is too long—Librarian error
Pointer to structure required on left side of -> or ->*—Compiler error
Possible unresolved external 'symbol' referenced from module 'module'—Linker warning
Possible use of 'identifier' before definition—Compiler error and warning
Possibly incorrect assignment—Compiler error and warning
Printf/Scanf floating point formats not linked—Runtime error
Public 'symbol' in module 'module1' clashes with prior module 'module2'—Librarian error
Public symbol 'symbol' defined in both module 'module1' and 'module2'—Linker message
Public symbol 'symbol' defined in both module 'module1' and 'module2'—Incremental Linker message

__published or __automated sections only supported for VCL classes—Compiler error
Published property access functions must use __fastcall calling convention—Compiler error
Pure virtual function called—Runtime error

## Q

Qualifier 'identifier' is not a class or namespace name—Compiler error

## R

Realloc of number bytes failed in module, line number—Incremental Linker error
'reason'—extended dictionary not created—Librarian warning
Record kind 'num' found, expected theadr or lheadr in module 'filename'—Librarian error
Record length 'len' exceeds available buffer in module 'module'—Librarian error
Record type 'type' found, expected theadr or lheadr in module—Librarian error
Recursive template function: '" instantiated "'—Compiler error
Redeclaration of property not allowed in __automated sedtion'—Compiler error
Redefinition of 'macro' is not identical—Compiler error or warning
Redefinition of target 'filename'—MAKE error
Reference initialized with 'type1', needs lvalue of type 'type2'—Compiler error
Reference member 'member' Initialized with a non-reference parameter—Compiler error
Reference member 'member' in class without constructors—Compiler error
Reference member 'member' is not initialized—Compiler error
Reference member 'member' needs a temporary for initialization—Compiler error
Reference variable 'variable' must be initialized—Compiler error
Register allocation failure—Compiler error
Restarting compile using assembly—Compiler warning
Results are safe in file 'filename'—Librarian warning
RLINK32 was not initialized—Incremental Linker error
RTTI not available for expression evaluation—Compiler error
Rule line too long—MAKE error

## S

Self relative fixupp overflowed in module 'module'—Linker warning
Side effects are not allowed—Compiler error
Size of 'identifier' is unknown or zero—Compiler error
Size of the type 'identifier' is unknown or zero—Compiler error
Size of the type is unknown or zero—Compiler error
sizeof may not be applied to a bit field—Compiler error
sizeof may not be applied to a function—Compiler error
Specialization after first use of template—Compiler error
'specifier' has already been included—Compiler error
STACK commit 'size' greater than reserve 'size'—Linker error
Stack overflow—Runtime error
Statement missing ;—Compiler error
Static data members not allowed in __published or __automated sections—Compiler error
Srorage class 'storage class' is not allowed here—Compiler error
Srorage specifier not allowed for array properties—Compiler error
String type not allowed with this operand—MAKE error
Structure packing size has changed—Compiler warning
Structure passed by value—Compiler error or warning
Structure required on left side of . or .*—Compiler error
Structure size too large—Compiler error
Style of function definition is now obsolete—Compiler error or warning
Subscripting missing ]—Compiler error
Superfluous & with function—Compiler error or warning
Suspicious pointer arithmetic—Compiler warning

Suspicious pointer conversion—Compiler error or warning
Switch selection expression must be of integral type—Compiler error
Switch statement missing (—Compiler error
Switch statement missing )—Compiler error
Symbol 'symbol' marked as __import in 'module' was public—Incremental Linker error

## T

T3 and T7 fixupps not allowed (module 'module')—Linker error
Target index of FIXUPP is 0 in module 'module'—Linker error
Template argument must be a constant expression—Compiler error
Template class nesting too deep: 'class'—Compiler error
Template function argument 'argument' not used in argument types—Compiler error
Template functions may only have 'type-arguments'—Compiler error
Templates and overloaded operators cannot have C linkage—Compiler error
Templates can only be declared at file level—Compiler error
Templates must be classes or functions—Compiler error
Templates not supported—Compiler error
Temporary used for parameter 'parameter'—Compiler warning
Temporary used for parameter 'number'—Compiler warning
Temporary used for parameter 'number' in call to 'function'—Compiler warning
Temporary used for parameter 'parameter'—Compiler warning
Temporary used for parameter 'parameter'—Compiler warning
Temporary used for parameter 'parameter' in call to 'function'—Compiler warning
Temporary used to initialize 'identifier'—Compiler error or warning
Terminated by user (32-bit)—Linker error
The '...' handler must be last—Compiler error
The combinations '+*' or '*+' are not allowed—Librarian error
The constructor 'constructor' is not allowed—Compiler error
The value for 'identifier' is not within the range of an int—Compiler error
'this' can only be used within a member function—Compiler error
THREAD fixup found in module 'module'—Linker error
Throw expression violates exception specification—Compiler warning
Too few arguments in template class name 'template'—Compiler error
Too few parameters in call—Compiler error
Too few parameters in call to function—Compiler error
Too many commas on command-line—Linker error
Too many decimal points—Compiler error
Too many default cases—Compiler error
Too many default libraries—Linker error
Too many error or warning messages—Compiler error
Too many errors—Linker error
Too many exponents—Compiler error
Too many file names—Linker error
Too many initializers—Compiler error
Too many LNAMEs—Linker error
Too many rules for target 'target'—MAKE error
Too many storage classes in declaration—Compiler error
Too many suffixes in .SUFFIXES list—MAKE error
Too many types in declaration—Compiler error
Too much global data defined in file—Compiler error
Trying to derive a far class from the huge base 'base'—Compiler error
Trying to derive a far class from the near base 'base'—Compiler error
Trying to derive a huge class from the far base 'base'—Compiler error
Trying to derive a huge class from the near base 'base'—Compiler error

## U

## V



## W

Wrong number of arguments in call of macro 'macro'—Compiler error

## Compiler table limit exceeded Compiler message

One of the compiler's internal tables overflowed.

This usually means that the module being compiled contains too many function bodies.

This limitation will not be solved by making more memory available to the compiler. You need to simplify the file being compiled.

## Irreducible expression tree Compiler message

An expression on the indicated line of the source file caused the code generator to be unable to generate code. Avoid using the expression. Notify Borland if an expression consistently reproduces this error.

## Register allocation failure

Compiler message

**Possible Causes**

An expression on the indicated line of the source file was so complicated that the code generator could not generate code for it.

**Solutions**

Simplify the expression. If this does not solve the problem, avoid the expression.

Notify Borland if an expression can consistently reproduce this error.

## Bad call of intrinsic function Compiler message

You have used an intrinsic function without supplying a prototype. You may have supplied a prototype for an intrinsic function that was not what the compiler expected.

## Unable to open input file 'filename' Compiler message

This error occurs if the source file can't be found.

Check the spelling of the name. Make sure the file is on the specified disk or directory.

Verify that the proper directory paths are listed. If multiple paths are required, use a semicolon to separate them.

## **Unable to create output file 'filename'** Compiler message

This error occurs if the work disk is full or write protected.

This error also occurs if the output directory does not exist.

**Solutions**

If the disk is full, try deleting unneeded files and restarting the compilation.

If the disk is write-protected, move the source files to a writeable disk and restart the compilation.

## Error writing output file Compiler message

A DOS error that prevents the C++ IDE from writing an .OBJ, .EXE, or temporary file.

**Solutions**

Make sure that the Output directory in the Directories dialog box is a valid directory.

Check that there is enough free disk space.

## Error directive: 'message'

Compiler message

This message is issued when an **#error** directive is processed in the source file.

`'message'` is the text of the **#error** directive.

## Out of memory Compiler message

The total working storage is exhausted.

This error can occur in the following circumstances:

- Not enough virtual memory is available for compiling a particular file. In this case, shut down any other concurrent applications. You may also try to reconfigure your machine for more available virtual memory, or break up the source file being compiled into smaller separate components. You can also compile the file on a system with more available RAM.
- The compiler has encountered an exceedingly complex or long expression at the line indicated and has insufficient reserves to parse it. Break the expression down into separate statements.

## Unable to open 'filename'

Compiler message

This error occurs if the specified file can't be opened.

Make sure the file is on the specified disk or directory. Verify the proper paths are listed. If multiple paths are required, use a semicolon to separate them.

## **Declaration syntax error** Compiler message

Your source file contained a declaration that was missing a symbol or had an extra symbol added to it. Check for a missing semicolon or parenthesis on that line or on previous lines.

## Wrong number of arguments in call of macro 'macro'

Compiler message

Your source file called the named macro with an incorrect number of arguments.

## Array size too large  Compiler message

The declared array is larger than 64K and the 'huge' keyword was not used.

If you need an array of this size, either use the 'huge' modifier, like this:

```
int huge array[70000L];  /* Allocate 140000 bytes */
```

or dynamically allocate it with farmalloc( ) or farcalloc( ), like this:

```
int huge *array = (int huge *) farmalloc (sizeof (int) * 70000); ?? Allocate
  140,000 bytes
```

## Invalid macro argument separator Compiler message

In a macro definition, arguments must be separated by commas.

The compiler encountered some other character after an argument name.

This is correct:

```
#define tri_add(a, b, c)  ((a) + (b) + (c))
```

This is incorrect:

```
#define tri_add(a  b. c)  ((a) + (b) + (c))
```

## Assembler statement too long Compiler message

Inline assembly statements can't be longer than 480 bytes.

## Macro argument syntax error Compiler message

An argument in a macro definition must be an identifier.

The compiler encountered some non-identifier character where an argument was expected.

## Bad file name format in include directive Compiler message

### OR Bad file name format in line directive

Include and line directive file names must be surrounded by quotes (`"filename.h"`) or angle brackets (`<filename.h>`).

The file name was missing the opening quote or angle bracket.

If a macro was used, the resulting expansion text is not surrounded by quote marks.

## **Invalid indirection** Compiler message

The indirection operator (*) requires a non-void pointer as the operand.

**Example**

```
int main (void)
{
  void *p;
  *p = 10;      /* ERROR: Invalid Indirection */
  return 0;
}
```

## Illegal use of pointer  Compiler message

Pointers can only be used with these operators:

- addition `(+)`
- subtraction `(-)`
- assignment `(=)`
- comparison `(==)`
- indirection `(*)`
- arrow `(->)`

Your source file used a pointer with some other operator.

**Example**

```
int main (void)
{
  char *p;
  p /= 7;      /* ERROR: Illegal Use of Pointer */
  return 0;
}
```

## Not an allowed type Compiler message

Your source file declared some sort of forbidden type; for example, a function returning a function or array.

## Incompatible type conversion Compiler message

The cast requested can't be done.

## **Misplaced decimal point** Compiler message

The compiler encountered a decimal point in a floating-point constant as part of the exponent.

## Incorrect use of default Compiler message

The compiler found no colon after the default keyword.

## Invalid use of dot Compiler message

An identifier must immediately follow a period operator (.).

**Example**

```
struct foo {
  int x;
  int y;
}p = 0,0;
int main (void)
{
  p.x++;          /* Correct */
  p.  y++;        /* Error: Invalid use of dot */
  return 0;
}
```

## Function call missing )  Compiler message

The function call argument list had some sort of syntax error, such as a missing or mismatched right parenthesis.

## Case statement missing : Compiler message

A case statement must have a constant expression followed by a colon.

The expression in the case statement either was missing a colon or had an extra symbol before the colon.

## Character constant must be one or two characters long Compiler message

Character constants can only be one or two characters long.

## Compound statement missing }

Compiler message

The compiler reached the end of the source file and found no closing brace.

This is most commonly caused by mismatched braces.

## Cannot modify a const object Compiler message

This indicates an illegal operation on an object declared to be const, such as an assignment to the object.

## Declaration missing ;  Compiler message

Your source file contained a struct or union field declaration that was not followed by a semicolon. Check previous lines for a missing semicolon.

## Define directive needs an identifier Compiler message

The first non-whitespace character after a **#define** must be an identifier.

The compiler found some other character.

## Too much global data defined in file Compiler message

The sum of the global data declarations exceeds 64K bytes. This includes any data stored in the DGROUP (all global variables, literal strings, and static locals).

**Solutions**

Check the declarations for any array that might be too large. You can also remove variables from the DGROUP.

Here's how:

- Declare the variables as automatic. This uses stack space.
- Dynamically allocate memory from the heap using calloc, malloc, or farmalloc for the variables. This requires the use of pointers.

Literal strings are also put in the DGROUP. Get the file farstr.zip from our BBS to extract literal strings into their own segment.

## Two consecutive dots Compiler message

Because an ellipsis contains three dots (...), and a decimal point or member selection operator uses one dot (.), two consecutive dots cannot legally occur in a C program.

## Too many storage classes in declaration Compiler message

A declaration can never have more than one storage class, either Auto, Register, Static, or Extern.

## Too many types in declaration Compiler message

A declaration can never have more than one of these basic types:

- char
- class
- int
- float
- double
- struct
- union
- enum
- typedef name

## Misplaced elif directive Compiler message

The compiler encountered an **#elif** directive without any matching **#if**, **#ifdef**, or **#ifndef** directive.

## Misplaced else directive Compiler message

The compiler encountered an **#else** directive without any matching **#if**, **#ifdef**, or **#ifndef** directive.

## Misplaced endif directive

Compiler message

The compiler encountered an **#endif** directive without any matching **#if**, **#ifdef**, or **#ifndef** directive.

## Enum syntax error

Compiler message

An **enum** declaration did not contain a properly formed list of identifiers.

## Unexpected end of file in comment started on 'line number'

Compiler message

The source file ended in the middle of a comment.

This is normally caused by a missing close of comment (*/).

## Unexpected end of file in conditional started on 'line number'  Compiler message

The source file ended before the compiler (or MAKE) encountered **#endif**.

The **#endif** either was missing or misspelled.

Every **#if** statement needs a matching **#endif** statement.

## Expression syntax Compiler message

This is a catch-all error message when the compiler parses an expression and encounters a serious error.

### Possible Causes

This is most commonly caused by one of the following:

- two consecutive operators
- mismatched or missing parentheses
- a missing semicolon on the previous statement.

### Solutions

If the line where the error occurred looks syntactically correct, look at the line directly above for errors.

Try moving the line with the error to a different location in the file and recompiling.

If the error still occurs at the moved statement, the syntax error is occurring somewhere in that statement.

If the error occurred in another statement, the syntax error is probably in the surrounding code.

## Too few parameters in call

Compiler message

This error message occurs when a call to a function with a prototype (via a function pointer) had too few arguments. Prototypes require that all parameters be given. Make certain that your call to a function has the same parameters as the function prototype.

## Too few parameters in call to 'function'

Compiler message

A call to the named function (declared using a prototype) has too few arguments.

Make certain that the parameters in the call to the function match the parameters of the function prototype.

## Illegal use of floating point  Compiler message

Floating-point operands are not allowed in these operators

- shift (SHL, SHR)
- bitwise Boolean (AND, OR, XOR, NOT)
- conditional (? :)
- indirection (*)
- certain others

The compiler found a floating-point operand with one of these prohibited operators.

## File name too long

Compiler message

The file name given in an **#include** directive was too long for the compiler to process.

File names in DOS must be no more than 79 characters long.

## Conflicting type modifiers Compiler message

This occurs when a declaration is given that includes more than one addressing modifier on a pointer or more than one language modifier for a function.

Only one language modifier (**cdecl** and **pascal**) can be given for a function.

One cannot multiply derive from a class declared to use the fast this pointer optimization, and one that was not. For example:

```
class __fastthis A { // one way to declare a class as using the
   myex();           // fast this optimization, note that
};                   // #pragma option -po- turns it off.
class B {
   twoex();
};
class c : A , B {}; // error
// note that __fastthis is only recognized in BC 4.0 or later
```

## Goto statement missing label Compiler message

The **goto** keyword must be followed by an identifier.

## Group overflowed maximum size: 'name'

Compiler message

The total size of the segments in a group (for example, DGROUP) exceeded 64K.

## Illegal character 'character' (0x'value')

Compiler message

The compiler encountered some invalid character in the input file.

The hexadecimal value of the offending character is printed.

This can also be caused by extra parameters passed to a function macro.

## Illegal initialization

Compiler message

Initializations must be one of the following:

- constant expressions
- the address of a global extern or static variable plus or minus a constant

## Unable to open include file 'filename' Compiler message

The compiler could not find the named file.

**Possible Causes**

- The named file does not exist.
- An **#include** file included itself.
- You do not have FILES set in CONFIG.SYS on your root directory.

**Solutions**

- Verify that the named file exists.
- Set FILES = 20 in CONFIG.SYS.

## No file name ending

Compiler message

The file name in an **#include** statement was missing the correct closing quote or angle bracket.

## Lvalue required Compiler message

The left side of an assignment operator must be an addressable expression.

Addressable expressions include the following:

- numeric or pointer variables
- structure field references or indirection through a pointer
- a subscripted array element

## Too many error or warning messages

Compiler message

There were more errors or warnings than allowed.

## statement missing (  Compiler message

In a do, for, if, switch, or while statement, the compiler found no left parenthesis after the while keyword or test expression.

## statement missing )

Compiler message

In a do, for, if, switch, or while statement, the compiler found no right parenthesis after the while keyword or test expression.

## do-while statement missing OR For statement missing ; Compiler message

In a do or for statement, the compiler found no semicolon after the right parenthesis.

## Type mismatch in redeclaration of 'identifier'

Compiler message

Your source file redeclared a variable with a different type than was originally declared for the variable.

**Possible Causes**

This can occur if a function is called and subsequently declared to return something other than an integer.

**Solutions**

If this has happened, you must declare the function before the first call to it.

## Default outside of switch

Compiler message

The compiler encountered a default statement outside a switch statement.

This is most commonly caused by mismatched braces.

## Macro expansion too long

Compiler message

A macro can't expand to more than 4,096 characters.

## Too many decimal points

Compiler message

The compiler encountered a floating-point constant with more than one decimal point.

## Too many exponents Compiler message

The compiler encountered more than one exponent in a floating-point constant.

## Too many initializers Compiler message

The compiler encountered more initializers than were allowed by the declaration being initialized.

## Inline assembly not allowed Compiler message

Your source file contains inline assembly language statements and you are compiling it from within the integrated environment.

You must use the BCC command to compile this source file from the DOS command line.

## Incorrect number format Compiler message

The compiler encountered a decimal point in a hexadecimal number.

## Numeric constant too large Compiler message

String and character escape sequences larger than hexadecimal or octal 77 can't be generated.

Two-byte character constants can be specified by using a second backslash. For example,

```
\\
```

represents a two-byte constant.

A numeric literal following an escape sequence should be broken up like this:

```
printf("" "12345");
```

This prints a carriage return followed by 12345.

## Illegal octal digit Compiler message

The compiler found an octal constant containing a non-octal digit (8 or 9).

## Type mismatch in parameter 'number' in call to 'function'

Compiler message

Your source file declared the named function with a prototype, and the given parameter number (counting left to right from 1) could not be converted to the declared parameter type.

When compiling C++ programs, this message is always preceded by another message that explains the exact reason for the type mismatch.

That other message is usually "Cannot convert 'type1' to 'type2'", but the mismatch might be due to many other reasons.

## Type mismatch in parameter 'parameter' in call to 'function'

Compiler message

Your source file declared the named function with a prototype, and the named parameter could not be converted to the declared parameter type.

When compiling C++ programs, this message is always preceded by another message that explains the exact reason for the type mismatch.

That other message is usually "Cannot convert 'type1' to 'type2'" but the mismatch might be due to many other reasons.

## Type mismatch in parameter 'number'

Compiler message

The function called, via a function pointer, was declared with a prototype.

However, the given parameter number (counting left to right from 1) could not be converted to the declared parameter type.

When compiling C++ programs, this message is always preceded by another message that explains the exact reason for the type mismatch.

That other message is usually "Cannot convert 'type1' to 'type2'" but the mismatch might be due to many other reasons.

## Type mismatch in parameter 'parameter'

Compiler message

Your source file declared the function called via a function pointer with a prototype.

However, the named parameter could not be converted to the declared parameter type.

When compiling C++ programs, this message is always preceded by another message that explains the exact reason for the type mismatch.

That other message is usually "Cannot convert 'type1' to 'type2'" but the mismatch might be due to many other reasons.

## Pointer to structure required on left side of -> or ->*

Compiler message

Nothing but a pointer is allowed on the left side of the arrow (->) in C or C++.

In C++ a -> operator is allowed.

## Invalid pointer addition Compiler message

Your source file attempted to add two pointers together.

## Illegal pointer subtraction Compiler message

This is caused by attempting to subtract a pointer from a non-pointer.

## Illegal structure operation

Compiler message

Structures can only be used with dot (.), address-of (&) or assignment (=) operators, or be passed to or from a function as parameters.

The compiler encountered a structure being used with some other operator.

## Bad 'directive' directive syntax Compiler message

A macro definition starts or ends with the ## operator, or contains the # operator that is not followed by a macro argument name.

An example of this might be:

```
Bad ifdef directive syntax
```

Note that an **#ifdef** directive must contain a single identifier (and nothing else) as the body of the directive.

Another example is:

```
Bad undef directive syntax
```

An **#undef** directive must also contain only one identifier as the body of the directive.

## Unterminated string or character constant Compiler message

The compiler found no terminating quote after the beginning of a string or character constant.

## Structure size too large Compiler message

Your source file declared a structure larger than 64K.

## Unknown preprocessor directive: 'identifier'

Compiler message

The compiler encountered a # character at the beginning of a line. The directive name that followed the # was not one of the following:

- define
- else
- endif
- if
- ifdef
- ifndef
- include
- line
- undef

## Undefined symbol 'identifier' Compiler message

The named identifier has no declaration.

**Possible Causes**

- actual declaration of identifier has been commented out.
- misspelling, either at this point or at the declaration.
- there was an error in the declaration of the identifier.

Tools to help track down the problem:

- CPP
- GREP

## Could not find file 'filename' Compiler message

The compiler is unable to find the file supplied on the command line.

## Non-portable pointer conversion Compiler warning

(Command-line option to suppress warning: **-w-rpt**)

An implicit conversion between a pointer and an integral type is required, but the types are not the same size. You must use an explicit cast.

This conversion might not make any sense, so be sure this is what you want to do.

## Extra parameter in call Compiler message

A call to a function, via a pointer defined with a prototype, had too many arguments.

## Extra parameter in call to function Compiler message

A call to the named function (which was defined with a prototype) had too many arguments given in the call.

## User break Compiler message

You typed a Ctrl+Break while compiling in the IDE.
(This is not an error, just a confirmation.)

## Multiple base classes require explicit class names

Compiler message

In a C++ class constructor, if there is more than one immediate base class, each base class constructor call in the constructor header must include the base class name.

## Member is ambiguous: 'member1' and 'member2' Compiler message

You must qualify the member reference with the appropriate base class name.

In C++ class 'class', member 'member' can be found in more than one base class, and it was not qualified to indicate which one you meant.

This applies only in multiple inheritance, where the member name in each base class is not hidden by the same member name in a derived class on the same path.

The C++ language rules require that this test for ambiguity be made before checking for access rights (private, protected, public).

It is possible to get this message even though only one (or none) of the members can be accessed.

## 'function1' cannot be distinguished from 'function2'

Compiler message

The parameter type lists in the declarations of these two functions do not differ enough to tell them apart.

Try changing the order of parameters or the type of a parameter in one declaration.

## Attempt to grant or reduce access to 'identifier'

Compiler message

A C++ derived class can modify the access rights of a base class member, but only by restoring it to the rights in the base class.

It can't add or reduce access rights.

## Array must have at least one element Compiler message

ANSI C and C++ require that an array be defined to have at least one element (objects of zero size are not allowed).

An old programming trick declares an array element of a structure to have zero size, then allocates the space actually needed with malloc.

You can still use this trick, but you must declare the array element to have (at least) one element if you are compiling in strict ANSI mode.

Declarations (as opposed to definitions) of arrays of unknown size are still allowed.

**Example**

```
char ray[];         /* definition of unknown size -- ILLEGAL */
char ray[0];        /* definition of 0 size -- ILLEGAL */
extern char ray[];  /* declaration of unknown size -- OK */
```

## operator -> must return a pointer or a class Compiler message

The C++ operator -> function must be declared to either return a class or a pointer to a class (or struct or union).

In either case, it must be something to which the -> operator can be applied.

## 'identifier' must be a previously defined class or struct Compiler message

You are attempting to declare 'identifier' to be a base class, but either it is not a class or it has not yet been fully defined.

Correct the name or rearrange the declarations.

## Misplaced break Compiler message

The compiler encountered a break statement outside a switch or looping construct.

You can only use break statements inside of switch statements or loops.

## Switch selection expression must be of integral type

Compiler message

The selection expression in parentheses in a **switch** statement must evaluate to an integral type (**char, short, int, long, enum**).

You might be able to use an explicit cast to satisfy this requirement.

## Cannot cast from 'type1' to 'type2' Compiler message

A cast from type 'ident1' to type 'ident2' is not allowed.

In C++, you cannot cast a member function pointer to a normal function pointer.

For example:

```
class A {
public:
   int myex();
};
typedef int (*fp)();
test()
{
   fp myfp - (fp) &A::myex; //error
```

The reason being that a class member function takes a hidden parameter, the this pointer, thus it behaves very differently than a normal function pointer.

A static member function behaves as normal function pointer and can be cast.

For example:

```
class A {
public:
   static int myex();
};
typedef int (*fp)();
test()
{
   fp myfp - (fp) &A::myex; //ok
```

However, static member functions can only access static data members of the class.

**In C**

- A pointer can be cast to an integral type or to another pointer.
- An integral type can be cast to any integral, floating, or pointer type.
- A floating type can be cast to an integral or floating type.

Structures and arrays can't be cast to or from.

You usually can't cast from a void type.

**In C++**

User-defined conversions and constructors are checked for. If one can't be found, the preceding rules apply (except for pointers to class members).

Among integral types, only a constant zero can be cast to a member pointer.

A member pointer can be cast to an integral type or to a similar member pointer.

A similar member pointer points to a data member (or to a function) if the original does. The qualifying class of the type being cast to must be the same as (or a base class of) the original.

## Constructor cannot have a return type specification

Compiler message

C++ constructors have an implicit return type used by the compiler, but you can't declare a return type or return a value.

## Misplaced continue Compiler message

The compiler encountered a continue statement outside a looping construct.

## Declaration terminated incorrectly Compiler message

A declaration has an extra or incorrect termination symbol, such as a semicolon placed after a function body.

A C++ member function declared in a class with a semicolon between the header and the opening left brace also generates this error.

## Need an identifier to declare Compiler message

In this context, an identifier was expected to complete the declaration.

This might be a typedef with no name, or an extra semicolon at file level.

In C++, it might be a class name improperly used as another kind of identifier.

## Default value missing Compiler message

When a C++ function declares a parameter with a default value, all of the following parameters must also have default values.

In this declaration, a parameter with a default value was followed by a parameter without a default value.

## Declare operator delete (void*) or (void*, size_t)

## Declare operator delete[] (void*) or (void*, size_t) Compiler message

Declare the operator delete with one of the following:

1. A single void* parameter, or
2. A second parameter of type size_t

If you use the second version, it will be used in preference to the first version.

The global operator delete can only be declared using the single-parameter form.

## operator delete must return void

## operator delete[] must return void Compiler message

This C++ overloaded operator delete was declared in some other way.

Declare the operator delete with one of the following:

1. A single void* parameter, or
2. A second parameter of type size_t

If you use the second version, it will be used in preference to the first version.

The global operator delete can only be declared using the single-parameter form.

## Destructor name must match the class name

Compiler message

In a C++ class, the tilde (~) introduces a declaration for the class destructor.

The name of the destructor must be same as the class name.

In your source file, the ~ preceded some other name.

## Destructor cannot have a return type specification

Compiler message

C++ destructors never return a value, and you can't declare a return type or return a value.

## Default value missing following parameter 'parameter' Compiler message

All parameters following the first parameter with a default value must also have defaults specified.

## Misplaced else Compiler message

The compiler encountered an **else** statement without a matching if statement.

**Possible Causes**

- An extra "else" statement
- An extra semicolon
- Missing braces
- Some syntax error in a previous "if" statement

## Unknown language, must be C or C++ Compiler message

In the C++ construction

```
extern "name" type func( /*...*/ );
```

the given "name" must be "C" or "C++" (use the quotes); other language names are not recognized.

You can declare an external Pascal function without the compiler's renaming like this:

```
extern "C" int pascal func( /*...*/ );
```

To declare a (possibly overloaded) C++ function as Pascal and allow the usual compiler renaming (to allow overloading), you can do this:

```
extern int pascal func( /*...*/ );
```

## 'function' was previously declared with the language 'language'

Compiler message

Only one language modifier (cdecl pascal) can be given for a function.

This function has been declared with different language modifiers in two locations.

## Parameter names are used only with a function body

Compiler message

When declaring a function (not defining it with a function body), you must use either empty parentheses or a function prototype.

A list of parameter names only is not allowed.

**Example declarations**

```
int func();             /* declaration without prototype -- OK */
int func(int, int);     /* declaration with prototype -- OK */
int func(int i, int j); /* parameter names in prototype -- OK */
int func(i, j);         /* parameter names only -- ILLEGAL */
```

## Extern variable cannot be initialized Compiler message

The storage class extern applied to a variable means that the variable is being declared but not defined here--no storage is being allocated for it.

Therefore, you can't initialize the variable as part of the declaration.

## Cannot initialize 'type1' with 'type2' Compiler message

You are attempting to initialize an object of type 'type1' with a value of type 'type2' which is not allowed. The rules for initialization are essentially the same as for assignment.

## Objects of type 'type' cannot be initialized with {} Compiler message

Ordinary C structures can be initialized with a set of values inside braces.

C++ classes can only be initialized with constructors if the class has constructors, private members, functions, or base classes that are virtual.

## Operator new must return an object of type void *

## Operator new[] must return an object of type void *

Compiler message

This C++ overloaded operator new was declared in some other way.

## 'function' must be declared with no parameters Compiler message

This C++ operator function was incorrectly declared with parameters.

## 'function' must be declared with one parameter Compiler message

This C++ operator function was incorrectly declared with more than one parameter.

## 'function' must be declared with two parameters Compiler message

This C++ operator function was incorrectly declared with other than two parameters.

## 'identifier' must be a member function or have a parameter of class type

Compiler message

Most C++ operator functions must have an implicit or explicit parameter of class type.

This operator function was declared outside a class and does not have an explicit parameter of class type.

## Only one of a set of overloaded functions can be "C" Compiler message

C++ functions are by default overloaded, and the compiler assigns a new name to each function.

If you wish to override the compiler's assigning a new name by declaring the function **extern** "C", you can do this for only one of a set of functions with the same name.

(Otherwise the linker would find more than one global function with the same name.)

## 'identifier' is not a parameter Compiler message

In the parameter declaration section of an old-style function definition, 'identifier' is declared but not listed as a parameter. Either remove the declaration or add 'identifier' as a parameter.

## Qualifier 'identifier' is not a class or namespace name Compiler message

The C++ qualifier in the construction qual::identifier is not the name of a struct or class.

## Two operands must evaluate to the same type Compiler message

The types of the expressions on both sides of the colon in the conditional expression operator (`?:`) must be the same, except for the usual conversions.

These are some examples of usual conversions

- **char** to **int**
- **float** to **double**
- void* to a particular pointer

In this expression, the two sides evaluate to different types that are not automatically converted.

This might be an error or you might merely need to cast one side to the type of the other.

When compiling C++ programs, this message is always preceded by another message that explains the exact reason for the type mismatch.

That other message is usually "Cannot convert 'type1' to 'type2'" but the mismatch might be due to many other reasons.

## 'this' can only be used within a member function Compiler message

In C++, "this" is a reserved word that can be used only within class member functions.

## Virtual function 'function1' conflicts with base class 'base'

Compiler message

A virtual function has the same argument types as one in a base class, but a different return type. This is illegal.

## Bit field too large Compiler message

This error occurs when you supply a bit field with more than 16 bits.

## Bit fields must contain at least one bit

Compiler message

You can't declare a named bit field to have 0 (or less than 0) bits.

You can declare an unnamed bit field to have 0 bits.

This is a convention used to force alignment of the following bit field to a byte boundary (or to a word boundary.

## Overloadable operator Compiler message

Almost all C++ operators can be overloaded.

These are the only ones that can't be overloaded:

- the field-selection dot (`.`)
- dot-star (`.*`)
- double colon (`::`)
- conditional expression (`?:`)

The preprocessor operators (# and ##) are not C or C++ language operators and thus can't be overloaded.

Other non-operator punctuation, such as semicolon (;), can't be overloaded.

## Default argument value redeclared Compiler message

When a parameter of a C++ function is declared to have a default value, this value can't be changed, redeclared, or omitted in any other declaration for the same function.

## Default argument value redeclared for parameter 'parameter' Compiler message

When a parameter of a C++ function is declared to have a default value, this value can't be changed, redeclared, or omitted in any other declaration for the same function.

## Declaration is not allowed here

Compiler message

Declarations can't be used as the control statement for while, for, do, if, or switch statements.

## Array allocated using 'new' may not have an initializer Compiler message

When initializing a vector (array) of classes, you must use the constructor that has no arguments.

This is called the default constructor, which means that you can't supply constructor arguments when initializing such a vector.

## Trying to derive a far class from the near base 'base'

Compiler message

If a class is declared (or defaults to) near, all derived classes must also be near.

## Trying to derive a near class from the far base 'base'

Compiler message

If a class is declared (or defaults to) far, all derived classes must also be far.

## Destructor for class is not accessible

Compiler message

The destructor for this C++ class is protected or private, and can't be accessed here to destroy the class.

If a class destructor is private, the class can't be destroyed, and thus can never be used. This is probably an error.

A protected destructor can be accessed only from derived classes.

This is a useful way to ensure that no instance of a base class is ever created, but only classes derived from it.

## Division by zero Compiler message

Your source file contains a divide or remainder in a constant expression with a zero divisor.

## Too many default cases Compiler message

The compiler encountered more than one default statement in a single switch.

## 'identifier' specifies multiple or duplicate access Compiler message

A base class can be declared public or private, but not both.

This access specifier can appear no more than once for a base class.

## Base class 'class' is included more than once Compiler message

A C++ class can be derived from any number of base classes, but can be directly derived from a given class only once.

## Duplicate case Compiler message

Each case of a switch statement must have a unique constant expression value.

## Member 'member' is initialized more than once Compiler message

In a C++ class constructor, the list of initializations following the constructor header includes the same member name more than once.

## Multiple declaration for 'identifier'  Compiler message

This identifier was improperly declared more than once.

This might be caused by conflicting declarations such as:

- `int a; double a;`
- a function declared two different ways, or
- a label repeated in the same function, or
- some declaration repeated other than an extern function or a simple variable

This can also happen by inadvertently including the same header file twice. For example, given:

```
//a.h
struct A { int a; };

//b.h
#include "a.h"

//myprog.cpp
#include "a.h"
#include "b.h"
```

myprog.cpp will get two declarations for the struct A. To protect against this, one would write the a.h header file as:

```
//a.h
#ifndef __A_H
#define __A_H

struct A { int a; };

#endif
```

This will allow one to safely include a.h several times in the same source code file.

## Base class 'class' is initialized more than once Compiler message

In a C++ class constructor, the list of initializations following the constructor header includes base class 'class' more than once.

## 'specifier' has already been included Compiler message

This type specifier occurs more than once in this declaration.

Delete or change one of the occurrences.

## Body has already been defined for function 'function' Compiler message

A function with this name and type was previously supplied a function body.

A function body can only be supplied once.

One cause of this error is not declaring a default constructor which you implement. For example:

```
class A {
public:
   virtual myex();
};
A::A() {} // error
```

Having not seen you declare the default constructor in the class declaration, the compiler has had to generate one, thus giving the error message when it sees one. this is a correct example:

```
class A {
public:
   A();
   virtual myex();
};
A::A() {}
```

## virtual specified more than once

Compiler message

The C++ reserved word "virtual" can appear only once in one member function declaration.

## File must contain at least one external declaration

Compiler message

This compilation unit was logically empty, containing no external declarations.

ANSI C and C++ require that something be declared in the compilation unit.

## 'member' is not accessible Compiler message

You are trying to reference C++ class member 'member,' but it is private or protected and can't be referenced from this function.

This sometimes happens when you attempt to call one accessible overloaded member function (or constructor), but the arguments match an inaccessible function.

The check for overload resolution is always made before checking for accessibility.

If this is the problem, try an explicit cast of one or more parameters to select the desired accessible function.

Virtual base class constructors must be accessible within the scope of the most derived class. This is because C++ always constructs virtual base classes first, no matter how far down the hierarchy they are. For example:

```
class A {
public:
   A();
};
class B : private virtual A {};

class C : private B {
public:
C();
};

C::C() {} // error, A::A() is not accessible
```

Since A is private to B, which is private to C, it makes A's constructor not accessible to C. However, the constructor for C must be able to call the constructors for its virtual base class, A. If B inherits A publicly, the above example would compile.

## Function defined inline after use as extern Compiler message

Functions can't become inline after they have already been used.

Either move the inline definition forward in the file or delete it entirely.

The compiler encountered something like:

```
myex();
twoex() { myex(); }
inline myex() { return 2; } // error
```

and already used the function as an extern before it saw that it was specified as inline. This would be correct:

```
myex();
inline myex() { return 2; }

twoex() { myex(); }
```

or better:

```
inline myex();
inline myex() { return 2; }

twoex() { myex(); }
```

## Undefined label 'identifier'

Compiler message

The named label has a **goto** in the function, but no label definition.

## Unexpected }  Compiler message

An extra right brace was encountered where none was expected. Check for a missing {.

**Useful Tip:**
The IDE has a mechanism for finding a matching curly brace. If you put the cursor on the '{' or '}' character, hold down control, hit 'Q' and then '{' or '}', it will position the cursor on the matching brace.

## Ambiguity between 'function1' and 'function2'

Compiler message

Both of the named overloaded functions could be used with the supplied parameters.

This ambiguity is not allowed.

## Class member 'member' declared outside its class

Compiler message

C++ class member functions can be declared only inside the class declaration.

Unlike nonmember functions, they can't be declared multiple times or at other locations.

## 'constructor' is not an unambiguous base class of 'class'

Compiler message

A C++ class constructor is trying to call a base class constructor 'constructor.'

This error can also occur if you try to change the access rights of 'class::constructor.'

Check your declarations.

## 'identifier' must be a member function

Compiler message

Most C++ operator functions can be members of classes or ordinary non-member functions, but these are required to be members of classes:

- operator =
- operator ->
- operator ( )
- type conversions

This operator function is not a member function but should be.

## Parameter 'number' missing name  Compiler message

In a function definition header, this parameter consisted only of a type specifier 'number' with no parameter name.

This is not legal in C.

(It is allowed in C++, but there's no way to refer to the parameter in the function.)

## No base class to initialize

Compiler message

This C++ class constructor is trying to implicitly call a base class constructor, but this class was declared with no base classes.

Check your declarations.

## Illegal to take address of bit field

Compiler message

It is not legal to take the address of a bit field, although you can take the address of other kinds of fields.

## Cannot initialize a class member here

Compiler message

Individual members of structs, unions, and C++ classes can't have initializers.

A struct or union can be initialized as a whole using initializers inside braces.

A C++ class can only be initialized by the use of a constructor.

## : expected after private/protected/private

Compiler message

When used to begin a private, protected, or public section of a C++ class, the reserved words "private," "protected," and "public" must be followed by a colon.

## No : following the ?

Compiler message

The question mark (?) and colon (:) operators do not match in this expression.

The colon might have been omitted, or parentheses might be improperly nested or missing.

## , expected Compiler message

A comma was expected in a list of declarations, initializations, or parameters.

This problem is often caused by a missing syntax element earlier in the file or one of its included headers.

## Cannot find 'class::class' ('class'&) to copy a vector Compiler message

### OR Cannot find 'class'::operator=('class'&) to copy a vector

**Cannot find class::class ...**

When a C++ class 'class1' contains a vector (array) of class 'class2', and you want to construct an object of type 'class1' from another object of type 'class 1', you must use this constructor:

```
class2::class2(class2&)
```

so that the elements of the vector can be constructed.

The constructor, called a copy constructor, takes just one parameter (which is a reference to its class).

Usually, the compiler supplies a copy constructor automatically.

However, if you have defined a constructor for class 'class2' that has a parameter of type 'class2&' and has additional parameters with default values, the copy constructor can't exist and can't be created by the compiler.

This is because these two can't be distinguished:

```
class2::class2(class2&)
class2::class2(class2&, int = 1)
```

You must redefine this constructor so that not all parameters have default values.

You can then define a reference constructor or let the compiler create one.

**Cannot find class::operator= ...**

When a C++ class 'class1' contains a vector (array) of class 'class2', and you want to copy a class of type 'class1', you must use this assignment operator:

```
class2::class2(class2&)
```

so that the elements of the vector can be copied.

Usually, the compiler automatically supplies this operator.

However, if you have defined an operator= for class 'class2' that does not take a parameter of type 'class2&,' the compiler will not supply it automatically--you must supply one.

## Declaration was expected

Compiler message

A declaration was expected here but not found.

This is usually caused by a missing delimiter such as a comma, semicolon, right parenthesis, or right brace.

## 'identifier' must be a previously defined enumeration tag

Compiler message

This declaration is attempting to reference 'ident' as the tag of an enum type, but it has not been so declared.

Correct the name, or rearrange the declarations.

## Expression expected Compiler message

An expression was expected here, but the current symbol can't begin an expression.

This message might occur where the controlling expression of an if or while clause is expected or where a variable is being initialized.

This message is often due to a symbol that is missing or has been added.

## Member identifier expected Compiler message

The name of a structure or C++ class member was expected here, but not found. The right side of a dot (.) or arrow (->) operator must be the name of a member in the structure or class on the left of the operator.

## 'identifier' is not a member of 'struct' Compiler message

You are trying to reference 'identifier' as a member of 'struct', but it is not a member.

Check your declarations.

## Friends must be functions or classes Compiler message

A friend of a C++ class must be a function or another class.

## Functions may not be part of a struct or union

Compiler message

This C struct or union field was declared to be of type function rather than pointer to function. Functions as fields are allowed only in C++.

## Identifier expected Compiler message

An identifier was expected here, but not found.

In C, an identifier is expected in the following situations:

- in a list of parameters in an old-style function header
- after the reserved words struct or union when the braces are not present, and
- as the name of a member in a structure or union (except for bit fields of width 0).

In C++, an identifier is also expected in these situations:

- in a list of base classes from which another class is derived, following a double colon (::), and
- after the reserved word "operator" when no operator symbol is present.

## Constant/Reference variable 'variable' must be initialized

Compiler message

This C++ object is declared constant or as a reference, but is not initialized.

It must be initialized at the point of declaration.

## Inline assembly not allowed in inline and template functions Compiler message

The compiler can't handle inline assembly statements in a C++ inline or template function.

You could eliminate the inline assembly code or, in the case of an inline function, make this a macro, and remove the inline storage class.

## { expected Compiler message

A left brace was expected at the start of a block or initialization.

## Attempting to return a reference to local variable 'identifier'

Compiler message

This C++ function returns a reference type, and you are trying to return a reference to a local (auto) variable.

This is illegal, because the variable referred to disappears when the function exits.

You can return a reference to any static or global variable, or you can change the function to return a value instead.

## ( expected Compiler message

A left parenthesis was expected before a parameter list.

## Reference member 'member' is not initialized Compiler message

References must always be initialized, in the constructor for the class.

A class member of reference type must have an initializer provided in all constructors for that class.

This means you can't depend on the compiler to generate constructors for such a class, because it has no way of knowing how to initialize the references.

## Could not find a match for argument(s)

Compiler message

No C++ function could be found with parameters matching the supplied arguments. Check parameters passed to function or overload function for parameters that are being passed.

## Use :: to take the address of a member function Compiler message

If f is a member function of class c, you take its address with the syntax

```
&c::f
```

Note the use of the class type name (not the name of an object) and the :: separating the class name from the function name.

(Member function pointers are not true pointer types, and do not refer to any particular instance of a class.)

## Cannot find default constructor to initialize base class 'class' Compiler message

Whenever a C++ derived class 'class2' is constructed, each base class 'class1' must first be constructed.

If the constructor for 'class2' does not specify a constructor for 'class1' (as part of 'class2's' header), there must be a constructor `class1::class1()` for the base class.

This constructor without parameters is called the default constructor.

The compiler will supply a default constructor automatically unless you have defined any constructor for class 'class1'.

In that case, the compiler will not supply the default constructor automatically--you must supply one.

```
class Base {
public:
   Base(int) {}
};
class Derived = public Base {
   Derived():Base(1) {}
}

// must explicitly call the Base constructor, or provide a
// default constructor in Base.
```

Class members with constructors must be initialized in the class' initializer list, for example:

```
class A {
public
   A( int );
};
class B {
public:
   A a;
   B() : a( 3 ) {}; //ok
};
```

## Cannot find default constructor to initialize member 'identifier' Compiler message

When the following occurs

1. A C++ class 'class1' contains a member of class 'class2,'

and

2. You want to construct an object of type 'class1' (but not from another object of type 'class1'). There must be a constructor class2::class2() so that the member can be constructed.

This constructor without parameters is called the default constructor.

The compiler will supply a default constructor automatically unless you have defined any constructor for class 'class2'.

In that case, the compiler will not supply the default constructor automaticallyyou must supply one.

## Use . or -> to call function

Compiler message

You attempted to call a member function without providing an object. This is required to call a member function.

```
class X {
   member func() {}
};
X x;
X*xp = new X;
X.memberfunc();
Xp-> memberfunc();
```

## missing ] Compiler message

This error is generated if any of the following occur:

- Your source file declared an array in which the array bounds were not terminated by a right bracket.
- The array specifier in an operator is missing a right bracket.
- The operator [ ] was declared as operator [.
- A right bracket is missing from a subscripting expression.

Add the bracket or fix the declaration.

Check for a missing or extra operator or mismatched parentheses.

### } expected Compiler message

A right brace was expected at the end of a block or initialization.

## Must take address of a memory location

Compiler message

Your source file used the address-of operator (&) with an expression that can't be used that way; for example, a register variable.

## ) expected Compiler message

A right parenthesis was expected at the end of a parameter list.

## Statement missing ;  Compiler message

The compiler encountered an expression statement without a semicolon following it.

## **Bit field cannot be static** Compiler message

Only ordinary C++ class data members can be declared static, not bit fields.

## Global anonymous union not static Compiler message

In C++, a global anonymous union at the file level must be static.

## Structure required on left side of . or .*

Compiler message

The left side of a dot (.) operator (or C++ dot-star operator, .*) must evaluate to a structure type. In this case it did not.

This error can occur when you create an instance of a class using empty parentheses, and then try to access a member of that 'object'.

## Constant expression required Compiler message

Arrays must be declared with constant size.

This error is commonly caused by misspelling a **#define** constant.

## Call of nonfunction

Compiler message

The name being called is not declared as a function.

This is commonly caused by incorrectly declaring the function or misspelling the function name.

## Case outside of switch Compiler message

The compiler encountered a case statement outside a switch statement.

This is often caused by mismatched braces.

## Member pointer required on right side of .* or ->* Compiler message

The right side of a C++ dot-star (.*) or an arrow star (->*) operator must be declared as a pointer to a member of the class specified by the left side of the operator.

In this case, the right side is not a member pointer.

## Type name expected Compiler message

One of these errors has occurred:

- In declaring a file-level variable or a struct field, neither a type name nor a storage class was given.
- In declaring a typedef, no type for the name was supplied.
- In declaring a destructor for a C++ class, the destructor name was not a type name (it must be the same name as its class).
- In supplying a C++ base class name, the name was not the name of a class.

## **union cannot have a base type** Compiler message

In general, a C++ class can be of union type, but such a class can't be derived from any other class.

## Value of type void is not allowed Compiler message

A value of type **void** is really not a value at all, so it can't appear in any context where an actual value is required.

Such contexts include the following:

- the right side of an assignment
- an argument of a function
- the controlling expression of an if, for, or while statement.

### do statement must have while Compiler message

Your source file contained a do statement that was missing the closing while keyword.

## Identifier 'identifier' cannot have a type qualifier Compiler message

A C++ qualifier class::identifier can't be applied here.

A qualifier is not allowed on the following:

- typedef names
- function declarations (except definitions at the file level)
- on local variables or parameters of functions
- on a class member--except to use its own class as a qualifier (redundant but legal).

## sizeof may not be applied to a bit field

Compiler message

sizeof returns the size of a data object in bytes, which does not apply to a bit field.

## sizeof may not be applied to a function

Compiler message

sizeof can be applied only to data objects, not functions.

You can request the size of a pointer to a function.

## Storage class 'storage class' is not allowed here Compiler message

The given storage class is not allowed here.

Probably two storage classes were specified, and only one can be given.

## Access can only be changed to public or protected

Compiler message

A C++ derived class can modify the access rights of a base class member, but only to public or protected.

A base class member can't be made private.

## Variable 'identifier' is once

Compiler message

This variable has more than one initialization. It is legal to declare a file level variable more than once, but it can have only one initialization (even if two are the same).

## Improper use of typedef 'identifier'  Compiler message

Your source file used a typedef symbol where a variable should appear in an expression.

Check for the declaration of the symbol and possible misspellings.

## Size of 'identifier' is unknown or zero Compiler message

This identifier was used in a context where its size was needed.

A struct tag might only be declared (the struct not defined yet), or an extern array might be declared without a size.

It's illegal then to have some references to such an item (like sizeof) or to dereference a pointer to this type.

Rearrange your declaration so that the size of 'identifier' is available.

## Size of the type 'identifier' is unknown or zero

Compiler message

This type was used in a context where its size was needed.

For example, a struct tag might only be declared (the struct not defined yet).

It's illegal then to have some references to such an item (like sizeof) or to dereference a pointer to this type.

Rearrange your declarations so that the size of this type is available.

## Size of the type is unknown or zero Compiler message

This error message indicates that an array of unspecified dimension nested within another structure is initialized and the **-A** (ANSI) switch is on. For example:

```
struct
{
  char a[];       //Size of 'a' is unknown or zero
}
  b = { "hello" };      //Size of the type is
                  //unknown or zero
```

## Conversion operator cannot have a return type specification Compiler message

This C++ type conversion member function specifies a return type different from the type itself.

A declaration for conversion function operator can't specify any return type.

## Linkage specification not allowed

Compiler message

Linkage specifications such as extern "C" are only allowed at the file level.

Move this function declaration out to the file level.

## Reference member 'member' needs a temporary for initialization

Compiler message

You provided an initial value for a reference type that was not an lvalue of the referenced type.

This requires the compiler to create a temporary for the initialization.

Because there is no obvious place to store this temporary, the initialization is illegal.

## 'identifier' cannot be declared in an anonymous union Compiler message

The compiler found a declaration for a member function or static member in an anonymous union. Such unions can only contain data members.

## Function 'function' cannot be static Compiler message

Only ordinary member functions and the operators new and delete can be declared static.

Constructors, destructors and other operators must not be static.

## Cannot overload 'main' Compiler message

main is the only function that can't be overloaded.

## Non-virtual function 'function' declared pure

Compiler message

Only virtual functions can be declared pure, because derived classes must be able to override them.

## Bad syntax for pure function definition

Compiler message

Pure virtual functions are specified by appending "= 0" to the declaration, like this:

```
class A { virtual void f () = 0;}
class B : public A { void f () {}; }
```

You wrote something similar, but not quite the same.

## Cannot create instance of abstract class 'class'  Compiler message

Abstract classes (those with pure virtual functions) can't be used directly, only derived from.

When you derive an abstract base class, with the intention to instantiate instances of this derived class, you must override each of the pure virtual functions of the base class exactly as they are declared.

For example:

```
class A {
public:
   virtual myex( int ) = 0;
   virtual twoex( const int ) const = 0;
};
class B : public A {
public:
   myex( int );
   twoex( const int );
};
B b;   // error
```

The error occurs because we have not overridden the virtual function in which `twoex` can act on const objects of the class. We have created a new one which acts on non-const objects. This would compile:

```
class A {
public:
   virtual myex( int ) = 0;
   virtual twoex( const int ) const = 0;
};
class B : public A {
public:
   myex( int );
   twoex( const int ) const;
};
B b;   // ok
```

## Cannot define a pointer or reference to a reference

Compiler message

It is illegal to have a pointer to a reference or a reference to a reference.

## Array of references is not allowed Compiler message

It is illegal to have an array of references, because pointers to references are not allowed and array names are coerced into pointers.

## void & is not a valid type

Compiler message

A reference always refers to an object, but an object cannot have the type void.

Thus, the type void is not allowed.

## 'identifier' is not a non-static member and can't be initialized here

Compiler message

Only data members can be initialized in the initializers of a constructor.

This message means that the list includes a static member or function member.

Static members must be initialized outside of the class, for example:

```
class A { static int i; };
int A::i = -1;
```

## Destructor for 'class' required in conditional expression Compiler message

If the compiler must create a temporary local variable in a conditional expression, it has no good place to call the destructor because the variable might or might not have been initialized.

The temporary can be explicitly created, as with classname(val, val), or implicitly created by some other code.

You should recast your code to eliminate this temporary value.

## The value for 'identifier' is not within the range of an int Compiler message

All enumerators must have values that can be represented as an integer.

You have attempted to assign a value that is out of the range of an integer.

If you need a constant of this value, use a const integer.

## Member 'member' cannot be used without an object Compiler message

This means that you have written class::member where 'member' is an ordinary (non-static) member, and there is no class to associate with that member.

For example, it is legal to write this:

```
obj.class::member
```

but not to write this:

```
class::member
```

## **Function should return a value** Compiler warning

`(Command-line option to suppress warning: **-w-rvl**)`

Your source file declared the current function to return some type other than **int** or **void**, but the compiler encountered a return with no value. This usually indicates some sort of error.

Functions declared as returning **int** are exempt because older versions of C did not support **void** function return types.

## **Undefined structure 'structure'** Compiler warning

(Command-line option to display warning = **-wstu**)

The named structure was used in the source file, probably on a pointer to a structure, but had no definition in the source file.

This is probably caused by a misspelled structure name or a missing declaration.

## 'base' is an indirect virtual base class of 'class'

Compiler message

You can't create a pointer to a C++ member of a virtual base class.

You have attempted to create such a pointer (either directly, or through a cast) and access an inaccessible member of one of your base classes.

## 'identifier' is not a public base class of 'classtype' Compiler message

The right operand of a .*, ->*, or ::operator was not a pointer to a member of a class that is either identical to (or an unambiguous accessible base class of) the left operand's class type.

## 'operator' must be declared with one or no parameters Compiler message

When operator ++ or operator -- is declared as a member function, it must be declared to take either:

- No parameters (for the prefix version of the operator), or
- One parameter of type int (for the postfix version)

## 'operator' must be declared with one or two parameters Compiler message

When operator ++ or operator -- is declared as a non-member function, it must be declared to take either:

- one parameter (for the prefix version of the operator), or
- two parameters (for the postfix version)

## 'virtual' can only be used with member functions Compiler message

A data member has been declared with the **virtual** specifier.

Only member functions can be declared virtual.

For example:

```
class myclass
{
public:
   virtual int a;    //error
};
```

## **Address of overloaded function 'function' doesn't match 'type'** Compiler message

A variable or parameter is assigned (or initialized with) the address of an overloaded function.

However, the type of the variable or parameter doesn't match any of the overloaded functions with the specified name.

## Ambiguous member name 'name'  Compiler message

Whenever a structure member name is used in inline assembly, such a name must be unique.

(If it is defined in more than one structure, all of the definitions must agree as to its type and offset within the structures).

In this case, an ambiguous member name has been used.

For example:

```
struct A
{
   int a;
   int b;
};
...
asm ax,.a;
```

## Assignment to 'this' not allowed, use X::operator new instead Compiler message

In early versions of C++, the only way to control allocation of class of objects was by assigning to the 'this' parameter inside a constructor.

This practice is no longer allowed, because a better, safer, and more general technique is to define a member function operator new instead.

For example:

```
this = malloc(n);
```

## Cannot allocate a reference Compiler message

You have attempted to create a reference using the new operator.

This is illegal, because references are not objects and can't be created through new.

## Cannot call near class member function with a pointer of type 'type' Compiler message

Member functions of near classes can't be called via a member pointer.

This also applies to calls using pointers to members.

(Remember, classes are near by default in the tiny, small, and medium memory models.)

Either change the pointer to be near, or declare the class as far.

## Cannot convert 'type1' to 'type2'

Compiler message

An assignment, initialization, or expression requires the specified type conversion to be performed, but the conversion is not legal.

In C++, the compiler will convert one function pointer to another only if the signature for the functions are the same. Signature refers to the arguments and return type of the function. For example:

```
myex( int );
typedef int ( *ffp )( float );
test()
{
   ffp fp = myex; //error
}
```

Seeing that `myex` takes an int for its argument, and `fp` is a pointer to a function which takes a float as argument, the compiler will not convert it for you.

In cases where this is what is intended, performing a typecast is necessary:

```
myex( int );
typedef int ( *ffp )( float );
test()
{
   ffp fp = (ffp)myex;  //ok
}
```

## Class 'class' may not contain pure functions

Compiler message

The class being declared cannot be abstract, and therefore it cannot contain any pure functions.

## Compiler could not generate copy constructor for class 'class' Compiler message

**OR Compiler could not generate default constructor for class 'class'**

**OR Compiler could not generate operator = for class 'class'**

Sometimes the compiler is required to generate a member function for the user.

Whenever such a member function can't be generated due to applicable language rules, the compiler issues one of these error messages.

## Constant/Reference member 'member' in class without constructors Compiler message

A class that contains constant or reference members (or both) must have at least one user-defined constructor.

Otherwise, there would be no way to ever initialize such members.

## Constructor/Destructor cannot be declared 'const' or 'volatile' Compiler message

A constructor or destructor has been declared as **const** or **volatile**.

This is not allowed.

## Default expression may not use local variables Compiler message

A default argument expression is not allowed to use any local variables or other parameters.

## Illegal use of member pointer Compiler message

Pointers to class members can only be passed as arguments to functions, or used with the following operators:

- assignment
- comparison
- `.*`
- `->*`
- `?:`
- `&&`
- `||`

The compiler has encountered a member pointer being used with a different operator.

In order to call a member function pointer, one must supply an instance of the class for it to call upon.

For example:

```
class A {
public:
   myex();
};
typedef int (A::*Amfptr)();
myex()
{
   Amfptr mmyex = &A::myex;
   (*mmyex)():  //error
}
```

This will compile:

```
class A {
public:
   myex();
};
typedef int (A::*Amfptr)();
foo()
{
   A a;
   Amfptr mmyex - &A::myex;
   (a.*mmyex)90;
}
```

## Implicit conversion of 'type1' to 'type2' not allowed

Compiler message

When a member function of a class is called using a pointer to a derived class, the pointer value must be implicitly converted to point to the appropriate base class.

In this case, such an implicit conversion is illegal.

## Last parameter of 'operator' must have type 'int' Compiler message

When a postfix operator ++ or operator -- is overloaded, the last parameter must be declared with the type int.

## Member function must be called or its address taken

Compiler message

A reference to a member function must be called, or its address must be taken with & operator.

In this case, a member function has been used in an illegal context.

For example:

```
class A
{
   void (A::* infptr)(void);
public;
   A();
   void myex(void);
};
A::A()
{
   infptr = myex;   //illegal - call myex or take address?
   infptr = A::& myex;   //correct
}
```

## Reference initialized with 'type1', needs lvalue of type 'type2'

Compiler message

A reference variable that is not declared constant must be initialized with an lvalue of the appropriate type.

In this case, the initializer either wasn't an lvalue, or its type didn't match the reference being initialized.

## Only member functions may be 'const' or 'volatile'

Compiler message

Something other than a class member function has been declared **const** or **volatile**.

## Operand of 'delete' must be non-const pointer

Compiler message

It is illegal to delete a variable that is not a pointer.

It is also illegal to delete a pointer to a constant.

For example:

```
const int x=10;
   const int * a = &x;
   int * const b = new int;
   int &c = *b;
   delete a;   //illegal - deleting pointer to constant
   delete b;   //legal
   delete c;   //illegal - operand not of pointer type
               //should use 'delete&c' instead
```

## Operator must be declared as function

Compiler message

An overloaded operator was declared with something other than function type.

For example:

```
class A
{
   A& operator +;    ..note missing parenthesis
};
```

In the example, the function operator '()' is missing, so the operator does not have function type and generates this error.

## Operators may not have default argument values Compiler message

It is illegal for overloaded operators to have default argument values.

## Overloaded 'function name' ambiguous in this context Compiler message

The only time an overloaded function name can be used or assigned without actually calling the function is when a variable or parameter of the correct function pointer type is initialized or assigned the address of the overload function.

In this case, an overloaded function name has been used in some other context, for example, the following code will generate this error:

```
class A{
   A(){myex;}            //calling the function
   void myex(int) {}     //or taking its address?
   void myex(float){}
};
```

## Type mismatch in default argument value Compiler message

The default parameter value given could not be converted to the type of the parameter.

The message "Type mismatch in default argument value" is used when the parameter was not given a name.

When compiling C++ programs, this message is always preceded by another message that explains the exact reason for the type mismatch.

That other message is most often "Cannot convert 'type1' to 'type2'" but the mismatch could be due to another reason.

## Type mismatch in default value for parameter 'parameter'

Compiler message

The default parameter value given could not be converted to the type of the parameter.

The message "Type mismatch in default argument value" is used when the parameter was not given a name.

When compiling C++ programs, this message is always preceded by another message that explains the exact reason for the type mismatch.

That other message is usually "Cannot convert 'type1' to 'type2'" but the mismatch might be due to many other reasons.

## union cannot be a base type Compiler message

A union can't be used as a base type for another class type.

## unions cannot have virtual member functions

Compiler message

A union can't have virtual functions as its members.

## Union member 'member' is of type class with constructor (or destructor, or operator =) Compiler message

A union can't contain members that are of type class with user-defined constructors, destructors, or operator =.

## Use . or -> to call 'member', or & to take its address

Compiler message

A reference to a non-static class member without an object was encountered.

Such a member can't be used without an object, or its address must be taken with the & operator.

## Void 'function' cannot return a value Compiler message

A function with a return type void contains a return statement that returns a value; for example, an int.

Default = displayed

## 'identifier' cannot start a parameter declaration Compiler message

An undefined 'identifier' was found at the start of an argument in a function declarator.

Often the type name is misspelled or the type declaration is missing. This is usually caused by not including the appropriate header file.

## < expected Compiler message

The keyword **template** was not followed by <.

Every template declaration must include the template formal parameters enclosed within < >, immediately following the template keyword.

## Attempting to return a reference to a local object Compiler message

You attempted to return a reference to a temporary object in a function that returns a reference type. This may be the result of a constructor or a function call.

This object will disappear when the function returns, making the reference illegal.

## Base class 'class' contains dynamically dispatchable functions

Compiler message

This error occurs when a class containing a DDVT function attempts to inherit DDVT functions from multiple parent classes.

Currently, dynamically dispatched virtual tables do not support the use of multiple inheritance.

## Bit fields must be signed or unsigned int  Compiler message

In ANSI C, bit fields may only be signed or unsigned int (not char or long, for example).

## Cannot add or subtract relocatable symbols

Compiler message

The only arithmetic operation that can be performed on a relocatable symbol in an assembler operand is addition or subtraction of a constant.

Variables, procedures, functions, and labels are relocatable symbols.

## Cannot have a non-inline function/static data in a local class Compiler message

All members of classes declared local to a function must be entirely defined in the class definition.

This means that local classes cannot contain any static data members, and all of their member functions must have bodies defined within the class definition.

## Case bypasses initialization of a local variable

Compiler message

In C++ it is illegal to bypass the initialization of a local variable.

This error indicates a case label that can transfer control past this local variable.

## Declaration does not specify a tag or an identifier Compiler message

This declaration doesn't declare anything.

This may be a struct or union without a tag or a variable in the declaration. C++ requires that something be declared.

For example:

```
struct
{
int a
};
//no tag or identifier
```

## Expression of scalar type expected Compiler message

The !, ++, and -- operators require an expression of scalar type.

Only these types are allowed:

- char
- short
- int
- long
- enum
- float
- double
- long double
- pointer

## Extra argument in template class name 'template' Compiler message

A template class name specified too many actual values for its formal parameters.

## Function definition cannot be a typedef'ed declaration Compiler message

In ANSI C, a function body cannot be defined using a typedef with a function Type.

Redefine the function body.

## Functions 'function1' and 'function2' both use the same dispatch number

Compiler message

This error indicates a dynamically dispatched virtual table (DDVT) problem.

## Goto bypasses initialization of a local variable

Compiler message

In C++ it is illegal to bypass the initialization of a local variable.

This error indicates a goto statement that can transfer control past this local variable.

## Illegal parameter to _ _emit_ _ Compiler message

There are some restrictions on inserting literal values directly into your code with the __emit__ function. For example, you cannot give a local variable as a parameter to __emit__.

## Invalid combination of opcode and operands

Compiler message

The built-in assembler does not accept this combination of operands.

**Possible Causes**

- There are too many or too few operands for this assembler opcode.
- The number of operands is correct, but their types or order do not match the opcode.

## Invalid register combination (e.g. [BP+BX]) Compiler message

The built-in assembler detected an illegal combination of registers in an instruction.

These are valid index register combinations:

- `[BX]`
- `[BP]`
- `[SI]`
- `[DI]`
- `[BX+SI]`
- `[BX+DI]`
- `[BP+SI]`
- `[BP+DI]`

Other index register combinations are not allowed.

## Invalid template argument list Compiler message

This error indicates that an illegal template argument list was found.

In a template declaration, the keyword **template** must be followed by a list of formal arguments enclosed within < and > delimiters.

## Invalid template qualified name 'template::name' Compiler message

When defining a template class member, the actual arguments in the template class name used as the left operand for the :: operator must match the formal arguments of the template class.

## Invalid use of template 'template'

Compiler message

You can only use a template class name without specifying its actual arguments inside a template definition.

Using a template class name without specifying its actual arguments outside a template definition is illegal.

## Matching base class function 'function' has different dispatch number

Compiler message

If a DDVT function is declared in a derived class, the matching base class function must have the same dispatch number as the derived function.

## Matching base class function 'function' is not dynamic Compiler message

If a DDVT function is declared in a derived class, the matching base class function must also be dynamic.

## Member 'member' has the same name as its class Compiler message

A static data member, enumerator, member of an anonymous union, or nested type cannot have the same name as its class.

Only a member function or a non-static member can have a name that is identical to its class.

## Memory reference expected Compiler message

The built-in assembler requires a memory reference.

You probably forgot to put square brackets around an index register operand.

## Nontype template argument must be of scalar type

Compiler message

A nontype formal template argument must have scalar type; it can have an integral, enumeration, or pointer type.

## operator new must have an initial parameter of type size_t

## Operator new[] Must Have an Initial Parameter of Type size_t Compiler message

Operator new can be declared with an arbitrary number of parameters.

It must always have at least one, the amount of space to allocate.

## Template argument must be a constant expression

Compiler message

A non-type template class argument must be a constant expression of the appropriate type.

This includes constant integral expressions and addresses of objects or functions with external linkage or members.

## Template class nesting too deep: 'class'

Compiler message

The compiler imposes a certain limit on the level of template class nesting. This limit is usually only exceeded through a recursive template class dependency.

When this nesting limit is exceeded, the compiler issues this error message for all of the nested template classes. This usually makes it easy to spot the recursion.

This error message is always followed by the fatal error "Out of memory".

## Template functions may only have 'type-arguments'

Compiler message

A function template was declared with a non-type argument.

This is not allowed with a template function, as there is no way to specify the value when calling it.

## Template function argument 'argument' not used in argument types Compiler message

The given argument was not used in the argument list of the function.

The argument list of a template function must use all of the template formal arguments; otherwise, there is no way to generate a template function instance based on actual argument types.

## Templates can only be declared at file level Compiler message

Templates cannot be declared inside classes or functions.

They are only allowed in the global scope, or file level.

For example:

```
template <class T, class U>
void foo(Ta,Tb)
{
    .
    .
    .
}
// error U is not used
```

## Templates must be classes or functions Compiler message

The declaration in a template declaration must specify either a class type or a function.

## The constructor 'constructor' is not allowed

Compiler message

Constructors of the form

```
X::(X)
```

are not allowed.

This is the correct way to write a copy constructor:

```
X::(const X&).
```

## Too few arguments in template class name 'template'

Compiler message

A template class name was missing actual values for some of its formal parameters.

## Trying to derive a far class from the huge base 'base'

Compiler message

If a class is declared (or defaults to) huge, all derived classes must also be huge.

## Trying to derive a huge class from the far base 'base'

Compiler message

If a class is declared (or defaults to) far, all derived classes must also be far.

## Trying to derive a huge class from the near base 'base' Compiler message

If a class is declared (or defaults to) near, all derived classes must also be near.

## Trying to derive a near class from the huge base 'base' Compiler message

If a class is declared (or defaults to) far, all derived classes must also be far.

## Type mismatch in parameter 'parameter' in template class name 'template'

Compiler message

The actual template argument value supplied for the given parameter did not exactly match the formal template parameter type.

When compiling C++ programs, this message is always preceded by another message that explains the exact reason for the type mismatch.

That other message is usually "Cannot convert 'type1' to 'type2'" but the mismatch might be due to many other reasons.

## Type mismatch in parameter 'number' in template class name 'template'

Compiler message

The actual template argument value supplied for the given parameter did not exactly match the formal template parameter type.

When compiling C++ programs, this message is always preceded by another message that explains the exact reason for the type mismatch.

That other message is usually "Cannot convert 'type1' to 'type2'" but the mismatch might be due to many other reasons.

## Bit fields must have integral type

Compiler message

In C++, bit fields must have an integral type. This includes enumerations.

## Cannot find default constructor to initialize array element of type 'class'

Compiler message

When declaring an array of a class that has constructors, you must either explicitly initialize every element of the array, or the class must have a default constructor.

The compiler will define a default constructor for a class unless you have defined any constructors for the class.

## Cannot generate 'function' from template function 'template'

Compiler message

A call to a template function was found, but a matching template function cannot be generated from the function template.

## Variable 'variable' has been optimized and is not available

Compiler message

You have tried to inspect, watch, or otherwise access a variable which the optimizer removed.

This variable is never assigned a value and has no stack location.

## Function should return a value Compiler message

This function was declared (maybe implicitly) to return a value.

The compiler found a return statement without a return value, or it reached the end of the function without finding a return statement.

Either return a value or change the function declaration to return void.

## Undefined structure 'structure' Compiler message

Your source file used the named structure on some line before where the error is indicated (probably on a pointer to a structure) but had no definition for the structure.

This is probably caused by a misspelled structure name or a missing declaration.

## Call to function 'function' with no prototype Compiler message

This message is given if the "Prototypes required" warning is enabled and you call function 'function' without first giving a prototype for that function.

## Initializing 'identifier' with 'identifier' Compiler warning

(Command-line option to suppress warning: **-w-bei**)

You're trying to initialize an **enum** variable to a different type.

For example, the following initialization will result in this warning, because 2 is of type int, not type enum count:

```
enum count zero, one, two x = 2;
```

It is better programming practice to use an enum identifier instead of a literal integer when assigning to or initializing enum types.

This is an error, but is reduced to a warning to give existing programs a chance to work.

## Initialization is only partially bracketed

Compiler warning

`(Command-line option to display warning: **-wpin**)`

When structures are initialized, braces can be used to mark the initialization of each member of the structure.

If a member itself is an array or structure, nested pairs of braces can be used.

This ensures that the compiler's idea and your idea of what value goes with which member are the same.

When some of the optional braces are omitted, the compiler issues this warning.

## 'identifier' is declared as both external and static Compiler warning

(Command-line option to suppress warning: **-w-ext**)

This identifier appeared in a declaration that implicitly or explicitly marked it as global or external, and also in a static declaration.

The identifier is taken as static.

You should review all declarations for this identifier.

## Declare 'type' prior to use in prototype

Compiler warning

(Command-line option to suppress warning: **-w-dpu**)

When a function prototype refers to a structure type that has not previously been declared, the declaration inside the prototype is not the same as a declaration outside the prototype.

For example,

```
int func(struct s *ps);  struct s  /* ... */ ;
```

Because there is no "struct s" in scope at the prototype for func, the type of parameter ps is pointer to undefined struct s, and is not the same as the "struct s" that is later declared.

This will result in later warning and error messages about incompatible types, which would be very mysterious without this warning message.

To fix the problem, you can move the declaration for "struct s" ahead of any prototype that references it, or add the incomplete type declaration "struct s;" ahead of any prototype that references "structs".

If the function parameter is a struct, rather than a pointer to struct, the incomplete declaration is not sufficient.

You must then place the struct declaration ahead of the prototype.

## Division by zero Compiler warning

(Command-line option to suppress warning: **-w-zdi**)

A divide or remainder expression had a literal zero as a divisor.

## Ambiguous operators need parentheses Compiler warning

(Command-line option to display warning: **-wamb**)

This warning is displayed whenever two shift, relational, or bitwise-Boolean operators are used together without parentheses.

Also, an addition or subtraction operator that appears without parentheses with a shift operator will produce this warning.

## Superfluous & with function  Compiler warning

(Command-line option to display warning: **-wamp**)

An address-of operator (&) is not needed with function name; any such operators are discarded.

## Parameter 'parameter' is never used Compiler warning

(Command-line option to suppress warning: **-w-par**)

The named parameter, declared in the function, was never used in the body of the function.

This might or might not be an error and is often caused by misspelling the parameter.

This warning can also occur if the identifier is redeclared as an automatic (local) variable in the body of the function.

The parameter is masked by the automatic variable and remains unused.

## Restarting compile using assembly Compiler warning

(Command-line option to suppress warning: **-w-asc**)

The compiler encountered an asm with no accompanying or **#pragma** inline statement.

The compile restarts using assembly language capabilities.

Default = On

## Unknown assembler instruction

Compiler warning

(Command-line option to suppress warning: **-w-asm**)

The compiler encountered an inline assembly statement with a disallowed opcode.

Check the spelling of the opcode.

This warning is off by default.

## Hexadecimal value contains more than three digits

Compiler warning

(Command-line option to suppress warning = **-w-big**)

Under older versions of C, a hexadecimal escape sequence could contain no more than three digits.

The ANSI standard allows any number of digits to appear as long as the value fits in a byte.

This warning results when you have a long hexadecimal escape sequence with many leading zero digits (such as \x00045).

Older versions of C would interpret such a string differently.

## Constant is long Compiler warning

(Command-line option to display warning: **-wcln**)

The compiler encountered one of the following:

- a decimal constant greater than 32,767 or
- an octal, hexadecimal, or decimal constant greater than 65,535 without a letter l or L following it

The constant is treated as a long.

## Non-portable pointer comparison Compiler warning

(Command-line option to suppress warning: **-w-cpt**)

Your source file compared a pointer to a non-pointer other than the constant 0.

You should use a cast to suppress this warning if the comparison is proper.

## Temporary used to initialize 'identifier'

Compiler warning

(Command-line option to suppress warning: **-w-lin**)

In C++, a variable or parameter of reference type must be assigned a reference to an object of the same type.

If the types do not match, the actual value is assigned to a temporary of the correct type, and the address of the temporary is assigned to the reference variable or parameter.

The warning means that the reference variable or parameter does not refer to what you expect, but to a temporary variable, otherwise unused.

**Related Topics**

Example

## Example for "Temporary used ..." error messages Compiler message

In this example, function f requires a reference to an int, and c is a char:

```
f(int&);
char c;
f(c);
```

Instead of calling f with the address of c, the compiler generates code equivalent to the C++ source code:

```
int X = c, f(X);
```

## Temporary used for parameter 'parameter' Compiler warning

**OR Temporary used for parameter 'number'**

**OR Temporary used for parameter 'parameter' in call to 'function'**

**OR Temporary used for parameter 'number' in call to 'function'**

(Command-line option to suppress warning: **-w-lvc**)

In C++, a variable or parameter of reference type must be assigned a reference to an object of the same type.

If the types do not match, the actual value is assigned to a temporary of the correct type, and the address of the temporary is assigned to the reference variable or parameter.

The warning means that the reference variable or parameter does not refer to what you expect, but to a temporary variable, otherwise unused.

## The constant member 'identifier' is not initialized Compiler warning

(Command-line option to suppress warning: **-w-nci**)

This C++ class contains a constant member 'member' that doesn't have an initialization.

Note that constant members can be initialized only; they can't be assigned to.

## Functions containing reserved words are not expanded inline Compiler warning

**Also:**

**Functions containing local destructors are not expanded inline in function 'function'**

(Command-line option to suppress warning: **-w-inl**)

**Reserved Words**

Functions containing any of these reserved words can't be expanded inline, even when specified as inline:

- break
- case
- continue
- do
- for
- goto
- switch
- while

The function is still perfectly legal, but will be treated as an ordinary static (not global) function.

A copy of the function will appear in each compilation unit where it is called.

**Local Destructors**

You've created an inline function for which the compiler turns off inlining. You can ignore this warning; the function will be generated out of line.

## Base initialization without a class name is now obsolete Compiler warning

(Command-line option to suppress warning: **-w-obi**)

Early versions of C++ provided for initialization of a base class by following the constructor header with just the base class constructor parameter list.

It is now recommended to include the base class name.

This makes the code much clearer, and is required when you have multiple base classes.

**Old way**

```
derived::derived(int i) : (i, 10) { ... }
```

**New way**

```
derived::derived(int i) : base(i, 10) { ... }
```

## Style of function definition is now obsolete Compiler warning

(Command-line option to suppress warning = **-w-ofp**)

In C++, this old C style of function definition is illegal:

```
int func(p1, p2) int p1, p2; { /* ... */ }
```

This practice might not be allowed by other C++ compilers.

## Overload is now unnecessary and obsolete Compiler warning

(Command-line option to suppress warning: **-w-ovl**)

Early versions of C++ required the reserved word "overload" to mark overloaded function names.

C++ now uses a "type-safe linkage" scheme, whereby all functions are assumed overloaded unless marked otherwise.

The use of the term "overload" should be discontinued.

## Assigning 'type' to 'enumeration'  Compiler warning

(Command-line option to suppress warning: **-w-eas**)

Assigning an integer value to an enum type.

This is an error in C++, but is reduced to a warning to give existing programs a chance to work.

## Non-const function 'function' called for const object

Compiler warning

(Command-line option to suppress warning = **-w-ncf**)

A non-const member function was called for a const object.

(This is an error, but was reduced to a warning to give existing programs a chance to work.)

## Non-volatile function 'function' called for volatile object Compiler message

(Command-line option to suppress warning: **-w-nvf**)

In C++, a class member function was called for a volatile object of the class type, but the function was not declared with volatile following the function header. Only a volatile member function can be called for a volatile object.

For example, if you have

```
class c
{
public:
  f() volatile;
  g();
};
volatile c vcvar;
```

it is legal to call `vcvar.f()`, but not to call `vcvar.g()`.

## 'function1' hides virtual function 'function2' Compiler warning

(Command-line option to suppress warning: **-w-hid**)

A virtual function in a base class is usually overridden by a declaration in a derived class.

In this case, a declaration with the same name but different argument types makes the virtual functions inaccessible to further derived classes.

## Possible use of 'identifier' before definition Compiler warning

(Command-line option to display warning: **-wdef**)

Your source file used the variable 'identifier' in an expression before it was assigned a value.

The compiler uses a simple scan of the program to determine this condition.

If the use of a variable occurs physically before any assignment, this warning will be generated.

Of course, the actual flow of the program can assign the value before the program uses it.

## Constant out of range in comparison Compiler warning

(Command-line option to suppress warning: **-w-rng**)

Your source file includes a comparison involving a constant sub-expression that was outside the range allowed by the other sub-expression's type.

For example, comparing an unsigned quantity to -1 makes no sense.

To get an unsigned constant greater than 32,767 (in decimal), you should either

- cast the constant to unsigned--for example, (unsigned) 65535, or
- append a letter u or U to the constant--for example, 65535u.

Whenever this message is issued, the compiler still generates code to do the comparison.

If this code ends up always giving the same result (such as comparing a char expression to 4000), the code will still perform the test.

## Redefinition of 'macro' is not identical

Compiler warning

(Command-line option to suppress warning: **-w-dup**)

Your source file redefined the macro 'ident' using text that was not exactly the same as the first definition of the macro.

The new text replaces the old.

## Array variable 'identifier' is near Compiler warning

(Command-line option to suppress warning: **-w-ias**)

When you use set the Far Data Threshold option, the compiler automatically makes any global variables that are larger than the threshold size be far.

When the variable is an initialized array with an unspecified size, its total size is not known when the compiler must decide whether to make it near or far, so the compiler makes it near.

The compiler issues this warning if the number of initializers given for the array causes the total variable size to exceed the data size threshold.

If the fact that the compiler made the variable be near causes problems, make the offending variable explicitly far.

To do this, insert the keyword "far" immediately to the left of the variable name in its definition.

## **Possibly incorrect assignment** Compiler warning

`(Command-line option to suppress warning: **-w-pia**)`

This warning is generated when the compiler encounters an assignment operator as the main operator of a conditional expression (part of an if, while, or do-while statement).

This is usually a typographical error for the equality operator.

If you want to suppress this warning, enclose the assignment in parentheses and compare the whole thing to zero explicitly.

For example, this code

```
if (a = b) ...
```

should be rewritten as

```
if ((a = b) != 0) ...
```

## Conversion may lose significant digits

Compiler warning

(Command-line option to display warning: **-wsig**)

For an assignment operator or some other circumstance, your source file requires a conversion from long or unsigned long to int or unsigned int type.

Because int type and long type variables don't have the same size, this kind of conversion might alter the behavior of a program.

## No declaration for function 'function' Compiler warning

`(Command-line option to display warning: -wnod)`

This message is given if you call a function without first declaring that function.

In C, you can declare a function without presenting a prototype, as in

```
int func();
```

In C++, every function declaration is also a prototype; this example is equivalent to

```
int func(void);
```

The declaration can be either classic or modern (prototype) style.

## Code has no effect Compiler warning

(Command-line option to suppress warning: **-w-eff**)

This warning is issued when the compiler encounters a statement with some operators that have no effect.

For example, the statement

```
a + b;
```

has no effect on either variable.

The operation is unnecessary and probably indicates a bug.

## 'identifier' is assigned a value that is never used Compiler warning

(Command-line option to suppress warning: **-w-aus**)

The variable appears in an assignment, but is never used anywhere else in the function just ending. The warning is indicated only when the compiler encounters the closing brace.

## **Ill-formed pragma** Compiler warning

`(Command-line option to suppress warning: **-w-ill**)`

A pragma does not match one of the pragmas expected by the compiler.

## Call to function with no prototype Compiler warning

(Command-line option to suppress warning: **-w-pro**)

This message is given if the "Prototypes required" warning is enabled and you call a function without first giving a prototype for that function.

## Suspicious pointer conversion Compiler warning

(Command-line option to suppress warning: **-w-sus**)

The compiler encountered some conversion of a pointer that caused the pointer to point to a different type.

You should use a cast to suppress this warning if the conversion is proper.

A common cause of this warning is when the C compiler converts a function pointer of one type to another (the C++ compiler generates an error when asked to do that). It can be suppressed by doing a typecast. Here is a common occurrence of it for Window programmers:

```
#define STRICT
#include <windows.h>

LPARAM _export WndProc( HWND , UINT , WPARAM , LPARAM );

test() {
   WNDCLASS wc;
   wc.lpfnWndProc = WndProc;  //warning
}
```

It is suppressed by making the assignment to lpfnWndProc as follows:

```
wc.lpfnWndProc = ( WNDPROC ) WndProc;
```

## **Unreachable code** Compiler warning

(Command-line option to suppress warning: **-w-rch**)

A break, continue, goto, or return statement was not followed by a label or the end of a loop or function.

The compiler checks while, do,, and for loops with a constant test condition, and attempts to recognize loops that can't fall through.

## Both return and return of a value used

Compiler warning

(Command-line option to suppress warning: **-w-ret**)

The current function has return statements with and without values.

This is legal C, but almost always an error.

Possibly a return statement was omitted from the end of the function.

## Nonportable pointer conversion Compiler message

A nonzero integral value is used in a context where a pointer is needed or where an integral value is needed; the sizes of the integral type and pointer are the same.

Use an explicit cast if this is what you really meant to do.

## Void functions may not return a value

Compiler warning

(Command-line option to suppress warning: **-w-voi**)

Your source file declared the current function as returning void, but the compiler encountered a return statement with a value.

The value of the return statement will be ignored.

## Structure passed by value Compiler warning

`(Command-line option to display warning: **-wstv**)`

This warning is generated any time a structure is passed by value as an argument.

It is a frequent programming mistake to leave an address-of operator (&) off a structure when passing it as an argument.

Because structures can be passed by value, this omission is acceptable.

This warning provides a way for the compiler to warn you of this mistake.

## Mixing pointers to different 'char' types Compiler warning

(Command-line option to display warning: **-wucp**)

You converted a signed char pointer to an unsigned char pointer, or vice versa, without using an explicit cast. (Strictly speaking, this is incorrect, but it is often harmless.)

## 'identifier' is declared but never used Compiler warning

(Command-line option to display warning: **-wuse**)

This message can occur in the case of either local or static variables. It occurs when the source file declares the named local or static variable as part of the block just ending, but the variable was never used.

In the case of local variables, this warning occurs when the compiler encounters the closing brace of the compound statement or function.

In the case of static variables, this warning occurs when the compiler encounters the end of the source file.

## Condition is always true OR Condition is always false Compiler warning

(Command-line option to suppress warning: **-w-ccc**)

Whenever the compiler encounters a constant comparison that (due to the nature of the value being compared) is always true or false, it issues this warning and evaluates the condition at compile time.

For example:

```
void proc(unsigned x){
  if (x >= 0)         /* always 'true' */
  {
     ...
  }
}
```

## Array size for 'delete' ignored Compiler warning

(Command-line option to suppress warning: **-w-dsz**)

The C++ IDE issues this warning when you've specified the array size when deleting an array.

With the new C++ specification, you don't need to make this specification. The compiler ignores this construct.

This warning lets older code compile.

## **Base class 'class1' is also a base class of 'class2'** Compiler warning

(Command-line option to suppress warning: **-w-ibc**)

A class inherits from the same base class both directly and indirectly. It is best to avoid this non-portable construct in your program code.

## Bit fields must be signed or unsigned int  Compiler warning

(Command-line option to display warning: **-wbbf**)

In ANSI C, bit fields may not be of type signed char or unsigned char.

When you're not compiling in strict ANSI mode, the compiler allows these constructs, but flags them with this warning.

## Overloaded prefix operator 'operator' used as a postfix operator

Compiler warning

`(Command-line option to suppress warning: **-w-pre**)`

With the latest C++ specification, it is now possible to overload both the prefix and postfix versions of the ++ and -- operators.

Whenever the prefix operator is overloaded, but is used in a postfix context, the compiler uses the prefix operator and issues this warning.

This allows older code to compile.

## Use qualified name to access member type 'identifier' Compiler warning

(Command-line option to suppress warning: **-w-nst**)

In previous versions of the C++ specification, **typedef** and tag names declared inside classes were directly visible in the global scope.

In the latest specification of C++, these names must be prefixed with `class::qualifier` if they are to be used outside of their class scope.

The compiler issues this warning whenever a name is uniquely defined in a single class. The compiler permits this usage without `class::`. This allows older versions of code to compile.

## Conversion to 'type' will fail for members of virtual base 'class'

Compiler warning

(Command-line option to suppress warning: **-w-mpc**)

This warning is issued only if the **-Vv** option is in use.

The warning may be issued when a member pointer to one type is cast to a member pointer of another type and the class of the converted member pointer has virtual bases.

Encountering this warning means that at runtime, if the member pointer conversion cannot be completed, the result of the cast will be a NULL member pointer.

## Stack overflow Runtime message

This error is reported when you compile a function with the Test Stack Overflow option on, but there is not enough stack space to allocate the function's local variables.

This error can also be caused by the following:

- infinite recursion, or
- an assembly language procedure that does not maintain the stack project
- a large array in a function

## Abnormal program termination

Runtime message

The program called abort because there wasn't enough memory to execute.

This message can be caused by memory overwrites.

## **Divide error** Runtime message

You tried to divide an integer by zero, which is illegal.

## Floating point error: Divide by 0



**OR Floating point error: Domain**

**OR Floating point error: Overflow**

These fatal errors result from a floating-point operation for which the result is not finite:

- Divide by 0 means the result is `+INF` or `-INF` exactly, such as `1.0/0.0`.
- Domain means the result is `NAN` (not a number), like `0.0/0.0`.
- Overflow means the result is `+INF` (infinity) or `-INF` with complete loss of precision, such as assigning `1e200*1e200` to a double.

## Floating point error: Partial loss of precision



**OR Floating point error: Underflow**

These exceptions are masked by default, because underflows are converted to zero and losses of precision are ignored.

## Floating point error: Stack fault

Runtime message

The floating-point stack has been overrun. This error may be due to assembly code using too many registers or due to a misdeclaration of a floating-point function.

The program prints the error message and calls abort and _exit.

These floating-point errors can be avoided by masking the exception so that it doesn't occur, or by catching the exception with signal.

## Type 'typename' may not be defined here Compiler message

Class and enumeration types may not be defined in a function return type, a function argument type, a conversion operator type, or the type specified in a cast.

You must define the given type before using it in one of these contexts.

**Note:** This error message is often the result of a missing semicolon (`;`) for a class declaration. You might want to verify that all the class declarations preceding the line on which the error occurred end with a semicolon.

## Cannot call 'main' from within the program Compiler message

C++ does not allow recursive calls of main( ).

## Application is running Runtime message

The application you tried to run is already running.

For Windows, make sure the message loop of the program has properly terminated.

```
PostQuitMessage(0);
```

## Maximum precision used for member pointer type 'type' Compiler warning

(Command-line option to suppress warning: **-w-mpd**)

When a member pointer type is declared, its class has not been fully defined, and the **-Vmd** option has been used, the compiler has to use the most general (and the least efficient) representation for that member pointer type. This can cause less efficient code to be generated (and make the member pointer type unnecessarily large), and can also cause problems with separate compilation; see the **-Vm** compiler switch for details.

## Include files nested too deep Compiler message

This message flags (directly or indirectly) recursive **#include** directives.

## Cannot take address of 'main' Compiler message

In C++, it is illegal to take the address of the main function.

## Ambiguous override of virtual base member 'base_function': 'derived_function'

Compiler message

A virtual function in a virtual base class was overridden with two or more different functions along different paths in the inheritance hierarchy. For example,

```
struct VB
{
  virtual f();
};

struct A:virtual VB
{
  virtual f();
};

struct B:virtual VB
  virtual f();
}
```

## Cannot throw 'type' -- ambiguous base class 'base'

Compiler message

It is not legal to throw a class that contains more than one copy of a (non-virtual) base class.

## Exception specification not allowed here Compiler message

Function pointer type declarations are not allowed to contain exception specifications.

## Exception handling variable may not be used here

Compiler message

An attempt has been made to use one of the exception handling values that are restricted to particular exception handling constructs, such as GetExceptionCode().

## Functions cannot return arrays or functions

Compiler message

A function was defined to return an array or a function. Check to see if either the intended return was a pointer to an array or function (and perhaps the * is missing) or if the function definition contained a request for an incorrect datatype.

## Duplicate handler for 'type1', already had 'type2' Compiler message

It is not legal to specify two handlers for the same type.

## The *name* handler must be last Compiler message

In a list of catch handlers, if the specified handler is present, it must be the last handler in the list (that is, it cannot be followed by any more catch handlers).

## VIRDEF name conflict for 'function' Compiler message

The compiler must truncate mangled names to a certain length because of a name length limit that is imposed by the linker. This truncation may (in very rare cases) cause two names to mangle to the same linker name. If these names happen to both be VIRDEF names, the compiler issues this error message. The simplest workaround for this problem is to change the name of 'function' so that the conflict is avoided.

## Goto into an exception handler is not allowed

Compiler message

It is not legal to jump into a try block, or an exception handler that is attached to a try block.

## 'catch' expected Compiler message

In a C++ program, a 'try' block must be followed by at least one 'catch' block.

## Type 'type' is not a defined class with virtual functions Compiler message

A dynamic_cast was used with a pointer to a class type that is either undefined, or doesn't have any virtual member functions.

## > expected Compiler message

A new-style cast (for example, dynamic_cast) was found with a missing closing ">".

## 'type' is not a polymorphic class type Compiler message

This error is generated if the -RT compiler option (for runtime type information) is disabled and either

- dynamic_cast was used with a pointer to a class

or

- you tried to delete a pointer to an object of a class that has a virtual destructor

## '__except' or '__finally' expected following '__try' Compiler message

In C, a '__try block' must be followed by a '__except' or '__finally' handler block.

## Reference member 'member' initialized with a non-reference parameter

Compiler message

An attempt has been made to bind a reference member to a constructor parameter. Since the parameter will cease to exist the moment the constructor returns to its caller, this will not work correctly.

## Non-ANSI keyword used: 'keyword' Compiler warning

(Command-line option to display warning: **-wnak**)

A non-ANSI keyword (such as '__fastcall') was used when strict ANSI conformance was requested via the -A option.

## Handler for 'type1' hidden by previous handler for 'type2'

Compiler warning

(Command-line option to suppress warning: **-w-hch**)

This warning is issued when a handler for a type 'D' that is derived from type 'B' is specified after a handler for B', since the handler for 'D' will never be invoked.

## Cannot create precompiled header: 'reason'

Compiler warning

(Command-line option to suppress warning: **-w-pch**)

This warning is issued when pre-compiled headers are enabled but the compiler could not generate one, for one of the following reasons:

| Reason | Explanation |
|---|---|
| write failed | The compiler could not write to the pre-compiled header file. This is usually due to the disk being full. |
| code in header | One of the headers contained a non-inline function body. |
| initialized data in header | One of the headers contained a global variable definition (in C, a global variable with an initializer; in C++ any variable not declared as 'extern'). |
| header incomplete | The pre-compiled header ended in the middle of a declaration, for example, inside a class definition (this often happens when there is a missing"}" in a header file). |

## Use '> >' for nested templates Instead of '>>'

Compiler warning

(Command-line option to suppress warning: **-w-ntd**)

Whitespace is required to separate the closing ">" in a nested template name, but since it is an extremely common mistake to leave out the space, the compiler accepts a ">>" with this warning.

## Initializing enumeration with type

Compiler message

You're trying to initialize an enum variable to a different type. For example,

```
enum count { zero, one, two } x = 2;
```

will result in this warning, because 2 is of type int, not type enum count. It is better programming practice to use an enum identifier instead of a literal integer when assigning to or initializing enum types.

This is an error, but is reduced to a warning to give existing programs a chance to work.

## Main must have a return type of int  Compiler message

In C++, function main has special requirements, one of which is that it cannot be declared with any return type other than int.

## 'ident' is obsolete Compiler warning

(Command-line option to suppress warning: **-w-obs**)

Issues a warning upon usage for any "C" linkage function that has been specified. This will warn about functions that are "obsolete".

Here's an example of it's usage:

```
#ifdef __cplusplus
extern "C" {
#endif
void my_func(void);
#ifdef __cplusplus
}
#endif

#pragma obsolete my_func

main()
{
  my_func();   // Generates warning about obsolete function
}
```

## Unexpected termination during compilation [Module Seg#:offset] Compiler message

### OR Unexpected termination during linking [Module Seg#:offset]

If either of these errors occur, it indicates a catastrophic failure of the Borland tools. You should contact Borland to report the problem and to find a potential work around for your specific case. By isolating the test case as well as possible, you will increase the chance for either Borland or yourself to find a work around for the problem.

Commonly, compiler failures can be worked around by moving the source code that is currently being compiled. Simple cases might be switching the order of variable declarations, or functions within the source module. Moving the scope and storage of variables also helps in many cases.

For linker failures, you can reduce the amount of debugging information that the linker has to work with. Try compiling only one or two modules with debug information instead of an entire project.

Similarly, switching the order in which object modules are handed to the linker can work around the problem. The IDE hands objects to the linker in the order that they are listed in the project tree. Try moving a source up or down in the list.

## Compiler stack overflow Compiler message

The compiler's stack has overflowed. This can be caused by a number of things, among them deeply nested statements in a function body (for example, if/else) or expressions with a large number of operands. You must simplify your code if this message occurs. Adding more memory to your system will not help.

## 286/287 instructions not enabled

Compiler message

Use the -2 command-line compiler option to enable 286/287 opcodes. Be aware that the resulting code cannot be run on 8086- and 8088-based machines.

## Incorrect option: Compiler message

An error has occurred in either the configuration file or a command-line option. The compiler may not have recognized the configuration file parameter as legal; check for a preceding hyphen (-), or the compiler may not have recognized the command-line parameter as legal.

This error can also occur if you use a **#pragma** option in your code with an invalid option.

### Unable to execute command 'command'

Compiler message

TLINK or TASM cannot be found, or possibly the disk is bad.

## Cannot evaluate function call Compiler message

The error message is issued if someone tries to explicitly construct an object or call a virtual function.

In integrated debugger expression evaluation, calls to certain functions (including implicit conversion functions, constructors, destructors, overloaded operators, and inline functions) are not supported.

## Not a valid expression format type Compiler message

Invalid format specifier following expression in the debug evaluate or watch window. A valid format specifier is an optional repeat value followed by a format character (c, d, f[n], h, x, m, p, r, or s).

## Cannot access an inactive scope

Compiler message

You have tried to evaluate or inspect a variable local to a function that is currently not active. (This is an integrated debugger expression evaluation message.)

## Unable to create turboc.$ln Compiler message

The compiler cannot create the temporary file TURBOC.$LN because it cannot access the disk or the disk is full.

## Conversion of near pointer not allowed

Compiler message

A near pointer cannot be converted to a far pointer in the expression evaluation box when a program is not currently running. This is because the conversion needs the current value of DS in the user program, which doesn't exist.

## 'new' and 'delete' not supported

Compiler message

The integrated debugger does not support the evaluation of the new and delete operators.

## Exception handling not enabled

Compiler message

A 'try' block was found with the exception handling disabled.

## No file names given

Compiler message

The command line contained no file names. You must specify a source file name.

## Multiple base classes not supported for VCL classes

Compiler message

VCL style classes cannot have multiple base classes.

Example:

```
struct __declspec(delphiclass) base1   {};
struct __declspec(delphiclass) base2   {};
struct derived : base1, base2          {}; // Error
```

## Overloaded function resolution not supported

Compiler message

In integrated debugger expression evaluation, resolution of overloaded functions or operators is not supported, not even to take an address.

## Templates not supported

Compiler message

An error has occurred while using the command-line utility H2ASH. See the online file "tsm_util.txt" for further information about this utility.

## No type information

Compiler message

The integrated debugger has no type information for this variable. Ensure that you've compiled the module with debug information. If it has, the module may have been compiled by another compiler or assembler.

## Virtual base classes not supported for VCL classes

Compiler message

VCL style classes cannot be derived virtually, not even from other VCL style classes.

Example:

```
struct __declspec(delphiclass) base  {};
struct derived : virtual base        {};    // Error
```

## Overlays only supported in medium, large, and huge memory models

Compiler message

Only programs using the medium, large, or huge memory models can be overlaid.

## Repeat count needs an lvalue Compiler message

The expression before the comma (,) in the Watch or Evaluate window must be an accessible region of storage. For example, expressions like this one are not valid:

```
i++,10d
x = y, 10m
```

## Can't inherit non-RTTI class from RTTI base

Compiler message

## OR Can't inherit RTTI class from non-RTTI base

When virtual functions are present, the RTTI attribute of all base classes must match that of the derived class.

## Side effects are not allowed Compiler message

Side effects such as assignments, ++, or -- are not allowed in the debugger watch window. A common error is to use x = y (not allowed) instead of x == y to test the equality of x and y.

## 'member' is not a valid template type member

Compiler message

A member of a template with some actual arguments that depend on the formal arguments of an enclosing template was found not to be a member of the specified template in a particular instance.

## '%s' requires runtime initialization/finalization

Compiler message

This message is issued when a global variable that is declared as __thread (a Win32-only feature) or a static data member of a template class is initialized with a non-constant initial value.

This message is also issued when a global variable that is declared as __thread (a Win32-only feature) or a static data member of a template class has the type class with constructor or destructor.

## Cannot use tiny or huge memory model with Windows Compiler message

This message is self-explanatory. Use small, medium, compact, or large instead.

## Declaration ignored Compiler warning

(Command-line option to suppress warning: **-w-dig**)

An error has occurred while using the command-line utility H2ASH. See the online file "tsm_util.txt" for further information about this utility.

Default = On

## Macro definition ignored Compiler warning

(Command-line option to suppress warning: **-w-nma**)

An error has occurred while using the command-line utility H2ASH. See the online file "tsm_util.txt" for further information about this utility.

## Constructor initializer list ignored Compiler warning

(Command-line option to suppress warning: **-w-ncl**)

An error has occurred while using the command-line utility H2ASH. See the online file "tsm_util.txt" for further information about this utility.

## Function body ignored Compiler warning

(Command-line option to suppress warning: **-w-nfd**)

An error has occurred while using the command-line utility H2ASH. See the online file "tsm_util.txt" for further information about this utility.

## Initializer for object 'x' ignored Compiler warning

`(Command-line option to suppress warning: **-w-nin**)`

An error has occurred while using the command-line utility H2ASH. See the online file "tsm_util.txt" for further information about this utility.

## Functions with exception specifications are not expanded inline

Compiler message

**Also: Functions taking class by value agruments are not expanded inline**

Exception specifications are not expanded inline: Check your inline code for lines containing exception specification.

Functions taking class-by-value argument(s) are not expanded inline: When exception handling is enabled, functions that take class arguments by value cannot be expanded inline.

**Note:** Functions taking class parameters by reference are not subject to this restriction.

## #undef directive ignored Compiler warning

`(Command-line option to suppress warning: **-w-nmu**)`

An error has occurred while using the command-line utility H2ASH. See the online file "tsm_util.txt" for further information about this utility.

## Declaration of static function *function* ignored Compiler warning

(Command-line option to suppress warning: **-w-nsf**)

An error has occurred while using the command-line utility H2ASH. See the online file "tsm_util.txt" for further information about this utility.

## Temporary used for parameter '???' Compiler message

In C++, a variable or parameter of reference type must be assigned a reference to an object of the same type. If the types do not match, the actual value is assigned to a temporary of the correct type, and the address of the temporary is assigned to the reference variable or parameter.

The warning means that the reference variable or parameter does not refer to what you expect, but to a temporary variable, otherwise unused.

In the following example, function f requires a reference to an int, and c is a char:

```
f(int &amp);
char c;
f(c);
```

Instead of calling f with the address of c, the compiler generates code equivalent to the C++ source code:

```
int X = c, f(X);
```

## Temporary used for parameter 2 in call to '???' Compiler message

In C++, a variable or parameter of reference type must be assigned a reference to an object of the same type. If the types do not match, the actual value is assigned to a temporary of the correct type, and the address of the temporary is assigned to the reference variable or parameter.

The warning means that the reference variable or parameter does not refer to what you expect, but to a temporary variable, otherwise unused.

In the following example, function f requires a reference to an int, and c is a char:

```
f(int &amp);
char c;
f(c);
```

Instead of calling f with the address of c, the compiler generates code equivalent to the C++ source code:

```
int X = c, f(X);
```

## Internal compiler error Compiler message

An error occurred in the internal logic of the compiler. This error shouldn't occur in practice, but is generated in the event that a more specific error message is not available.

## DPMI programs must use the large memory model

Compiler message

DPMI programs can only use large memory model. Tiny, Small, Medium, Compact, and Huge memory models are not allowed.

## Recursive template function: '" instantiated "

Compiler message

The compiler has detected a recursive template function instance. For example:

```
template<class T> void f(T x)
{
   f((T*)0);   // recursive template function!
}

void main()
{
   f(0);
}
```

The compiler will issue one message per each nesting of the recursive instantiation, so it is usually quite obvious where the recursion has occurred. To fix a recursive template, either change the dependencies, or provide a specialized version that will stop the recursion. For example, adding the following function definition to the above program will remove the endless recursion:

```
void f(int **)
{
}
```

## Class 'classname' is abstract because of 'member = 0' Compiler message

This message is issued immediately after the "Cannot create instance of abstract class 'classname' error message and is intended to make it easier to figure out why a particular class is considered abstract by the compiler.

For example, consider the following example of an illegal attempt to instantiate an abstract class:

```
struct VB
{
  virtualvoid  f() = 0;
  virtualvoid  g() = 0;
  virtualvoid  h() = 0;
};
struct D1 : virtual VB
{
    void   f();
};
struct D2 : virtual VB
{
    void   h();
};
struct DD : D1, D2
{
}
  v;   // error 'DD' is an abstract class
```

The above code will cause the following two error messages:

```
Error TEST.CPP 21: Cannot create instance of abstract class 'DD'
Error TEST.CPP 21: Class 'DD' is abstract because of 'VB::g() = 0'
```

## User-defined message Compiler message

The error message on which you have requested Help, is a user-defined warning.

In C++Builder code, user-defined messages are introduced by using the **#pragma** message compiler syntax.

**Note:** In addition to messages that you introduce with the **#pragma** message compiler syntax, user-defined warnings can be introduced by third party libraries. Should you require Help about a third party warning, please contact the vendor of the header file that issued the warning.

## Internal code generator error Compiler message

An error has occurred in the internal logic of the code generator. Contact Borland technical support.

## Conversions of class to itself or base class not allowed Compiler message

You tried to define a conversion operator to the same class or a base class.

## Cannot declare or define 'identifier' here

Compiler message

You tried to declare a template in an illegal place or a namespace member outside of its namespace.

## Invalid use of namespace 'identifier' Compiler message

A namespace identifier was used in an illegal way, for example, in an expression.

## Cannot define 'identifier' using a namespace alias

Compiler message

You cannot use a namespace alias to define a namespace member outside of its namespace.

## Namespace name expected Compiler message

The name of a namespace symbol was expected.

## Namespace member 'identifier' declared outside its namespace

Compiler message

Namespace members must be declared inside their namespace. You can only use explicit qualification to define a namespace member (for example, to give a body for a function declared in a namespace). The declaration itself must be inside the namespace.

## Invalid template member definition Compiler message

After the declarator of a template member, either a semicolon, an initialization, or a body was expected, but some other, illegal token was found. This message appears when a template member is declared outside of the template, but the syntax was wrong.

## Cannot use local type 'identifier' as template argument Compiler message

A local type was used in an actual template type argument, which is illegal.

## CodeGuarded programs must use the large memory model and be targeted for Windows Compiler message

Only issued by the 16-bit compiler. Programs that have CodeGuard enabled must use the large memory model and be targeted for Windows.

## Local data exceeds segment size limit

Compiler message

The local variables in the current function take up more than 64K.

## RTTI not available for expression evaluation

Compiler message

Expressions requiring RTTI are not supported by the expression evaluator in the integrated debugger. This error message is only issued by the expression evaluator (if you try to Inspect, Watch, or Evaluate), not by the compiler.

## Specialization after first use of template Compiler message

A new ANSI C++ rule requires that a specialization for a function template be declared before its first use. This error message is only issued when the ANSI conformance option (-A) is active.

## Earlier declaration of 'identifier'

Compiler message

This error message only shows up after the messages "Multiple declaration for 'identifier'" and "Type mismatch in redeclaration of 'identifier'". It tells you where the previous definition of the identifier in question was found by the compiler, so you don't have to search for it.

## Templates and overloaded operators cannot have C linkage Compiler message

You tried to use a linkage specification with a template or overloaded operator. The most common cause for this error message is having the declaration wrapped in an extern "C" { linkage specification.

## Structure packing size has changed Compiler warning

(Command-line option to suppress warning: **-w-pck**)

This warning message is issued when the structure alignment is different after including a file than it was before including that file.

The intention is to warn you about cases where an include file changes structure packing, but by mistake doesn't restore the original setting at the end. If this is intentional, you can give a `#pragma nopackwarning` directive at the end of an include file to disable the warning for this file.

The warning can be disabled altogether by `#pragma warn -pck`.

## Continuation character \ found in // comment

Compiler warning

(Command-line option to suppress warning: **-w-com**)

This warning message is issued when a C++ // comment is continued onto the next line with backslash line continuation.

The intention is to warn about cases where lines containing source code unintentionally become part of a comment because that comment happened to end in a backslash.

If you get this warning, check carefully whether you intend the line after the // comment to be part of the comment. If you don't, either remove the backslash or put some other character after it. If you do, it's probably better coding style to start the next comment line with // also.

The warning can be disabled altogether with `#pragma warn -com`.

## Assembler stack overflow

Compiler message

The assembler ran out of memory during compilation. Review the portion of code flagged by the error message to ensure that it uses memory correctly.

## Call to undefined function 'function' Compiler message

Your source file declared the current function to return some type other than **void** in C++ (or **int** in C), but the compiler encountered a return with no value. All **int** functions are exempt in C because in old versions of C, there was no **void** type to indicate functions that return nothing.

## Null pointer assignment Runtime message

When a small or medium memory model program exits, a check is made to determine if the contents of the first few bytes within the program's data segment have changed. These bytes would never be altered by a working program. If they have been changed, this message is displayed to inform you that (most likely) a value was stored to an uninitialized pointer.

The program might appear to work properly in all other respects; however, this is a serious bug which should be attended to immediately. Failure to correct an uninitialized pointer can lead to unpredictable behavior (including locking the computer up in the large, compact, and huge memory models).

You can use the integrated debugger to track down null pointers.

## Pure virtual function called Runtime message

This is a runtime error. It is generated if the body of a pure virtual function was never generated and somehow the compiler tried to call it.

## Invalid 'expression' in scope override

Compiler message

The evaluator issues this message when there is an error in a scope override in an expression you are watching or inspecting. You can specify a symbol table, a compilation unit, a source file name, etc. as the scope of the expression, and the message will appear whenever the compiler cannot access the symbol table, compilation unit, or whatever.

## String literal not allowed in this context

Compiler message

This error message is issued by the evaluator when a string literal appears in a context other than a function call.

## The function 'function' is not available

Compiler message

You tried to call a function that is known to the evaluator, but which was not present in the program being debuggedfor example, an inline function.

## Missing 'identifier' in scope override Compiler message

The syntax of a scope override is somehow incomplete. The evaluator issues this message.

## Cannot take address of member function 'function'

Compiler message

An expression takes the address of a class member function, but this member function was not found in the program being debugged. The evaluator issues this message.

## Invalid function call Compiler message

A requested function call failed because the function is not available in the program, a parameter cannot be evaluated, and so on. The evaluator issues this message.

## Printf/Scanf floating-point formats not linked

Runtime message

Floating-point formats contain formatting information that is used to manipulate floating-point numbers in certain runtime library functions, such as scanf() and atof(). Typically, you should avoid linking the floating-point formats (which take up about 1K) unless they are required by your application. However, you must explicitly link the floating-point formats for programs that manipulate fields in a limited and specific way.

Refer to the following list of potential causes (listed from most common to least common) to determine how to resolve this error:

- **CAUSE**: Floating point set to None. You set the floating-point option to None when it should be set to either Fast or Normal.

  **FIX**: Set Floating Point to Fast or Normal.

- **CAUSE**: Either the compiler is over-optimizing or the floating-point formats really do need to be linked. You need the floating-point formats if your program manipulates floats in a limited and specific way. Under certain conditions, the compiler will ignore floating-point usage in scanf(). For example, this may occur when trying to read data into a float variable that is part of an array contained in a structure.

  **FIX**: Add the following code to one source module:

```
extern _floatconvert;
#pragma extref _floatconvert
```

- **CAUSE**: You forgot to put the address operator & on the scanf variable expression. For example:

```
float foo;
scanf("%f", foo);
```

  **FIX**: Change the code so that the & operator is used where needed. For example, change the above code to the following:

```
float foo;
scanf("%f", &foo);
```

## Illegal number suffix

A numeric literal is followed by a suffix that is not recognized by the compiler.

Example:

```
int i = 1234i15;   // Error: no i15 suffix
int j = 1234i16;   // OK
```

## Circular property definition

Indicates that a property definition relies directly or indirectly on itself.

Example:

```
struct pbase
{
     int __property ip1 = {read = ip2, write = ip2};
     int __property ip2 = {read = ip1, write = ip1};
};
```

The above code sample will cause this error message on any usage of ip1 or ip2.

## access specifier of property *property* must be a member function

Only member functions or data members are allowed in access specifications of properties.

Example:

```
int GlobalGetter(void)
{
     return 0;
}

struct pbase
{
     int MemberGetter(void) {return 1;}

     int __property ip1 = { read = GlobalGetter }; // Error
     int __property ip2 = { read = MemberGetter }; // OK
};
```

## Parameter mismatch in access specifier *specifier* of property *property*

The parameters of the member function used to access a property don't match the expected parameters.

Example:

```
struct pbase
{
        void Setter1(void)      {}
        void Setter2(int){}

     int __property ip1 = { write = Setter1 }; // Error
     int __property ip2 = { write = Setter2 }; // OK
};
```

## Storage specifier not allowed for array properties

Array properties cannot have a storage specification.

Example:

```
struct pbase
{
    int __property ap[char *] =
                            { stored = false }; // Error
};
```

## VCL style classes must be constructed using operator new

VCL style classes cannot be statically defined. They have to be constructed on the heap.

Example:

```
voidfoo(void)
{
     Tobject   o1;     // Error;
     Tobject  *o2 = new TObject();
}
```

## VCL style classes must be caught by pointer

It is only possible to catch a VCL style object by pointer.

Example:

```
void foo(TObject *p)
{
     try
     {
          throw(p);
     }

     catch    (TObject o)    // Error
     {
     }

     catch  (TObject *op)    // OK
     {
     }
}
```

## VCL classes have to be derived from VCL classes

It is not allowed to derive a VCL style class from a non-VCL style class.

Example:

```
struct base // base not a VCL style class
{
  int  basemem;
};

struct __declspec(delphiclass) derived : base // or
{
  int  derivedmem;
};
```

## *Specifier* requires VCL style class type

The stored, default, and nodefault storage specifiers are only allowed within property declarations of VCL style class types.

Example:

```
struct regclass
{
     int __property ip1 = { stored  = false }; // Error
     int __property ip2 = { default = 42    }; // Error
     int __property ip3 = { nodefault       }; // Error
};

struct __declspec(delphiclass) vclclass
{
     int __property ip1 = { stored  = false }; // OK
     int __property ip2 = { default = 42    }; // OK
     int __property ip3 = { nodefault       }; // OK
};
```

## VCL style classes require exception handling to be enabled

If you are using VCL style classes in your program, you cannot turn off exception handling (compiler option **-x-**) when compiling your source code.

## __classid requires definition of TClass as a pointer type

To use __classid, there needs to be a definition for *TClass* which can be found in vcl.h.

Example:

```
// #include <vcl/vcl.h> missing

struct __declspec(delphiclass)bar
{
     virtual int barbara(void);
};

void *foo(void)
{
     return classid(bar);    // Error
}
```

## __published or __automated sections only supported for VCL classes

The compiler needs to generate a special kind of vtable for classes containing __published and __automated sections. Therefore, these sections are only supported for VCL style classes.

Example:

```
struct regclass
{
  int  mem;
__published:      // Error: no VCL style class
  int __property ip = { read = mem, write = mem };
};

struct __declspec(delphiclass) vclclass
{
  int  mem;
__published:      // OK
  int __property ip = { read = mem, write = mem };
};
```

## Static data members not allowed in __published or __automated sections

Only nonstatic data members and member functions are allowed in __published or __automated sections.

Example:

```
struct __declspec(delphiclass) vclclass
{
__published:
  static int      staticDataMember; // Error
};
```

## Illegal type *type* in __automated section

Only certain types are allowed in __automated sections.

Example:

```
struct __declspec(delphiclass) vclclass
{
__automated:
  int  __fastcall fooInt(int);      // OK
  long __fastcall fooLong(long);    // Error: long illegal
};
```

## Data member definition not allowed in __automated section

Only member function declarations are allowed in __automated sections.

Example:

```
struct __declspec(delphiclass) vclclass
{
__automated:
  int  __fastcall fooInt(int);      // OK
  int  memInt;                      // Error
};
```

## Only __fastcall functions allowed in __automated section

The calling convention for functions declared in an __automated section must be __fastcall.

Example:

```
struct __declspec(delphiclass) vclclass
{
__automated:
  int  __fastcall fooInt(int);      // OK
  int  __cdecl    barInt(int);      // Error
};
```

## Constructors and destructors not allowed in __automated section

Compiler error

Only member function declarations are allowed in __automated sections.

Example:

```
struct __declspec(delphiclass) vclclass
{
__automated:
  int  __fastcall fooInt(int);      // OK
  vclclass()  {}                    // Error
};
```

## Redeclaration of property not allowed in __automated section Compiler error

If you declare a property in an __automated section it has be a new declaration. Property hoisting is not allowed.

Example:

```
struct __declspec(delphiclass) vclbaseclass
{
  int  __fastcall Get(void);
  void   __fastcall Set(int);
  int __property ip1 = { read = Get, write = Set };
};

struct vclderivedclass : vclbaseclass
{
  int  __fastcall NewGetter(void);
__automated:
  __property ip1; // Error
  int __property ip2 = { read = Get, write = Set }; // OK
};
```

## Only read or write clause allowed in property declaration in __automated section

Compiler error

Storage specifiers stored, default, and nodefault are not allowed in property declarations in __automated sections.

Example:

```
struct __declspec(delphiclass) vclclass
{
  int  __fastcall Get(void);
__automated:
  int __property ip1 = { read = Get };                 // OK
  int __property ip2 = { read = Get, default = 42 }; // Error
};
```

## Dispid *number* already used by *identifier*  Compiler error

Dispids must be unique and the compiler checks for this.

Example:

```
struct __declspec(delphiclass) vclclass
{
__automated:
  int  __fastcall foo1(void) __dispid(42);      // OK
  int  __fastcall foo2(void) __dispid(42);      // Error
};
```

## Dispid only allowed in __automated sections

Compiler error

The definition of dispids is only permitted in __automated sections.

Example:

```
struct __declspec(delphiclass) vclclass
{
  int  __fastcall foo1(void) __dispid(42);      // Error
__automated:
  int  __fastcall foo2(void) __dispid(43);      // OK
};
```

## Suspicious pointer arithmetic Compiler error

This message indicates an unintended side effect to the pointer arithmetic (or array indexing) found in an expression.

Example:

```
#pragma warn +spa

int array[10];

int foo(__int64 index)
{
  return array[index];
}
```

The value of index is 64 bits wide while the address of array is only 32 bits wide.

## No type OBJ file present. Disabling external types option.

Compiler warning

(Command-line option to suppress warning: **-w-nto**)

A precompiled header file references a type object file, but the type object file cannot be found. This is not a fatal problem but will make your object files larger than necessary.

## Comparing signed and unsigned values Compiler warning

(Command-line option to suppress warning: **-w-csu**)

Since the ranges of signed and unsigned types are different the result of an ordered comparison of an unsigned and a signed value might have an unexpected result.

Example:

```
#pragma warn +csu

bool foo(unsigned u, int i)
{
  return u < i;
}
```

## Negating unsigned value Compiler warning

(Command-line option to suppress warning: **-w-ngu**)

Basically, it makes no sense to negate an unsigned value because the result will still be unsigned.

Example:

```
#pragma warn +ngu

unsigned foo(unsigned u)
{
  return -u;
}
```

## Throw expression violates exception specification

Compiler warning

(Command-line option to suppress warning: **-w-thr**)

The function contains a throw expression whose type is not specified in the exception specification in the function header.

Example:

```
#pragma warn +thr

void foo(int i) throw(double)
{
  throw i;
}
```

## Simple type name expected Compiler error

To ensure interoperability between Delphi and C++ Builder, there are restrictions on the types names mentioned in the parameter lists of published closure types. The parameter types have to be simple type names with optional const modifier and pointer or reference notation.

Example:

```
struct __declspec(delphiclass) foo
{
     typedef void __fastcall (__closure *foo1)(SomeTemplateType<int> *);

     typedef SomeTemplateType<int> SimpleTypeName;
     typedef void __fastcall (__closure *foo2)(SimpleTypeName *);

published:

     __property foo1 prop1;  // Error
     __property foo2 prop2;  // OK
};
```

## Function call terminated by unhandled exception 'value' at address 'addr'

Compiler error

This message is emitted when an expression you are evaluating while debugging includes a function call that terminates with an unhandled exception. For example, if in the debugger's evaluate dialog, you request an evaluation of the expression `foo()+1` and the execution of the function `foo()` causes a GP fault, this evaluation produces the above error message.

You may also see this message in the watches window because it also displays the results of evaluating an expression.

## VCL style classes need virtual destructors Compiler error

Destructors defined in VCL style class have to be virtual.

Example:

```
struct __declspec(delphiclass) vclclass1
{
  ~vclclass1() {}           // Error
};

struct __declspec(delphiclass) vclclass2
{
  virtual ~vclclass2() {}   // OK
};
```

## = expected Compiler error

The compiler expected an equal sign in the position where the error was reported but there was none. This is usually a syntax error or typo.

## Published property access functions must use __fastcall calling convention

Compiler error

The calling convention for access functions of a property (read, write, and stored) declared in a __published section must be __fastcall. This also applies to hoisted properties.

Example:

```
struct __declspec(delphiclass) vclclass
  {
    int  __fastcall Getter1(void);
    int  __cdecl    Getter2(void);
  __published:
    int __property ip1 = {read = Getter1};    // OK
    int __property ip2 = {read = Getter2};    // Error
  };
```

## **Bad character in parameters -> 'char'** Linker message

One of the following characters was encountered in the command line or in a response file:

" * < = > ? [ ] |

or any control character other than horizontal tab, line feed, carriage return, or Ctrl+Z.

## Unable to open file 'filename' Linker message

The named file can't be found, possibly because it does not exist or is misspelled, or it resides in a different directory than those being searched.

If you are using the IDE, make sure you have set the appropriate directory paths.

## Unresolved type external *type* referenced from module *module* Linker message

This warning means that the compiler emitted debug information in a compilation unit which referred to an external type. The external type definition could not be found in another module or in a type OBJ. This can happen if you have Pascal code compiled with debug linked to Pascal code that is compiled without debug. If you are linking Pascal code into your project, it is best to compile it all with debug information if you plan to debug the Pascal code. Otherwise, compile all of the Pascal code without debug information.

## **Debug information overflow in module *module* near file *offset*** Linker message

This error message happens when you compiled with debug information, and are linking with debug information and a module contained more than 64K lines. You need to turn off debug information in the specified module, or break the module into multiple modules with fewer lines. The *offset* is the file offset in the OBJ where the problem occurred.

## Unknown Optimization Linker message

This error message refers to an UNDOCUMENTED optimization feature.

## Self relative fixup overflowed in module *module* Linker message

This message appears if a self-relative reference (a call or a jump) is made from one physical segment to another. This often happens when employing assembler code, but can occur if you use the segment-naming options in the compiler. If the reference is from one code segment to another, you are safe. If, however, the reference is from a code segment to a data segment, you have probably made a mistake in some assembler code.

You can suppress this warning with the **/w-srf** command-line option.

## Using based linking for DLLs may cause the DLL to malfunction Linker message

This warning occurs if you use the **/B** switch when linking a DLL. In almost every case, this is an error that will prevent the application from running.

You can suppress this warning with the **/w-bdl** command-line option.

## Public symbol 'symbol' defined in both module 'module1' and 'module2' Linker message

There is a conflict between two public symbols. This usually means that a symbol is defined in two modules.

- An error occurs if both are encountered in the .OBJ file(s), because TLINK doesn't know which is valid.
- A warning results if TLINK finds one of the duplicated symbols in a library and finds the other in an .OBJ file; in this case, TLINK uses the one in the .OBJ file.

## Import record does not match previous definition Linker message

This warning usually occurs if an IMPDEF record appears in an import library when the import in question is also imported from a .DEF file. If the description of the imports differ in internal name or ordinal, this warning appears, and the first definition is used.

You can suppress this warning with the **/w-imt** command-line option.

## Multiple stack segments found. The most recent one will be used. Linker message

This warning occurs when two stack segments of different names are defined in the object modules. The startup code defines a stack segment for the application.

You can suppress this warning with the **/w-msk** command-line option.

## Empty LEDATA record in module 'module' Linker message

This warning can happen if the translator emits a data record containing data. If this should happen, report the occurrence to the translator vendor. There should be no bad side effects from the record.

## Explicit stacks are ignored for PE images Linker message

Win32 apps are PE format applications, which do not have explicit stacks. The stack segment will be linked into the image, but it will not be used as the application stack. Instead, the stack size parameter will be used to set the stack size, and the operating system will allocate a stack for the application.

## Export 'symbol' is duplicated Linker message

This warning will occur if two different functions with the same name are exported by the use of _export. The linker cannot resolve which definition to export, and will use the first symbol.

## Old debug information in module 'module' will be ignored

Linker message

Debug information in the OBJ is incompatible with this linker, and will be ignored.

## Extern 'symbol' was not qualified with __import in module 'module' Linker message

Win32 applications which make reference to imported symbols need to make indirections to get to the data. For calls, this is handled automatically by the linker. For references to imported DATA, the compiler must generate an indirection, or the application will function incorrectly. The compiler knows to generate the indirection when the symbol is qualified with __import. If the linker sees a segment relative reference to a symbol which is imported, and if the symbol was not qualified with __import, you will get this message.

You can suppress this warning with the **/w-inq** command-line option.

## Debug information enabled, but no debug information found in OBJs message

Linker

No part of the application was compiled with debug information, but you requested that debug information be turned on in the link.

## 'filename' is not a valid library Linker message

This error happens when the first record in a library file is not the LIBSTART record.

This can happen when something that is not a library is passed as a library file to the linker, or if the library file was corrupted.

## Too many LNAMEs Linker message

TLINK32 supports up to 255 LNAMEs appearing in an OBJ file.

## Illegal ACBP byte in SEGDEF in module 'module' Linker message

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## Illegal component to GRPDEF in module 'module' Linker message

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## Unsupported option 'string'

Linker message

You specified an unknown option on the command line.

### Too many commas on command-line Linker message

You specified an invalid entry on the command line. Check the command you entered.

## Malformed command-line

Linker message

You specified an invalid entry on the command line. Check the command you entered.

## Bad file name 'filename' Linker message

An invalid file name was passed to the linker.

## Failed read from 'filename'

Linker message

The linker was unable to read from the file.

## **Failed write to 'filename'** Linker message

The linker was unable to write to the file.

## Terminated by user

Linker message

The user pressed Ctrl+Break.

## Too many errors Linker message

The linker got more errors than the maximum number of errors specified by the user.

## Bad object file 'filename' near file offset 'offset'

Linker message

The linker has found a bad OBJ file. This is usually caused by a translator error.

## No output file specified Linker message

You did not specify an output file.

## Image base address must be a multiple of 0x10000

Linker message

Based images must be aligned on 64K boundaries.

## General error in module 'module'

Linker message

TLink32 emits a General Error and crashes if the user specifies a map file that cannot be opened. This usually occurs when the directory (or drive) for the command-line specified map file doesn't exist. This linker is trying to emit the following message to the map file:

```
Fatal: cannot open mapfile...
```

Make sure that the map file destination drive and directory is correct.

## General error

Linker message

An unhandled exception occurred in the linker. Inform Borland Technical Support of the circumstances.

## Mixed common types in module 'module'. Cannot mix COMDEFs and VIRDEFs.

Linker message

You cannot mix both COMDEFs and VIRDEFs. Turn off the -Fc switch to stop generating COMDEFs, or turn on the -Vs switch to stop generating VIRDEFs.

## Out of memory for block 'block'

Linker message

The linker ran out of memory. Try reducing the size of disk caches and/or RAM drives.

## **Bad LF_POINTER in module 'module'** Linker message

This is typically caused by bad debug information in the OBJ file. Borland Technical Support should be informed.

## Target index of FIXUP is 0 in Module 'module'

Linker message

This is a translator error.

## Bad field list in debug information in module 'module'

Linker message

This is typically caused by bad debug information in the OBJ file. Borland Technical Support should be informed.

## Couldn't load DLL 'dll'  Linker message

The linker was not able to load the specified DLL. Check to make sure that the DLL is on your path.

## Couldn't get procedure address from DLL 'dll'

Linker message

The linker was not able to get a procedure from the specified DLL. Check to make sure that you have the correct version of the DLL.

## Incorrect version of RLINK32.DLL

Linker message

You don't have a the right version of RLINK32.DLL. Check to make sure that you have the correct version of the DLL. If not, delete that DLL and reinstall C++Builder.

## Too many default libraries

Linker message

The linker can handle a maximum of 128 default libraries.

## 16-bit segments not supported in module 'module'

Linker message

16-bit segments are not supported for Win32 applications. Check to make sure that you have not compiled the application with the 16-bit compiler.

## Multiple entry points defined Linker message

Multiple entry points were defined for the application. This can happen if you have specified the startup code twice, or if you are making use of assembler code which defines a starting address for the application.

## Duplicate ordinal for exports: 'string' ('ordval1') and 'string' ('ordval2')

Linker message

Two exports share the same ordinal. The linker cannot resolve which export should get which ordinal.

## Attempt to export non-public symbol 'symbol'

Linker message

The EXPORTS section in the .DEF file specified the name of a symbol which has no public definition.

## Fixup to zero length segment in module *module* Linker message

A reference has been made past the end of an image segment. This reference would end up accessing an invalid address, and has been flagged as an error.

## No internal name for IMPORT in .DEF file Linker message

The .DEF file has a semantic error. You typed an illegal import sequence or forgot to put the internal name for an import before the module name. For example:

```
IMPORTS
     _foo.1
```

Here, _foo was to be the function to be imported, but the proper syntax is:

```
IMPORTS
     _foo=mydll.1
```

## Converting reference to external type *type* to void

Linker message

This warning means that the linker was unable to resolve a reference within the debug information to a legitimate type. Try deleting your precompiled header files and type OBJ files (*.#*) and rebuilding your project. In the message, type is the name of the type that couldn't be converted.

## Debug info switch ignored for COM files Linker message

C++Builder does not include debug information for .COM files. See the description of the **/v** option.

## Export 'symbol' has multiple ordinal values: 'value1' and 'value2' Linker message

The linker encountered one symbol with multiple ordinal values. The linker cannot resolve which value to export, and will use the first value.

Check the EXPORTS section of the module definition file and make sure that the export name applies only once.

## Bad secondary target for fixup in module 'module'

Linker message

The linker encountered an object file with an incompatible .OBJ file format.

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## THREAD fixup found in module 'module'

Linker message

The linker encountered an object file with an incompatible .OBJ file format.

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## Far COMDEFs are not supported

Linker message

The linker encountered an object file with an incompatible .OBJ file format.

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## Illegal type of entry point

Linker message

The linker encountered an object file with an incompatible .OBJ file format.

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## Unsupported COMENT OMF extension 'extension' Linker message

The linker encountered an object file with an incompatible .OBJ file format.

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## Fixupps found for an LIDATA record Linker message

The linker encountered an object file with an incompatible .OBJ file format.

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## Bad loc for fixupp in module 'module' near file offset 'offset'

Linker message

The linker encountered an object file with an incompatible .OBJ file format.

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## Bad *type* debug info in module *module*

Linker message

The linker encountered an object file with an incompatible .OBJ file format.

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## Block overflow for block 'block'

Linker message

This message results from one of the linker's internal tables overflowing.

The internal linker tables contain information such as linker or debugging data. The tables are set to a default size which was not sufficient for the data in your application.

When this error occurs, the linker emits an INI file called TLINK32.INI and includes the name of the block that overflowed along with its size. Edit the INI file and increase the size.

The .DEF file is ill formed. You typed an illegal import sequence in the .DEF file.

## STACK/HEAP commit 'size' greater than reserve 'size'  Linker message

The heap or stack commit size has to be less than or equal to the stack or heap reserve size. Change the stack and heap reserve or commit sizes by changing the 32-bit linker values in the IDE or by changing HEAPSIZE and STACKSIZE in the module definition file.

## Invalid stack reserve/commit size 'size'

Linker message

The heap or stack commit size specified is not valid. The default and minimum values for reserve and commit sizes are shown here.

| Reserve | Default | Minimum |
|---|---|---|
| **Stack** | 1Mb | 4K |
| **Heap** | 1Mb | 0K |

| Commit | Default | Minimum |
|---|---|---|
| **Stack** | 8K | 4K |
| **Heap** | 4K | 0K |

Change the stack and heap reserve or commit sizes by changing the 32-bit linker values in the IDE or by changing HEAPSIZE and STACKSIZE in the module definition file.

## Invalid file/object alignment value 'value'  Linker message

You specified an incorrect value for file or object alignment in the 32-bit linker options. The value must be a number (either decimal or hex) that is a power of 2. The smallest allowable file alignment value is 16. The smallest allowable object alignment value is 4096.

## Image linked as EXE, but with DLL extension  Linker message

The linker generates this warning when an executable file has been generated and stored in a file with a .DLL extension.

This usually occurs when you intended to build a .DLL but forgot to specify a .DLL target with the **/T** linker option (or you forgot the **/T** option altogether) on the command line.  If you want to generate a DLL as a target, use the appropriate **/T** option.

## Possible unresolved external 'symbol' referenced from module 'module'

Linker message

This warning appears only for static data members of classes that have been declared but not defined.

## Images fixed at specific addresses typically will not run under Win32s

Linker message

Windows 32s loads all applications in a single address space. It's possible to predict where your application is going to be loaded, because other 32-bit applications might have been loaded before yours.

## Non-existent segment 'segment' in SEGMENTS section of .DEF file Linker message

You specified a segment name in the SEGMENTS section of the .DEF file which doesn't exist in any of the object files or library files included in the link.

## .DEF file stack/heap reserve size < 64K; 1MB default will be used

Linker message

The reserve size for either the STACK or the HEAP specified in the .DEF file is less than 64K. The linker emits this warning to inform you that it will use the default of 1MB instead of the value specified.

## Multiple public definitions for symbol 'symbol' in module 'module;' link case sensitively Linker message

The specified symbol was encountered twice in the same module. This is usually caused by the use of case-sensitive symbols. Try linking case sensitively.

## Unresolved external 'symbol' referenced from module 'module'

Linker message

The named symbol is referenced in the given module but is not defined anywhere in the set of object files and libraries included in the link.

Check to make sure the symbol is spelled correctly.

You will usually see this error from the linker for C or C++ symbols if any of the following occur:

- You did not properly match a symbol's declarations of pascal and cdecl type in different source files.
- You have omitted the name of an .OBJ file your program needs.
- You did not link in the emulation library.

If you are linking C++ code with C modules, you might have forgotten to wrap C external declarations in `extern "C" {...}`.

You could also have a case mismatch between two symbols.

## T3 and T7 fixupps not allowed (module 'module') Linker message

The linker encountered an object file with an incompatible .OBJ file format.

This error generally occurs due to an incompatible .OBJ file format. If you're linking .OBJ files compiled using another compiler, these files may be incompatible with TLINK32. You'll need to recompile all modules using one of the C++Builder compilers.

This can also occur if the machine was rebooted during a compile, or if a compiler did not delete its output object file when Ctrl+Break was pressed. Recompile.

If the error persists, call Borland Technical Support.

## Could not find precompiled type object file *filename*.

The linker could not find the OBJ file that contains some of the debug information needed for the link. This most commonly happens if your library path (-L option) doesn't point to the place where your precompiled header files are. Make sure that the path is correct and try again. This warning may also appear if the type OBJ files were deleted, and the precompiled header file was not. In this case, delete the precompiled header files, and rebuild the application.

## **Unknown option 'option'** Linker message

A forward slash character (/), hyphen (-), or DOS switch character was encountered on the command line or in a response file without being followed by one of the allowable options. You might have used the wrong case to specify an option.

## Invalid target /T 'target' Linker message

The command-line linker found an invalid target. Valid targets are 'w' and 'd.'

## Unknown Goodie Linker message

An unsupported option was supplied to the command-line linker.

## Invalid size specified for segment packing Linker message

A non-decimal number was provided on the command line for the segment packing size limit.

## Invalid size specified for segment alignment

Linker message

This error occurs if an invalid value is specified for the Segment Alignment setting. The value specified must be an integral multiple of 2 and less than 64K. Common values are 16 and 512. This error only occurs when linking Windows applications.

### Invalid map filename: 'filename'

Linker message

The map file name had an incorrect extension, such as .OBJ, .EXE, .DLL, .LIB, .DEF, or .RES.

### **Invalid exe filename: 'filename'** Linker message

The .EXE file name has an incorrect extension such as .OBJ, .MAP, .LIB, .DEF, or .RES.

## Too many file names Linker message

This error occurs if the linker encounters more than 64K characters in the response file. The linker only handles response files up to 64K.

You'll need to shorten the response file, shorten the pathnames, or chunk the .OBJs into a library.

## End of system input buffer encountered Linker message

The input line you typed is too long. Instead of typing all you're object and library files on the command line, put them into a response file.

## Invalid overlay switch specification

Linker message

You specified an overlay option but omitted the file name or names. Delete the switches or add the names of the files containing the overlays.

## No program entry point Linker message

This warning message appears if no starting execution point was defined in the application. This usually happens if you forget to link in the startup code.

You can suppress this warning with the **/w-ent** command-line option.

## No DEF file Linker message

This warning message appears if the module definition file is missing from the project.

## Out of memory in block *address*

The linker has run out of memory.

**Solutions**

You can try reducing size of active RAM disks and/or disk caches.

Close one or more applications to free memory.

## Malloc of *number* bytes failed in *module*, line *number*

The linker has run out of memory.

**Solutions**

You can try reducing size of active RAM disks and/or disk caches.

Close one or more applications to free memory.

## Realloc of *number* bytes failed in *module*, line *number*

The linker has run out of memory.

**Solutions**

You can try reducing size of active RAM disks and/or disk caches.

Close one or more applications to free memory.

## Attempt to realloc NULL pointer in *module*, line *number*

This message is typically caused by bad debug information in the OBJ file. You should note the circumstances under which this message occurred and inform Borland Technical Support.

## Attempt to free NULL pointer in *module*, line *number*

Internally, an attempt was made to free a block of memory that was not allocated. Try using Build All to recompile the application. If you still receive this message, note the circumstances under which this message occurred and inform Borland Technical Support.

## Out of memory

The linker has run out of dynamically allocated memory needed during the link process. The total working storage is exhausted.

This error is a catchall for running into a limit on memory usage. This usually means that too many modules, externals, groups, or segments have been defined by the object files being linked together.

**Solutions**

You can try reducing size of active RAM disks and/or disk caches.

Close one or more applications to free memory.

## Assertion failed: *module* at "*address*\", line *number*

This is an internal error. You should note the circumstances under which this message occurred and inform Borland Technical Support.

## Access violation.  Program terminated.

You should note the circumstances under which this message occurred and inform Borland Technical Support.

## Illegal *type* fixup index in module *module*

The object file contains an invalid fixup index. This can result if the compiler emits the wrong the index. It can also happen if the object file is corrupted. Try to recompile that object file. If this message still persists, contact Borland Technical Support.

## User break. Link aborted.

You typed Ctrl+Break while in the linker.  The link was aborted. (This is not an error, just a confirmation.)

## Out of disk space

Your disk is full. Check to ensure that you have write access to the disk.

## Cannot write to disk

Writing to the specified disk failed.  Check to see if the disk is full.

## *module*: ILINK32 does not support segmentation - use TLINK32

A module processed by ILINK32 contained multiple user-defined code segments or user-defined data segments of different classes. ILINK32 does not support this. You must either eliminate these additional segments or use TLINK32.

## Undefined symbol *symbol* referenced from *module*

The named symbol is referenced in the given module but is not defined anywhere in the set of object files and libraries included in the link. Check to make sure the symbol is spelled correctly.

You will usually see this error from the linker for C or C++ symbols if any of the following occur:

- You did not properly match a symbol's declarations of **__pascal** and **__cdecl** types in different source files.
- You have omitted the name of an .OBJ file your program needs.
- You did not link in the emulation library.

If you are linking C++ code with C modules, you might have forgotten to wrap C external declarations in extern "C".

You could also have a case mismatch between two symbols.

## Attempt to export non-public symbol *symbol*

A symbol name was listed in the EXPORTS section of the module definition file, but no symbol of this name was found as public in the modules linked.

If compiling in C++Builder, this is usually caused by the name mangling that occurs as a result of C++Builder type safe linkage. Inserting the **_export** keyword in the function prototype and function definition is required for all Windows callback functions.

Language-independent causes result from a mistake in spelling or case, case-sensitive exports, or a procedure with this name that was not defined.

If you are using case-sensitive exports, the Pascal calling convention used by Windows requires these symbols to be all uppercase characters.

## Import by ordinal not supported by ILINK32

You attempted to import by ordinal in the IMPORTS section of a .DEF file. This is not supported by ILINK32. ILINK32 only supports importing by name. Import by name or else use the **-o** flag in TLINK32.

## Public symbol '*symbol*' defined in both module *module1* and *module2*

There is a conflict between two public symbols. This means that a symbol is defined in two modules.

## Could not find object file *filename*

The compiler is unable to find the file supplied on the command line.

## Error processing module *module*

The incremental linker is unable to process the module named in the error message.

## Unable to open file *filename*

The named file can't be found, possibly because it does not exist or is misspelled, or it resides in a different directory than those being searched.

If you are using the IDE, make sure you have set the appropriate directory paths in the Options | Directories dialog box.

## RLINK32 was not initialized

The RLINK32.DLL failed to initialize. The .DLL is either corrupted or missing.

## Could not load RLINK32.DLL

The linker was not able to load the specified DLL. Check to make sure that the DLL is in a directory that is in your path.

## Could not get procedure address from RLINK32.DLL

A procedural export is missing from RLINK32.DLL. You don't have the right version of RLINK32.DLL in the C++Builder BIN directory. You need to delete that version of the DLL and replace it with the correct version from the C++Builder source. You can either copy it from another system or reinstall Borland C++ Builder.

## Incompatible version of RLINK32.DLL

You don't have the right version of RLINK32.DLL in the C++Builder BIN directory. You need to delete that version of the DLL and replace it with the correct version from the C++Builder source. You can either copy it from another system or reinstall Borland C++ Builder.

## Unknown RLINK32 error

Error processing resources (general error).Try using Build All to recompile the application. If you still receive this message, note the circumstances under which this message occurred and inform Borland Technical Support.

## Undefined external type *data-type*

This happens if you mix object files containing debug information with object files that do not have debug information. It means that the debugger will not be able to display type-information (objects of that type will be “opaque” in the debugger). The solution is to compile with debug information turned on the module containing that type definition.

## Could not open *filename* (error code *number*)

The incremental linker was unable to open *filename*.

## Could not create *filename* (error code *number*)

The linker could not create the file *filename.*

## Could not open *filename* (program still running?)

Unable to open *filename.* Is the program still running?

## Could not open *filename* (project already open in IDE?)

Unable to open *filename.* Is the project already open in the IDE?

## Failed to create map file *filename* (error code *number*)

Attempt to create map file *filename* failed.

## Symbol *symbol* marked as __import in *module* was public

This happens if you compile one object module using imports (-D_RTLDLL, for example), and another module using static binding (no such compilation flag). The result is that one object will expect global variables within the module to require an indirection in the assembly code (because it is an import), while the other object will expect to reference the data directly.

The solution is to either compile all object modules to using imports, or compile them all to link statically.

## Internal failure -- Retrying link...

An error occurred in the internal logic of the linker.

This error shouldn't occur, but is listed for completeness in the event that a more specific error isn't generated.

If this error persists, write down the errorcode number and contact Borland Technical Support.

### State invalid: *module* at "*address*\", line line

Internal linker error. Call Borland Technical Support if you receive this error.

### *module* contains invalid OMF record, type 0x*HH*

The object file is corrupt. Regenerate it, or contact Borland Technical Support.

## General error in link set

An unhandled exception occurred in the linker. Inform Borland Technical support of the circumstances.

## General linker message

**OR** **General error in library file** *filename* **in module** *module* **near module file offset '0xyyyyyyyy'**

**OR** **General error in module** *module* **near module file offset '0xyyyyyyyy'**

The linker gives as much information as possible about what processing was happening at the time of the unknown fatal error.

Call Borland Technical Support with information about the .OBJ or .LIB files.

## Error parsing .DEF file

An error occurred while parsing the .DEF file. Check the .DEF file syntax.

## Unsupported 16-bit segment(s) in module *module*

You cannot link 16-bit segments into 32-bit applications. The only way you can code 16-bit segments is using the assembler (tasm32.exe).

## Bad OMF record type 'type' encountered in module 'module' Librarian message

The librarian encountered a bad Object Module Format (OMF) record while reading through the object module.

Because the librarian has already read and verified the header records in 'module', the object module is probably corrupt. Recreate it.

## Could not allocate memory for per module data

Librarian message

The librarian has run out of memory.

## Could not create list file 'filename'

Librarian message

The librarian could not create a list file for the library. This could be due to lack of disk space.

## Could not write output Librarian message

The librarian could not write the output file.

## **Error opening 'filename'** Librarian message

The librarian cannot open the specified file.

## Error opening 'filename' for output

Librarian message

The librarian cannot open the specified file for output.

## Error renaming 'filename' to 'filename'

Librarian message

This error occurs when the librarian is building a temporary library file and renaming the temporary file to the target library file name.

The error indicates that the target file is read only.

### Library too large, restart with library page size 'size'

Librarian message

The library being created could not be built with the current library page size.

## Not enough memory for command-line buffer

Librarian message

This error occurs when the librarian runs out of memory.

## Object module 'filename' is invalid

Librarian message

The librarian could not understand the header record of the object module being added to the library. The librarian assumes that it is an invalid module.

## Out of memory

Librarian message

For any number of reasons, the librarian or the IDE ran out of memory while building the library. For many specific cases, a more detailed message is reported.

Close one or more applications.

## Out of memory creating extended dictionary

Librarian message

The librarian ran out of memory while creating an extended dictionary for a library.

The library is created but will not have an extended dictionary.

## Out of memory reading LE/LIDATA record from object module Librarian message

The librarian is attempting to read a record of data from the object module, but it cannot get a large enough block of memory.

If the module being added has a large data segment or segments, try adding this module before other modules.

## Out of space allocating per module debug struct Librarian message

The librarian ran out of memory while allocating space for the debug information associated with a particular object module.

Try removing debugging information from the modules being added to the library to resolve the problem.

## Output device is full Librarian message

The output device is full. This error usually means that there is no space left on the disk.

### 'path' - path is too long Librarian message

This error occurs when the length of any of the library file or module file's 'path' is greater then 64.

## Public 'symbol' in module 'module1' clashes with prior module 'module2'

Librarian message

A public symbol can only appear once in a library file. A module, which is being added to the library, contains a public 'symbol' that is already in a module of the library and cannot be added.

The command-line message reports the module2 name.

## Record kind 'num' found, expected theadr or lheadr in module 'filename'

Librarian message

The librarian could not understand the header record of the object module being added to the library and has assumed that it is an invalid module.

## Record length 'len' exceeds available buffer in module 'module'

Librarian message

This error occurs when the record length 'len' exceeds the available buffer to load the buffer in module 'module'.

This occurs when the librarian runs out of dynamic memory.

## The combinations '+*' or '*+' are not allowed

Librarian message

It is not legal to add and extract an object module from a library in one action.

## User break, library aborted

Librarian message

You pressed Cancel while compiling in the IDE. The library was not created.

(This is not an error, just a confirmation.)

## Added file 'filename' does not begin correctly, ignored Librarian message

The librarian has decided that the file being added is not an object module. It will not try to add it to the library.

The library is created anyway.

## Cannot write GRPDEF list, extended dictionary aborted Librarian message

The librarian cannot write the extended dictionary to the end of the library file. There may not be enough space on the disk.

## 'filename' couldn't be created, original won't be changed

Librarian message

You tried to extract an object, but the librarian cannot create the object file into which to extract the module.

Either the object already exists and is read only, or the disk is full.

## 'reason' - extended dictionary not created Librarian message

### OR Library contains COMDEF records - extended dictionary not created

If the Library contains COMDEF records message is displayed, an object record being added to a library contains a COMDEF record. This is not compatible with the extended dictionary option.

## 'filename' file not found Librarian message

The IDE creates the library by removing the existing library and rebuilding it. If any of the specified objects do not exist, the library is incomplete and this error is generated.

If the IDE reports that an object does not exist, either the source module has not been compiled or there were errors during compilation.

## Invalid page size value ignored

Librarian message

The librarian encountered an invalid page size.

The page size must be an integer that is a power of 2.

## Memory full listing truncated! Librarian message

The librarian ran out of memory while creating a library listing file. An incomplete list file will be created.

## Results are safe in file 'filename'

Librarian message

The librarian has successfully built the library into a temporary file, but it cannot rename the file to the desired library name.

The temporary file will not be removed (so that the library can be preserved).

## Unknown command line switch 'X' ignored Librarian message

A forward slash character (/) was encountered on the command line or in a response file without being followed by one of the allowed options.

### @ seen, expected a response-files name

Librarian message

The response file name is not given immediately after @.

## Import 'symbol' in module 'module' clashes with prior module Librarian message

An import symbol can appear only once in a library file. A module that is being added to the library contains an import that is already in a module of the library and it cannot be added again.

## 'module' already in LIB, not changed! Librarian message

This warning indicates that you attempted to use the + action on the library, but an object with the same name already exists in the library. To update the module, the action should be +–. The library was not modified.

## Bad GRPDEF type encountered, extended dictionary aborted Librarian message

The librarian has encountered an invalid entry in a group definition (GRPDEF) record in an object module while creating an extended dictionary.

The only type of GRPDEF record that the librarian and the linker support is segment index type. If any other type of GRPDEF is encountered, the librarian can't create an extended dictionary. It is possible that an object module created by products other than Borland tools can create GRPDEF records of other types. A corrupt object module can also generate this warning.

## Bad header in input LIB Librarian message

When adding object modules to an existing library, the librarian found a bad library header. Rebuild the library.

## Can't grow LE/LIDATA record buffer Librarian message

Command-line error.

The librarian is attempting to read a record of data from the object module, but it cannot get a large enough block of memory.

If the module being added has a large data segment or segments, try adding this module before other modules.

## Couldn't get LE/LIDATA record buffer

Librarian message

Command-line error.

The librarian is attempting to read a record of data from the object module, but it cannot get a large enough block of memory.

If the module being added has a large data segment or segments, try adding this module before other modules.

## Duplicate file 'filename' in list, not added! Librarian message

When building a library module, you specified an object file more than once.

Error changing file buffer size

## Error changing file buffer size Librarian message

The librarian is attempting to adjust the size of a buffer used while reading or writing a file, but there is not enough memory. You'll need to free up a lot of system memory to resolve this error.

## 'filename' file not found

Librarian message

Command-line error.

The command-line librarian attempted to add a nonexisting object but created the library anyway.

## Ignored 'module', path is too long

Librarian message

The path to a specified .OBJ or .LIB file is greater than 64 characters. The maximum path to a file for the librarian is 64 characters.

## 'module' not found in library Librarian message

An attempt to perform either a remove (-) or an extract (*) on a library has occurred and the indicated object does not exist in the library.

## Record type 'type' found, expected theadr or lheadr in 'module'

Librarian message

The librarian encountered an unexpected type instead of the expected THEADR or LHEADER record in the specified module.

## Unable to open 'filename' for output Librarian message

The librarian cannot open the specified output file. This is usually due to lack of disk space for the target library, or a listing file.

## Unable to rename 'filename1' to 'filename2' Librarian message

The librarian builds a library into a temporary file (filename1) and then renames the temporary file to the target library file name (filename2). This message appears if an error occurs during the renaming process, such as insufficient disk space.

## Unexpected char X in command line Librarian message

The librarian encountered a syntactical error while parsing the command line.

## Use /e with TLINK to obtain debug information from library

Librarian message

The library was built with an extended dictionary and also includes debugging information. TLINK cannot extract debugging information if it links using an extended dictionary.

To obtain debugging information in an executable from this library, use the /e switch to cause the linker to ignore the extended dictionary.

**Note:** The IDE linker does not support extended dictionaries; therefore, no settings need to be altered in the IDE.

## Resource Linker Error Messages and Warnings

## Internal software error! Resource Linker message

The resource linker encountered unexpected data. Restart the resource link. If the error persists, contact Borland Technical Support.

## Out of memory! Resource Linker message

Not enough memory is available for compiling a particular file. In this case, shut down any other concurrent applications. You may also try to re-configure your machine for more available memory, or break up the source file being compiled into smaller separate components. You can also compile the file on a system with more available RAM.

## Bad EXE header format in file. Resource Linker message

Also: **Executable format not recognized in file.**

The executable file contained invalid information in its header. The file might not be a valid executable or might contain corrupted data.

## Bad EXE segment table in file. Resource Linker message

The executable file contained invalid information in its segment table. The file might not be a valid executable or might contain corrupted data.

## Error in packing preload area. Turn off preload packing. Resource Linker message

An error occurred while the resource linker was trying to optimize how resources and segments are arranged in the executable file. (This error only occurs for 16-bit resources.)

**Solution**

To correct this error, turn off the Pack Fastload Area option in the Resources Options.

## Error creating file. Resource Linker message

An error occurred when the resource linker tried to create a file. This error occurs if the work disk is full or write-protected. It can also occur if the output directory does not exist.

**Solutions**

- If the disk is full, try deleting unneeded files and restarting the resource link.
- If the disk is write-protected, direct the output to a writeable disk and restart the resource link.

## Error creating temporary file in directory. Resource Linker message

An error occurred when the resource linker tried to create a temporary file. This error occurs if the work disk is full or write-protected. It can also occur if the output directory does not exist.

**Solutions**

- If the disk is full, try deleting unneeded files and restarting the resource link.
- If the disk is write-protected, direct the output to a writeable disk and restart the resource link.

## **Error deleting file.** Resource Linker message

An error occurred when the resource linker tried to delete a file. This error occurs if the file is marked as read-only or does not exist.

**Solutions**

- If the disk is read-only, change its attributes so that it can be deleted.

## Error in CURSDIR. Cannot find CURS.

Resource Linker message

An entry was found in the cursor directory that had no corresponding cursor resource. The resource file is probably corrupted.

## **Error in EXE's resource table format.** Resource Linker message

There is invalid information in the executable files resource table. The executable file might contain invalid resource information or be corrupt.

## **Error in FONTDIR. Cannot find FONT.** Resource Linker message

An entry was found in the font directory that had no corresponding font resource. The resource file is probably corrupted.

## Error in ICONDIR. Cannot find ICON. Resource Linker message

An entry was found in the icon directory that had no corresponding icon resource. The resource file is probably corrupted.

## Error in RES format.

Resource Linker message

There is invalid information in the binary resource file. The resource file might contain invalid resource information or be corrupt. Try recompiling the resources and restart the resource link.

## Error in RES format. Cannot find NAMEDIR resource.

Resource Linker message

An entry was found in the name-table that had no corresponding resource. Verify that you have included the appropriate resource in your resource script. Name-tables are not used for Windows version 3.1 or greater.

## Error in resource binary length (bad format?).

Resource Linker message

There is an error in the size of a binary resource. The format of the resource might be invalid or the resource has been corrupted.

**Error in STRINGTABLE format.** Resource Linker message

The resource linker (RLINK) should not generate this message.

**Error making absolute file name.** Resource Linker message

The resource linker (RLINK) should not generate this message.

## **Error opening file.** Resource Linker message

An error occurred when the resource linker tried to open a file. This error occurs if the file does not exist, another process has denied access to the file, the path or filename is incorrect, or there are no more available file handles.

## Error positioning file. Resource Linker message

An error occurred trying to seek to a location in a file. This file could be truncated or corrupted. Try verifying the disk integrity using CHKDSK or recompile the resource files and restart the resource link.

**Error positioning file.** Resource Linker message

The resource linker (RLINK) should not generate this message.

## **Error reading file.** Resource Linker message

An error occurred when the resource linker tried to read a file. This error typically occurs when there is a disk error while the file is being read.

## Error renaming file.

Resource Linker message

An error occurred when the resource linker tried to rename a file. This error occurs if the file is marked as read-only or does not exist or a file already exists having the name that the resource linker is trying to use.

**Solutions**

- If the disk is read-only, change its attributes so that it can be deleted.
- If a file having the name already exists you can either delete that file or choose another name.

## Error sizing file. Resource Linker message

Also: **Error getting size of file.**

A disk error occurred when trying to determine the file size.

## Error writing file. Resource Linker message

An error occurred when the resource linker tried to write to a file. This error occurs if the work disk is full or write-protected.

**Solutions**

- If the disk is full, try deleting unneeded files and restarting the resource link.
- If the disk is write-protected, direct the output to a writeable disk and restart the resource link.

## Error. EXE alignment too small for packing resources tooResource Linker message

You have pre-packing turned on and the resources will not fit with the current image alignment. Try increasing the alignment, re-link, and restart the resource link.

## Error. Missing NAME resource. RES not for Windows 3. Resource Linker message

The binary resource file is missing the NAME resource. The resource file is probably not a Windows 3 resource file.

## FONTDIR resource too big to link.

Resource Linker message

Also **FONTDIR too large to handle.**

The directory of fonts table size has been exceeded. Try splitting your fonts into multiple FON files.

## NAMEDIR resource too big to link. Resource Linker message

The name-table maximum size has been exceeded. Name-table entries are not used in Windows 3.1 or later.

## Not a Windows format EXE file.

Resource Linker message

The executable file you tried to bind resources to is not a valid Windows or Win32 executable file.

**Program execution arrived at yet-to-be-written code.** Resource Linker message

The resource linker (RLINK) should not generate this message.

**Resultant EXE size too big.** Resource Linker message

The resource linker (RLINK) should not generate this message.

## Too many files to open. Resource Linker message

You have exceeded the resource linkers limit on files. Try combining some of your individual resource files into a single resource.

## **Too many resources to handle.** Resource Linker message

You have too many resources for the resource linker to handle. Try reducing the total number of resources you are trying to link.

## Too many STRINGTABLEs to link. Resource Linker message

The resource linker (RLINK) should not generate this message.

You have too many string tables for the resource linker to handle. Try reducing the total number of string tables in your resource.

## **Unsupported EXE RC version.** Resource Linker message

The executable you are attempting to link the resources to already has resources attached. The version number of these resources is not recognized by this resource linker. Try removing the resources or relinking.

## 32-bit format in resource file. Please recompile. Resource Linker message

The compiled resource (.RES) file you are trying to use with your application contains 32-bit resources, but the target type of your application is for 16-bit Windows. Recompile the resource file for Windows 3.1 or change the target type for your application to Win32.

**Reporting error.** Resource Linker message

The resource linker encountered a problem while trying to report an error.

## Error. Expecting RES file, not EXE. File: <filename>

Resource Linker message

The resource linker was.expecting a compiled resource (.RES) file, but found an executable (.EXE) instead. Verify that you have the correct node and file types specified.

## Windows version is set to Win32, but target type is Win16

Resource Linker message

The version for your resources is set to Win32, but the target type for your project is for a 16-bit Windows application. 16-bit Windows applications cannot use 32-bit resources.

Either change the version for your resources to Windows 3.1 or change the target type for your application to Win32.

## **Warning. Duplicate resources.** Resource Linker message

If multiple resource files are linked in to the image, the user could have duplicate resources in the resource files. For example, you might have the same ICON in two different RES files. The resource linker flags this and the second resource is removed.

**Note:** The resources must have the same type and identifier to be declared duplicates.

**No resources.** Resource Linker message

This warning message occurs if the resource linker is given a resource file that contains not resources.

## Error seeking point in file.

Resource Linker message

An error occurred trying to seek to a location in a file. This file could be truncated or corrupted. Try compiling the resource files and restart the resource link.

## Resource format not recognized in file.

Resource Linker message

The format of a .RES file that you are attempting to use contains a resource with an unknown format. This is normally due to a corrupt resource file. Make sure that you are binding a legitimate resource file, and rebuild the .RES file, if necessary.

### 'filename' does not exist: don't know how to make it

MAKE message

The build sequence includes a nonexistent file name, and no rule exists that would allow the file name to be built.

## Unable to execute command: 'command'  MAKE message

A command failed to execute. This might be because the command file could not be found or was misspelled, there was no disk space left in the specified swap directory, the swap directory does not exist, or (less likely) the command itself exists but has been corrupted.

### Incorrect command line argument: 'argument'

MAKE message

You've used incorrect command-line arguments. Reenter the command and arguments.

## **Unable to open makefile** MAKE message

The current directory does not contain a file named MAKEFILE or MAKEFILE.MAK, or it does not contain the file you specified with –f.

## Not enough memory MAKE message

All of your working storage has been exhausted.

## Error directive: 'message'

MAKE message

MAKE has processed an #error directive in the source file, and the text of the directive is displayed in the message.

## Unable to redirect input or output MAKE message

MAKE was unable to open the temporary files necessary to redirect input or output. If you are on a network, make sure you have access rights to the current directory.

## No terminator specified for in-line file operator MAKE message

The makefile contains either the && or << command-line operators to start an inline file, but the file is not terminated.

## Circular dependency exists in makefile

MAKE message

The makefile indicates that a file needs to be up-to-date before it can be built. Take, for example, the explicit rules

```
filea:  fileb
fileb:  filec
filec:  filea
```

This implies that filea depends on fileb, which depends on filec, and filec depends on filea. This is illegal because a file cannot depend on itself, indirectly or directly.

## Macro substitute text 'string' is too long MAKE message

The macro substitution text string overflowed MAKE's internal buffer of 512 bytes.

## Macro replace text 'string' is too long MAKE message

The macro replacement text string overflowed MAKE's internal buffer of 512 bytes.

## 'macroname' - ')' missing in macro invocation

MAKE message

The macro you entered is missing a right parenthesis.

## Cycle in include files: 'filename'

MAKE message

This error message is issued if a makefile includes itself in the make script.

## FATAL ERROR: GP FAULT MAKE message

Your program caused a general protection fault and exited fatally. This type of error is caused by various reasons such as attempting to access or write to out of bound memory. For best results, use CodeGuard to locate the error.

## Cannot write a string option MAKE message

The –W MAKE option writes a character option to MAKE.EXE. If there's any string option, this error message is generated. For example, the following string option generates this message:

```
-Dxxxx="My_foo" or -Uxxxxx
```

## Cannot find MAKE.EXE MAKE message

The MAKE command-line tool cannot be found. Be sure that MAKE.EXE is in either the current directory or in a directory contained in your directory path.

### Unable to open file 'filename' MAKE message

This error occurs if the named file does not exist or is misspelled.

### 'filename' not a MAKE  MAKE message

The file you specified with the –f option is not a makefile.

## Write error on file 'filename'

MAKE message

MAKE couldn't open or write to the file specified in the makefile. Check to ensure that there's enough space left on your disk, and that you have write access to the disk.

## **Command arguments too long** MAKE message

The arguments to a command exceeded the 511-character limit imposed by DOS.

## Unexpected end of file in conditional started at line 'line number'

MAKE message

The source file ended before MAKE encountered an !endif. The !endif was either missing or misspelled.

## Unknown preprocessor statement MAKE message

A ! character was encountered at the beginning of a line, and the statement name following it was not error, undef, if, elif, include, else, or endif.

## Bad filename format in include statement MAKE message

Include file names must be surrounded by quotes or angle brackets. The file name was missing the opening quote or angle bracket.

## **Filename too long** MAKE message

The path name in an !include directive overflowed MAKE's internal buffer (512 bytes).

## No filename ending MAKE message

The file name in an !include statement is missing the correct closing quote or angle bracket.

## **Unable to open include file 'filename'** MAKE message

The compiler could not find the named file. This error can also be caused if an !include file included itself, or if you do not have FILES set in CONFIG.SYS on your root directory (try `FILES=20`). Check whether the named file exists.

## Macro expansion too long MAKE message

A macro cannot expand to more than 4,096 characters. This error often occurs if a macro recursively expands itself. A macro cannot legally expand to itself.

## If statement too long MAKE message

An If statement has exceeded 4,096 characters.

## **Rule line too long** MAKE message

An implicit or explicit rule was longer than 4,096 characters.

## Bad undef statement syntax MAKE message

An !undef statement must contain a single identifier and nothing else as the body of the statement.

## Misplaced else statement MAKE message

An !else directive is missing a matching !if directive.

## Command syntax error MAKE message

This message occurs if

- The first rule line of the makefile contains any leading whitespace.
- An implicit rule does not consist of .ext.ext:.
- An explicit rule does not contain a name before the : character.
- A macro definition does not contain a name before the = character.

## **Misplaced elif statement** MAKE message

An !elif directive is missing a matching !if directive.

## Misplaced endif statement MAKE message

An !endif directive is missing a matching !if directive.

## Illegal character in constant expression 'expression'

MAKE message

MAKE encountered a character not allowed in a constant expression. If the character is a letter, this probably indicates a misspelled identifier.

## Expression syntax error in !if statement MAKE message

The expression in an !if statement is badly formed—it contains a mismatched parenthesis, an extra or missing operator, or a missing or extra constant.

## Illegal octal digit MAKE message

An octal constant containing a digit of 8 or 9 was found.

## Character constant too long MAKE message

A char constant in an expression is too long.

## Division by zero MAKE message

A division or remainder operator in an !if statement has a zero divisor.

## Redefinition of target 'filename'

MAKE message

The named file occurs on the left side of more than one explicit rule.

## Bad macro output translator MAKE message

Invalid syntax for substitution within macros.

## Too many suffixes in .SUFFIXES list MAKE message

The suffixes list can include up to 255 suffixes.

## Use of : and :: dependents for target 'target'

MAKE message

You tried to use the target in both single and multiple description blocks (using both the : and :: operators). Examples:

```
filea: fileb
filea:: filec
```

## Ifdef statement too long MAKE message

An Ifdef statement has exceeded 4,096 characters.

## Ifndef statement too long

MAKE message

An Ifndef statement has exceeded 4,096 characters.

## No match found for wildcard 'expression'  MAKE message

No files match the wildcard expression that you want MAKE to expand. For example, the following causes MAKE to send this error message if there are no files with the extension .OBJ in the current directory:

```
prog.exe: *.obj
```

## No closing quote MAKE message

A string expression is missing a closing quote in an !if or !elif expression.

## String type not allowed with this operand MAKE message

You tried to use an operand that is not allowed for comparing string types. Valid operands are ==, !=, <, >, <=, and >=.

### Int and string types compared MAKE message

You tried to compare an integer operand with a string operand in an !if or !elif expression.

## Colon expected MAKE message

Your implicit rule is missing a colon at the end.

```
.c.obj:      # Correct
.c.obj       # Incorrect
```

## Only <<KEEP or <<NOKEEP MAKE message

You specified something besides KEEP or NOKEEP when closing a temporary inline file.

## Unexpected end of file MAKE message

The end of the makefile was reached before closing a temporary inline file.

## Cannot have path list for target

MAKE message

You can only specify a path list for dependents of an explicit rule. For example, an invalid and a valid path list are shown here:

```
{path1;path2}prog.exe: prog.obj      # Invalid
prog.exe: {path1;path2}prog.obj      # Valid
```

## Cannot have multiple paths for implicit rule MAKE message

You can have only one path for each of the extensions in an implicit rule; for example, `{path}.c.obj`. Multiple path lists are allowed only for dependents in an explicit rule.

## No macro before =

MAKE message

You must name a macro before you can assign it a value.

## Too many rules for target 'target'  MAKE message

MAKE can't determine which rules to follow when building a target because you've created too many rules for the target. For example, the following makefile generates this error message:

```
abc.exe : a.obj
  bcc -c a.c

abc.exe : b.obj

abc.exe : c.obj
  bcc -c b.c c.c
```

## Illegal/invalid option in CMDSWITCHES directive 'option'

MAKE message

The !CMDSWITCHES preprocessing directive turns on or off one or more command-line options. Specify an operator, either a plus sign (+) to turn options on, or a minus sign (-) to turn options off, followed by one or more letters specifying options. An invalid or illegal option is specified in the !CMDSWITCHES directive.

## Unknown CMDSWITCHES operator 'operator'  MAKE message

The !CMDSWITCHES directive is supposed to turn on command-line options using a plus sign (+) followed by the option and turn them off using a minus sign (-) followed by the option. An operator other than + or - is specified in the !CMDSWITCHES directive.

## MAKE errors

Following are the error messages that can occur while using MAKE to compile a program:

Bad filename format in include statement
Bad macro output translator
Bad undef statement syntax
Cannot find MAKE.EXE
Cannot have multiple paths for implicit rule
Cannot have path list for target
Cannot write a string option
Character constant too long
Circular dependency exists in makefile
Colon expected
Command arguments too long
Command syntax error
Cycle in include files: 'filename'
Division by zero
Error directive: 'message'
Expression syntax error in !if statement
FATAL ERROR: GP FAULT
filename' does not existdon't know how to make it
'filename' not a MAKE
Filename too long
If statement too long
Ifdef statement too long
Ifndef statement too long
Illegal character in constant expression 'expression'
Illegal octal digit
Illegal/invalid option in CMDSWITCHES directive 'option'
Incorrect command line argument:
Int and string types compared
Macro expansion too long
Macro replace text 'string' is too long
Macro substitute text 'string' is too long
'macroname'—')' missing in macro invocation
Misplaced elif statement
Misplaced else statement
Misplaced endif statement
No closing quote
No filename ending
No macro before =
No match found for wildcard 'expression'
No terminator specified for in-line file operator
Not enough memory
Only <<KEEP or <<NOKEEP
Redefinition of target 'filename'
Rule line too long
String type not allowed with this operand
Too many rules for target 'target'
Too many suffixes in .SUFFIXES list

Unable to execute command: 'command'
Unable to open file 'filename'
Unable to open include file 'filename'
Unable to open makefile
Unable to redirect input or output
Unexpected end of file in conditional started at line 'line number'
Unexpected end of file
Unknown CMDSWITCHES operator 'operator'
Unknown preprocessor statement
Use of : and :: dependents for target 'target'
Write error on file 'filename'

## Resource Workshop error messages and warnings

Click here for an alphabetical listing of Resource Workshop errors: Errors

## Memory lock failed

Resource Workshop could not lock memory. Exit Resource Workshop immediately, without saving files. Start Windows again.

## Could not allocate memory

Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files. To free memory for use by Resource Workshop, exit other applications.

## Memory unlock failed

Resource Workshop could not unlock global memory. Exit Resource Workshop immediately, without saving files. Start Windows again.

## File creation failed

Resource Workshop could not create a file.

Verify that the specified file does not already exist and that there is sufficient directory or disk space for the file. Retry the operation that caused the error.

## Could not open file

Resource Workshop could not open the specified file. This error can happen when you create resources, then attempt to save them to an executable file (an .EXE or .DLL) when the executable file does not exist to which to bind the resources.

To correct the problem, verify that the file exists, then retry the operation that caused the error.

## File seek failed

Resource Workshop failed in seeking to a location in a file.

The file may be corrupted. Retry the operation that caused the error. Try running CHKDSK on the disk.

## File read failed

Resource Workshop could not read the specified file.

Verify that the file exists and is readable. Retry the operation that caused the error.

## File write failed

Resource Workshop could not write to the specified file.

Verify that the file exists and can be written to. Check that there is enough free disk space then retry the operation that caused the error.

## Virtual table allocation failed

Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files.

To free up more memory for use by Resource Workshop, try exiting other applications, or running Windows in Enhanced mode.

## Virtual table put failed

Resource Workshop's internal database is probably corrupt. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Virtual table read failed

Resource Workshop could not read the specified file.

Verify that the file exists and is readable. Retry the operation that caused the error.

## Virtual table get failed

Resource Workshop's internal database is probably corrupt. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Virtual table create failed

Resource Workshop's internal database is probably corrupt. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Virtual table write failed

Resource Workshop could not write to the specified file.

Verify that the file exists and can be written to. Retry the operation that caused the error.

## Virtual table lock failed

Resource Workshop's internal database is probably corrupt. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Virtual buffer allocation failed

Resource Workshop could not obtain memory for an operation.

Exit Resource Workshop immediately, without saving files. To free memory for Resource Workshop, exit other applications.

## Virtual buffer lock failed

Resource Workshop's internal database is probably corrupt. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Binary too large

A binary data item (resource or field) that is too large for Resource Workshop could not be compiled.

## Unexpected NULL pointer encountered

Resource Workshop encountered an unexpected NULL pointer.

## Input source stack overflow

Resource Workshop could not open an include or rcinclude file or expand a **#define**. Too many files are open or too many **#defines** are nested.

## Cannot find resource

Resource Workshop could not find the selected resource. Exit Resource Workshop immediately, without saving files.

## Unexpected file format

This error can occur when Resource Workshop:

- decompiles a binary resource. In this case, the error means that Resource Workshop could not match the binary data with the resource type definition. Resource Workshop skips the resource.
- saves a program or dynamic link library. The error means that the file is non-standard. This error most often occurs when you try to save Microsoft applications, such as Word for Windows, that use a non-standard executable file format.

## Unreleased version format

The version of the file you are opening is greater than Resource Workshop supports.

## Software error!

Resource Workshop encountered unexpected data. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Not implemented

The selected function is not implemented in this release of Resource Workshop.

## You cannot use this identifier. It is a keyword, resource type name or resource name

You are trying to create an identifier whose name conflicts with a keyword, resource type name, or resource name. Choose a unique name.

## Bad character in source input

The specified source file contains an unrecognizable character.

## #define text too long

The definition for the specified **#define** is too long for Resource Workshop to store. A **#define** definition must be less than 2000 characters.

## Invalid preprocessor directive

Resource Workshop has encountered a # (pound sign) character that is not followed by a valid preprocessor directive name.

## Symbol already defined. Redefinition is not the same

Resource Workshop encountered a **#define** whose name is a keyword, or whose definition is not the same as a previous definition.

Although duplicate definitions of the same **#define** are ignored, two different definitions for the same **#define** are not allowed.

## Expecting #define identifier

Resource Workshop encountered a **#define** followed by an illegal name. **#define** names must begin with a letter and contain only letters, digits, and underscores.

## '#else' before '#if'

Resource Workshop encountered an **#else** or an **#elif** directive without a corresponding **#if** directive. Use a text editor to correct this syntax error.

## '#endif' before '#if'

Resource Workshop encountered an **#endif** directive without a corresponding **#if** directive. Use a text editor to correct this syntax error.

## Unexpected end of file

An end of file was encountered when processing a compiler directive (**#if**, **#ifdef**, etc.).

## Expecting resource name or resource type name

Resource Workshop encountered an undefined identifier or integer expression, which it classified as a resource name or ID. It expects the next token to be a resource type name or ID, but encountered something else.

## Expecting ')'

A numeric expression contains unbalanced parentheses.

## Expecting identifier

Resource Workshop encountered an illegal token in an **#ifdef** or **#if** preprocessor directive.

## Expecting constant expression

Resource Workshop could not evaluate an integer expression.

## Expecting filename

A quoted or unquoted filename is expected.

## Expecting filename in quotes

An **#include** statement was not followed by a filename surrounded by quotes or angle brackets.

## Expecting a number or '('

Resource Workshop encountered an unexpected token when attempting to parse an integer expression. This error is frequently caused by an error in an identifier name.

## Expecting BEGIN

Resource Workshop encountered an unexpected token when searching for the BEGIN keyword. This error is frequently caused by a typo in an identifier name.

## Expecting END

Resource Workshop encountered an unexpected token when searching for the END keyword. This error is frequently caused by a typo in an identifier name.

## Not a positive short integer

Resource and resource type IDs must be positive short integers.

## HEXSTRING over 255 bytes long

A hexstring data item was over 255 bytes long. This syntax extension in Resource Workshop is not supported by the Microsoft Resource Compiler.

## Invalid value in HEXSTRING

A hexstring data type (a Resource Workshop syntax extension not supported by the Microsoft Resource Compiler) is a series of hex digits and white space surrounded by single quotes.

Resource Workshop encountered a character within the single quotes that cannot be interpreted as a hex digit.

## Field too large

Resource Workshop encountered a data field larger than 32K.

## PASCAL string over 255 bytes

A Pascal format string must be less than 256 bytes in length.

## Cannot open file:<filename>

Resource Workshop could not open the specified file. You may have insufficient rights to the file.

## Conflicting memory options

A resource definition with conflicting memory options has been encountered.

## String ID is already used

ID values in stringtable resources must be unique within a single project.

**Note:** If you're in Win32 mode, duplicate strings are allowed in stringtables that use different languages.

## Resource of that name/ID, and type already exists

Resource names or IDs must be unique within type.

## Incomplete expression

Resource Workshop could not completely evaluate an expression. There may be a typo or missing information.

## A MENU or POPUP must have at least one item

A MENU and POPUP definition must contain at least one menu item or menu separator. Either add more menu items under the popup or change the popup to a menu item.

## An ACCELERATORS table must have at least one item

You cannot define an empty accelerator table.

## Invalid icon format

Resource Workshop could not compile an icon resource due to an invalid file format.

## Invalid cursor format

Resource Workshop could not compile a cursor resource due to an invalid format.

## Invalid bitmap format

Resource Workshop could not compile a bitmap resource due to an invalid format.

## Expecting filename or BEGIN

Resource Workshop could not compile a file or resource. It encountered a token that was not a file name, curly brace, or a BEGIN keyword.

## #undef is not supported

Resource Workshop does not support **#undef**. If the **#undef** define is required in this .H file in order to satisfy C requirements, surround the statement with the **#ifndef** directive.

## L string prefix is not allowed

Resource Workshop only supports L-quoted strings inside an RCDATA statement. Make sure the L-quoted string is inside an RCDATA statement.

## Expecting signed short integer

Resource Workshop's incremental compiler expects to see a signed short (16-bit) integer or integer expression in this field. This error is usually caused by an undefined identifier.

## Expecting unsigned short integer

Resource Workshop's incremental compiler expects to see an unsigned short (16-bit) integer or integer expression in this field. This error is usually caused by an undefined identifier.

## Expecting signed long integer

Resource Workshop's incremental compiler expects to see a signed long (32-bit) integer or integer expression in this field. This error is usually caused by an undefined identifier.

## Expecting unsigned long integer

Resource Workshop's incremental compiler expects to see an unsigned long (32-bit) integer or integer expression in this field. This error is usually caused by an undefined identifier.

## Expecting quoted string

Resource Workshop's incremental compiler expects to see a quoted string in this field.

## Expecting resource ID or name

Resource Workshop expects a positive short integer or an unquoted alphanumeric literal for a resource name or ID.

## Expecting resource type

Resource Workshop expects a positive short integer or an unquoted alphanumeric literal for a resource type name or ID.

## Expecting class name or ID

A dialog control requires a quoted name or unsigned character with a value > 0x7F. The unsigned character syntax is reserved for use by standard windows controls.

## Expecting HEXSTRING (hex digits surrounded by single quotes)

The CTLDATA keyword for dialog controls must be followed by a HEXSTRING data field.

## Invalid menu option

The valid options for POPUP or MENUITEM statements are:

- CHECKED
- MENUBREAK
- MENUBARBREAK
- INACTIVE
- GRAYED
- HELP
- SEPARATOR (valid for MENUITEMs only)

## Invalid accelerator option

The valid accelerator options are: ASCII, VIRTKEY, SHIFT, ALT, CONTROL, and NOINVERT.

## Invalid accelerator key value

Accelerator key values must contain:

- unsigned characters
- a quoted string that includes an unsigned character and a ^ to indicate control keys

## Expecting caption: quoted string or unsigned integer

A dialog control caption must be either a quoted string or an unsigned short (16-bit) integer.

## String too long

Strings in string tables can be no longer than 255 bytes.

## Expecting control window style

The control specification on this line is not an unsigned long (32-bit) integer or expression. This error is usually caused by an undefined style identifier.

## Expecting menu text (quoted string) or SEPARATOR

The MENUITEM keyword must be followed either by a quoted string (the item text) or the keyword SEPARATOR.

## Compile initialization failed

The compiler could not complete initialization. Exit and restart both Windows and Resource Workshop.

## Input reset failed

The compiler could not complete initialization. Exit and restart both Windows and Resource Workshop.

## Source input stack overflow (too many nested includes?)

The sum of your nested includes or **#define** exceeds 63.

## Parser stack overflow

The nesting level of a recursive resource definition exceeds a Resource Workshop limitation. For example, this error can occur if the number of nested pop-ups exceeds 63.

## Expression stack overflow

An integer expression is too complex for Resource Workshop to evaluate. The maximum nesting depth is 32. Try simplifying the expression by removing parentheses.

## File IO error

A read error occurred on a file. Check to see if the file is corrupt.

## New symbol failed

Resource Workshop could not create a **#define.** This error is usually caused by a lack of memory.

To find out how much system memory is available, chose Start|Settings|Control Panel and choose System. Then choose the Performance tab.

## New field instance failed

In the process of creating a field record, Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files.

To free up more memory for use by Resource Workshop, exit other applications.

## Allocate failed

Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files.

To free up more memory for use by Resource Workshop, exit other applications.

## Memory lock failed

Resource Workshop could not lock memory. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## VTMgr allocation error

Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files.

To free up more memory for use by Resource Workshop, exit other applications.

## VTMgr lock error

Resource Workshop could not lock memory. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## VBuff allocation error

Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files.

To free up more memory for use by Resource Workshop, exit other applications.

## VBuff lock error

Resource Workshop could not lock memory. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Internal software error

Resource Workshop encountered unexpected data. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Device dependent bitmap does not match current display. Cannot convert

You tried to open a Windows 2.0 resource file containing a device-dependent bitmap in a format that does not match the current display device. Use another tool to convert the bitmap.

## Preprocessor directives not allowed

Preprocessor directives are not allowed when editing a resource as text. To add preprocessor directives, edit the file that contains the resource as a text file.

## Too much data for 1 field

You entered more data in this field than the incremental compiler allows. This error may also be caused by an invalid character that was parsed as an extra token. Delete the extra data.

## Cannot convert Windows 2 image: file is read-only

You opened a resource file containing a Windows 2.0 format bitmap resource. However, the file is read-only. Change the file's attributes to examine the file in Resource Workshop.

## Too many controls

A dialog resource contains more than 255 controls. Only 255 controls are allowed per dialog template.

## Invalid escape sequence

A backslash character in a string was not followed by a valid escape code.

## Too many digits in a number

The number contains too many digits to be represented by an unsigned 32-bit integer.

## Expecting unit keyword

Resource Workshop encountered an unrecognized token when it was expecting a unit keyword. Make sure that the include or unit file is syntactically correct by compiling it.

## Expecting semicolon

Resource Workshop encountered an unrecognized token when it was expecting a semicolon. Make sure that the include or unit file is syntactically correct by compiling it.

## Expecting interface keyword

Resource Workshop encountered an unrecognized token when it was expecting an interface keyword. Make sure that the include or unit file is syntactically correct by compiling it.

## Must be first token on a line

A preprocessor directive or constant name must be the first token on a source line. Use an external text editor to correct the source.

## Pascal syntax error (unrecognized token)

Resource Workshop encountered an unrecognized token. Make sure that the include or unit file is syntactically correct by compiling it.

## Unexpected const keyword

Resource Workshop encountered an unexpected const keyword. Make sure that the include or unit file is syntactically correct by compiling it.

## Unexpected operator

Resource Workshop encountered an unexpected operator. Make sure that the include or unit file is syntactically correct by compiling it.

## #error directive encountered: <error message>

The Resource Workshop compiler encountered an **#error** directive. The text of the user-defined message is displayed.

## Not a valid identifier name

The name you selected for a **#define** or constant is not syntactically correct. Identifier names must start with a letter and contain only letters, digits, and underscores.

## Resource binary too large

The binary resource file you're trying to read into Resource Workshop is too large.

## Syntax error

A syntax error has occurred. Check the resource script syntax and recompile.

## Token is too large for scanner (unbalanced quotes?)

The scanner tried to create a token for a string or hexstring, but couldn't find the closing single or double quote. Check the resource script for unmatched quotes.

## Unterminated string or hexdecimal constant

A string value or hexadecimal constant in one of your resources is missing either an ending double quotation mark or a continuation (/) character at the end of a line.

## Expecting "]"

The resource compiler found a syntax error in the .RC file.

## Expecting field type

The resource compiler found a syntax error in the .RC file.

## Expecting RESEND

The resource compiler found a syntax error in the .RC file.

## Expecting field name, field type or function keyword

The resource compiler found a syntax error in the .RC file.

## Expecting RESPARENT or RESITEMCOUNT

The resource compiler found a syntax error in the .RC file.

## Field name already used

The resource compiler found a semantics error in the .RC file.

## Object is read only

Resource Workshop could not update a file because it is marked read-only or is in use by another process.

## Only one unnamed item allowed

The resource compiler found a semantics error in the .RC file.

## Optional field keyword already used

The resource compiler found a semantics error in the .RC file.

## Duplicate EDITDATA keyword

The resource compiler found a semantics error in the .RC file.

## Duplicate RESBIN keyword

The resource compiler found a semantics error in the .RC file.

## Duplicate RESITEM keyword

The resource compiler found a semantics error in the .RC file.

## Positional optional fields must precede keyword optional fields

The semantics of the flagged line is not correct. Check for misspellings or a typing error.

## Item keyword already used

The semantics of the flagged line is not correct. Check for misspellings or a typing error.

## Field definitions not allowed

The semantics of the flagged line in the .RC file is not correct. Check for misspellings or a typing error.

## RESBIN definitions not defined

The semantics of the flagged line is not correct. Check for misspellings or a typing error.

## RESBIN definitions not allowed

The semantics of the flagged line is not correct. Check for misspellings or a typing error.

## RESITEM definitions not allowed

The resource compiler found a semantics error in the .RC file.

## '|=' default value not allowed for an array

The resource compiler found a semantics error in the .RC file.

## '|=' default value not supported for this field type

The resource compiler found a semantics error in the .RC file.

## Item names cannot be reserved words

The resource compiler found a syntax error in the .RC file.

## Nested default values not allowed

The resource compiler found a semantics error in the .RC file.

## Expecting a valid file extension (3 letters or less)

The resource compiler found a semantics error in the .RC file.

## Fonts must have numerics resource IDs

Font names must have numeric resource ids. Other resources can use either a number or string resource identifier.

## Invalid font format

The file specified in the FONT resource is not a valid font file, or it has been corrupted.

## Expecting signed character

The resource compiler found a syntax error in the .RC file.

## κExpecting unsigned character

The resource compiler found a syntax error in the .RC file.

## Invalid font specification

One of the fonts in the font file is invalid. Try reinstalling the fonts.

## Fatal error

The error is too severe for the resource compiler to continue processing the resource scripting.

## Identifier is out of scope

An identifier used in a macro has been **#undef**'ed.

## Expecting window rectangle (4 signed long integers)

A dialog control requires 4 unsigned integers to specify its location and size.

The resource compiler did not find enough values in the resource statement. Dialog boxes and the controls in them require the x and y position of the upper left corner followed by the width and height of the resource in integers.

## User break

The Cancel button was pressed during the resource compile.

## Height must be from 1 to 5000

You entered an incorrect height value. Enter a value from 1 to 5000.

## Bitmap must contain less than <some number> of pixels

This bitmap is too large to be edited in Resource Workshop.

## Width must be from 1 to 5000

You entered an incorrect width value. Enter a value from 1 to 5000.

## Unable to load Bitmap editor

Resource Workshop is unable to load the Bitmap editor because the dll is missing, or memory is extremely low.

## Horizontal (x) must be from 1 to 31

When specifying a hotspot for a cursor, the x coordinates must range from 1 to 31. That is, you must choose a point that is on the cursor bitmap.

## Vertical (y) must be from 1 to 31

When specifying a hotspot for a cursor, the y coordinates must range from 1 to 31. That is, you must choose a point that is on the cursor bitmap.

## Unable to load Cursor editor

Resource Workshop is unable to load the Bitmap editor because the dll is missing or memory is extremely low.

## The file <filename> is already in this project. You cannot use the file.

A file with the name you specified for the new resource already exists. It is also already referenced in the current resource project. If you want to change the existing resource file, find its entry in the project and edit it. If you want to create a new resource, change the filename to something unique.

## Icon dimensions must be from 1 to 255

Icons cannot be larger than 255x255. Bitmaps do not have this restriction.

## Unable to load Icon editor

Resource Workshop is unable to load the Bitmap editor because the dll is missing or memory is extremely low.

## This image has too many colors to edit

This image contains more than 256 colors.

## This image has too many colors to edit on this display

This image contains more colors than can be edited on this display. Use the Control Panel to change the number of colors supported. Depending on the display, you may need to change the resolution.

## An image of this type exists. Continue?

A version of the icon or cursor already exists that matches the specified width, height, and number of colors.

## This image has an unknown format

This image is not a standard Windows bitmap format.

### <filename> does not exist. Do you wish to create it?

The specified file does not exist in the given path. Check the path and the spelling of the file name.

## Resource project has changed. Do you wish to save it?

The resource you're working on has changed. If you don't compile the changes, they will be discarded. Choose Cancel to return to Resource Workshop without losing any changes or saving any files.

## Delete resource: <resource name>

The selected resource will be deleted. Click Yes if you're sure you want to delete it. Click No if you don't want to delete it.

## Could not load library

Resource Workshop could not load one of the libraries it needs.

If the error persists, check to see how much memory is available. If memory is low, close another application.

## Could not load one of the Dialog editor libraries

Resource Workshop could not load one of the libraries it needs because it could not find it or the computer is very low on memory.

## <filename> does not exist

The specified program or dynamic link library file does not exist. Resource Workshop will not create program files, but will only add resources to those files.

## A resource of that name already exists

The selected name is already assigned to another resource. Choose another name.

## Identifier already exists

The name you selected for this identifier is already in use. Use a unique name.

Note that C/C++ **#define** statements are case-sensitive, but Pascal constants are not.

## Selected file <filename) already exists. Do you wish to overwrite it?

A file of the type and name specified already exists in this project. Resource names and IDs must be unique by type and in a project. Use a name or ID that is not used for a resource of this type.

**Note:** In Win32 mode, resource names and IDs only need to be unique for a given type and language.

## <filename> exists. Overwrite?

You are saving to an existing file. Choose Yes to replace the contents of that file. Choose Cancel to abort the process and return to Resource Workshop.

## <filename> does not exist. Create?

The specified file does not exist. Click Yes if you want to create it.

## Error creating file <filename>

Resource Workshop could not create the new file. If the filename name is valid, try the operation again. Check to see that there is room on the destination drive, or for drive problems.

If the error persists, check to see how much memory is available. If memory is low, close another application.

## You cannot undo this action! Select cancel to stop

There is no undo for the action you are about to perform. This is your last chance to change your mind.

## Remove contents of <filename> from this project?

If you answer Yes, all resources and/or identifiers in the named file and in any file referenced by the file will be removed from this project.

Because you cannot undo the action, Resource Workshop displays a Warning dialog box when you answer Yes.

The warning message reads:

> You cannot undo this action! Select cancel to stop

## Division by zero is not allowed

You specified a constant expression or a remainder function that contains a divide by zero. This is not allowed - you must change the expression or function.

## Please close a window

Resource Workshop allows 10 editor windows to be open at the same time. To open another window, you need to close a currently open window. To close the current editor window, press Ctrl+F4.

## Resource type already exists

You are trying to create a new resource type that already exists. Use a unique name or ID for the resource type.

Note that user-defined resource type IDs should be greater than 256, since the numbers 1 to 256 are reserved for use by the operating system vendor.

## Invalid addon module (check installation.)

Could not start Resource Workshop. Either the installation is missing files or registry entries or the computer is extremely low on memory.

## Out of memory (increase swap file or RAM size.)

If the error persists, check to see how much memory is available. If memory is low, close another application.

## Invalid range for value, must be between <number> and <number>

The value entered for one of the Environment|Project options is out of range.

## Missing Help file <filename>

The Help file could not be found. It may have been deleted or is in a directory other than \HELP.

## Unexpected end of file in conditional started on line <line number>

An **#ifdef** or **#ifndef** on the specified line number does not have a matching **#endif**.

## User Break

You typed a Ctrl+Break while compiling a program. (This is not an error, just a confirmation.)

## Unknown error (# errornum)

Internal compiler error. Contact Borland Technical Support.

## Link terminated by user

You canceled the link.  (This is not an error, just a confirmation.)

## Fatal Error: Cannot Load Linker: linker

You are trying to load a version of the linker which is not compatible with this version of C++Builder. Try reinstalling C++Builder or contact Borland Technical Support.

## Fatal Error: Linker missing CREATE Entry Point

You are trying to load a version of the linker which is not compatible with this version of C++Builder. Try reinstalling C++Builder or contact Borland Technical Support.

## Fatal Error: Linker missing DESTROY Entry Point

You are trying to load a version of the linker which is not compatible with this version of C++Builder. Try reinstalling C++Builder or contact Borland Technical Support.

## Fatal Error: Linker CREATE Failed

You are trying to load a version of the linker which is not compatible with this version of C++Builder. Try reinstalling C++Builder or contact Borland Technical Support.

## Cannot find tasm program: tasm32.exe

You are trying to build code that includes assembly language directives but C++Builder cannot locate the assembler on your system. Borland's Turbo Assembler (tasm.32) is a separate product, available separately from C++Builder. If you have the Turbo Assembler, make sure it is in your path.

## Unknown fatal error

Internal resource compiler error. C++Builder loaded the compiler, but it was unable to run. The compilation process was abruptly terminated. Call Borland Technical Support.

## Unable to load RW32CORE.DLL

Internal resource compiler error. C++Builder was unable to load the resource compiler DLL (RW32CORE.DLL) because it is not there, there is not enough memory, or the DLL is damaged in some way. Contact Borland Technical Support.

## Cannot locate file I/O hook functions for resource compiler

C++Builder loaded a version of RW32CORE.DLL, but for some reason, it is not the correct version. This could happen if you inadvertantly loaded the Borland C++ 5.0 or 5.01 version of RW32CORE.DLL instead of the one provided with C++Builder.  You may need to reinstall C++Builder.