Programming_Languages

COLLABORATORS

	TITLE : Programming_Languages		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 2, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Programming_Languages

1.1 Programming Languages - A Comparison

```
                This document contains a comparison of some programming languages, ↩
                    including
C/C++, Basic, and Pascal.


                 Speed comparison
                   - Execution speed, size of executable


                 General comparison
                 - Capabilities, hints


                 Test programs
                      - Listings


                 Author
                             - Address
```

1.2 Speed comparison

```
                Tables


                 Test #1 (for loop)

                 Test #2 (CLI window output)

                 Abbreviations
                Notes


                 How the execution times have been measured
```

Notes concerning the execution speeds

Notes concerning the program lengths

1.3 Test #1 (for loop)

A loop counting from 1 to 1,000,000 results in the following execution times and program lengths:

Language	Bytes	Ticks
Assembler A68k	48	5.00
ACE 2.0 - ASM	4,568	6.00
BB2 1.9 - ASM	6,228	6.50
StromCPP 1.1 reg.	480	7.00
MaxonC++ 3.1 reg.	708	7.00
Struct 1.0	224	8.00
BCF 77	16,748	8.00
TurboDEX	132	8.00
Dice 2.0 reg.	2,744	8.00
GCC 2.7.0	2,332	9.00
PCQ 1.2d (Phx)	2,308	12.00
Oberon-A	980	14.00
Cyclone 0.94 reg./inl.	496	14.00
HighSpeed Pascal 1.2	1,776	14.00
MaxonPASCAL 3.0	4,588	14.00
StormCPP 1.1	496	15.00
GFA Basic	9,900	16.00
PDC	6,288	16.00
Struct 1.0	224	17.00
MaxonC++ 3.1	752	17.00
StormCPP 1.0 Demo	976	18.00
SAS C 6.56	1,404	18.00
E 2.1b	500	18.50
Cyclone 0.92	500	19.00
M2Amiga 4.1 Demo	1,512	19.00
Turbo Modula-2	2,764	20.00
E 3.2a	524	21.00
BB2 2.1	328	28.00
Oberon_2	728	28.50
ACE 2.37 (Phx)	19,604	34.00
ACE 2.0	42,568	41.00
Maxon/HisoftBasic 3.0	15,768	53.00
Cursor 1.7	32,468	283.00
EXECREXX	884	829.00

1.4 Test #2 (CLI window output)

Writing "Hello, world" 1,000 times in a CLI window results in the following execution times and program lengths:

Language	Bytes	Ticks

StormCPP 2.0 Demo	6,512	786
MaxonC++ 3.1	856	830
ACE 2.40 (Phx)	28,760	831
BB2 (optimized version)	2,132	834
HighSpeed Pascal 1.2	4,116	836
SAS C 6.56	38,724	841
GCC 2.7.0	58,444	849
Turbo Modula-2	10,324	850
Cyclone 0.92	1,612	860
MaxonPASCAL 3.0	4,628	864
BB2 2.1	4,084	865
PCQ Pascal 1.2d (Phx)	3,004	893
Maxon/Hisoft Basic 3.0	16,032	873
Oberon-A	4,360	925

1.5 Abbreviations

The following abbreviations have been used:

Abbreviation	Explanation
reg.	register variables have been used
inl.	inline functions have been used
Phx	Phx assembler and linker have been used
ASM	internal assembler has been used
BB2	Blitz Basic 2

1.6 How

The program »exe_timer« has been used to calculate the execution time of the test programs in intuition ticks (1/50 second). »exe_timer« was written using HiSoft Basic 2 (Maxon Basic 3).

The results stated above have been measured on an Amiga 4000/040 25 MHz with 16 MB fast and 2 MB graphics RAM.

All programs have been copied to RAM: before testing them. 2 ticks have been subtracted from the results computed by »exe_timer« because about 2 ticks are necessary to load a program using »exe_timer«.

1.7 Notes concerning the execution speeds

DEX uses register variables automagically.

Struct has been tested twice: using register variables and without using register variables.

SAS C eliminates empty loops - this results in 1,636 bytes and only 1 tick execution time for test #1. SAS C is not faster if register variables

```
are used.

Blitz Basic programs with error checks turned on need about 160 ticks for
test #1.

Maxon/Hisoft Basic programs with error checks turned on need 388 ticks for
test #1.
```

1.8 Notes concerning the program lengths

```
Oberon_2 requires the »garbagecollector.library« (11,088 bytes).

EXECREXX requires the »rexxapp.library« (1,568 bytes) and  ARexx.

Cursor executables can load the »basic.library« (30 KB) alternatively.

Maxon/Hisoft Basic executables can load the »hbasic2.library« (50 KB)
alternatively, resulting in an executable of 592 bytes for test #1.

Oberon-A: Startup and System/IO/Errors need about 9 KB.

StormCPP executables are significantly shorter if the C++ mode is turned
on.

Cyclone executables are shorter and slower if inlines are turned off.

GCC executables can load the »ixemul.library« (150 KB) alternatively.
```

1.9 General comparison

```
                This section shows the capabilities of some programming languages ↩
                   and gives
you some hints.


                 Assembler
                         - A68k, Blink, Phx


                 Basic
                             - ACE, Maxon/Hisoft Basic, GFA Basic, Blitz Basic 2


                 C/C++
                             - SAS C, GNU C++, Maxon C++, Storm CPP


                 Pascal
                            - PCQ, Maxon Pascal, Hisoft Pascal


                 Modula and Oberon
```

- Turbo Modula, Cyclone, Oberon-A, M2Amiga

Other languages
- E, BCF 77 Fortran, Struct, CanDo, Helm

Conclusion
- How to select your programming language

1.10 assembler

For the speed comparisons, the A68k assembler has been used. It is placed in the public domain. The blink linker has been used to link the object code generated by A68k.

The Phx assembler does a whole bunch of optimizations to ensure that the resulting object files are as fast as possible. To shorten the file length of your executables, you should use the Phx linker.

Some programming languages do not produce object code directly. They create an assembly source file and use an assembler to generate an object file. The object file is then linked to a startup code and a run-time library. In such cases you should try to use the Phx assembler and linker mentioned above.

A couple of programming languages offer an integrated assembler. This is usefull if you write a program in for example C and want to include some assembler instructions for a time critical procedure. This way, there is no need for a seperate assembler package any longer.

1.11 basic

The Cursor Basic compiler generates executables directly. It is able to compile allmost all Amiga Basic source codes. Unfortunately, the resulting executables are very slow. Screens created by Cursor executables may have up to 8 bitplanes.

ACE (= Amiga Basic Compiler with Extras) is placed in the public domain. ACE uses A68k and the blink linker to generate executables. The intermediate assembler source code it generates is fairly understandable. You should use the Phx assembler and linker to get shorter and faster executables (in this case, you must not use the '$' char in SUB names; use SUB STRING trim(a$) instead of SUB trim$(a$), for instance).

ACE supports allmost all Amiga Basic statements and has the following features:

- integrated assembler (via A68k or PhxAss)
- very good support of the serial device
- gadgets, requesters, speech, gadtools menus, turtle graphics, sound, IFF
- interprocess communication, error handling

· include files, external submodules
· random access files can be handled very easily
· programming environment AIDE
· graphical user interface (GUI) creator

Maxon/Hisoft Basic is fully compatible with Amiga Basic and partially with Microsoft Quick Basic for the PC. It comes with include files for Amiga OS 3.1 and supports AGA screens. The programming environment is very good (like Hisoft Pascal). The documentation supplied with this Basic compiler is very good, too.

GFA Basic generates very fast and short executables. Unfortunately, it does not support Amiga OS 2.0+ directly. ECS/AA screen modes are not possible. The GFA-Basic interpreter cannot be run wit Amiga OS 2.0+ unless you switch ECS/AA screen modes and processor caches off. GFA Basic executables are known to throw a lot of Enforcer hits.

Blitz Basic supports all library functions of the Amiga OS directly; you do not need to declare any library function. Blitz Basic is usefull if you want your programs to have a nice user interface. The support for graphics and game creation is excellent. Blitz Basic executables can get very long if you use a lot of graphics statements, windows, or requesters. The resulting executables are quite slow, especially if you need to access files. The AGA chipset is supported. A GUI creator is included.

Test #2 shows that ACE executables are the fastest ones in writing text to a CLI window. The Hisoft Basic executable is quite long and slow.

(If you use the library function putstr, the Blitz Basic program needs only 834 ticks and the size of the executable file is only 2,132 bytes.)

I prefer using ACE and Maxon Basic, but I also continue using GFA and Blitz Basic for some programs.

1.12 C/C++

SAS/C is the standard C compiler for the Amiga. You should not use SAS/C to compile C++ source codes because these are converted to C by SAS/C and thus the resulting executables are very long. (The size of the executable for test #2 is about 38 KB.)

The PDC compiler does not support all features of ANSI C. VBCC is better. Both are placed in the public domain.

GNU C/C++ (GCC) is compatible with UNIX C compilers. Using GCC, you can write UNIX and Amiga programs. C++ executables are even longer than those created by SAS/C, but they are also faster. GCC is intended for freeware programmers.

The best C/C++ compilers are Maxon C++ and StormCPP. Maxon C++ executables are slightly more efficient, but StormCPP executables are compatible with the new pOS operating system and the Power PC processor chip. StormCPP has the best programming environment, too.

StormCPP includes a fast ANSI C library and a MUI class library. Maxon C++ includes an Intuition class library for an easy creation of C++ programs.

The separate Storm Wizard package is used to create graphical user interfaces very easily.

Test #2 shows that Maxon C++ creates the shortest executable files, but StormCPP creates the fastest ones. SAS C and GCC executables are very long, but they are quite fast, too.

I prefer using StormCPP.

1.13 pascal

PCQ is a public domain Pascal compiler. It uses A68k and blink to generate executables. The new version supports the Phx assembler and linker and is faster than version 1.2b.

Maxon Pascal and Hisoft Pascal are commercial Pascal compilers. Hisoft Pascal has the better programming environment and does even support nearly all statements of Turbo Pascal 5.0 for MS-DOS. This is very usefull if you are learning Turbo Pascal at school or university and want to write your own programs. Thus you can run Turbo Pascal on your Amiga and do not have to wait 5 minutes until Windows 95 has booted and Turbo Pascal has been loaded...

The graphics support of Hisoft Pascal is excellent because you are able to use the graphics capabilities of Turbo Pascal. If you want to write a program which creates business graphics, you should use Hisoft Pascal.

Test #2 shows that HighSpeed Pascal executables provide the fastest output to CLI windows.

I prefer using Hisoft Pascal.

1.14 Modula and Oberon

M2Amiga supports all features of the Amiga operating system. The resulting executables are short and fairly fast.

Turbo Modula is a freeware compiler. It uses DICE C to generate the executables, thus being able to call functions of the standard C run-time library.

Cyclone is a new object orientated Modula-2 compiler and is giftware. It supports multi-threaded executables and is able to compile shared libraries. Cyclone features register access, static lists and C++ exceptions, for instance. The new version creates shorter executables than version 0.92.

Oberon_2 is a commercial Oberon-2 compiler. The newest version is Oberon 3.0.

Oberon-A is a free Oberon-2 compiler. It comes with a lot of documentation and sample source codes, including an Oberon operating system. Oberon-A is intended for freeware programmers.

Test #2 shows that the Turbo Modula-2 executable is the fastest one. (Note that this is also the longest file.) The Oberon-A executable is very slow.

I prefer using Cyclone.

1.15 Other languages

E is a language similar to C and Pascal. The resulting executables are fast and short. It is possible to compile shared libraries using E. E has been in the public domain until version 2.1b. E 3.0+ is shareware.

DEC is similar to but not as powerfull as E.

Struct is a very restricted programming language. The resulting executables are very short and fast.

EXECREXX is an ARexx "compiler". The resulting executable still needs the ARexx interpreter and thus is very slow.

BCF 77 is a Fortran compiler. It is shareware. The resulting executables are very long.

The authoring systems CanDo and Helm are usefull for computer based training. CanDo executable files are quite long (about 150 KB as for version 2.5), but they can call the "cando.library" alternatively. The programs generated by Helm and CanDo are quite slow.

1.16 conclusion

Generally, C/C++ executables are faster than Pascal/Modula/Oberon executables, which in turn faster are than Basic executables.

On the other side, C++ and Basic executable files can get very long in some cases.

You should select a programming language which fits your needs concerning the execution speed and the size of the resulting executable file. If you are a beginner, I suggest to use Pascal, Modula-2 or Oberon.

If you need fast executables, you should use the library functions provided by the operating system instead of commands offered by your programming language. (This is very important if you use a Basic compiler like Blitz Basic 2.)

1.17 Basic

Test #1

```
  FOR i& = 1 TO 1000000
  NEXT i&
```

Test #2

```
  FOR t& = 1 TO 1000
    PRINT "Hello, world"
  NEXT t&
```

Optimized version for Blitz Basic 2

```
  a$ = "Hello, world" + CHR$(10) ;add carriage return
  FOR t.l = 1 TO 1000
    PutStr_ a$                   ;call Amiga OS library function
  NEXT t
```

1.18 C/C++

Test #1

```
  #include <stdio.h>

  main()
  {
    register long t;

    for( t = 1; t < 1000000; t++)
    {
    }
  }
```

Test #2

```
  #include <iostream.h>

  main()
  {
    int i;

    for(i = 1; i < 1000; ++i)
    {
    cout << "Hello, world\n";
    }
    return(0);
  }
```

Notes

Listing #1 is a C program, whereas listing #2 is a C++ program.

1.19 Pascal

```
Test #1

  program test;

  var
    t: longint;

  begin
    for t := 1 to 1000000 do
    begin
    end;
  end.


Test #2

  program hello;

  var
    t: integer;

  begin
    for t := 1 to 1000 do writeln('Hello, world');
  end.
```

1.20 Modula / Oberon

```
Test #1

  MODULE test;

  VAR i: LONGINT;

  BEGIN
    FOR i := 1 TO 1000000 DO
    END;
  END test.


Test #2

  MODULE cm;

  FROM InOut IMPORT WriteString, WriteLn;

  VAR
    i: LONGINT;

  BEGIN
    FOR i := 1 TO 1000 DO
      WriteString("Hello World!"); WriteLn;
    END;
  END cm.
```

1.21 Test programs

```
                The test program #1 consists of a loop counting from 1 to ↩
                   1,000,000 using
long integer variables.

Test program #2 writes "Hello, world" 1,000 times and is run in the CLI.

Error checks have been disabled. If necessary, Motorola 68000 object code
generation has been selected.


                 Basic

                 C/C++

                 Pascal

                 Modula and Oberon
```

1.22 author

```
This document may be freely distributed. I hope that it is helpfull for
people looking for a programming language which fits their needs.

Send bug reports and / or suggestions to:
```

```
Frank Reibold
Ottberger Weg 13
D-31737 Rinteln

GERMANY

eMail: Peter.Reibold@T-Online.de
```