

Linked

| |
|----------------------|
| COLLABORATORS |
|----------------------|

| | | |
|------------------|--------------------------|-----------------|
| | <i>TITLE :</i> Linked | |
| <i>ACTION</i> | <i>NAME</i> | <i>DATE</i> |
| WRITTEN BY | | August 24, 2022 |
| <i>SIGNATURE</i> | | |

| |
|-------------------------|
| REVISION HISTORY |
|-------------------------|

| NUMBER | DATE | DESCRIPTION | NAME |
|--------|------|-------------|------|
| | | | |

Contents

| | | |
|----------|--------------------------------|----------|
| 1 | Linked | 1 |
| 1.1 | Linked List V1.10 | 1 |
| 1.2 | addelement | 2 |
| 1.3 | changecurrentelement | 2 |
| 1.4 | clearlist | 3 |
| 1.5 | countlist | 3 |
| 1.6 | firstelement | 3 |
| 1.7 | insertelement | 3 |
| 1.8 | killelement | 4 |
| 1.9 | lastelement | 4 |
| 1.10 | listbase | 4 |
| 1.11 | listindex | 4 |
| 1.12 | nextelement | 4 |
| 1.13 | previouselement | 5 |

Chapter 1

Linked

1.1 Linked List V1.10

PureBasic - Linked List library V1.10

'Linked List' signifie 'Liste chaînée', c'est à dire un ensemble d'éléments similaires qui sont indépendants les uns des autres mais liés entre eux par grâce à des 'chaînes'. Ces listes servent à stocker des données, un peu à la manière des tableaux, mais ont l'énorme avantage d'être dynamiques. De plus on peut rapidement supprimer, insérer ou ajouter un élément à n'importe quelle position de la liste, contrairement aux tableaux.

Commandes disponibles:

AddElement
ChangeCurrentElement
ClearList
CountList
FirstElement
InsertElement
KillElement
LastElement
ListBase
ListIndex
NextElement
PreviousElement

```

AddElement

CountList

FirstElement

KillElement

LastElement

ListBase

ListIndex

NextElement

PreviousElement
Exemple:

```

Linked List demo

1.2 addelement

Syntaxe

```
AddElement(linkedlist())
```

Résumé

Ajoute un élément vide après la position courante dans la liste indiquée. Il devient l'élément courant de la liste.

1.3 changecurrentelement

Syntaxe

```
ChangeCurrentElement(linkedlist(), *NewElement)
```

Résumé

Change l'élément courant de la liste chaînée spécifiée par l'élément donné (*NewElement). Cet élément doit être un pointeur vers un élément de cette liste chaînée qui existe déjà. Cette fonction est très utile pour rappeler un élément spécifique après une manipulation complexe de la liste chaînée.

Exemple:

```

*Old_Element = @mylist()      ; Stocke l'adresse de l'élément courant dans * ←
  Old_Element

ResetList(mylist())          ; Recherche de tous les éléments dont le nom est ←
  'John'

While(NextItem(mylist()))    ;
  If mylist()\name = "John" ; A la fin de la recherche, l'élément courant ←
    est le dernier

```

```
        mylist()\name = "J"    ; élément de la liste.  
    EndIf                    ;  
Wend                          ;  
  
ChangeCurrentElement(mylist(), *Old_Element) ; Remettons notre ancien élément ←  
    courant  
                                                ; comme élément courant.
```

1.4 clearlist

Syntaxe

```
ClearList(linkedlist())
```

Résumé

Efface tous les éléments d'une liste chaînée et libère la mémoire associée. ←

Cette

liste chaînée est toujours utilisable, mais elle est vide, comme après un
NewList().

1.5 countlist

Syntaxe

```
NbElems.l = CountList(linkedlist())
```

Résumé

Retourne le nombre d'éléments que contient la liste indiquée. Cette commande
ne change pas la position de l'élément courant de la liste.

1.6 firstelement

Syntaxe

```
FirstElement(linkedlist())
```

Résumé

Change l'élément courant de liste par le premier élément de cette liste.

1.7 insertelement

Syntaxe

```
InsertElement(linkedlist())
```

Résumé

Insère un élément vide avant l'élément courant de la liste chaînée. Ce nouvel
élément devient l'élément courant.

1.8 killelement

Syntaxe

```
KillElement(linkedlist())
```

Résumé

Efface l'élément courant de la liste et libère la mémoire qu'il occupait. Le nouvel élément courant est l'élément qui suivait celui que l'on vient d'effacer ou NULL si l'élément qu'on a effacé était le dernier de la liste.

1.9 lastelement

Syntaxe

```
LastElement(linkedlist())
```

Résumé

Change l'élément courant de liste par le dernier élément de cette liste.

1.10 listbase

Syntaxe

```
*ListBase = ListBase(linkedlist())
```

Résumé

Retourne l'adresse mémoire de la liste chaînée (Structure 'List' de l'AmigaOS). Très utile pour les programmeurs expérimentés qui veulent manipuler davantage les listes chaînées.

1.11 listindex

Syntaxe

```
Index.l = ListIndex(linkedlist())
```

Résumé

Retourne la position de l'élément courant dans la liste en considérant que la position du premier élément est 1, du deuxième est 2, etc..

1.12 nextelement

Syntaxe

```
Résultat = NextElement(linkedlist())
```

Résumé

Change l'élément courant de la liste par l'élément suivant. Si le résultat est NULL, alors on est au bout de la liste (il n'y a plus d'éléments suivants), ← sinon il représente l'adresse mémoire de l'élément pour les programmeurs expérimentés.

1.13 previousement

Syntaxe

```
Result = PreviousElement(linkedlist())
```

Résumé

Change l'élément courant de la liste par l'élément précédent. Cette fonction ↔
retourne
la valeur NULL si il n'y a pas d'élément précédent (si on est déjà sur le ↔
premier
élément), sinon elle renvoie l'adresse de l'élément précédent.