Linked

COLLABORATORS

	TITLE : Linked	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 23, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Linked

1.1 Linked List V1.00

```
Pure Basic - Linked List library V1.00

  Die verknüpften Listen sind Objekte, welche dynamisch entsprechend
  Ihrem Bedarf reserviert werden. Sie sind eine Liste von Elementen,
  wobei jedes Element vollkommen unabhängig von einem anderen ist.
  Sie können beliebige Elemente hinzufügen, Elemente an der von Ihnen
  gewünschten Stelle einfügen, andere löschen und vieles mehr.
  Diese Art des Datenmanagements wird im AmigaOS sehr oft benutzt,
  da es der beste Weg ist, mit Daten umzugehen, wo deren Anzahl und
  Größe nicht bekannt ist.

Befehlsübersicht in alphabetischer Reihenfolge:

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

Beispiel:

   Linked List Demo
```

1.2 addelement

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

```
  COMMAND
Fügt ein neues Listenelement nach der aktuellen Position ein. Dieses neue
Element wird auch das aktuelle Element der Liste.
```

1.3 changecurrentelement

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

  COMMAND
Ändert das aktuelle Element der angegebenen Liste auf das angegebene
*NewElement. Das *NewElement muß ein Zeiger (Pointer) auf ein anderes
in der Liste existierendes Element sein. Diese Funktion ist sehr nützlich,
um sich ein Element zu merken und es nach der Verarbeitung anderer
Befehle wieder aufzurufen.

Beispiel:

  *Old_Element = @mylist()      ; Ermittelt die Adresse des aktuellen Elements

  ResetList(mylist())           ; Durchführen einer Suchschleife nach allen
  While(NextItem(mylist())      ; Elementen mit Name "John" und Änderung dieser
      If mylist()\name = "John" ; in "J"
          mylist()\name = "J"   ;
      EndIf                     ;
  Wend                          ;

  ChangeCurrentElement(mylist(), *Old_Element) ; Wiederherstellen unseres  ↩
     Elements
                                               ; vor der Suche
```

1.4 clearlist

```
  SYNTAX
ClearList(linkedlist())

  COMMAND
Löscht alle Elemente in dieser Liste und gibt deren Speicherplatz frei.
Nach diesem Aufruf ist die Liste noch benutzbar, es befinden sich aber
keine Elemente mehr in der Liste.
```

1.5 countlist

```
  SYNTAX
CountList(linkedlist())

  COMMAND
Zählt, wieviele Elemente in der verknüpften Liste vorhanden sind. Es
ändert nicht das aktuelle Listenelement.
```

1.6 firstelement

```
  SYNTAX
FirstElement(linkedlist())

  FUNCTION
Ändert das aktuelle Listenelement auf das erste Listenelement.
```

1.7 insertelement

```
  SYNTAX
InsertElement(linkedlist())

  COMMAND
Fügt ein neues Element vor der aktuellen Position ein. Diese neue Element
wird das aktuelle Element der Liste.
```

1.8 killelement

```
  SYNTAX
KillElement(linkedlist())

  FUNCTION
Entfernt das aktuelle Element aus der Liste. Nach dem Aufruf dieser
Funktion wird das - auf das gelöschte Element folgende - Element das
neue aktuelle Element. Haben Sie das letzte Element in der Liste
gelöscht, wird das aktuelle Element auf NULL gesetzt.
```

1.9 lastelement

```
  SYNTAX
LastElement(linkedlist())

  FUNCTION
Ändert das aktuelle Listenelement auf das letzte Listenelement.
```

1.10 listbase

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

  FUNCTION
Gibt die Adresse der ListBase Struktur ('List' im AmigaOS) zurück.
```

1.11 listindex

```
  SYNTAX
Index = ListIndex(linkedlist())

  STATEMENT
Gibt die Position des aktuellen Listenelements zurück, das erste Element
in der Liste befindet sich dabei an Position 1.
```

1.12 nextelement

```
  SYNTAX
Result = NextElement(linkedlist())

  STATEMENT
Ändert das aktuelle Element auf das nächste Listenelement und gibt
dessen Adresse zurück, oder NULL wenn keine weiteren Elemente existieren.
```

1.13 previouselement

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

  STATEMENT
Ändert das aktuelle Element auf das vorhergehende Element und gibt
dessen Adresse zurück, oder NULL wenn das aktuelle Element bereits das
erste Element war.
```