

Einfach verkettete Listen

Datenstruktur definieren	<pre>typedef struct sData { int Index; double Value; struct sData *Next; };</pre>
--------------------------	---

C

By **TimSch**

cheatography.com/timsch/

Published 18th March, 2018.

Last updated 18th March, 2018.

Page 1 of 100.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Einfach verkettete Listen (cont)

Globale Zeiger auf `extern TData *First = NULL;`
Listenanfang und `extern TData *Last = NULL;`
Listenende
Schlüsselwort extern, damit die Zeiger über
alle Quelltextdateien verfügbar sind. Alternativ
in main definieren.

Einfach verkettete Listen (cont)

Funktionen (bspw. in list.c/h auslagern)



By **TimSch**

cheatography.com/timsch/

Published 18th March, 2018.

Last updated 18th March, 2018.

Page 2 of 100.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Einfach verkettete Listen (cont)

```
Neues      int append InL ist (TData *Neu)
Element    {
anhängen   //prüfen, ob neues Element existiert
            if (Neu == NULL)
                return 0;

            //Neues Element ist neues Listenende
            Neu->Next = NULL;

            if (First == NULL)
                //Fall 1: Liste ist leer
                First = Last = Neu;
            else
                //Fall 2: Liste ist nicht leer
                Last = Last->Next = Neu;
            return 1;
        }
```

Einfach verkettete Listen (cont)

```
Neues      int insert InL ist (TData *Neu)
Element    {
(sortiert) TData *tmp = First;
einfügen   //prüfen, ob neues Element existiert
            if (Neu == NULL)
                return 0;

            if (First == NULL)
            { //Fall 1: Liste ist leer
                Neu->Next = NULL;
                First = Last = Neu;
                return 1;
            }

            if (First->Value >= Neu->Value)
            { //Fall 2: am Listen anfang einfügen
                Neu->Next = First;
                First = Neu;
                return 1;
            }

            if (Last->Value <= Neu->Value)
            { //Fall 3: am Listenende anhängen
                Neu->Next = NULL;
                Last = Last->Next = Neu;
                return 1;
            }

            //Fall 4: zwischen zwei Elemente einfügen
            while (tmp->Next != NULL)
            { //prüfen, ob neues Listen element vor

                if (tmp->Next->Value >= Neu->Value)
                {
                    Neu->Next = tmp->Next;
                    tmp->Next = Neu;
                    return 1;
                }
            }
            return 0;
        }
```

Einfach verkettete Listen (cont)

```

Element   TData *remov eFr omL ist(int delIndex)
löschen   {
    TData tmp = NULL, prev = NULL;
    // Fall 1: Liste leer?
    if (First == NULL)
        return NULL;

    // Fall 2: Listen anfang entfernen?
    if (First ->Index == delIndex)
    {
        tmp = First;
        // nur ein Element in Liste?
        if (Last == First)
            Last = NULL;
        First = First->Next;
        return tmp;
    }

    // Fall 3: zu entfer nendes Element suchen
    prev = First;
    tmp = prev-> Next;
    while (tmp != NULL)
    {
        if (tmp-> Index == delIndex)
        {
            prev->Next = tmp->Next;
            if (tmp == Last) // letztes Element entfernt?
                Last = prev;
            return tmp;
        }
        prev = tmp;
        tmp = tmp->Next;
    }
    return NULL;
}

```

Doppelt verkettete Listen

```

An Liste   int append InL ist (TData *Neu)
anhängen   {
    //prüfen, ob neues Element existiert
    if (Neu == NULL)
        return 0;

    if (First == NULL)
        //Fall 1: Liste ist leer
        Neu->Next = Neu->Prev = NULL;
        First = Last = Neu;
    else
        //Fall 2: Liste ist nicht leer
        Neu->Next = NULL;
        Neu->Prev = Last;
        Last = Last->Next = Neu;
    return 1;
}

```

Doppelt verkettete Listen (cont)

```

In Liste      int insert InL ist (TData *Neu)
(sortiert)    {
einfügen      TData *tmp = First;

               //p rufen, ob neues Element existiert
               if(Neu == NULL)
                   return 0;

               if( First == NULL)
               { //Fall 1: Liste ist leer
                   Neu ->Next = Neu->Prev = NULL;
                   First = Last = Neu;
                   return 1;
               }

               if (First ->Value >= Neu->V alue(
               { //Fall 2: am Listen anfang einfügen
                   Neu ->Next = First;
                   Neu ->Prev = NULL;
                   First = First- >Prev = Neu;
                   return 1;
               }

               if (Last- >Value <= Neu->V alue)
               { //Fall 3: am Listenende anhängen
                   Neu ->Prev = Last;
                   Neu ->Next = NULL;
                   Last = Last->Next = Neu;
                   return 1;
               }

               //Fall 4: zwischen zwei Elemente einfügen
               while (tmp->Next != NULL)
               { //prüfen, ob neues Listen element vor dem nächsten Listen element eingefügt werden mus
                   if (tmp-> Nex t-> Value >= Neu->V alue)
                   {
                       Neu ->Next = tmp->Next;
                       Neu ->Prev = tmp;
                       tmp ->Next = tmp-> Nex t->Prev = Neu;
                       return 1;
                   }
                   tmp = temp-> Next;
               }
               return 0;
           }

```

Doppelt verkettete Listen (cont)

```

Aus Liste     TData *remov eFr omL ist(int delIndex)
entfernen     {
               TData *tmp = NULL, *prev = NULL;
               // Fall 1: Liste leer?
               if (First == NULL)
                   return NULL;

               // Fall 2: Listen anfang entfernen?
               if (First ->Index == delIndex)
               {
                   tmp = First;
                   First = First- >Next;
                   // nur ein Element in Liste?
                   if (First == NULL)
                       Last = NULL;
                   else
                       Fir st- >Prev = NULL;
                   return tmp;
               }

               // Fall 3: Listenende entfernen?
               if (Last- >Index == delIndex)
               {
                   tmp = Last;
                   Last = Last-> Prev;
                   Las t->Next = NULL;
                   return tmp;
               }

               //Fall 4: zu entfer nendes Element such
               n
               tmp = First- >Next;
               while (tmp != NULL)
               {
                   if (tmp-> Index == delIndex)
                   {
                       prev = tmp->Prev;
                       pre v->Next = tmp->Next;
                       pre v-> Nex t->Prev = prev;
                       return tmp;
                   }
                   tmp = tmp->Next;
               }
               return NULL;
           }

```



By **TimSch**

cheatography.com/timsch/

Published 18th March, 2018.

Last updated 18th March, 2018.

Page 5 of 100.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Doppelt verkettete Listen (cont)

```
Alle      typedef enum {forward, backward} TDirection;
Elemente
auflisten void printList (TDirection Direction)
{
    TData *tmp = (Direction == forward) ? First : Last;
    int AnzahlElemente = 0;

    printf ("\n Index ");
    while (tmp)
    {
        printf ("| %3i ", tmp->Index);
        tmp = (Direction == forward) ? tmp->Next : tmp->Prev;
        AnzahlElemente++;
    }

    printf ("\n --- --- ");
    while (AnzahlElemente-->0)
        printf ("-- --- -");

    printf ("\nWert ");
    tmp = (Direction == forward) ? First : Last;
    while (tmp != NULL)
    {
        printf ("| %3.0f ", tmp->Value);
        tmp = (Direction == forward) ? tmp->Next : tmp->Prev;
    }
    printf ("\n ");
}
```



By TimSch

cheatography.com/timsch/

Published 18th March, 2018.

Last updated 18th March, 2018.

Page 6 of 100.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>