11. Übung

1. Auswertung Hausaufgaben
2. Tipps zur Hausaufgabenserie Zusatz
3. Listen, Formale Spezifikationen, Realisierungsmöglichkeiten
4. Alte Klausuraufgabe
5. Übungen
6. Lösungen zu 5

1. Auswertung Hausaufgabenserie 4

1. Schreiben Sie ein rekursives Programm, das alle Permutationen von n Elementen ausgibt. (D.h., alle möglichen unterschiedlichen Reihenfolgen der Elemente.) Das Programm soll die Zahl n als Kommandozeilenargument akzeptieren (Sie können die Methode hierfür von hanoi.c kopieren). Sie können davon ausgehen, dass n nicht größer als 26 ist, so dass Sie die Elemente mit Buchstaben bezeichnen können. Ihr Programm soll damit beispielsweise folgende Ausgabe liefern:

```
$ ./perm 3
1: ABC
2: BAC
3: CAB
4: ACB
5: BCA
6: CBA
6 permutations
$
```

Tipp: Heap's Algorithmus. (12 Punkte)

1. Auswertung Hausaufgabenserie 4

https://en.wikipedia.org/wiki/Heap's_algorithm

```
procedure generate(n : integer, A : array of any):
    if n = 1 then
          output(A)
    else
        for i := 0; i < n - 1; i += 1 do
            generate(n - 1, A)
            if n is even then
                swap(A[i], A[n-1])
            else
                swap(A[0], A[n-1])
            end if
        end for
        generate(n - 1, A)
    end if
```

1. Auswertung Hausaufgabenserie 4

```
#include <stdio.h>
#include <stdlib.h>

char elems[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int np;

void swap(char *x, char *y) {
          char tmp = *x; *x = *y; *y = tmp;
}
void ausgabe(char a[]){
    printf("%d: %s\n",++np, elems);
}

void generate(int n, char a[]) {
          int i;
          if(n==1) ausgabe(a); /* printf("%d: %s\n",++np,elems); */
          else {
                    for(i=0;i < (n-1);i++) {
                              generate(n-1, a);
                              if (n%2==0)
                                        swap(&elems[i],&elems[n-1]);
                               else
                                        swap(&elems[0],&elems[n-1]);
                    }
                    generate(n-1, a);
          }
}
```

1. Auswertung Hausaufgabenserie 4

```
int main(int argc, char *argv[]) {
            int n;
            if(argc < 2 || (n = atoi(argv[1])) < 0) {
            printf("Usage: %s <number-of-elements>\n",argv[0]);
            return -1;
            }
            np = 0;
            elems[n] = '\0';
            generate(n,elems);
            printf("%d permutations\n",np);
            return 0;
}
```

1. Auswertung Hausaufgabenserie 4

Musterloesung Prof Kirste:

```
#include <stdio.h>
#include <stdlib.h>
char elems[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int np=0;
void swap(int i, int j) {
            char tmp = elems[i];
            elems[i] = elems[j];
            elems[j] = tmp;
}
void perm(int n) {
            int i;
            if(n==0) printf("%d: %s\n",++np,elems);
            else {
                        perm(n-1);
                        for(i=n-2;i>=0;--i) {
                                    swap(i,n-1);
                                    perm(n-1);
                                    swap(i,n-1);
                        }
            }
}
int main(int argc, char *argv[]) {
            int n;
            if(argc < 2 || (n = atoi(argv[1])) < 0) {
                        printf("Usage: %s <number-of-elements>\n",argv[0]);
                        return -1;
            }
            elems[n] = '\0';
            perm(n);
            printf("%d permutations\n",np);
            return 0;
}
```

1. Auswertung Hausaufgabenserie 4

2. Sortieren. Entwickeln Sie eine Funktion zum Sortieren eines Feldes von Zahlen mit Hilfe der Funktion swap aus der Vorlesung.
Die Idee ist einfach:

- *Wenn zwei benachbarte Zahlen im Feld in der falschen Reihenfolge sind, dann werden diese vertauscht.*
- *Dies machen Sie so lange, bis keine Vertauschungen mehr erforderlich sind.*

15 Punkte

1. Auswertung Hausaufgabenserie 4

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define NMAX 20
int nswap; /* zum zaehlen der Vertauschungen */
int ff[NMAX]; / * globales Array als bekannt benutzen */

void init() {
  int i;
  nswap=0;
  for(i=0;i<NMAX;i++)
    ff[i] = rand();
}
void ausgabe() { /*  Schoene Ausgabe 10 Zahlen in einer Reihe */
  int i;
  int j=0;
  for(i=0;i<NMAX;i++) {
    printf("%9d, ", ff[i]);
                    j++;
                    if(j==10) {
                                    printf("\n");
                                    j=0;
                    }
  }
  printf("\n\n\n");
}
```

1. Auswertung Hausaufgabenserie 4

```
void swap(int *a, int *b) {
  int tmp;
  tmp = *a;
  *a = *b;
  *b = tmp;
  nswap++;
}
void sort() {
  int swapped = 1, i;
  while(swapped) { /* Solange bis keine Vertauschung aufgetreten ist */
    swapped=0;
    for(i=0;i<(NMAX-1);i++)
      if(ff[i] > ff[i+1]) {
                swapped = 1; /* Vertauschung, deshalb noch ein Durchlauf */
                swap(ff+i,ff+(i+1));
      }
  }
}
int main(int argc, char *argv[]) {
  int i, r;
  r = atoi(argv[1]);
  printf(" %d Versuche zum testen des Programmes \n",r);
  for(i=0;i<r;i++) {
      init();
      ausgabe();
      sort();
      printf(" %d Vertauschungen notwendig\n",r);
      ausgabe();
  }
  return 0;
}
```

1. Auswertung Hausaufgabenserie 4

```
/* alternative Version */
#include <stdio.h>
void swap(int *a, int *b) {
 int tmp = *a;
 *a = *b;
 *b = tmp;

void sortiere(int a[], int n) {
 int swapped = 1; /* Tausch durchgeführt oder nicht */
 int i; /* Schleifenzähler */

 while (swapped == 1) {
  swapped = 0;
  for (i = 0; i < n - 1; i++) {      /* Vergleiche aktuelles Element mit Folgelement */
    if (a[i] > a[i + 1]) {         /*  Nutzung der swap Funktion zum Tausch der Elemente */
             swap(&a[i], &a[i + 1]);
             swapped = 1; /* Tausch durchgeführt */
    }
  }
 }
}
int main() {  /* Test-Array */  int a[] = {3, 2, 8, 1, 9, 7, 0, 5, 4, 6}; /* unsortiertes Array */
 int size = sizeof(a) / sizeof(int); /* Anzahl Elemente */
 int i; /* Schleifenzähler */
 printf("Unsortiert: ");
 for (i = 0; i < size; i++)
  printf("%d ", a[i]);
 sortiere(a, size); /* rufe Soritierfunktion mit Test-Array auf */
 printf("\nSortiert:   ");
 for (i = 0; i < size; i++)
  printf("%d ", a[i]);
 return 0;
}
```

1. Auswertung Hausaufgabenserie 4

4. Schreiben Sie ein Programm, das Text in „Pseudo-Englisch“ erzeugen kann. Dazu gehen Sie folgendermaßen vor:

• Bauen Sie ein dreidimensionales Feld auf, das für jedes Zeichen c (Buchstaben sowie Leerzeichen, Komma, und Punkt) die bedingte Wahrscheinlichkeit beinhaltet, dass das Zeichen c auf die Zeichen a und b folgt.

Diese Tabelle können Sie beispielsweise durch die Analyse von Herman Melvilles „Moby Dick“ erzeugen. (Geben Sie an, woher Sie den Text haben!) (10 Punkte)

• Entwickeln Sie ein Verfahren, mit dem Sie aus einem Vektor von Werten (v_1, ... , v_n), für die Sie einen Vektor von Wahrscheinlichkeiten (p_1, ... , p_n) gegeben haben, einen Wert v_i zufällig wählen können, so dass die Wahrscheinlichkeit, dass Sie v_i bekommen eben genau p_i entspricht. Sie können davon ausgehen, dass gilt

$$\sum_{i=1}^{n} p_i$$

10 Punkte

• Erzeugen Sie mit Hilfe des dreidimensionalen Feldes einen Zufallstext dadurch, dass Sie sich jeweils die beiden zuletzt generierten Zeichen a und b merken und dann ein neues Zeichen c zufällig wählen, wobei Sie die Wahrscheinlichkeit eines bestimmten Zeichens c gegeben die Vorgänger a und b aus dem Feld entnehmen. Um das erste Zeichen zu erzeugen können Sie annehmen dass die Vorgängerzeichen der Punkt und das Leerzeichen sind.

10 Punkte

1. Auswertung Hausaufgabenserie 4

Lösung Prof Kirste:

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
/* a..z + SPACE + COLON + STOP */
#define N 29
#define C_SPACE 26
#define C_COMMA 27
#define C_STOP 28
#define drand48()(rand()*(1.0/RAND_MAX))

char chars[]="abcdefghijklmnopqrstuvwxyz ,.?";
double Pc_ab[N][N][N];
double F_ab[N][N];
double F;

int getcc() { /* siehe Aufgabe 3 getcc liest ein Zeichen ein, wobei Leer- und Trennzeichen überlesen
und nur ein Leerzeichen beruecksichtigt wird */
           int c, haveSpace = 0;
           while(isspace(c=getchar()))
           haveSpace = 1;
           if(haveSpace) return (ungetc(c,stdin),C_SPACE);
           else if(c>='a' && c<='z') return c-'a';
           else if(c>='A' && c<='Z') return c-'A';
           else if(c==',') return C_COMMA;
           else if(c=='.') return C_STOP;
           else if(c==EOF) return EOF;
           else return getcc();
}
```

1. Auswertung Hausaufgabenserie 4

Lösung Prof Kirste:

```
void fcount() {
        int a, b, c;
        for(a=getcc(),b=getcc(),c=getcc(); c!=EOF; a=b, b=c, c=getcc()){
                Pc_ab[a][b][c]++;
                F_ab[a][b]++;
                F++;
        }
        fprintf(stderr,"%f triples collected\n",F);
                for(a=0;a<N;a++)
                        for(b=0;b<N;b++)
                                if(F_ab[a][b] > 0.0) // Es gibt unmoegliche Kombinationen
                                        for(c=0;c<N;c++)
                                        Pc_ab[a][b][c] /= F_ab[a][b];
}

void generate(int n) {
        int a, b, c;
        double s, r;
        for(a=C_STOP,b=C_SPACE; n; a=b,b=c,n--) {
                for( s=0,r=drand48(),c=0; s+=Pc_ab[a][b][c],s<r; c++) ;
                        putchar(chars[c]);
        }
}
int main(int argc, char *argv[]) {
        fcount();
        generate(atoi(argv[1]));
return 0;
}
```

1. Auswertung Hausaufgabenserie 4

```
Lösung Übung:
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>

/* a..z + SPACE + COLON + STOP*/
#define N 29
#define C_SPACE 26
#define C_COMMA 27
#define C_STOP 28
#define C_EXCLAMATION 29
#define C_QUESTION 30
#define drand48()( rand()*(1.0 /RAND_MAX    ))

char chars[] = "abcdefghijklmnopqrstuvwxyz ,.!?:;";
double Pc_ab[N][N][N];
double F_ab[N][N];
double F;

int possiblechar(int c) {
          if(isalpha(c)||c=='.'||c==','||c=='!'||c=='?')
           return 1 ;
          else return 0;
}
```

1. Auswertung Hausaufgabenserie 4

Lösung Übung:

```
int getcc() {
        int c, anz = 0;
        while (((c = getchar()) != EOF) && (!possiblechar(c))) {
                anz++;
        }
        if (anz > 0) { // letztes Zeichen zurueckschieben
                ungetc(c, stdin);
                return C_SPACE; // Ein Leerzeichen zaehlen
        } else { // Zeichen auswerten
                        if (c >= 'a'&&c <= 'z') return c - 'a';
                        else if (c >= 'A'&&c <= 'Z')return c - 'A';
                        else if (c == ',') return C_COMMA;
                        else if (c == '.') return C_STOP;
        }
        return EOF; // war wohl EOF
}
int maxIndex( double *f){
        int i, k;
        double m = f[0];
        k = 0;
        for(i=0; i<N;i++){
                if(f[i]>m) {
                        m = f[i];
                        k = i;
                }
        }
        return k;
}
```

1. Auswertung Hausaufgabenserie 4

Lösung Übung:

```
void fcount() {
        int a, b, c;
        a = getcc();
        b = getcc();
        for (c = getcc(); c != EOF; c = getcc()) {
                Pc_ab[a][b][c]++;
                F_ab[a][b]++;
                F++;
                a = b;
                b = c;
        }
        printf("%ftriples collected \n", F);
        // Jetzt Berechnung relativer Haeufigkeiten Ha Teil 1
        for (a = 0; a<N; a++)
                for (b = 0; b<N; b++)
                        if (F_ab[a][b] > 0.0) {
                                /*Es gibt unmoegliche Kombinationen */
                                for (c = 0; c < N; c++)
                                Pc_ab[a][b][c] /= F_ab[a][b];
                        }
}
```

1. Auswertung Hausaufgabenserie 4

Lösung :

```
void generate(int n) {
        int a, b, c, k;
        int p = 0;
        double s, r;
        for (a = C_STOP, b = C_SPACE; n >= 0; a = b, b = c, n--) {
                p = 0;
                for (s = 0, r = drand48(), c = rand()%N; s<r; c=(c+1)%N) {
                /*for (s = 0, r = drand48(), c = 0; s<r; c++)         { */
                        if ((F_ab[a][b] != 0) && ((s + Pc_ab[a][b][c])<r))
                                s += Pc_ab[a][b][c];
                        else {
                                break;
                                p = 1;
                        }
                }
                /*
                if (p==1)
                        putchar(chars[c]);
                else {
                        int k = maxIndex(Pc_ab[a][b]);
                        putchar(chars[k]);
                        c = k;
                }
                */
                putchar(chars[c]);
        }
}
```

1. Auswertung Hausaufgabenserie 4

Lösung :

```
int main(int argc, char** argv)
{
          srand(time(NULL));
          rand();
          rand();
          char cc;
          fcount();
          generate(atoi(argv[1]));
          return 0;
}
```

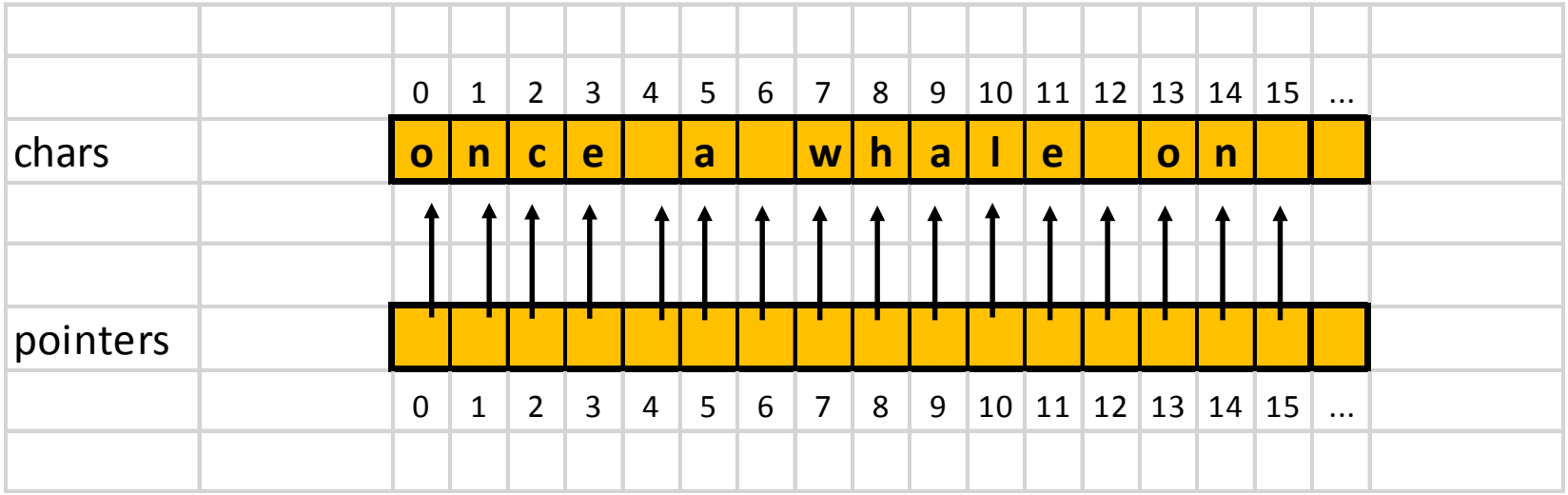
2. Ideen Hausaufgabenserie 5 (40 Zusatzpunkte)

1. Selbstplagiate: Schreiben Sie ein Programm, das in einem Text, der von der Standardeingabe gelesen wird, die längste duplizierte Passage herausfindet. Sie können davon ausgehen, dass die Standardeingabe aus einer Datei kommt.
Hinweis: Sie können ein solches Programm ziemlich effizient mit Hilfe der Funktion qsort realisieren:
(a) Lesen Sie den kompletten Text in einen String ein.
(b) Bauen Sie ein Pointer-Array auf, das genau so viele Pointer enthält, wie der String Zeichen.
(c) Speichern Sie im Pointer-Array an der Stelle i einen Verweis auf das String-Zeichen i. Also: pointers[i] = string + i; Danach zeigt jeder Eintrag von pointers auf einen String.
(d) Sortieren Sie das Pointer-Array mit Hilfe von qsort, wobei Sie zum Vergleichen die Funktion strcmp verwenden.
(e) Finden Sie die längste Übereinstimmung zwischen zwei aufeinanderfolgenden Strings in pointers. Die Funktionen fseek und ftell können Ihnen bei der anfänglichen Bestimmung der Textgröße helfen.
Verwenden Sie Ihr Programm, um die längste duplizierte Passage im „Moby Dick“ zu finden (vgl. Übungsblatt 6, Aufgabe 2).
40 Zusatzpunkte

2. Ideen Hausaufgabenserie 5

char *chars;
char **pointers;

Textausschnitt aus Mobi Dick : „once a whale“



Hier längstes Duplikat „on“
Idee Sortiere Feld pointers alphabetisch (Nutzung Funktion qsort aus <stdlib.h>), dann vergleiche benachbarte Felder bis abweichender Buchstabe! Wenn neues Maximum (Selbstplagiat), dann ausgeben.

2. Ideen Hausaufgabenserie 5

qsort –

The C library function void sorts an array.

Declaration

Following is the declaration for qsort() function.
void **qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *, const void*))**

Parameters

base − This is the pointer to the first element of the array to be sorted.

nitems − This is the number of elements in the array pointed by base.

size − This is the size in bytes of each element in the array.

compar − This is the function that compares two elements.

Return Value:

This function does not return any value.

2. Ideen Hausaufgabenserie 5

```
/* Funktion fuer den 4. Parameter definieren im Quelltext*/


int cmpf(const void *a, const void *b) {
        /* alphabetischer Vergleich von Zeigern auf Buchstaben */
        return strcmp(*(char **)a,*(char **)b);
}




qsort(pointers,nch,sizeof(char *),cmpf);
```

3. Typedefs in C

Wir kennen Datentypen mit Definition struct:

```
struct _person {
        char first[20];
        char last[20];
        int matr;
        int year;
} pers, personen[10]; /* Variablendefinition */
```

Die Variablen pers und das Array personen sind gleich mit der Datentypdefinition angelegt worden, weitere Variablen können/müssen über Datentyp „struct _person” definiert werden.

```
struct _person p; /* weitere Variable */
struct _person *z; /* Variable als Zeiger auf Struktur struct _person */
```

Kann man die Typdefinition nicht einfacher schreiben?
Ja mit typedef !

3. Typedefs in C

In C gibt es das vordefinierte Schlüsselwort typedef
Typedef fordert erst eine bestehenden Typdefinition und dann einen neuen Namen.
Beispiel wir wollen struct _person in den Datentyp PersonRec und in den Zeiger Person umbenennen.

```
typedef struct _person {
        char first[20];
        char last[20];
        int matr;
        int year;
} PersonRec, *Person; /* Typdefinitionen, keine Variablendefinition!!! */
```

Hinweis solche Definitionen funktionieren für beliebige Datentypen. Wir könnten z.B. den Datentyp Elem einführen, der tatsächlich einen int-Wert speichert:

```
typedef int Elem ; /* weitere Bezeichnung fuer int ! */
```

3. Typedefs in C

```
#include <stdio.h>
#include <string.h>

typedef struct _person{
        char first[20];
        char last[20];
        int matr;
        int year;
}PersonRec, *Person;
int rekursive_eingabe() {
        int nn = 0;
        PersonRec pers;
        printf("Bitte Vorname Name Matrikelnummer, Jahr eingeben:\n");
        if(scanf("%s %s %d %d",pers.first,pers.last,&pers.matr,&pers.year) == 4){
                nn = rekursive_eingabe() + 1;
                printf("%d %s %s %d Laenge des
Namens:%d\n",pers.matr,pers.last,pers.first,pers.year, strlen(pers.last));
        }
        return nn;
}
int main() {
        printf("Read %d records\n",rekursive_eingabe());
return 0;
}
```

```
$ TypePerson.exe
Bitte Vorname Name Matrikelnummer, Jahr eingeben:
Wally Wusel 234567 2019
Bitte Vorname Name Matrikelnummer, Jahr eingeben:
Egon Emsig 423567 2018
Bitte Vorname Name Matrikelnummer, Jahr eingeben:
Susi Sorglos 542673 2109
Bitte Vorname Name Matrikelnummer, Jahr eingeben:
a a a a
542673 Sorglos Susi 2109 Laenge des Namens:7
423567 Emsig Egon 2018 Laenge des Namens:5
234567 Wusel Wally 2019 Laenge des Namens:5
Read 3 records
```

3. Dynamischer Speicher in C

Wir wollen eine variable Anzahl von Informationen zu speichern!

Problem bei Feldern: feste Größen,

Problembehandlung bei Überlauf: größeres Feld und alte Elemente müssen umgespeichert werden.

Problem bei Strings (Länge des char-Arrays für String).

Dynamische Listen:
Inhaltlich bestehen sie aus 2 Teilen: Bereich zum Speichern einer Informationseinheit und einem Bereich zum Verketten von Informationen.

Speicher wird zur Laufzeit dynamisch angefordert! (malloc)

Beliebig erweiterbar, aber es besteht Platzbedarf für Zeiger.

3. Dynamischer Speicher in C

Idee: Wir erzeugen dynamische eine Variable vom Typ PersonRec!
Idee: wir stellen Puffer zur Eingabe eines Names bereit, legen dann eine Variable in korrekter Länge an und speichern die Information um!

```
#include <stdio.h>
#include <malloc.h>
#include<string.h>

typedef struct _person{
         char *last;
         int matr;
} PersonRec, *Person;
 ...

         int matnr;
         char name[20];
         Person pzeig;
         printf("Bitte Name und Matrikelnummer eingeben:\n");
         while(scanf("%s %d",name,&matnr,) == 2){
           printf("%d %s\n",matnr,name);
           pzeig = (Person) malloc(sizeof (PersonRec));/*dynamisch Speicherplatz */
           pzeig->last = (char*) malloc(strlen(name)+1);/* Platz fuer String Namen */
           strcpy(pzeig->last, name); /* Kopiere Inhalte von Name zur Struktur*/
           pzeig->matr = matnr;
...
```

4. Realisierungen von Listen

Liste mit Array-Implementierung

Realisierung mit Array in C:

```
int f[]= {37, 42, 17, 29, 4, -9};
```

f[0]	f[1]	f[2]	f[3]	f[4]	f[5]
37	42	17	29	4	-9

Keine Probleme bei Einfügen, wenn weniger als 6 Elemente (Kapazität) eingegeben

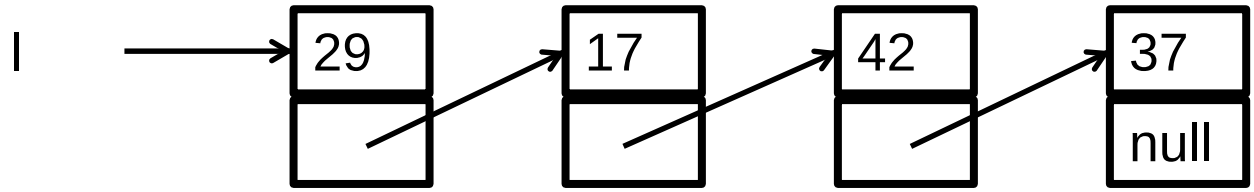
Was tun bei mehr als 6?
(Größeres Feld anlegen alle Elemente kopieren)

Delete 17
Alle Elemente ab Löschposition verschieben!

4. Realisierungen von Listen

Liste mit einfacher Verkettung

Beispiel Liste aus Zahlen 37, 42, 17, 29 in eine Kette



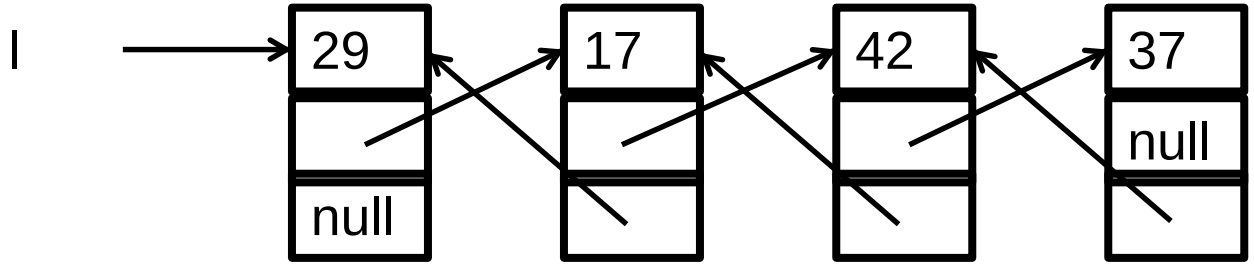
Keine Probleme bei Einfügen!
Neue Zelle anlegen, Nachfolger ist alte Zelle, neuen Anfang definieren.

Löschen von 17
Vorgänger bestimmen (29) oder merken, um dort den Nachfolger der gelöschten Zelle (42) einzutragen, dann ist 42 der Nachfolger von 29

4. Realisierungen von Listen

Liste mit doppelter Verkettung

Beispiel Liste aus Zahlen 37, 42, 17, 29 in einer doppelt verketteten Liste



Keine Probleme bei Einfügen!
Neue Zelle anlegen, Nachfolger ist alte Zelle, neuen Anfang definieren.

Löschen von 17
Vorgänger bestimmen, um dort den Nachfolger der gelöschten Zelle eintragen, dann ist 42 der Nachfolger von 29 und 29 der Vorgänger von 42)

4. Realisierungen von Listen

Liste mit Verkettungen

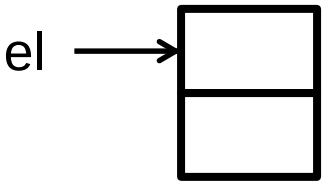
```
#include <stdio.h>
#include <malloc.h>

typedef struct _lelem{ /* einfach verkette Liste */
            int wert;
            struct _lelem *next;
} ListRec, *List;

int main (void) {
            int i, a;
            List el_anf, el, l;
            ListRec q ={37, NULL}; /* Weitere Variable */
            l = &q; /* Zeiger auf eine Informationselement */
            printf("% Ausgabe des Wertes d\n", l->wert);
            /*  Aufbau einer Liste */
            l = NULL;
            for(i=0;i<3;i++){
                        printf("Bitte Wert eingeben");
                        scanf("%d", &a);
                        el = (List) malloc(sizeof (ListRec));/*dynamisch erzeugter Speicherplatz */
                        el->wert = a; /* Wert einspeichern */
                        el->next = l; /* Alter Anfang ist Nachfolger */
                        l = el; /* Liste hat neuen anfang */
            }

            /* Ausgabe der Liste */
            while(l!=NULL){
                        printf("%d, ", l->wert);
                        l = l->next;
            }
}
```

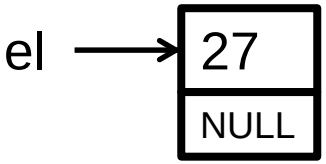
4. Realisierungen von Listen

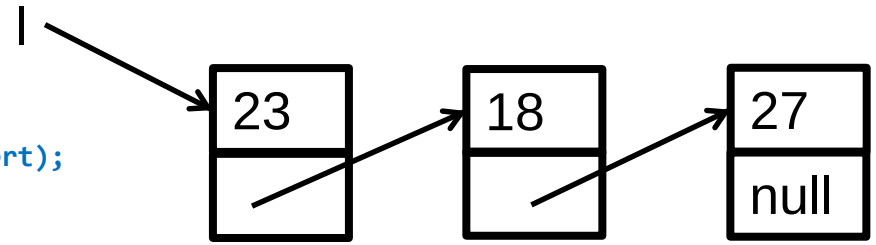
Liste mit Verkettungen

```
#include <stdio.h>
#include <malloc.h>

typedef struct _lelem{ /* einfach verkette Liste */
            int wert;
            struct _lelem *next;
} ListRec, *List;

int main (void) {
            int i, a;
            List el_anf, el, l;
            ListRec q ={37, NULL}; /* Weitere Variable */
            l = &q; /* Zeiger auf eine Informationselement */
            printf("% Ausgabe des Wertes d\n", l->wert);
            /*  Aufbau einer Liste */
            l = NULL;
            for(i=0;i<3;i++){
                        printf("Bitte Wert eingeben");
                        scanf("%d", &a);
                        el = (List) malloc(sizeof (ListRec));/*dynamisch erzeugter Speicherplatz */
                        el->wert = a; /* Wert einspeichern */
                        el->next = l; /* Alter Anfang ist Nachfolger */
                        l = el; /* Liste hat neuen anfang */
            }

            /* Ausgabe der Liste */
            while(l!=NULL){
                        printf("%d, ", l->wert);
                        l = l->next;
            }
}
```

l = NULL







5. Alte Klausuraufgabe

1.1 Parameterübergabe in C

15 Punkte

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct _list {
   int wert;
   struct _list *next;
} Elem, *List;

List insert(int v, List l) {
   List e = malloc(sizeof(Elem));
   e->wert = v;
   e->next = l;
   return e;
}

List change(int *x, int y, int *z, List m) {
   if (m->next != NULL) {
      *x = *x + 2;
      y = y - 4;
      *z += *x;
      m->wert = y;
      m = m->next;
   } else {
      (*x) -= y;
   }
   return m;
}

void main() {
   int a = 10, b = 5;
   int *c;
   c = &b;
   List d = insert(4,insert(1,insert(3,NULL)));
   List e = insert(3,NULL);

   printf("%d,%d,%d,%d,%d\n", a, b, *c, d->wert, e->wert);

   change(&a, b, c, d);
   printf("%d,%d,%d,%d,%d\n", a, b, *c, d->wert, e->wert);

   d = change(&a, b, c, e);
   printf("%d,%d,%d,%d,%d\n", a, b, *c, d->wert, e->wert);
}
```

6. Aufgaben in Übung

```
typedef struct _person{
          char *first;
          char *last;
          int matr;
          int year;
          struct _person *next;
}PersonRec, *Person;
```

1. Programmieren sie in der Liste der Personen (Einfachverkettung) folgende Methoden

   Vorgabe: Eine Funktion Eingabe der Liste

   1. Ausgabe von Werten der Listenelemente
   2. Zählen der Anzahl der Listenelemente
   3. Bestimmen des maximalen Immatrikulationsjahres
   4. Bestimmen des mittleren Elementes der Liste (Zeiger auf Mitte)
   5. Bestimmen des kleinsten Namens (Alphabetische Ordnung) - Nutzung der Funktion strcmp zum vergleich von Strings in C (aus #include <string.h>)!

```
int strcmp(const char *s1, const char *s2);
```

```
Sind beide Strings identisch, gibt diese Funktion 0 zurück. Ist der String s1 kleiner als s2, ist der
Rückgabewert kleiner als 0; und ist s1 größer als s2, dann ist der Rückgabewert größer als 0.
```

6 Vorgabe– typePersonB.c

```
#include <stdio.h>
#include <malloc.h>
#include<string.h>
typedef struct _person{
	char *first;
	char *last;
	int matr;
	int year;
	struct _person *next;
}PersonRec, *Person;
int eingabe(Person *lAnf) {/* Referenzparameter Person ! */
	int n=0;
	char name[20], vorname[20];
	int matnr, jahr;
	Person pzeig;
	printf("Bitte Vorname Name Matrikelnummer und Jahr eingeben:\n",vorname, name, matnr, jahr);
	while(scanf("%s %s %d %d",vorname,name,&matnr,&jahr) == 4){
			printf("%d %s %s %d\n",matnr,name,vorname,jahr);
			pzeig = (Person) malloc(sizeof (PersonRec));//dynamisch erzeugter Speicherplatz
			pzeig->first = (char*) malloc(strlen(vorname)+1);//dynamisch erzeugter Speicherplatz
			pzeig->last = (char*) malloc(strlen(name)+1);//dynamisch erzeugter Speicherplatz
			strcpy(pzeig->first, vorname);
			strcpy(pzeig->last, name);
			pzeig->matr = matnr;
			pzeig->year = jahr;
			pzeig->next= *lAnf;
			*lAnf = pzeig;
			n++;
	printf("Bitte Vorname Name Matrikelnummer und Jahr eingeben:\n",vorname, name, matnr, jahr);
	}
	return n;
}
```