Teilklausur I in Informatik – TEL08GR1 + TELK08 (TEL08GR2) – Lewandowski

Es sind insgesamt 84 Punkte zu erreichen (80 Punkte werden als 100% gewertet)
(Jeder Punkt entspricht einer Bearbeitungszeit von etwa 1 Minute)

Aufgabe 1: Gerade Zahlen (10 Punkte) – Struktogramm
Aufgabe 2: Ausgewürfelt (30 Punkte) – DF, PAD/PN, C/C++
Aufgabe 3: Eine Datenstruktur (10 Punkte) – DH, C/C++
Aufgabe 4: Wissensfragen und anderes (14 Punkte)
Aufgabe 5: Einfach verkettete Listen (20 Punkte) – C/C++

Bitte kennzeichnen Sie jedes Blatt lesbar mit Ihrem Namen und Ihrer Matrikelnummer

Zugelassene Hilfsmittel: Keine

Lesen Sie die Aufgaben in Ruhe durch!
Viel Erfolg!

Aufgabe 1 — 10 Punkte

(10 Punkte) **Gerade Zahlen:** Entwerfen Sie ein kleines Programm, das von einem Benutzer Zahlen einliest und dabei zählt, wieviel gerade Zahlen eingelesen wurden. Das Ende erfolgt durch eine Eingabe einer Zahl kleiner oder gleich 0 durch den Benutzer. Das Programm soll danach ausgeben, wieviel gerade Zahlen vom Benutzer eingegeben wurden.

Beispiel: Der Benutzer gibt folgende Zahlen ein (jeweils getrennt durch die `<RETURN>`-Taste): `42 11 7 21 18 4 3 -2`, dann soll die Ausgabe lauten:
`Es wurden 3 gerade Zahlen eingegeben.`

Geben Sie Ihren Entwurf als Struktogramm an (10 Punkte). Der Entwurf soll so detailliert sein, dass eine Umsetzung 1:1 in ein C/C++-Programm möglich wäre, d.h., dass alle Fallunterscheidungen und Schleifen auch im Entwurf als solche vorkommen müssen. Beachten Sie: Eine Umsetzung in C/C++ ist hier *nicht* gefordert.

Aufgabe 2 — 30 Punkte

(2+16+12 = 30 Punkte) **Ausgewürfelt:** In dieser Aufgabe ist eine Funktion `markiere_max(.,.)` innerhalb eines Spiels zu entwerfen und in C- oder C++-Code umzusetzen.

Die Funktion soll dabei zwei Arrays als Parameter bekommen: Das eine Array enthält `int`-Werte (die Werte seien hier aus dem Bereich 1 bis 6), ein zweites Array enthält Wahrheitswerte, die angeben, ob der Wert im `int`-Array zu berücksichtigen ist. Die Funktion soll aus dem ersten Array den größten zu berücksichtigenden Wert heraussuchen und dann das Wahrheitswerte-Array so modifizieren, dass danach nur noch genau die Stellen wahr sind, die zuvor auch wahr waren und dessen zugehöriger Wert dem zuvor bestimmten Maximalwert entsprechen. (Dies könnte z.B. ein wiederholtes Würfeln in einem Spiel simulieren, bei dem die Spieler nach und nach ausscheiden (nur, wer im zweiten Array markiert ist, ist noch am Spiel beteiligt).)

Gegeben sind die C/C++-Definitionen:

```
const int MAX_SPIELER=7;
typedef int werte[MAX_SPIELER];
typedef bool ist_dabei[MAX_SPIELER];
```

Ein Beispiel: seien zwei Arrays wie folgt definiert:

```
werte     zahlen  = {3,4,5,2,4,1,4};
ist_dabei spieler = {0,1,0,1,1,1,0}; // true entspricht 1, false entspricht 0
```

Dann soll der Aufruf `markiere_max(zahlen,spieler)` das Array `spieler` so verändern, dass es danach die Werte `0,1,0,0,1,0,0` enthält (4 ist der größte zu berücksichtigende Wert (`zahlen[2]` ist zwar 5, aber `spieler[2]` ist 0) und nur an den Indizes 1 und 4 war zuvor `spieler[.]` als wahr markiert und `zahlen[.]` gleich 4). Sind alle Einträge im Array `spieler[]` gleich 0, so soll die Ausgabe „`Keiner mehr dabei!`“ erfolgen.

Geben Sie das Datenflussdiagramm (DF) der Ebene 0 an (Funktion als ein Kästchen) (2 Punkte), sowie einen Programmablaufplan mit Daten (PAD) oder ein Programmnetz (PN) für die von Ihnen entworfene Funktion mit den nötigen Verfeinerungen (16 Punkte). Der Entwurf soll so detailliert sein, dass eine Umsetzung 1:1 in ein C/C++-Programm möglich ist, d.h., dass alle Fallunterscheidungen und Schleifen auch im Entwurf als solche vorkommen müssen. Beachten Sie: die Parameter sind an die Funktion zu übergeben und werden von dieser zurückgeliefert, Sie müssen sich *nicht* um die Ein- und Ausgabe der Werte kümmern. Geben Sie schließlich noch die Umsetzung der Funktion in C/C++ an (12 Punkte).

Aufgabe 3 — 10 Punkte

(5+5 = 10 Punkte) In dieser Aufgabe ist eine **Datenstruktur** zu entwerfen, als Datenhierarchiediagramm darzustellen und in C/C++-Code umzusetzen. Die Datenstruktur soll zur Verwaltung der Studierenden dienen und soll dabei die Daten für einen Studierenden beinhalten. Folgende Daten sollen gespeichert werden: Vorname, Nachname, Matrikelnummer und das Geburtsdatum (speichern Sie für das Datum jeweils Zahlen für den Tag, Monat und das Jahr). Das Datum soll dabei ein eigener Datentyp sein.

Geben Sie ein Datenhierarchiediagramm für Ihre Datenstruktur an (5 Punkte) und geben Sie die Umsetzung in C/C++-Code an (5 Punkte).

Aufgabe 4 — 14 Punkte

1. (4 Punkte) Beschreiben Sie in 4 kurzen Sätzen, was ein Programm von einem Algorithmus unterscheidet.

2. (5 Punkte) Geben Sie mindestens 3 gute Gründe an, warum eine Aufteilung des Source-Codes eines Projektes in einzelne Module sinnvoll ist (kurze Begründungen reichen jeweils).

3. Die `switch`-Anweisung wird gerne für große Fallunterscheidungen benutzt: ein Beispiel-Code (`i` sei eine `int`-Variable):

```
switch(i) {
  case 17: cout << "Oh, eine Primzahl";
  case 31: cout << "Was für eine schöne Primzahl"; break;
  case 91: cout << "Das ist doch keine Primzahl"; break;
  default: cout << "Was soll das denn?";
}
```

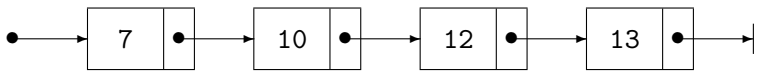
   (5 Punkte) Beschreiben Sie die Syntax der `switch`-Anweisung als EBNF oder als Syntaxdiagramm. Treten dabei Nichtterminale für Variablennamen, Typnamen, Arithmetische Ausdrücke, Boolesche Ausdrücke oder Kommandos auf, so müssen Sie diese *nicht* weiter spezifizieren.

Aufgabe 5 — 20 Punkte

(4+16 = 20 Punkte) **Einfach verkettete Listen:** In dieser Aufgabe beschäftigen wir uns mit einfach verketteten Listen, der Anker zeige dabei auf das erste Element.

1. (4 Punkte) Geben Sie eine C- oder C++-Typdefinition für eine einfach verkettete Liste aus `int`-Zahlen an. Der Datentyp für den Anker soll dabei `intListenPtr` heißen.

2. (16 Punkte) Schreiben Sie eine C/C++-Funktion `void arithm(intListenPtr liste)`, die als Parameter den Anker auf eine einfach verkettete Liste bekommt. Ist die Liste leer, so soll die Meldung „`Liste ist leer.`“ ausgegeben werden, ansonsten das arithmetische Mittel (Summe der Elemente geteilt durch deren Anzahl). Zu der Liste



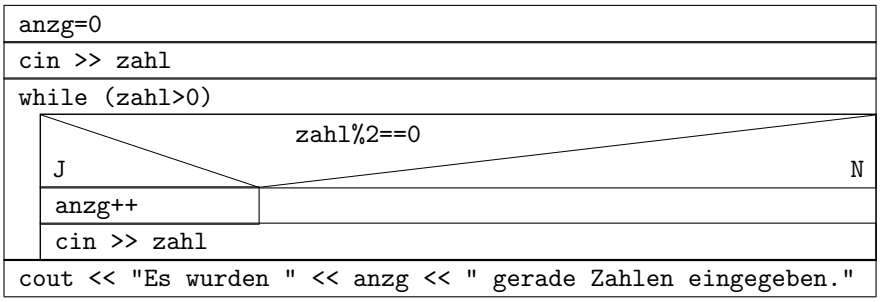
   soll die Ausgabe lauten: „`Das arithmetische Mittel beträgt 10.5`“

Beachten Sie: die Parameter sind an die Funktion zu übergeben, Sie müssen sich *nicht* um die Eingabe der Werte kümmern.

Teilklausur I in Informatik – Lösungshinweise

Aufgabe 1 — Gerade Zahlen

Hierbei war lediglich zu beachten, dass die zuletzt eingelesene Zahl (im Beispiel die `-2`) nicht mitgezählt wird. Bei der Bearbeitung ist unerheblich, ob die Befehle (wie hier) C-Syntax haben oder Pseudo-Code sind.

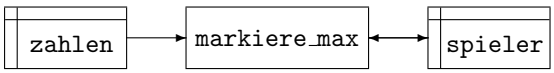


Mit einer „`do { ... } while (...)`“-Schleife geht's natürlich auch, die `if`-Abfrage ist dann aber unnötig komplizierter.

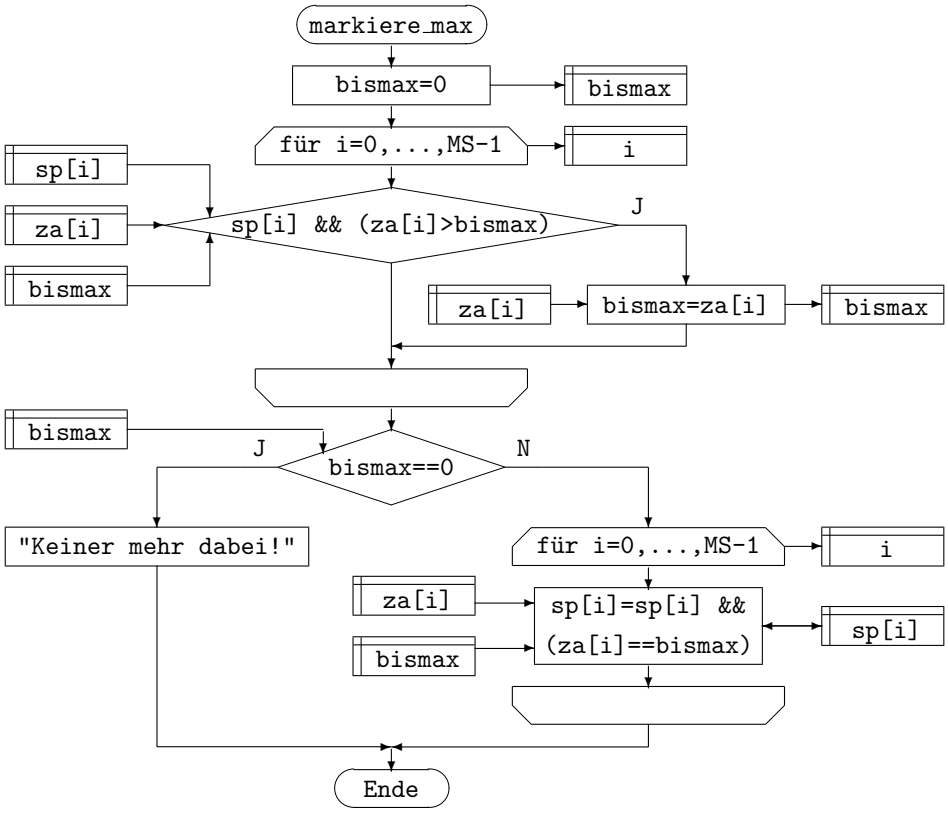
Aufgabe 2 — Ausgewürfelt

Zwei Schritte: Finde Maximalwert (oder stelle fest, dass keiner mehr mitspielt), verändere danach das Array `spieler`. Da die Komplexität der beiden Teile sehr gering ist, ist eine Aufteilung nicht zwingend erforderlich.

Das Datenflussdiagramm:



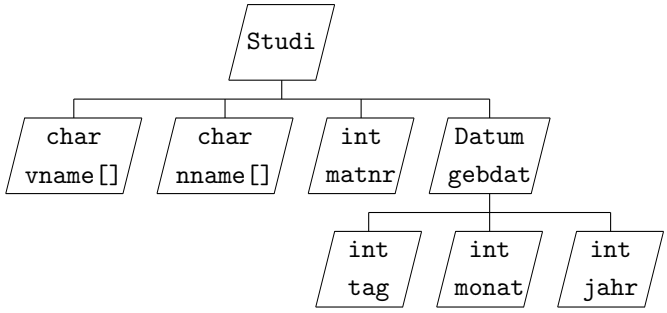
Ein Programmablaufplan mit Daten (beim Programmnetz wäre pro Variable nur ein Datensymbol vorhanden): Da vorab bekannt ist, dass die Werte im Array `zahlen` alle zwischen 1 und 6 liegen, kann hier das Maximum mit 0 initialisiert werden und so auch erkannt werden, wenn keiner mehr mitspielt. In der zweiten Schleife ist das Setzen des neuen Wahrheitswertes kompakt geschrieben (natürlich genauso korrekt wäre die Fallunterscheidung `if (spieler[i] && (zahlen[i]==bismax)) {spieler[i]=true;} else {spieler[i]=false;}` , auch korrekt `if (spieler[i] && (zahlen[i]<bismax)) {spieler[i]=false;}` oder noch kürzer `if (zahlen[i]<bismax) {spieler[i]=false;}`). Im Diagramm sind `MAX_SPIELER` mit `MS`, `zahlen` mit `za`, und `spieler` mit `sp` abgekürzt.



Die Umsetzung in C/C++ (wobei hier das `const` wichtig ist; die Schreibberechtigung auf das Array wäre hier auch ohne das `&` gewährleistet, da C/C++ Arrays automatisch als Pointer übergibt):

```
void markiere_max(const werte zahlen, ist_dabei & spieler) {
  int i;
  int bismax;
  bismax=0;
  for(i=0;i<=MAX_SPIELER-1;i++) {
   if (spieler[i] && (zahlen[i]>bismax)) {
     bismax=zahlen[i];
   }
  }
  if (bismax==0) {
    cout << "Keiner mehr dabei!" << endl;
  } else {
    for(i=0;i<=MAX_SPIELER-1;i++) {
      spieler[i] = spieler[i] && (zahlen[i]==bismax);
    }
  }
}
```

Aufgabe 3 — Eine Datenstruktur



Bei der Umsetzung in C/C++ ist das Vorgehen von unten nach oben zu beachten:

```
struct Datum {
  int tag;
  int monat;
  int jahr;
};
struct Studi {
  char vname[20];
  char nname[30];
  int matnr;
  Datum gebdat;
};
```

Aufgabe 4 — Wissensfragen

Siehe Skript:

- Programm vs. Algorithmus: 1.2.2
- Vorteile modularen Programmierens: 1.6.8

Zu EBNF/Syntaxdiagrammen: diese stellen die Syntax und somit die allgemeine Struktur dar. Eine Umsetzung eines konkreten Beispiels macht hier keinen Sinn. Ziel war eine Darstellung analog zur `if`-Anweisung in der Vorlesung (die Anführungsstriche sind hier wichtig, um zwischen Klammern von C und Klammern der EBNF unterscheiden zu können):

```
<switch-cmd> ::= "switch" "(" <Variablenname> ")" "{"
                    { "case" <Ausdruck> ":" <Kommando> }
                    [ "default" ":" <Kommando> ]
                 "}"
```

Syntaxdiagramm: Iteration (geschweifte Klammern) und optionale Elemente (eckige Klammern) als „Schleifen“ bzw. „Umfahrung“ analog zum Beispiel der `if`-Anweisung im Skript 1.4.

Aufgabe 5 — Einfach verkettete Listen

Der einzige kleine Haken in dieser Aufgabe ist die Berechnung des Mittelwerts (summiert werden `int`-Werte, das Ergebnis ist aber ein `float`-Wert – die Umwandlung erfolgt hier im Beispiel im C++-Stil (d.h., `float(wert)` statt `(float)wert`, was natürlich auch korrekt ist)). Der Rest ist ein einfaches Durchlaufen einer Liste mit einem einfachen Zugriff auf den jeweiligen Wert.

```
struct intListe;
typedef intListenPtr;
struct intListe {
  int value;
  intListenPtr next;
};

void arithm(intListenPtr liste) {
  if (liste==NULL) {
    cout << "Liste ist leer." << endl;
  } else {
    int anzahl=0;
    int summe=0;
    while(liste!=NULL) {
      anzahl++;
      summe+=liste->value;
      liste=liste->next;
    }
    cout << "Das arithmetische Mittel beträgt " << float(summe)/float(anzahl);
  }
}
```

Teilklausur I in Informatik – Nachschreibetermin – TEL08GR1 – Lewandowski

Es sind insgesamt 84 Punkte zu erreichen (80 Punkte werden als 100% gewertet)
(Jeder Punkt entspricht einer Bearbeitungszeit von etwa 1 Minute)

Aufgabe 1: Zahlensummen (12 Punkte) – Struktogramm
Aufgabe 2: Ausgewürfelt (30 Punkte) – DF, PAD/PN, C/C++
Aufgabe 3: Eine Datenstruktur (6 Punkte) – DH, C/C++
Aufgabe 4: `struct` und `typedef` (9 Punkte)
Aufgabe 5: Doppelt verkettete Listen (27 Punkte) – C/C++

Bitte kennzeichnen Sie jedes Blatt lesbar mit Ihrem Namen und Ihrer Matrikelnummer

Zugelassene Hilfsmittel: Keine

Lesen Sie die Aufgaben in Ruhe durch!
Viel Erfolg!

Aufgabe 1 — 12 Punkte

(12 Punkte) **Zahlensummen:** Entwerfen Sie ein kleines Programm, das von einem Benutzer Zahlen einliest und diese summiert, bis diese Summe exakt 42 beträgt. Gibt der Benutzer eine Zahl ein, so dass die Summe größer 42 würde, so soll diese Zahl in der Summe ignoriert werden. Das Programm soll danach ausgeben, wieviel Zahlen vom Benutzer eingegeben wurden.

Beispiel: Der Benutzer gibt folgende Zahlen ein (jeweils getrennt durch die `<RETURN>`-Taste): `20 15 10 5 3 2`, dann soll die Ausgabe lauten:
`Es wurden 6 Zahlen eingegeben.` (20+15=35, die 10 wird zwar bei der Anzahl berücksichtigt, aber in der Summe ignoriert, da 35+10 größer 42 ist, 35+5=40, die 3 wird nochmals in der Summe ignoriert, Abbruch nach Eingabe der 2, da 40+2=42.)

Geben Sie Ihren Entwurf als Struktogramm an (12 Punkte). Der Entwurf soll so detailliert sein, dass eine Umsetzung 1:1 in ein C/C++-Programm möglich wäre, d.h., dass alle Fallunterscheidungen und Schleifen auch im Entwurf als solche vorkommen müssen. Beachten Sie: Eine Umsetzung in C/C++ ist hier *nicht* gefordert.

Aufgabe 2 — 30 Punkte

(2+16+12 = 30 Punkte) **Ausgewürfelt:** In dieser Aufgabe ist eine Funktion `ausgabe_max(.,.)` innerhalb eines Spiels zu entwerfen und in C- oder C++-Code umzusetzen.

Die Funktion soll dabei zwei Arrays als Parameter bekommen: Das eine Array enthält `int`-Werte (die Werte seien hier aus dem Bereich 1 bis 6), ein zweites Array enthält Wahrheitswerte, die angeben, ob der Wert im `int`-Array zu berücksichtigen ist. Die Funktion soll aus dem ersten Array den größten zu berücksichtigenden Wert heraussuchen und die Indizes ausgeben, deren zugehöriger Wert dem zuvor bestimmten Maximalwert entsprechen. (Dies könnte z.B. ein wiederholtes Würfeln in einem Spiel simulieren, bei dem die Spieler nach und nach ausscheiden (nur, wer im zweiten Array markiert ist, ist noch am Spiel beteiligt).)

Gegeben sind die C/C++-Definitionen:

```
const int MAX_SPIELER=7;
typedef int werte[MAX_SPIELER];
typedef bool ist_dabei[MAX_SPIELER];
```

Ein Beispiel: seien zwei Arrays wie folgt definiert:

```
werte     zahlen  = {3,4,5,2,4,1,4};
ist_dabei spieler = {0,1,0,1,1,1,0}; // true entspricht 1, false entspricht 0
```

Dann soll der Aufruf `ausgabe_max(zahlen,spieler)` die Indizes `1` und `4` ausgeben (4 ist der größte zu berücksichtigende Wert (`zahlen[2]` ist zwar 5, aber `spieler[2]` ist 0, ebenso ist `6` nicht auszugeben, da `spieler[6]` 0 ist). Sind alle Einträge im Array `spieler[]` gleich 0, so soll die Ausgabe „`Keiner mehr dabei!`“ erfolgen.

Geben Sie das Datenflussdiagramm (DF) der Ebene 0 an (Funktion als ein Kästchen) (2 Punkte), sowie einen Programmablaufplan mit Daten (PAD) oder ein Programmnetz (PN) für die von Ihnen entworfene Funktion mit den nötigen Verfeinerungen (16 Punkte). Der Entwurf soll so detailliert sein, dass eine Umsetzung 1:1 in ein C/C++-Programm möglich ist, d.h., dass alle Fallunterscheidungen und Schleifen auch im Entwurf als solche vorkommen müssen. Beachten Sie: die Parameter sind an die Funktion zu übergeben und werden von dieser zurückgeliefert, Sie müssen sich *nicht* um die Ein- und Ausgabe der Werte kümmern. Geben Sie schließlich noch die Umsetzung der Funktion in C/C++ an (12 Punkte).

Aufgabe 3 — 6 Punkte

(3+3 = 6 Punkte) In dieser Aufgabe ist eine **Datenstruktur** zu entwerfen, als Datenhierarchiediagramm darzustellen und in C/C++-Code umzusetzen. Die Datenstruktur soll zur Verwaltung der Vorlesungen dienen und soll dabei die Daten für eine Vorlesung beinhalten. Folgende Daten sollen gespeichert werden: Titel und Dauer (in Minuten) der Vorlesung, Vor- und Nachname des Dozenten.

Geben Sie ein Datenhierarchiediagramm für Ihre Datenstruktur an (3 Punkte) und geben Sie die Umsetzung in C/C++-Code an (3 Punkte).

Aufgabe 4 — 9 Punkte

Was bewirken jeweils die folgenden Zeilen C/C++-Code? (1–2 Sätze reichen jeweils)

- Variante 1:

```
struct Datum {
  int tag;
  int monat;
  int jahr;
};
```

- Variante 2:

```
struct {
  int tag;
  int monat;
  int jahr;
} Datum;
```

- Variante 3:

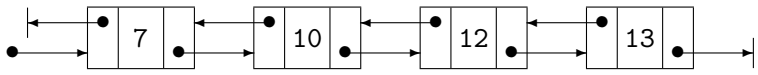
```
typedef struct {
  int tag;
  int monat;
  int jahr;
} Datum;
```

Aufgabe 5 — 27 Punkte

(5+20+2 = 27 Punkte) **Doppelt verkettete Listen:** In dieser Aufgabe beschäftigen wir uns mit doppelt verketteten Listen, der Anker zeige dabei auf das erste Element. Bei doppelt verketteten Listen hat jedes Listenelement sowohl einen Zeiger auf das nachfolgende als auch auf das vorhergehende Element der Liste.

1. (5 Punkte) Geben Sie eine C- oder C++-Typdefinition für eine doppelt verkettete Liste aus `int`-Zahlen an. Der Datentyp für den Anker soll dabei `intListenPtr` heißen.

2. (20 Punkte) Schreiben Sie eine C/C++-Funktion `void loesche(intListenPtr & liste, const int i)`, die als Parameter den Anker auf eine doppelt verkettete Liste und eine ganze Zahl bekommt. Hat die Liste weniger als `i` Elemente, so soll die Meldung „`Liste ist kürzer.`“ ausgegeben werden, ansonsten soll das `i`-te Element aus der Liste gelöscht werden. Zu der Liste

   (der Pointer `anker` zeige auf die `7`) soll beim Aufruf `loesche(anker,3)` die `12` entfernt werden – beim Aufruf `loesche(anker,5)` soll die Ausgabe lauten: „`Liste ist kürzer.`“

   Hinweis: Beachten Sie beim Löschen die Randfälle! Das zu löschende Element könnte keinen Nachfolger und/oder keinen Vorgänger haben.

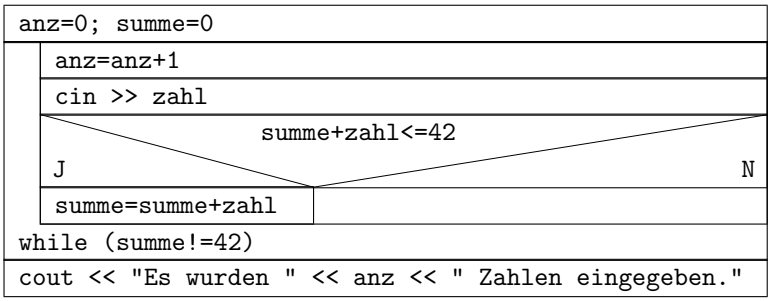
3. (2 Punkte) Erläutern Sie kurz (1–2 Sätze genügen), für welchen Fall der Parameter `liste` als Referenzparameter („`&`“) übergeben werden muss.

Beachten Sie: die Parameter sind an die Funktion zu übergeben, Sie müssen sich *nicht* um die Eingabe der Werte kümmern.

Teilklausur I in Informatik – Nachschreibetermin – Lösungshinweise

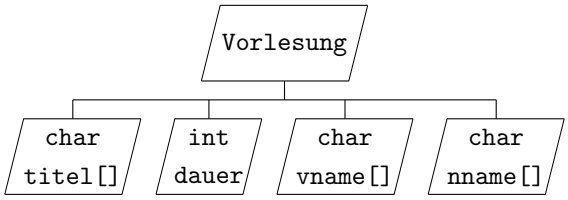
Aufgabe 1 — Zahlensummen

Bei der Bearbeitung ist unerheblich, ob die Befehle (wie hier) C-Syntax haben oder Pseudo-Code sind.



Mit einer `while`-Schleife (anstatt `do {...} while`) geht's natürlich auch.

Aufgabe 3 — Eine Datenstruktur

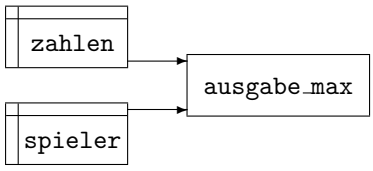



```
struct Vorlesung {
  char titel[20];
  int dauer;
  char vname[20];
  char nname[30];
};
```

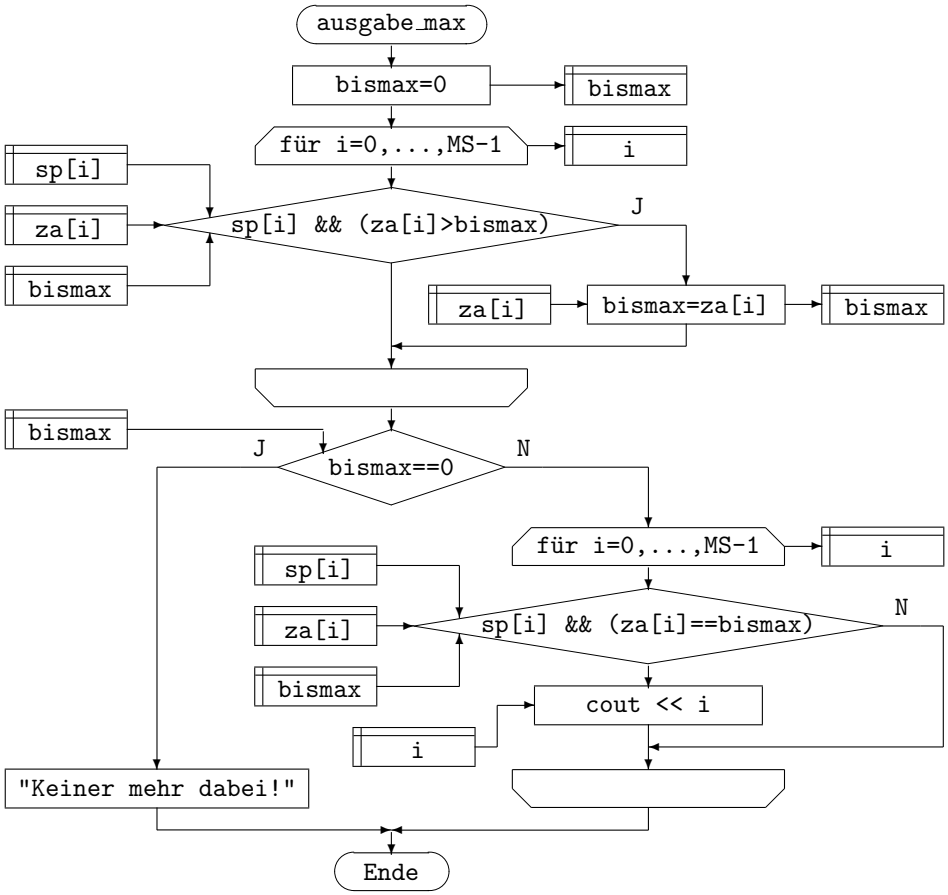
Aufgabe 2 — Ausgewürfelt

Zwei Schritte: Finde Maximalwert (oder stelle fest, dass keiner mehr mitspielt), gib die entsprechenden Indizes aus. Da die Komplexität der beiden Teile sehr gering ist, ist eine Aufteilung nicht zwingend erforderlich.

Das Datenflussdiagramm:



Ein Programmablaufplan mit Daten (beim Programmnetz wäre pro Variable nur ein Datensymbol vorhanden): Da vorab bekannt ist, dass die Werte im Array `zahlen` alle zwischen 1 und 6 liegen, kann hier das Maximum mit 0 initialisiert werden und so auch erkannt werden, wenn keiner mehr mitspielt. Im Diagramm sind `MAX_SPIELER` mit `MS`, `zahlen` mit `za`, und `spieler` mit `sp` abgekürzt.



Die Umsetzung in C/C++:

```
void ausgabe_max(const werte zahlen, ist_dabei & spieler) {
  int i;
  int bismax;
  bismax=0;
  for(i=0;i<=MAX_SPIELER-1;i++) {
   if (spieler[i] && (zahlen[i]>bismax)) {
     bismax=zahlen[i];
   }
  }
  if (bismax==0) {
    cout << "Keiner mehr dabei!" << endl;
  } else {
    for(i=0;i<=MAX_SPIELER-1;i++) {
      if (spieler[i] && (zahlen[i]==bismax)) {
       cout << i;
      }
    }
  }
}
```

Aufgabe 4 — `struct` und `typedef`

Variante 1 und 3 erzeugen einen Datentyp `Datum` (wobei bei Variante 3 der Datentyp in C und C++ gleichermaßen direkt verwendet werden kann, während bei Variante 1 in C eine Variable mittels `struct Datum <varname>;` angelegt werden muss), Variante 2 erzeugt hingegen nur eine Variable `Datum`. (siehe auch im Skript Abschnitt 1.6.6)

Aufgabe 5 — Doppelt verkettete Listen

Die Hauptschwierigkeit in dieser Aufgabe ist die Beachtung der Randfälle (der Rest ist ein einfaches Durchlaufen einer Liste, wobei die Elemente gezählt werden). Hat das zu löschende Element einen Vorgänger, so wird dessen Nachfolger auf den Nachfolger des zu löschenden Elements gesetzt (insbesondere auch, wenn dieser `NULL` ist); analog, wenn es einen Nachfolger gibt. Ist das zu löschende Element das erste der Liste (es hat dann keinen Vorgänger), wird der Anker verändert – daher ist der Parameter als Referenzparameter zu übergeben.

```
struct intDListe;
typedef intListenPtr;
struct intDListe {
  int value;
  intListenPtr prev, next;
};

void loesche(intListenPtr & liste, const int i) { // i muss >0 sein
  int anzahl=1;
  intListenPtr lst=liste;
  while(lst!=NULL && anzahl!=i) {
    anzahl++;
    lst=lst->next;
  } // lst zeigt jetzt auf das i-te Element (falls vorhanden)
  if (anzahl!=i || lst==NULL) { // die Liste könnte i-1 Elemente haben!
    cout << "Liste ist kürzer.";
  } else {
    intListenPtr tbd=lst;
    if (lst->next!=NULL) {
      lst->next->prev=lst->prev;
    }
    if (lst->prev!=NULL) {
      lst->prev->next=lst->next;
    } else { // das zu löschende Element ist das erste der Liste!
      liste=lst->next; // Anker der Liste anpassen
    }
    delete tbd;
  }
}
```