Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Dr. K. Simon

Studiengang Maschinenbau und Verfahrenstechnik

1. Vordiplom, Informatik I

Frühling 2002

Freitag, 8. März 2002

Name: _____

Vorname: _____

Legi-Nummer: _____

Unterschrift: _____

Aufgabe	Maximale Punktzahl	Erreichte Punktzahl	Visum
1. Syntax, Typen, Datenstrukturen	30		
2. Logik	26		
3. Funktionen und Prozeduren	27		
4. Rekursion	30		
5. Listen	29		
Total	142		
Note			

Allgemeine Hinweise

- Zugelassene Hilfsmittel: **keine**.
- Legen Sie zu Anfang der Prüfung Ihre Legi neben sich auf den Tisch.
- Die Prüfung besteht aus 5 Aufgaben. Kontrollieren Sie, ob Sie alle Aufgaben erhalten haben.
- Benutzen Sie keine rote Farbe!
- Verwenden Sie für jede Aufgabe ein separates Blatt.
- Schreiben Sie auf jedes Blatt Ihre Legi-Nummer (und nur diese!).
- Schreiben Sie deutlich, und benützen Sie keinen Bleistift.
- Pro Aufgabe darf höchstens ein gültiger Lösungsversuch abgegeben werden. Ungültige Lösungsversuche müssen klar durchgestrichen sein.
- In jeder Aufgabenstellung ist eine Lösungsstruktur angegeben. Nichtbeachten der Lösungsstruktur führt zu Punktabzug.
- Es empfiehlt sich *unbedingt*, zuerst alle Fragen durchzulesen.
- Abzugeben sind das vollständig ausgefüllte Deckblatt, die Aufgabenblätter und Ihre Lösungen.

Aufgabe 1: Syntax, Typen, Prozeduren und Funktionen *(30 Punkte)*

a) In der folgenden Aufgabe testen wir anhand einer typischen Situation beim Programmieren (wie Sie sie oft genug bei der Lösung Ihrer Übungen erlebt haben) Ihr Vermögen, Fehler zu finden, zu identifizieren und zu korrigieren. Unterstützt werden Sie dabei durch Informationen, die Ihnen der Computer gibt.

Worum geht's? Jedes der folgenden Programmstücke sollte eigentlich $3^7 = 2187$ berechen. In jedem Programmstück ist aber genau ein Fehler enthalten. Dieser Fehler tritt entweder beim Kompilieren oder bei der Ausführung auf. Wir geben Ihnen die Fehlermeldung des Compilers oder die Fehlermeldung bei der Ausführung an.

Aufgabe Kennzeichnen Sie in den folgenden Programmen, wo der Fehler genau auftritt. Beschreiben Sie, was der Fehler ist, und geben Sie einen Verbesserungsvorschlag an, der das Programm *auf irgendeine Weise fehlerlos* macht.

Lösungsstruktur Ihre Lösung enthält (1) die tatsächliche Zeilenangabe des Fehlers, (2) eine Beschreibung des Fehlers in Ihren eigenen Worten, (3) den Verbesserungsvorschlag.

Massstab Bewertet werden je Unteraufgabe die korrekte Angabe der Zeile (0.5 Punkte), die korrekte Beschreibung des Fehlers (0.5 Punkte), ein korrekter Verbesserungsvorschlag (1 Punkt). Jede Unteraufgabe gibt 2 Punkte. Die gesamte Teilaufgabe ist 8 Punkte wert.

i)
```
1   #include <iostream.h>
2
3   int main()
4   {
5     int a,b,c;
6
7     a= 3; b= 7; c= 1;
8     while (b>0) {
9       c= c*a; b--
10    }
11    cout << c << endl;
12  }
```

Fehler beim Kompilieren

```
syntax1.C: In function ‘int main()’:
syntax1.C:10: parse error before ‘}’
```

ii)
```
1   #include <iostream.h>
2
3   int main()
4   {
5     int a,b,c[8];
6
7     a= 3; b= 7; c[b]= 1;
8     while (b>0) {
9       c[b-1= c[b]*a; b--;
10    }
11    cout << c[0] << endl;
12  }
```

Fehler beim Kompilieren

```
syntax2.C: In function ‘int main()’:
syntax2.C:9: non-lvalue in assignment
syntax2.C:9: parse error before ‘;’
```

iii)
```
1   #include <iostream.h>
2
3   int main()
4   {
5     int *a,*b,*c;
6
7     a= new(int); b= new(int); c= new(int);
8
9     *a= 3; *b= 7; *c= 1;
10    while ((*b)>0) {
11      *c= (*c)*(*a);
12      b--;
13    }
14    cout << *c << endl;
15
16    delete(a); delete(b); delete(c);
17  }
```

Semantischer Fehler Ausgegeben wird 3, eigentlich sollte aber 2187 ausgegeben werden.

iv)
```
1   #include <iostream.h>
2
3   int main()
4   {
5     int a,b,c;
6
7     a= 3; b= 7; c= 1;
8     while (b>0) {
9       c= c*a;
10      a--;
11    }
12    cout << c << endl;
13  }
```

Endlosschleife Das Programm wird nicht beendet. Es wird nichts ausgegeben.

$\Longrightarrow$

b) In der Vorlesung haben Sie gelernt, wie man Variablen, deklariert. In dieser Aufgabe sollen Sie diese Fähigkeit demonstrieren.

Aufgabe Wir präsentieren Ihnen zwei korrekt codierte C++-Programmstücke (i-ii), aus denen hervorgeht, welchen Typ die „Namen“ a, b und c haben. Sie sollen für jedes Programmstück diese Programmelemente deklarieren.

Lösungsstruktur Wir erwarten uns pro Programmteil je eine Deklaration für *alle* a, b und c in korrekter C++-Syntax. Für Feldgrössen setzen sie irgendeine Zahl ein.

Beispiel

```
1   a= 3;
2   b= '2';
3   c= (char(a)=='2');
```

Dieser Programmteil verlangt folgende Deklaration der Objekte a, b, und c:

```
1   int a;
2   char b;
3   bool c;
```

Massstab Bewertet werden nur *notwendige* Deklarationen. Korrekte einfache Datentypen erhalten 1 Punkt. Korrekte erweiterte Datentypen werden mit bis zu 3 Punkten bewertet. Die Teilaufgabe ist 10 Punkte wert.

Die Programmstücke

i)
```
1   a= &b;
2   b= c[12] || (2>=1);
```

ii)
```
1   a= b.c*3.1415;
2   b.a= char(b.c%2) == c;
```

c) Die folgende Aufgabe behandelt Ihr Wissen im Bereich Prozeduren und Funktionen.

Aufgabe Ergänzen Sie auf sinnvolle Art die Kopfzeile der folgenden C++-Routinen und geben Sie einen sinnvollen Beispielaufruf an.

- Nehmen Sie dabei an, dass keine globalen Variablen definiert sind. Die Übergabe von Parametern erfolgt allein über den Aufruf.
- Wo eine Rückgabe sinnvoll erscheint (weil sonst Werte verloren gehen), *muss* eine Rückgabe erfolgen. Der gegebene Anweisungsteil der Routine darf dabei aber nicht abgeändert werden.
- Falls für eine verwendete Variable kein eindeutiger Typ bestimmbar ist, wählen Sie einen der Kandidaten aus. Achten Sie aber auf die Schlüssigkeit (Konsistenz) Ihrer Lösung.

Lösungsstruktur Ihre Antwort enthält *eine Kopfzeile* für die gegebene C++-Routine und *einen Aufruf* mit Deklaration aller verwendeten Variablen. Für den Aufruf müssen die übergebenen Variablen nicht initialisiert werden. Gegebenenfalls sind auch *Typen zu deklarieren*.

Bewertung Bewertet wird jeder korrekte Parameter (1 Punkt), die korrekte Einordnung und Handhabung von Funktion/Prozedur (1 Punkt), jede notwendige und korrekte Typendeklaration (1 Punkt) und jede notwendige und korrekte Variablendeklaration (0.5 Punkte). Die ganze Unteraufgabe ist 12 Punkte wert.

Die C++-Routinen

i)
```
1   ... skalarprodukt(...)
2   {
3     int i;
4
5     erg= 0.0;
6     for (i=0; i<n; i++) {
7       erg= erg+vec1[i]*vec2[i];
8     }
9   }
```

ii)
```
1   ... arg(...)
2   {
3     double winkel;
4     if (zahl.Re != 0) {
5       winkel= acos(zahl.Im/zahl.Re);
6     } else if (zahl.Im>0) {
7       winkel= 3.141592653589793/2.0;
8     } else {
9       winkel= -3.141592653589793/2.0;
10    }
11    return(winkel);
12  }
```

Aufgabe 2: Logik *(26 Punkte)*

In dieser Aufgabe werden Ihre Fähigkeiten bezüglich Logik geprüft. Sie müssen aus logischen Ausdrücken Wahrheitstabellen erstellen und logische Ausdrücke umformen.

Das Problem Sie haben bereits gelernt, dass die UND-Verknüpfung und ODER-Verknüpfung zusammen mit der Negation ausreichen, um jeden logischen Ausdruck zu formulieren. Es geht jedoch noch besser, nämlich mit einer einzien Operation. Diese heisst NAND, wird mit dem Zeichen $\bar{\wedge}$ dargestellt und ist durch $A \mathbin{\bar{\wedge}} B \stackrel{\text{def}}{=} !(A \&\& B)$ definiert.

Ihre Wahrheitstabelle ist

A	B	$A \bar{\wedge} B$
0	0	1
0	1	1
1	0	1
1	1	0

a) Formulieren Sie

i) die Negation ($!A$)
ii) die UND-Verknüpfung ($A \;\&\&\; B$)
iii) die ODER-Verknüpfung ($A \;||\; B$)

unter *alleiniger* Verwendung von NAND.

Lösungsstruktur Pro Unteraufgabe erwarten wir einen zu $!A$, $A \;\&\&\; B$ und $A \;||\; B$ äquivalenten Ausdruck in den Variablen A und B, der ausschliesslich die Verknüpfung $\bar{\wedge}$ enthält.

Massstab Für die korrekte Formulierung der Negation gibt es 1 Punkt, für die UND-Verknüpfung 3 Punkte, für die ODER-Verknüpfung 4 Punkte. Die Unteraufgabe ist 8 Punkte wert.

b) Gegeben ist folgender logischer Ausdruck in den Variablen A, B und C:

$$(A \;\&\&\; B \;\&\&\; !C) \;||\; (A \;\&\&\; !B \;\&\&\; C) \;||\; (A \;\&\&\; !B \;\&\&\; !C)$$

i) Stellen Sie zu diesem Ausdruck eine Wahrheitstabelle auf.

Lösungsstruktur Ihre Lösung enthält eine vollständige Wahrheitstabelle.

Massstab Bewertet werden die Zeilen sowie das Layout der Tabelle. Die Aufgabe ist 9 Punkte wert.

ii) Vereinfachen Sie den Ausdruck aufgrund der Wahrheitstabelle.

Lösungsstruktur Wir erwarten einen möglichst kurzen logischen Ausdruck in den gegebenen Variablen, der die Operationen && ||, und ! enthalten darf und zum gegebenen Ausdruck äquivalent ist.

Massstab Bewertet werden die Klammerung und der korrekte Einsatz der Verknüpfungen. Für die korrekte Lösung gibt es 4 Punkte.

iii) Formulieren Sie den vereinfachten oder gegebenen Ausdruck unter alleiniger Verwendung von NAND.

Lösungsstruktur Wir erwarten einen logischen Ausdruck in den gegebenen Variablen, der ausschliesslich die Operation $\bar{\wedge}$ enthält.

Massstab Jede korrekte NAND Verknüpfung und jede korrekte Klammerung ist 1 Punkt wert. Für die vollständige, korrekte Umformulierung erhalten sie 5 Punkte.

Aufgabe 3: Funktionen und Prozeduren *(27 Punkte)*

In dieser Aufgabe geht es um Funktionen und Prozeduren. Sie zeigen in der Aufgabe, dass Sie Funktionen und Prozeduren definieren und aufrufen können. Für die Aufgaben b) und c) brauchen Sie zudem Ihr Wissen über erweiterte Datenstrukturen.

Pythagoräische Tripel Ein Tripel (a, b, c) von natürlichen Zahlen heisst pythagoräisches Tripel, wenn gilt:

$$a^2 + b^2 = c^2.$$

Beispiel $3^2 + 4^2 = 5^2$

a) Implementieren Sie eine C++-Funktion `TestTripel`, welche für zwei natürliche Zahlen a und b testet, ob eine natürliche Zahl c existiert, so dass (a, b, c) ein pythagoräisches Tripel ist. Wenn (a, b, c) ein pythagoräisches Tripel ist, soll die Funktion die gefundene Zahl c zurückgeben. Anderenfalls soll der Wert 0 zurückgegeben werden.

Hinweis: Für den verlangten Test brauchen Sie die Quadratwurzel. Verwenden Sie dafür die Funktion

```
double sqrt(double z),
```

welche den Wert von $\sqrt{z}$ zurückgibt.

Lösungsstruktur Erwartet wird eine vollständige, mit Kommentaren versehene C++-Funktion, welche den verlangten Test durchführt.

Massstab Bewertet wird:

- korrekter Umgang mit Parametern: 4 Punkte
- korrekter Anweisungsteil: 3 Punkte
- Kommentare: 1 Punkt

Die Unteraufgabe ist 8 Punkte wert.

b) Entwerfen Sie in C++-Syntax eine Datenstruktur, in der genau ein pythagoräisches Tripel gespeichert werden soll.

Lösungsstruktur Abzugeben ist eine in C++-Syntax formulierte Datenstruktur, die mit Kommentaren versehen ist.

Massstab Bewertet wird der Einsatz einer sinnvollen Datenstruktur (1 Punkt), der richtige Typ (1 Punkt) und Kommentare (1 Punkt). Die Teilaufgabe ist insgesamt 3 Punkte wert.

c) Implementieren Sie eine C++-Prozedur `PythTripel`, welche alle pythagoräischen Tripel (a, b, c) mit $a < b \leq n$ in einem Feld `p` von demjenigen Datentyp abspeichert, den Sie in Aufgabe b) entworfen haben. Die Zahl n soll dabei als Parameter übergeben werden. Für die Berechnung der Tripel soll die Funktion `TestTripel` aus Aufgabe a) verwendet werden.

Lösungsstruktur Abzugeben ist eine vollständige und kommentierte C++-Prozedur, welche mit der Funktion aus Aufgabe a) arbeitet und die gestellte Aufgabe erfüllt.

Massstab Bewertet wird

- korrekter Umgang mit den Parametern: 3 Punkte
- korrekte Verwendung des Feldes: 3 Punkte
- korrekter Aufruf der Funktion aus Aufgabe a): 1 Punkt
- algorithmische Korrektheit: 8 Punkte
- Kommentare: 1 Punkt

Die Unteraufgabe ist 16 Punkte wert.

Aufgabe 4: Rekursion *(30 Punkte)*

In dieser Aufgabe prüfen wir am Beispiel der Zerlegung einer Zahl, ob Sie in der Lage sind, einen rekursiven Zusammenhang zu erkennen und ihn in einen Algorithmus umzusetzen.

Das Problem Additionen verschiedener natürlicher Zahlen können zum selben Ergebnis führen. So ergeben z.B. drei verschiedene additive Rechenvorschriften $1+1+1+1+1+1$, $3+2+1$ und $3+3$ alle die Zahl 6. Wir nennen solche verschiedenen Additionen, die alle die gleiche Summe n ergeben, eine *Zerlegung* der positiven ganzen Zahl n.

Beispiele von Zerlegungen

Alle Zerlegungen von 1**:** 1

Alle Zerlegungen von 2**:** $1+1$, 2

Alle Zerlegungen von 3**:** $1+1+1$, $1+2$, $2+1$, 3

Die Aufgaben

a) Geben Sie alle Zerlegung der Zahl 4 an. Überlegen Sie sich dazu ein *rekursives* Vorgehen, mit dem Sie die Gewissheit haben, dass sie keine Zerlegung vergessen. Es ist Teil der Aufgabe herauszufinden, wieviele Zerlegungen es gibt. Achten Sie ausserdem darauf, dass innerhalb einer Zerlegung die Reihenfolge der Zahlen eine Rolle spielt (wie beim Beispiel der Zahl 3: $1+2$ und $2+1$ sind zwei Zerlegungen).

Lösungsstruktur Anzugeben ist in drei bis vier eigenen Sätzen eine Beschreibung eines rekursiven Vorgehens, nach dem Sie alle Zerlegungen der Zahl 4 finden wollen, sowie die Zerlegungen in der Reihenfolge Ihrer beschriebenen Vorgehensweise.

Massstab Bei der Beschreibung Ihres Vorgehens legen wir Wert auf die korrekte Behandlung des Rekursionsschrittes (max. 2 Punkte), die Abbruchbedingung (1 Punkt) sowie die Vollstaendigkeit der Beschreibung (1 Punkt). Sowohl korrekte Zerlegungen als auch die Reihenfolge gemäss Ihrer Beschreibung wird insgesamt mit bis zu 5 Punkten bewertet. Die Unteraufgabe ist 9 Punkte wert.

b) Entwerfen Sie eine rekursive Prozedur,

```
void zerlege(int zahl, int zerlegung[], int anzahl)
```

die zu einer gegebenen positiven ganzen Zahl `zahl` alle Zerlegungen auf dem Bildschirm ausgibt. Das Array `zerlegung` kann dabei zur Speicherung der ersten `anzahl` Summanden des aktuellen Zwischenresultats dienen.

Lösungsstruktur Abzugeben ist eine *vollständige, kommentierte* C++-Prozedur, welche die Aufgabenstellung erfüllt. Versehen Sie Ihre Prozedur mit Kommentaren.

Massstab Wir legen Wert auf die Behandlung der Rekursion (max. 6 Punkte) und die Lösung des Problems (max. 2 Punkte). Sinnvolle Kommentare werden mit 1 Punkt bewertet. Die Unteraufgabe ist insgesamt 9 Punkte wert.

c) Verändern Sie die rekursive Prozedur aus Aufgabe b) so, dass sie Zerlegungen nur einmal ausgibt, welche sich ausschliesslich durch die Reihenfolge der Summanden unterscheiden. Für `zahl` gleich 3 gibt diese Prozedur also beispielsweise nur $1+1$, $1+2$ und $2+2$ aus, *nicht aber* $2+1$.

Lösungsstruktur Abzugeben ist eine *vollständige, kommentierte* C++-Prozedur, die die Aufgabenstellung erfüllt. Versehen Sie Ihre Prozedur mit Kommentaren.

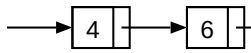
Massstab Wir legen Wert auf die Behandlung der Rekursion (max. 7 Punkte) und die Lösung des Problems (max. 4 Punkte). Sinnvolle Kommentare werden mit 1 Punkt bewertet. Die Unteraufgabe ist 12 Punkte wert.

Aufgabe 5: Dynamische Datenstrukturen: Listen *(29 Punkte)*

In dieser Aufgabe geht es um Listen und Prozeduren. Diese Konzepte sollen Sie im Zusammenhang mit der Repräsentation ganzer Zahlen und deren Addition einsetzen.

Das Problem Nicht negative ganze Zahlen sollen als Listen dargestellt werden. Das hat den Vorteil, dass Zahlen beliebig lange sein können. Ein Listenelement nimmt dabei nur eine Ziffer der zu speichernden Zahl auf.

Hinweis: Um die folgenden Teilaufgaben einfacher lösen zu können, wird die führende Ziffer als letztes Element der Liste gespeichert.

D.h. 

$\hat{=} 4 \cdot 10^0 + 6 \cdot 10^1 = 64.$

a) Entwerfen Sie in C++ eine listenartige Datenstruktur für die Speicherung der Ziffern beliebig langer, nicht negativer ganzer Zahlen.

Lösungsstruktur Abzugeben ist ein Datentyp in C++-Code, der eine beliebig lange, nicht negative ganze Zahl als Liste speichern kann. Spezifizieren Sie das Aussehen einer Ziffer. Geben Sie ausserdem an, wie die gesamte Zahl repräsentiert wird. Verwenden Sie aussagekräftige Namen und versehen Sie Ihren Code mit Kommentaren.

Massstab Wir legen Wert auf aussagekräftige Namen (1.5 Punkte), korrekte Datenstruktur (2.5 Punkte), Umgang mit Zeigern (1 Punkt) und Kommentare (1 Punkt). Die Teilaufgabe gibt 6 Punkte.

b) Erläutern Sie *zeichnerisch* und schrittweise die Addition zweier Zahlen 79 und 53, die in der von Ihnen definierten Struktur gespeichert sind. Das Ergebnis der Addition soll dabei in der ersten Zahl gespeichert werden.

Lösungsstruktur Abzugeben ist eine Folge von Zeichnungen, deren Reihenfolge gekennzeichnet ist. Zeichnen Sie für jede Änderung in einer Liste ein eigenes Bild, erläutern Sie die Veränderungen und geben Sie ggf. den Wert von Hilfsvariablen an.

Massstab Bewertet wird: Veränderungen der Liste (4 Punkte), die Verwendung von Hilfsvariablen (3 Punkte) und die schlussendliche Veränderung zum Endergebnis (1 Punkt). Die Teilaufgabe ist 8 Punkte wert.

c) Entwerfen Sie eine C++-Prozedur zur Addition von zwei *gleich langen* Zahlen. Die Prozedur verändert den ersten Parameter: Er enthält nach dem Aufruf das Ergebnis der Addition.

Lösungsstruktur Abzugeben ist eine C++-Prozedur, die die gegebene Aufgabe erfüllt. Versehen Sie Ihre Prozedur mit Kommentaren.

Massstab Bewertet werden der Umgang mit der dynamischen Datenstruktur und Zeigervariablen (7 Punkte), Problemlösung (6 Punkte), Prozedurschnittstelle (1 Punkt) und Kommentare (1 Punkt). Die gesamte Teilaufgabe ist 15 Punkte wert.

***** *Viel Erfolg!* *****

lj, März 2002