Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Dr. K. Simon

Studiengang Maschinenbau und Verfahrenstechnik

1. Vordiplom, Informatik I

Musterlösung

Frühling 2001

Freitag, 9. März 2001

Aufgabe 1: Fehlermeldungen, Typen & Konstanten, Kontrollstrukturen *(24 Punkte)*

a) i)

1. Der Fehler tritt in Zeile 7 auf.
2. Eine schliessende Klammer („}“) wurde am Ende vergessen.
3. `if (j%2==0) { cout << i << endl; }`

ii)

1. Der Fehler tritt in Zeile 7 auf.
2. Es passiert eine Division durch 0 in der if-Bedingung.
3. `if (j%2==0) { cout << i << endl; }`

iii)

1. Der Fehler tritt in Zeile 11 auf.
2. Die runden Klammern bei `p.m(0)` sind falsch. Sie sollten eckig sein.
3. `p.m[0]= 'M'; p.m[1]='a';`

iv)

1. Der Fehler tritt in Zeile 10 auf.
2. Es wurde versucht auf ein Element von `p` zuzugreifen. Aber `p` existiert gar nicht.
3. Zwischen Zeile 9 und Zeile 10 eine neue Zeile mit: `p= new(Datum)`.

b) i)

```
1  char a[100];
2  double b;
3  char c;
```

ii)

```
1  struct { bool a; double c; } a;
2  int c;
3  char b;
```

c) i)

```
1  i= 0;
2  while (i<b) {
3    a= a*a;
4    i++;
5  }
```

ii)

```
1  i= 0;
2  if (i<b) {
3    do {
4      a= a*a;
5      i++;
6    } while (i<b);
7  }
```

Aufgabe 2: Prozeduren und Funktionen: Komplexe Zahlen *(25 Punkte)*

a) **Addition**

```
1   Complex add(Complex a, Complex b)
2     // Addiert die komplexen Zahlen a und b und gibt das Ergebnis zurueck
3   {
4     Complex c;                      // Ergebniswert
5     c.real= a.real+b.real;          // Realteil der Addition
6     c.imag= a.imag+b.imag;          // Imaginaerteil der Addition
7     return(c);                      // Ergebnis zurueckgeben
8   }
```

Aufruf

```
1   Complex a,b,c;
2
3   a.real= 1; a.imag= 1;
4   b.real= 1; b.imag= 1;
5
6   c= add(a,b);
7
8   // c.real= 2; c.imag= 2;
```

b) **Multiplikation, Variante 1**

```
1   void mult(Complex a, Complex b, Complex &c)
2     // Multipliziert die komplexen Zahlen a und b und gibt das Erg. in c zurueck
3   {
4     c.real= a.real*b.real-a.imag*b.imag;  // Realteil gemaess Formel berechnen
5     c.imag= a.real*b.imag+b.real*a.imag;  // Imaginaerteil nach Formel berechnen
6   }
```

Aufruf, Variante 1

```
1   Complex a,b,c;
2
3   a.real= 2.0; a.imag= 3.0;
4   b.real= 1.0; b.imag= 2.0;
5
6   mult(a,b,c);
7
8   // c.real= -4.0; c.imag= 7.0;
```

Multiplikation, Variante 2

```
1   void mult(Complex a, Complex &b)
2     // Multipliziert die komplexen Zahlen a und b und gibt das Erg. in b zurueck
3   {
4     double temp;                          // tempoerare Variable
5
6     temp= b.real;                         // Realteil merken
7     b.real= a.real*b.real-a.imag*b.imag;  // Realteil gemaess Formel berechnen
8     b.imag= a.real*b.imag+temp*a.imag;    // Imaginaerteil nach Formel berechnen
9   }
```

Aufruf, Variante 2

```
1   Complex a,b;
2
3   a.real= 2.0; a.imag= 3.0;
4   b.real= 1.0; b.imag= 2.0;
5
6   mult(a,b);
7
8   // b.real= -4.0; b.imag= 7.0;
```

c)
- Ein funktionaler Aufruf entspricht mehr der mathematischen Konvention als ein prozeduraler Aufruf.
- Für die funktionale Version muss eine lokale Variable deklariert werden.
- Bei der prozeduralen Version kann ein Eingabeparamter auch ein Ausgabeparameter sein
- Eine Funktion muss immer mit `return` einen Rückgabewert zurückgeben.

Aufgabe 3: Records und Felder *(28 Punkte)*

a) i)

```
1   struct Beziehung {    // gibt Beziehung zwischen Einheiten e1 und e2 an.
2     double z1;          // Masszahl der ersten Einheit
3     int e1;             // Kodierung der ersten Einheit: z.B. 1=cm, 2=m, 3=inch ...
4     double z2;          // Masszahl der zweiten Einheit
5     int e2;             // Kodierung der zweiten Einheit
6   };
```

ii)

```
1   Beziehung umwandlung(Beziehung b1, Beziehung b2, double einheit1)
2   {
3      Beziehung erg;
4
5      if (b1.e1==b2.e1) {
6        erg.z2= einheit1;
7        erg.z1= einheit1*b1.z1/b1.z2*b2.z2/b2.z1;
8      } else if (b1.e2 == b2.e2) {
9        erg.z1= einheit1;
10       erg.z2= einheit1*b1.z2/b1.z1*b2.z1/b2.z2;
11     } else if (b1.e2 == b2.e1) {
12       erg.z1= einheit1;
13       erg.z2= einheit1*b1.z2/b1.z1*b2.z2/b2.z1;
14     } else {  // (b1.e1 == b2.e2)
15       erg.z2= einheit1;
16       erg.z1= einheit1*b1.z1/b1.z2*b2.z1/b2.z2;
17     }
18     return(ergebnis);
19  }
```

⟹

b)
```
1   void vollstaendig(double tabelle[N][N])
2   {
3     int i,j;                                          // Zaehlvariablen
4
5     for (j= 1; j<N; j++) {                            // Fuer alle restl. Spalten
6       for (i= 0; i<N; i++) {                          // Fuer alle Zeilen
7         if (i<j) {                                    // oberes Dreieck:
8           tabelle[i][j]= 1.0/tabelle[j][i];           // Reziprokwert
9         } else if (i==j){                             // Diagonale:
10          tabelle[i][j]= 1.0;                         // 1
11        } else {                                      // unteres Dreieck:
12          tabelle[i][j]= tabelle[i][0]*tabelle[0][j]; // gegegebene Formel
13        }
14      }
15    }
16  }
```

Aufgabe 4: Dynamische Datenstrukturen *(22 Punkte)*

a)
```
1   struct Polygon {  // Datenstruktur (zyklisch) fuer ein Polygon
2     double x,y;     // Koordinaten des Punktes
3     Polygon *next;  // Naechster Punkt
4   };
```

b)
```
1   void Divide(Polygon *p1, Polygon *p2)
2   {
3     Polygon *t1, *t2;
4
5     t1= new(Polygon);    // Neuer Punkt fuer p1
6     t1->x= p2->x;        // p2-Koordinaten kopieren
7     t1->y= p2->y;
8     t1->next= p2->next;  // Nachfolger von p2 kopieren
9
10    t2= new(Polygon);    // Neuer Punkt fuer p2
11    t2->x= p1->x;        // p1-Koordinaten kopieren
12    t2->y= p1->y;
13    t2->next= p1->next;  // Nachfolger von p1 kopieren
14
15    p1->next= t1;        // t1 in p1 einhaengen
16    p2->next= t2;        // t2 in p2 einhaengen
17  }
```

⟹

c)
```
1   void connect(Polygon & *p1, Polygon *p2)
2   {
3     Polygon *t1, *t2;                       // temporaere Variablen
4
5     if (p1 == 0) {                          // Falls p1 leer ist
6       p1= p2;                               // Wird p1 zu p2
7     } else if (p2 != 0) {                   // Falls p2 nicht leer ist
8
9       t1= p1;                               // t1 wird der
10      while (t1->next!= p1) t1= t1->next;   // letzte Punkt in p1
11
12      t2= p2;                               // t2 wird der
13      while (t2->next!= p2) t2= t2->next;   // letzte Punkt in p2
14
15      t2->next= p1;                         // Der Nachfolger von t2 ist p1
16      t1->next= p2;                         // Der Nachfolger von t1 ist p2
17    }
18  }
```

Aufgabe 5: Logik *(25 Punkte)*

a)
```
1   bool wetter;       // Das Wetter ist schoen
2   bool hausaufgaben; // Es sind Hausaufgaben zu erledigen
3   bool zimmer;       // Das Zimmer muss aufgeraeumt werden
```

b)
```
1   bool darfspielen(bool wetter; bool hausaufgaben; bool zimmer)
2
3     // wetter: Das Wetter ist schoen
4     // hausaufgaben: Du musst noch Hausaufgaben erledigen
5     // zimmer: Das Zimmer muss noch aufgeraeumt werden
6
7   {
8     return(!((!wetter && hausaufgaben) || // "das Wetter schlecht ist und du noch
9                                           //  Hausaufgaben zu erledigen hast"
10            (!hausaufgaben && zimmer) || // "keine hausaufgaben zu erledigen hast,
11                                          // aber das zimmer noch aufgeraeumt werden muss"
12            (wetter && hausaufgaben)));  // "das wetter zwar schoen ist, aber du noch
13                                          //  hausaufgaben erledigen musst"
14  }
```

c)

wetter	hausaufgaben	zimmer	darfspielen
false	**false**	**false**	**true**
false	**false**	**true**	**false**
false	**true**	**false**	**false**
false	**true**	**true**	**false**
true	**false**	**false**	**true**
true	**false**	**true**	**false**
true	**true**	**false**	**false**
true	**true**	**true**	**false**

d) `!hausaufgaben && !zimmer`

lj, Februar 2001