Hochschule Deggendorf Dr. Peter Jüttner	
Vorlesung: Einführung in die Programmierung	WS 2012
Übung 14	Termin 22.1.13

Wiederholung - Musterlösung

Datentypen

1. Welche Werte haben die Variablen uc, c1, c2, i1 am Ende des folgenden Programms (jeweils als Zahlen ihres Typs interpretiert) ?

```
void main(void)
{ unsigned char uc;
  char c1, c2;
  c1 = -1;
  unsigned long l = 258;
  int i1 = 6.2;

  i1 = i1 / 2;
  uc = c1;
  c1 = l;
  c2 = -'b';
}
```

uc hat den Wert 255 (2^8-1), da -1 binär den Wert 11111111 (als char) hat. In einer unsigend Variablen wird der Wert als vorzeichenlos interpretiert.
c1 hat den Wert 2, da 258 als long hexadezimal den Wert 0x00000102 hat. Bei der Zuweisung an eine char Variable werden die ersten drei Bytes „abgeschnitten".
c2 hat den Wert -98 (negierter Wert des ASCII-Codes des Buchstaben b
i1 hat den Wert 3, da 6.2 bei der Zuweisung an eine int Variable zu 6 umgewandelt wird (Nachkommateil wird abgeschnitten)

Shiftoperationen

2. Welchen Wert haben die Variablen i1, i2, i3 und i4 am Ende des folgenden Programms?

```
void main(void)
{ int i1, i2, i3, i4;

  i1 = 100;
```

```
  i2 = 10;
  i3 = -100;
  i4 = -10;
  i1 = i1 >> 2;
  i2 = i2 << 4;
  i3 = i3 >> 2;
  i4 = i4 << 4;

}
```

i1 = 25, da der Rechtsshift um 2 Stellen einer ganzzahligen Division durch 4 (22) entspricht (die erste Bitstelle (Vorzeichen, hier 0) wird links nachgezogen).
i2 = 160, da der Linksshift um 4 Bitstellen einer Multiplikation mit 16 (=24) entspricht. .
Bei den negativen Zahlen wird dies analog durchgeführt mit dem Unterschied, dass beim Rechtsshift das Vorzeigenbit (in diesem Fall 1) nachgezogen wird.
i3 hat also den Wert -25 und i4 den Wert -160.

Bitweise Operatoren

3. Schreiben Sie ein Programm, das feststellt, ob das vorletzte Bit einer Integervariablen gesetzt (d.h. gleich 1) ist.

```
int main()
{
  /* Feststellen ob vorletztes Bit gesetzt ist  */
  /* mittels einer Bitweise Und-Verknüpfung */
  int zahl;
  printf("Bitte eine Zahl eingeben\n");
  scanf("%d",&zahl);

  if (zahl & 0x00000002)
    printf("Vorletztes Bit gesetzt\n");
  else printf("Vorletztes Bit nicht gesetzt\n");

  system("PAUSE");
}
```

4. Schreiben Sie ein Programm, das das erste Bit einer Integervariablen auf 1 setzt (alle anderen Bits sollen unverändert bleiben)

```
int main()
{
  /* Setzen des ersten Bits einer Integer    */
  /* mittels einer Bitweise Oder-Verknüpfung */
  int zahl;
  printf("Bitte eine Zahl eingeben\n");
  scanf("%d",&zahl);

  zahl = zahl | 0x80000000;
```

```
  printf("Zahl nach Setzten : %d\n", zahl);

  system("PAUSE");
}
```

Kontrollstrukturen

5. Transformieren Sie die folgenden switch-case.Anweisung in eine äquivalente if-else-Anweisungsstruktur

```
int i;
scanf(“%d“, &i)
switch (i)
{
  case 0 : printf(“Es wurde die Zahl 0 eingegeben\n“); break;
  case 1 : printf(“Es wurde die Zahl 1 eingegeben\n“); break;
  default : printf(„Es wurde weder 0 noch 1 eingegeben\n“); break;
}
```

```
if (i == 0)
  printf(“Es wurde die Zahl 0 eingegeben\n“);
else if (i == 1)
        printf(“Es wurde die Zahl 1 eingegeben\n“);
else printf(„Es wurde weder 0 noch 1 eingegeben\n“);
```

6. Transformieren Sie die folgende for-Schleife in eine äquivalente while-Schleife

```
int z;
for (z=0; z < a*b; z++)
{ a = a--;
  b = b++;
}
```

```
int z = 0;
while (z<a*b)
{ a = a--;
  b = b++;
  z++;
};
```

Vektoren und Zeichenketten

7. Schreiben Sie ein C-Programm, das aus einer Zeichenkette alle Leerzeichen (Blanks) entfernt.

   Lösungsansatz: Die Zeichenkette wird durchlaufen, wobei jedes Zeichen ungleich Leerzeichen in eine zweite Zeichenkette umkopiert wird.

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

```
int main()
{
  /* Leerzeichen werden beim Kopieren in eine zweite Zeichenkette entfernt */
  char zk1[100]; /* Zeichenkette aus der Leerzeichen entfernt werden */
  char zk2[100]; /* Zeichenkette ohen Leerzeichen */
  printf("Bitte Zeichenkette eingeben (max. 99 Zeichen)\n");
  gets(zk1);
  int zk2_index = 0; /*Index für Schreiben in zk2 */
  for (int i = 0; i <= strlen(zk1); i++)
  { if (zk1[i] != ' ')
    { zk2[zk2_index] = zk1[i];
      zk2_index++;
    }
  };
  printf("Originale Zeichenkette: %s\n", zk1);
  printf("Ohne Leerzeichen: %s\n",zk2);
  system("PAUSE");
}
```

8. Schreiben Sie ein C-Programm, das für eine Menge M {1, 3, 5, 7, 9} von Zahlen alle paarweisen Produkte berechnet und ausgibt. Dabei soll kein Produkt doppelt vorkommen.
   Tipp: Speichern Sie die Werte in einem Vektor und führen Sie die Produktberechnung mittels for-Schleifen durch.

   Lösungsansatz: Zahlen werden in einem Vektor der Länge 5 gespeichert. Dann werden die ersten vier Zahlen mit den nachfolgenden Zahlen multipliziert. Codiert wird dies in zwei geschachtelten for-Schleifen. Die äußere läuft durch die ersten vier Zahlen, die innere durch die nachfolgenden.

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

int main()
{ int zvektor[5] = {1,3,5,7,9};

  for (int i=0; i<4; i++)
    for (int j=i+1; j < 5; j++)
      printf("%d mal %d = %d\n", zvektor[i], zvektor[j], zvektor[i] * zvektor[j]);

  system("PAUSE");

  return 0;
}
```

Pointer

9. Welchen Wert haben die Variablen a und b am Ende (vor der return-Anweisung) des folgenden Programms?

```
int main()
{ int a = 3;
  int b = 5;
  int c = 7;
  int *ip1 = &a;      /* ip1 zeigt auf a */
  int *ip2 = &c;      /* ip2 zeigt auf b */
  a = a * *ip1;       /* entspr. a = a*a, d.h. a hat Wert 9 */
  c = a * *ip2;       /* entspr. c = a*b, d.h. c hat den Wert 63 */
  ip1 = &b;           /* ip1 zeigt jetzt auf b */
  ip2 = &c;           /* ip2 zeigt jetzt auf c */
  *ip1 = b + *ip2;    /* entspricht b = b + c, b hat den Wert 68 */

  return 0;
}
```

Prozeduren und Funktionen

10. Schreiben Sie eine C-Prozedur, die nacheinander 5 nicht negative Zahlen (unsigned int) in einen entsprechenden Vektor einliest und die beiden größten Zahlen ermittelt und per Parameter zurückgibt. Eingabefehler müssen nicht berücksichtigt werden.

```
void zweimax(int *max1, int *max2)
{ int zahlenvektor[5] = {0,0,0,0,0};
  int m1 = 0, m2 = 0;
  /* m1 ist groeßte Zahl, m2 zweitgroeßte Zahl */
  int zaehler = 0;
  /* Einlesen der Zahlen */
  for (zaehler = 0; zaehler < 5; zaehler++)
  {
    printf("Bitte %d-te Zahl eingeben\n", zaehler+1);
    scanf("%d", &(zahlenvektor[zaehler]));
  }
  /* Einlesen beendet, jetzt ermitteln der beiden größten Zahlen */
  /* zunächst m1 und m2 mit Anfangswerten nach Größe belegen */
  if (zahlenvektor[0] > zahlenvektor[1])
  {
    m1 = zahlenvektor[0];
    m2 = zahlenvektor[1];
  }
  else
  {
    m1 = zahlenvektor[1];
    m2 = zahlenvektor[0];
  }
  /* Jetzt Durchlaufen des reatlichen Vektors */
  for (zaehler = 2; zaehler <5; zaehler++)
  {
    if ((zahlenvektor[zaehler] > m1))
    { /* Zahl im Vektor ist größer als die bisher größte */
      m2 = m1;
```

```
      m1 = zahlenvektor[zaehler];
    }
    else if (zahlenvektor[zaehler] > m2)
      /* Zahl im Vektor ist größer als die bisher zweitgrößte */
          m2 = zahlenvektor[zaehler];
  }
  *max1 = m1;
  *max2 = m2;
}
```

11. Schreiben Sei eine C-Funktion, die als Funktionswert zurückliefert, ob ein Buchstabe in einer Zeichenkette genau zweimal vorkommt. Sowohl die Zeichenkette als auch das Zeichen sollen per Parameter übergeben werden. Die Funktion soll 1 zurückgeben, falls das gesuchte Zeichen genau zweimal vorkommt, ansonsten soll 0 zurückgegeben werden. Der Zugriff auf die Elemente der Zeichenkette soll über Pointer und nicht über Indizierung erfolgen.

```
int zweimal(char zk[], char zeichen)
{
   int vorkommen = 0; /* Zaehlt, wie oft zeichen in zk vorkommt */
   for (int i = 0; i< strlen(zk); i++)
   { /* Zahlen, wie oft zeichen in zk vorkommt */
     if (*(zk+i) == zeichen)
       vorkommen++;
   }
   if (vorkommen == 2)
     return 1;
   else return 0;
}
```

Aufzählunsgtypen und Datenstrukturen

12. Definieren Sie in C einen Aufzählungstyp für die verschiedenen Arten von Wirbeltieren.

```
enum Wirbeltier {Saeugetier, Fisch, Echse, Amphibium, Vogel};
```

alternativ:

```
typedef enum Wirbeltier {Saeugetier, Fisch, Echse, Amphibium, Vogel};
```

13. Datenstruktur Vereinsmitglied
    a.) Definieren Sie in C eine Datenstruktur zur Erfassung der Daten eines Mitglieds in einem Sportverein. Folgende Daten eines Mitglieds sollen in getrennten Komponenten der Datenstruktur mit geeigneten Standarddatentypen und Vektoren gespeichert werden:
    - Vorname
    - Nachname
    - Mitgliedsnummer (6-stellig)
    - Anschrift (Postleitzahl, Ort, Strasse und Hausnummer)

- Sparte (Fußball, Handball, Tennis, Ski, Gymnastik, Volleyball)

b.) Schreiben Sie eine Folge von C-Anweisungen, in der eine Variable der Datenstruktur angelegt wird und eine beliebige Komponente von Tastatur eingelesen wird.

```
enum Sparte {Fussball, Handball, Tennis, Ski, Gymnastik, Volleyball};

typedef struct Vereinsmitglied
{
   char Vorname[NAMENSLAENGE];
   char Nachname[NAMENSLAENGE];
   char Mitgliedsnummer[NR_LAENGE];
   char Anschrift[ANSCHRIFTLAENGE];
   Sparte s;
};

Vereinsmitglied vml;

/* Einlesen der Daten */
printf("Bitte Vorname eingeben (max. %d Zeichen\n", NAMENSLAENGE);
gets(vm.Vorname);
printf("Bitte Nachname eingeben (max. %d Zeichen\n", NAMENSLAENGE);
gets(vm.Nachname);
printf("Bitte Nummer eingeben (max. %d Zeichen\n", NR_LAENGE);
gets(vm.Mitgliedsnummer);
printf("Bitte Anschrift eingeben (max. %d Zeichen\n", ANSCHRIFTLAENGE);
gets(vm.Anschrift);
printf("Bitte Sparte eingeben\n");
scanf("%d",&( vm.s));
```