Prof. Dr. Carsten Meyer
Fachbereich Informatik & Elektrotechnik
Fachhochschule Kiel

Programmieren in C++ – Sommersemester 2015

Aufgabenblatt 1

Abgabe: Donnerstag, 23.04.2015, 11:30

Aufgabe 1 (Kontrollstrukturen, Zufallszahlen):

Schreiben Sie ein Programm, das Zufallszahlen zwischen 0 und 1 generiert, und zwar so lange, bis die generierte Zufallszahl größer als 0.85 ist. Die Zufallszahlen sollen auf dem Bildschirm ausgegeben werden. Weiterhin sollen von den generierten Zufallszahlen Summe, Mittelwert, Minimum und Maximum berechnet und auf dem Bildschirm ausgegeben werden (zusammen mit der Anzahl der gezogenen Zufallszahlen). Das Programm soll insgesamt 5 Durchläufe generieren.

Die Lösung der Aufgabe soll sowohl die Implementierung als auch eine Bildschirmausgabe des kompletten Programms (5 Durchläufe) umfassen.

Hinweise:

1.) Zur (Pseudo)Zufallszahlenberechnung verwenden Sie die Funktion **`rand()`**, die eine Integralzahl zwischen 0 und RAND_MAX zurückgibt. Diese (Pseudo-) Zufallszahlenberechnung muß initialisiert werden, damit nicht bei jedem erneuten Programmaufruf dieselbe Sequenz von Zufallszahlen generiert wird. Hierzu rufen Sie einmalig gleich zu Beginn der **`main()`**-Funktion die folgende Funktion auf:

```
srand( time(NULL) );
```

Achtung: Unter Linux muß für **`srand`** ggf. die **`cstdlib`** eingebunden werden!

2.) Zur Ausgabe der Zahlen verwenden Sie entweder die C-Funktion "**`printf`**" oder die C++ Ausgabe über "**`cout`**".
Beispiel: Eine Gleitpunktzahl z kann ebenso wie eine Integer-Zahl i und ein String s ausgegeben werden mit

```
cout << z << endl;
cout << i << endl;
cout << s << endl;
cout << "Hallo" << endl;
```

Das "**`endl`**" sorgt dabei für einen Zeilenumbruch (mehr später in der Vorlesung)

3.) Falls Sie Typumwandlungen benötigen, benutzen Sie bitte die Typumwandlungsoperatoren von C++!

Aufgabe 2 (Struktur, sizeof-Operator):

a) Mit dem **`sizeof`**-Operator kann die Anzahl an Bytes ermittelt werden, die Variablen eines bestimmten Datentyps im Hauptspeicher belegen. Beispielsweise hat **`sizeof(char)`** den Wert 1.
Geben Sie in einem C++ - Programm für jeden elementaren Datentyp die Größe des benötigten Speicherplatzes (zusammen mit dem Datentyp) auf dem Bildschirm aus.

b) Legen Sie eine Struktur "student" an mit folgenden Elementen:

- id
- name
- studiengang
- note
- bestanden

Überlegen Sie sich hierfür geeignete Datentypen für die einzelnen Elemente.
Geben Sie mit dem sizeof-Operator den Speicherplatz für die Struktur student aus.

c) Legen Sie ein Feld von 3 Variablen des Typs **`struct student`** an und initialisieren Sie die einzelnen Strukturelemente der Feldvariablen über eine Initialisierungsliste. Legen Sie anschließend zwei weitere Variablen vom Typ **`struct student`** an. In diesem Fall sollen die Werte der jeweiligen Stukturelemente jedoch einzeln über entsprechende Zuweisungen gespeichert werden.

Geben Sie anschließend die Werte der Elemente für alle 5 Strukturvariablen auf dem Bildschirm aus.

Die Lösung dieser Aufgabe soll sowohl die Implementierung als auch die Bildschirmausgabe des Programms umfassen.

Aufgabe 3 (Referenzen):

a) Gegeben sei folgendes Programm:

```
int main ( int argc, char* argv[] )
{
  int x = 5;
  int& y = x;

  y = 9;
  int a[100];
  int &b = a[55];

  b = y;

  int *p = &y;
  int& z = *p;

  z = 33;
  …
}
```

Geben Sie die Werte von x, y, z und b am Ende des Programms an.

b) Legen Sie die Struktur „Student“ aus Aufgabe 2b an sowie ein Feld von 3 Variablen des Typs **`struct student`**, das über eine Initialisierungsliste initialisiert wird (hierzu können Sie den Code aus Aufgabe 2b verwenden).

Legen Sie anschließend folgende Referenzen an:

i. Eine Referenz auf das Element „note“ des ersten Feldelements,
ii. Eine Referenz auf das mittlere Feldelement,
iii. Eine Referenz auf das Element „bestanden“ der Referenz aus ii.

Verwenden Sie diese Referenzen, um die entsprechenden Werte der Variablen auf dem Bildschirm auszugeben (Hinweis: Bei der Referenz auf die Struktur in ii. müssen die einzelnen Strukturelemente dereferenziert und einzeln ausgegeben werden).

Aufgabe 4 (Referenzen als Aufrufparameter):

Schreiben Sie ein C++-Programm, welches aus der Funktion **`main`** und der Funktion **`addiere`** besteht. In der Funktion **`main`** sollen drei lokale **`int`**-Variablen angelegt und mit einer Zufallszahl zwischen 1 und 100 initialisiert werden; die Werte der drei Variablen sollen auf dem Bildschirm ausgegeben werden. Anschließend wird die Funktion **`addiere`** aufgerufen. **`addiere`** hat drei Aufrufparameter (Parameter 1: der Wert der ersten Zufallszahl-Variablen, Parameter 2: die *Adresse* der zweiten Zufallszahl-Variablen, Parameter 3: eine Referenz auf die dritte Zufallszahl-Variable) und gibt die Summe der Werte zurück. Außerdem werden lokal in der Funktion **`addiere`** (*nach* Berechnung der Summe) alle Werte verdoppelt und die verdoppelten Werte auf dem Bildschirm ausgegeben. Nach Aufruf der Funktion **`addiere`** wird dann in der **`main`** –Funktion der Wert der Summe ausgegeben sowie die Werte der drei Variablen.

Diskutieren Sie kurz das Ergebnis.