Dr. Peter Lachmann
Martin Pietsch

SS 2015
19.05.2015

Programmieren - Blatt 3

Diskussion:

- *Vorbereitung Aufgabe 1 - Call by Reference*:
  - Was ist mit *Call-by-value* und *Call-by-reference* gemeint?
  - Wofür braucht man das Konzept *Call-by-reference*?
  - Was geben die folgenden Programme aus?

```
#include <iostream>

using namespace std;

void set_to_zero(int a){
  a = 0;
}

int main(){
  int a = 5;
  set_to_zero(a);
  cout << a << endl;
  return 0;
}
```

```
#include <iostream>

using namespace std;

void set_to_zero(int& a){
  a = 0;
}

int main(){
  int a = 5;
  set_to_zero(a);
  cout << a << endl;
  return 0;
}
```

- *Vorbereitung Aufgabe 2 - Arrays*:
  - Was ist ein Array?
  - Wie wird ein (statistisches) Array der Länge n vom Typ **double** angelegt?
  - Wie findet man heraus, wie lang ein Array ist?

Präsenz-Aufgabe 1: *(Call-by-Reference)* **[$\leq$ 5 min]**

Schreiben Sie eine Funktion **tausche(..)**, welche die Werte zweier Variablen a und b (z.B. vom Typ **int** oder **double**) vertauscht und testen Sie diese in Ihrem Hauptprogramm.

Präsenz-Aufgabe 2: *(Arrays)*

(a) Schreiben Sie ein Programm, welches den Mittelwert $\bar{x} := \frac{1}{n}\sum_{i=1}^{n} x_i$ und die empirische Varianz $s^2 := \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$ von n mit **rand()** erzeugten Zufallsvariablen $x_i, ..., x_n \in [0, 1]$ berechnet und ausgibt. Dabei soll n zur Laufzeit eingelesen und die Variablen $x_1, ..., x_n$ in einem Array gespeichert werden.

(b) Lagern Sie die Berechnung von $\bar{x}$ und s^2 jeweils in eine Funktion aus. Was für Parameter müssen diesen Funktionen übergeben werden?

Präsenz-Aufgabe 3: *(Modularisierung)*

In der ersten Aufgabe von Blatt 2 erstellten Sie ein Programm zur Berechnung einer Fakultät. Modularisieren Sie nun Ihr Programm, indem Sie die Deklaration der Funktion von ihrer Implementierung trennen. Im Detail:

- Schreiben Sie die Deklaration in eine sogenannte *Header*-Datei **fakultaet.h**. Diese sollte alles beinhalten, was für andere Benutzer sichtbar sein muss (in unserem Fall nur der Funktionskopf).

- Schreiben Sie die eigentliche Implementierung der Funktion in eine Datei **fakultaet.cpp**. Üblicherweise wird hier am Anfang gleich unser Header **fakultaet.h** eingebunden (auch wenn dies eigentlich nur netwendig ist, wenn darin noch andere Deklarationen stehen die benötigt werden):

Listing 1: fakultaet.cpp

```
#include "fakultaet.h"

long fakultaet(...){
  ...
}
```

In dieser Quelltext-Datei steht nun alles, was für die Implementierung notwendig ist, aber nicht notwendigerweise nach außen sichtbar sein muss.

- Binden Sie die Header-Datei in Ihrer ursprünglichen Programm-Datei ein, welche die *main*-Funktion enthält.
- Kompilieren Sie das Ganze mit folgendem Befehl:
  g++ -Wall -o fakultaet.o main.cpp fakultaet.cpp
- Versuchen Sie folgende Fragen zu beantworten:
  - Was passiert beim Einbinden von Header-Dateien?
  - Welche Schritte werden beim Aufruf eines Kompilierbefehls durchgeführt, z.B. beim Aufruf **g++ -Wall -o fakultaet.o main.cpp fakultaet.cpp**?
  - Was passiert, wenn eine Funktion zwar deklariert, aber nicht definiert wird?

(Hinweis: Eine sehr gute Erklärung zu dem Thema findet sich z.B. hier: http://www.learncpp.com/cpp-tutorial/19-header-files/)