Tag 4a - Felder und Funktionen

Aufgabe 1: Felder

Ziel dieser Aufgabe ist es, die Benutzung von Feldern einzuüben.

(a) Lege ein Feld `ifeld` vom Typ `int` an. Initialisiere es mit den Zahlen `{97, 43, 24, 2, 7, 12}`

(b) Lege ein Feld `cfeld` vom Typ `char` an. Initialisiere es mit der Zeichenkette `"Willkommen zum Vorkurs!"`.

(c) Was passiert, wenn man die Felder mittels `std::cout«ifeld;` bzw. `std::cout«cfeld;` ausgibt?

(d) Eine Erklärung für das Verhalten aus (c) ist, dass in C Zeichenketten traditionell als Felder vom Typ `char` gespeichert werden. Das Ende der Zeichenkette wird dabei durch eine 0 angezeigt. Schreibe eine Funktion `verschluesseln(char feld[])`, welche eine Zeichenkette als Eingabe nimmt und den Wert jedes Zeichens um 1 erhöht, bis die abschließende 0 erreicht wird. Was wird aus unserer Zeichenkette aus (b)?

Aufgabe 2: Felder und Funktionen

In dieser Aufgabe betrachten wir Felder vom Typ int mit variabler Länge. Schreibe jeweils eine Funktion, die

(a) das kleinste Element (Minimum) ausgibt.

(b) das größte Element (Maximum) ausgibt.

(c) alle Elemente ausgibt, die durch 5 teilbar sind.

(d) alle Elemente ausgibt, die durch eine beliebige Zahl p teilbar sind. Dabei soll p der Funktion als zusätzliches Argument übergeben werden.

Schreibe zudem ein Programm um obige Funktionen zu testen. Lies dazu eine natürliche Zahl n von der Kommandozeile ein. Lege dann ein Feld für n Zahlen an und initialisiere dies mit den Werten $17, \ldots, (n-1) * 3 + 17$. Was gibt das Programm aus ($p = 7$)?

Aufgabe 3: Funktionen und Modularisierung

(a) Schreibe eine Funktion `void malZweiA(int x)`, welche die übergebene Variable `x` mit 2 multipliziert und dann ausgibt.

(b) Schreibe eine Funktion `void malZweiB(int &x)`, die das gleiche macht, jedoch *per Referenz* aufgerufen wird.

(c) Was passiert, wenn beide Funktionen auf eine Variable `int y=4;` angewendet werden? Schreibe ein kleines Testprogramm.

(d) Schreibe eine Funktion `int Differenz(int x, int y)`, welche die Differenz `x-y` ausrechnet und zurückgibt. Verteile nun im Sinne eines *modularen Programmdesigns* diese Funktion auf drei Dateien. Verwende eine für die Deklaration (`sub.h`), eine für die Definition (`sub.cpp`) und eine für den Aufruf (`sub-programm.cpp`).

Aufgabe 4: Berechnung der Fakultät

Die Fakultät ist für natürliche Zahlen $n \in \mathbb{N}$ wie folgt definiert:

$$n! := 1 \cdot \ldots \cdot n = \prod_{i=1}^{n} i, \qquad n > 0$$

wobei $0! := 1$ für das leere Produkt gelten soll. Berechne die Fakultät wie folgt:

(a) Implementiere die Fakultät als rekursiven Prozess.

(b) Implementiere die Fakultät als iterativen Prozess unter Verwendung einer Schleife.

(c) Implementiere die Fakultät als iterativen Prozess ohne Verwendung einer Schleife.

(d) Teste die Funktionen durch Verwendung in einem Programm

Aufgabe 5: Berechnung der Fibonaccizahlen

Die Fibonaccizahlen sind wie folgt definiert (vgl. Seite 78 im Skript):

$$\text{fib}(n) := \begin{cases} 0, & \text{n=0} \\ 1, & \text{n=1} \\ \text{fib(n-1)+fib(n-2)}, & \text{n>1} \end{cases}$$

In dieser Aufgabe sollen diese auf verschiedene Weisen berechnet werden:

(a) Schreibe eine Funktion, welche die n. Fibonaccizahl rekursiv berechnet.

(b) Schreibe eine Funktion, welche die n. Fibonaccizahl iterativ berechnet. Verwende dazu eine Schleife.

(c) Schreibe eine weitere Funktion, welche die n. Fibonaccizahl iterativ berechnet. Löse jedoch die Schleife aus b) auf, indem Du sie durch eine rekursive Funktion ersetzt.

(d) Vergleiche, wie lange die drei Varianten benötigen, um z.B. `fib(35)`, `fib(40)` und `fib(45)` zu berechnen. (Hinweis: Ein Programm in Eclipse kann durch einen Klick auf das rote Viereck in der Konsole abgebrochen werden.)

Viel Erfolg!