Dr. Michael J. Winckler
IWR, Raum 302
INF 368
69120 Heidelberg
Michael.Winckler@iwr.uni-heidelberg.de
http://www.iwr.uni-heidelberg.de/teaching/Mathe-Star/

Arbeitsblatt 02 *Variablen, Datentypen, Seuerkonstrukte* (Version 1.10)

Theorie

Übungsziele: Erstes Ziel dieses Übungsblatts ist der sichere Umgang mit Variablen und einfachen Datentypen. Als Grundlage arbeiten wir nur mit den beiden Datentypen `int` und `float`, die für die meisten einfachen Anwendungen ausreichend sind. Benennung von Variablen, Umgang mit binären Operatoren und Ausgabe von Variableninhalten sind die Übungsinhalte dieser Lektion.

Zur Einführung in die Kontrollstrukturen betrachten wir zwei unterschiedliche Ablaufkontraollen: mit `if`–`then`–`else` steht uns ein Befehl zur Bedingten Verzweigung zur Verfügung (Selektion), während `for`–Schleifen eine erste Iteratorvariante darstellen.

Im Bereich der systematischen Compilerarbeit werden wir uns mit ersten Unterprogrammansätzen beschäftigen.

Übung 2.1

Schreibe ein Programm, das zwei fest codierte Zahlen `a` und `b` darauf testet, ob `b` ein Teiler von `a` ist. Gib auf dem Bildschirm eine entsprechende Meldung aus:

```
Erste Zahl:    a = 18
Zweite Zahl:   b =  6

 6 ist ein Teiler von 18.
18 geteilt durch 6 ist 3 Rest 0
```

Erweitere dein Programm dahingehend, dass du nicht b fest vorgibst, sondern in einer Schleife alle Zahlen von 1 bis `a` durchläufst und jeweils ausgibst:

```
Verwendete Zahl: a = 18

 1 ist  ein Teiler von 18.
 2 ist  ein Teiler von 18.
 3 ist  ein Teiler von 18.
 4 ist kein Teiler von 18.
 5 ist kein Teiler von 18.
 6 ist  ein Teiler von 18.
 ...
```

Übung 2.2

Wir wollen nun ein erstes Unterprogramm (eine Subroutine) anlegen. Diese Subroutine testet eine Zahl auf ihre Primeigenschaft. Die Schnittstelle dieser Subroutine ist `int IstPrim(int n)`.

Mache dir klar, welche Werte an die Subroutine übergeben werden und welches Resultat zu erwarten ist. Verwende dann deine Ergebnisse aus Aufgabe 2.1, um diese Subroutine zu realiiseren.

Schreibe ein Hauptprogramm, das alle Primzahlen zwischen 1 und 1000 ausgibt. Lass dir vom Programm mitzählen, wieviele Primzahlen das sind.

Übung 2.3

Schreibe ein monolithisches Programm `sinus1`, in dem zwei Variablen `argument` und `ergebnis` definiert werden. Zudem brauchst du noch eine Zählschleifenvariable `i`.

Lasse mit Hilfe einer Zählschleife die Werte der Sinusfunktion berechnen. Gehe dabei von 0° bis 180° in Schritten von 1 Grad vor. Gebe untereinander aus:

```
Der Sinus von   0.00 Grad betr"agt 0.000000
Der Sinus von   1.00 Grad betr"agt 0.017452
...
```

Beachte den Unterschied zwischen Gradmass und Bogenmass! Die Sinusfunktion findet sich in der Mathematik–Bibliothek. Ihr Headerfile ist `math.h`.

Schreibe ein zweites Programm `sinus2`, das dem ersten bis auf die Ausgabe völlig gleicht. Gebe statt des Satzes Sternchen aus, 30 Sternchen für einen Sinuswert von 1.00, 0 Sternchen für einen Sinuswert von 0. Dafür brauchst Du eine Variable `sternzahl` vom Typ `int`, die Sie aus dem gewonnenen Sinuswert berechnen. Ferner tut eine zweite Schleife hier gute Dienste!

Läuft dein Programm auch von 0° bis 360°? Wie könnte man es passend erweitern?

Übung 2.4

Schreibe ein Programm, das ein Schachbrett aus den Zeichen `#` und `.` auf den Bildschirm ausgibt. Erweitere dein Programm so, daß für jedes Feld 3x3 Zeichen ausgegeben werden.

Strukturiere das Programm so, daß zwei Routinen `MyKleinBrett` und `MyGrossBrett` die beiden Funktionalitäten bereitstellen.

Kannst du beiden Routinen so ändern, daß sie verschieden grosse Bretter ausgeben können?

Übung 2.5

Schreibe ein Programm `sinus3`, das wie `sinus1` arbeitet, aber bei einem beliebigen Winkel beginnt und endet und immer 40 Teilpunkte ausgibt. Verwende dazu eine adäquate Steuerkonstruktion.

Übung 2.6

Schreibe ein Programm `fakultaet`, das nach Eingabe einer (kleinen!) ganzen Zahl `n` den Ausdruck `n!=n*(n-1)*...*2*1` ausrechnet (`n!` spricht sich „*n Fakultät*).

Hierzu benötigst du zuerst einen kurzen Hilfetext, der die Aufgabe des Programms erläutert. Gebe die vom Programm gewählte Zahl aus. Danach kannst du eine freie Variable `fakult` mit dem Wert 1 vorbelegen und in einer Zähl-Schleife diese Variable nacheinander mit 1, 2, 3, ... multiplizieren lassen. Wie lange muß die Schleife laufen und welches Schleifenkonstrukt sollten Sie verwenden.

Vorsicht: Der mathematische Ausdruck `n!` wächst extrem schnell. Teste das Programm nur an kleinen Beispielen.

Erweitere das Programm dahingehend, dass du `fakult` vom Datentyp `long int` definierst. Probiere aus, welche Eingaben noch richtig bearbeitet werden.

Übung 2.7

Schreibe ausgehend von `fakultaet` ein neues Programm `floatfakultaet`, das `fakult` mit dem Typ `float` deklariert. Wann bekommst du nun Probleme?

Kannst du die Schleife zur Aufmultiplikation auch rückwärts laufen lassen (d.h. so rechnen, wie die Definition oben angegeben ist)?