

2 Variablen

2.1 Gültigkeitsbereich von Variablen

Variable werden dann verwendet, wenn das Programm Daten für die spätere Weiterverarbeitung speichern soll.

Wenn man in einer Hochsprache, und damit auch in C bzw. C#, eine Variable deklariert, so stellt das System abhängig vom Datentyp der Variablen Speicherplatz bereit.

Im Modul 403 haben Sie die Programmiersprache C kennen gelernt. Sie haben gesehen, dass bei der Deklaration von Variablen immer der Datentyp angegeben werden muss. Die wichtigsten Datentypen sind:

- char Zeichen
 - int Ganzzahliger Wert
 - double Wert mit Kommastellen
- usw.

Als Gültigkeitsbereich einer Variablen bezeichnet man den die zeitliche Dauer während diese Variable verwendet werden kann. Man unterscheidet dabei zwischen «lokalen» und «globalen» Variablen. Eine globale Variable gilt während des ganzen Programms und wird ausserhalb des Hauptprogramms (der main-Funktion) deklariert. Eine lokale Variable wird innerhalb einer Funktion bzw. eines Blocks deklariert und gilt nur innerhalb der Funktion bzw. des Blocks. Ein Block wird durch geschweifte Klammern {...} definiert.

Aufgabe 01: Globale und lokale Variablen

Analysieren Sie das untenstehende Beispiel und beantworten Sie die folgenden Fragen!

Lokale Variable:	Globale Variable:
<pre>#include <stdio.h> #include <stdlib.h> int main() { double Umfang = 0.0; Umfang = 11.45; printf("Umfang = %lf", Umfang); system("pause"); return 0; }</pre>	<pre>#include <stdio.h> #include <stdlib.h> double Umfang = 0.0; int main() { Umfang = 11.45; printf("Umfang = %lf", Umfang); system("pause"); return 0; }</pre>

1. Können beide Programme ohne Fehler kompiliert werden?
2. Geben beide Programme das gleiche aus?
3. Welche Vor- und Nachteile haben die beiden Lösungen?

[illegible]

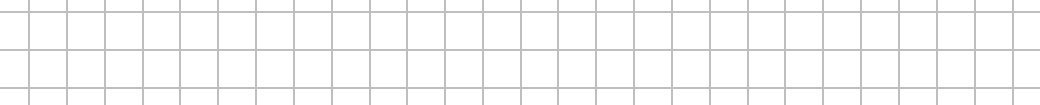
Aufgabe 02: Gültigkeit von Variablen

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

int Wert = 0;

int main()
{
    if (Wert >= 0) {
        double Wert = 15.5;
        printf("Wert = %lf", Wert);
    } else {
        printf("Der Wert ist negativ!");
    }
    system("pause");
    return 0;
}
```

Analysieren Sie das obestehende Programm und schreiben Sie auf, ob dieses Programm kompiliert werden kann oder nicht! Falls es kompiliert werden könnte, was gäbe das Programm aus?



Tippen Sie nun das Programm ab und bringen Sie es zum Laufen. Erstellen Sie dazu eine leere Projektmappe «B02_Variablen» mit dem Projekt «Aufgabe02».

Begründen Sie nun warum Sie bei der Beantwortung vorher richtig bzw. falsch lagen:

[illegible]

Aufgabe 03: Umrechnung von Celsius in Fahrenheit

Schreiben Sie ein Programm mit welchem nach der Eingabe der Temperatur in Celsius, z.B. 31.5° Celsius, in die in Amerika übliche Fahrenheit-Skala umrechnet. Die Formel zur Umrechnung lautet:

$$^{\circ}F = \frac{^{\circ}C \cdot 9.0}{5.0} + 32.0$$

Die Temperatur in Celsius muss über die Tastatur eingegeben werden können. Nach der Umrechnung geben Sie die beiden Temperaturen aus.

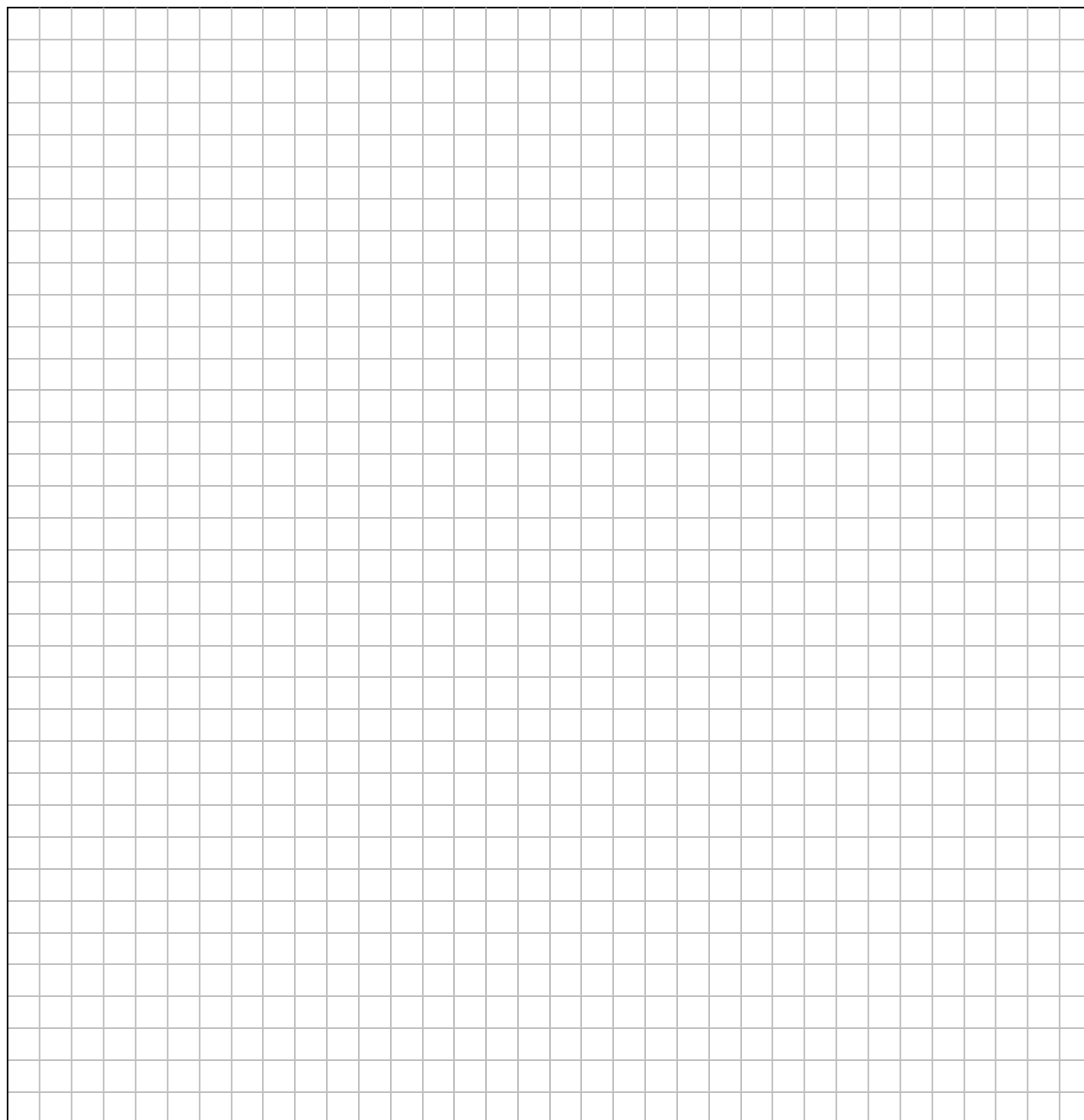
Tipp: Überprüfen Sie die Berechnung mit den Umrechnungsmöglichkeiten des Windows-Rechners.

Erstellen Sie die Aufgabe in drei Varianten

Aufgabe03a: Verwenden Sie für die erste Variante nur ein Hauptprogramm

Aufgabe03b: Lagern Sie die Umrechnung in eine Funktion «CelsiusToFahrenheit» aus. Die Funktion hat keine Argumente und keinen Rückgabewert.

Aufgabe03c: Ändern Sie die Aufgabe03b jetzt so ab, dass Ihr Programm keine globalen Variablen mehr verwendet.



2.2 Adressen von Variablen

Bis jetzt kennen wir Variablen, die durch einen Namen und einen Typ definiert sind. Das folgende Beispiel zeigt die Anwendung von solchen Variablen.

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

int main()
{
    int Anzahl = 24;

    printf ("Inhalt von Anzahl = %i\n", Anzahl);
    printf ("Adresse von Anzahl = %p\n", &Anzahl);
    system("pause");
    return 0;
}
```

Mit der Deklaration der ganzzahligen Variablen «Anzahl» reserviert das System einen Speicherplatz von 32 Bits = 4 Bytes (die Grösse des reservierten Speichers ist dabei vom Betriebssystem bzw. den Compilereinstellungen abhängig). Das System merkt sich dabei nur die Adresse und den Datentyp der Variablen, aber nicht deren Namen. Der Name wird nur für den Programmierer gebraucht, da wir uns Namen besser als Zahlen merken können (Analog wie beim Domain-Name und der dazugehörigen IP-Adresse)

Bringt man das obige Programm zum Laufen, so erzeugt es die folgende Ausgabe:

```
Inhalt von Anzahl = 24
Adresse von Anzahl = 0032FA20
```

Mit dem Adress-Operator «&» wird die Adresse der Variable «Anzahl» bestimmt. Damit wir die Adressen und den Inhalt von Variablen während dem Programmlauf kontrollieren können, verwenden wir den Debugger vom Visual Studio.

Aufgabe 04: Adresse von Variablen

Erstellen Sie eine Windows-Konsolenanwendung «Aufgabe04» in der Projektmappe «B02_Variablen», mit den folgenden Anforderungen:

- Eine ganzzahlige Zahl über die Tastatur eingeben und in der Variablen «Zahl» abspeichern.
- Den Wert der Variable «Zahl» durch drei dividieren und in der «Double»-Variablen "Drittel" abspeichern
- Den Inhalt und die Adressen der beiden Variablen «Zahl» und «Drittel» ausgeben.

Kontrollieren Sie die Variablen, die erhaltenen Ausgaben und den Arbeitsspeicher mit dem Debugger.

Was stellen Sie fest, wenn Sie den Inhalt der Variablen im Speicher anschauen?

