7. Strukturen

7.1.Definition von struct

Bis jetzt haben wir nur die in der C++ Standard Library definierte Typen benutzt. Es ist aber manchmal praktisch, sie selbst zu definieren. Das macht man mit dem Schlüsselwort *struct*. Hier ein kleiner Beispiel:

Beispiel 1

```
struct adresse
{
  std::string Anrede;
  std::string Vorname;
  std::string Nachname;
  std::string Strasse;
  int Hausnummer;
  int Postleitzahl;
  std::string Ort;
  std::string Land;
};
```

adresse ist hier der Name des neuen Datentyps – das, was Sie angeben müssen, falls Sie es benutzen wollen (wie z.B. float oder char). Danach kommt die Typdefinition selbst: Namen und Typen von den Feldern des neuen Datentyps.

Definitionen neuer Datentypen müssen immer außerhalb von Funktionen erfolgen. Das Schlüsselwort *struct* darf daher nie innerhalb einer Funktion stehen. Beachten Sie außerdem, dass hinter der geschlossenen geschweiften Klammer einer Struktur-Definition ein Semikolon stehen muss. Andernfalls meldet der Compiler einen Fehler.

Und hier ein Beispiel, wie man einen neudefinierten Typ benutzt:

Beispiel 2

```
#include <string>

struct adresse
{
  std::string Anrede;
  std::string Vorname;
  std::string Nachname;
  std::string Strasse;
  int Hausnummer;
  int Postleitzahl;
  std::string Ort;
  std::string Land;
};

int main()
{
  adresse MeineAdresse;
```

```
  MeineAdresse.Anrede = "Herr";
  MeineAdresse.Vorname = "Boris";
  MeineAdresse.Nachname = "Schaeling";
  MeineAdresse.Strasse = "Schlossallee";
  MeineAdresse.Hausnummer = 1;
  MeineAdresse.Postleitzahl = 12345;
  MeineAdresse.Ort = "Entenhausen";
  MeineAdresse.Land = "Deutschland";
}
```

7.2.Enumeration

Während mit dem Schlüsselwort *struct* Datentypen erstellt werden können, die Variablen zu einer Gruppe zusammenfassen, werden mit dem Schlüsselwort *enum* ganz bestimmte und gültige Werte für einen neuen Datentypen definiert.

Hier ein Beispiel:

Beispiel 3

```
enum land
{
  Deutschland, Oesterreich, Schweiz
};
```

Mit dem Schlüsselwort *enum* beginnt die Definition einer Enumeration. Hinter *enum* folgt der Name des neuen Datentyps - im obigen Beispiel lautet er *land*. Danach folgen wie bei der Struktur geschweifte Klammern, zwischen denen die eigentliche Definition des Datentypen steht. Beachten Sie, dass auch bei einer Enumeration hinter der geschlossenen geschweiften Klammer ein Semikolon angegeben werden muss.

Was muss nun zwischen den geschweiften Klammern stehen? Eine Enumeration ist ein Datentyp, der auf ganz bestimmte neue Werte gesetzt werden kann. Die Werte, die für den neuen Datentyp gültig sind, werden einfach zwischen geschweiften Klammern angegeben und jeweils per Komma getrennt. Für den Datentyp land sind demnach die Werte Deutschland, Oesterreich und Schweiz gültig.

Nachdem der Datentyp definiert ist, könnte er wie folgt in einem Programm verwendet werden.

Beispiel 4

```
enum land
{
  Deutschland, Oesterreich, Schweiz
};

int main()
{
  land MeinLand = Deutschland;
}
```

Intern besteht eine Enumeration aus dem Datentyp *int*. Im obigen Beispiel ist also MeinLand genaugenommen eine *int*-Variable. Den in einer Enumeration definierten Werten wird einfach intern jeweils eine Zahl zugeordnet. Der erste Wert erhält hierbei die Zahl 0, der zweite die Zahl 1, der dritte die Zahl 2 und so weiter. Wenn Sie möchten, dass die interne Darstellung der Werte nicht bei 0, sondern bei 100 beginnt, so können Sie dies direkt bei der Definition der Enumeration angeben.

Beispiel 5

```
enum land
{
  Deutschland = 100, Oesterreich, Schweiz
};
```

Der erste Wert wird intern als Zahl 100 gehandhabt. Dem nächsten Wert Oesterreich wird die nächsthöhere Zahl zugewiesen - in diesem Fall also 101. Möchten Sie, dass Oesterreich eine andere Zahl erhält, müssen Sie diese wieder explizit angeben.

Beispiel 6

```
enum land
{
  Deutschland = 100, Oesterreich = 150, Schweiz
};
```

Die Tatsache, dass eine Enumeration intern lediglich ein int ist, bedeutet, dass einer Variablen vom Typ einer Enumeration auch jede beliebige Zahl zugewiesen werden kann.

Beispiel 7

```
enum land
{
  Deutschland, Oesterreich, Schweiz
};

int main()
{
  land MeinLand = 0;
}
```

Ob Sie der Variablen MeinLand im obigen Beispiel die Zahl 0 zuweisen oder den Wert Deutschland spielt keine Rolle - das Ergebnis ist das gleiche. Sie können der Variablen jedoch auch jede beliebige andere Zahl zuweisen, solange diese vom Datentyp int gespeichert werden kann. Daher kompiliert auch folgendes Programm fehlerfrei, auch wenn die zugewiesene Zahl keinem Wert in der Enumeration entspricht.

Beispiel 8

```
enum land
{
  Deutschland, Oesterreich, Schweiz
};
```

```
int main()
{
  land MeinLand = 100;
}
```

Variablen vom Typ einer Enumeration können nicht nur als lokale oder globale Variablen definiert werden, sondern selbstverständlich auch innerhalb einer Struktur Verwendung finden.

7.3.Aufgaben

Mini-Projekt: Implementieren Sie ein Telefon- und Adressbuch Programm.

Die Daten sollen in ASCII-Dateien gespeichert werden und wieder abrufbar sein.

Für die Daten von einem Menschen benutzen Sie Konstrukte. Die Einzelheiten des Projekts und der Implementation wird in der Übung besprochen.