Herzlich willkommen!

Dozent: Dipl.-Ing. Jürgen Wemheuer

Teil 6: Zusammenfassung und Beispiele

Mail: wemheuer@ewla.de
Online: http://cpp.ewla.de/

Zusammenfassung (1)

1. Programm in Maschinensprache (Bytefolge)

- Maschinenprogramm = Daten & Befehle (Sprache des Computers)
- Maschinenbefehl = Operation & Operand(en)
- Wie das konkrete Byte interpretiert wird, ob als Datum oder als Befehl, wird allein von der Struktur der Bytefolge im Hauptspeicher bestimmt
- Jedes Byte im Hauptspeicher ist über seine Adresse eindeutig erreichbar

2. Programm in Hochsprache (hier: C++)

2.1 Daten

2.1.1 Variable vom Typ: [unsigned] (short) int, long, char, bool, float, double

2.1.2 Konstante vom Typ: Literal, Symbol, Aufzählung (enum)

2.1.3 Array vom Typ: [unsigned] (short) int, long, char, bool, float, double

Zusammenfassung (2)

2.2 Befehle

2.2.1 Anweisungen

- Ausdrücke und Anweisungen mit verschiedenen Operatoren und Klammern

2.2.2 Verzweigungen

- Definiert als if- und switch- Anweisung

2.2.3 Schleifen

- Definiert als for-, while und do-while-Schleife, oft gefolgt von {Block}

2.2.4 Funktionen

- Deklariert durch: Prototyp - Semikolon
- Aufruf durch: Funktionsname
- Definiert durch: Quellcode der Funktion

Beispiele: Daten (1)

1. Variable
   - Die Erzeugung einer Variablen erfolgt durch ihre Definition
   - Definition: Typ - mind. ein Leerzeichen - Name - Semikolon
   - Beispiel: `long int Var1;  // 4 Byte-Reservierung im HS`
   - Beispiel: `double Var2;  // 8 Byte-Reservierung im HS`
2. Array
   - Die Erzeugung eines Feldes erfolgt durch seine Definition
   - Definition: Typ - mind. ein Leerzeichen - Name - [Index] - Semikolon
   - Beispiel: `int Projekt[9];`

Beispiele: Daten (2)

3. Konstante

- Die Erzeugung einer Konstanten erfolgt durch ihre Definition
- Definition des Literals:
  Typ - Leerzeichen - Name - Gleichheitszeichen - Zahlenwert - Semikolon
  - Beispiel: `int AnzTage = 77;`
    - Die 77 ist eine literale Konstante
- Definition des Symbols als „Präprozessor-Konstante“:
  #define - Leerzeichen - Name - Leerzeichen - Wert (- und kein Semikolon!)
  - Beispiel: `#define TageProProjekt 99`
    - Der Präprozessor setzt überall im Programm, wo **`TageProProjekt`** steht, die Zahl 99 in den Quellcode
      ~> der Compilerlauf sieht in der 99 ein Literal
- Definition des Symbols als „const-Variable“:
  Bei Variablen-Deklaration "const" davor und Initialisierung dahinter
  - Beispiel: `const int AnzTage = 77;`

Beispiele: Befehle (1)

1. Anweisungen
   - Ausdruck: Wert = Vari1 + Vari2;
   - Beispiel: **`Saldo = Haben - Soll;`**
2. Verzweigungen mit if und switch
   - if-Anweisung:
     - **if** (Ausdruck einer Bedingung) Anweisung1; **else** Anweisung2;
   - switch-Anweisung:
     - **switch** (Ausdruck) {
       case Wert1: Anweisung; **break**;
       case Wert2: Anweisung; **break**;
       ...
       case Wertn: Anweisung; **break**;
       default: Anweisung; }

Beispiele: Befehle (2)

3. Schleifen mit for, while und do - while
   - for-Schleife:

     ```
     for(Initialisierung; Test; Inkrementierung) Anweisung;
     ```

     oder

     ```
     for(Initialisierung; Test; Inkrementierung) {Block;}
     ```

   - while-Schleife:

     ```
     while (Test) {Blockanweisung;}
     ```

   - do-while-Schleife:

     ```
     do {Blockanweisung} while (Test);
     ```

Beispiele: Befehle (3)

4. Funktionen

- Deklariert durch: Rückgabetyp - Leerzeichen - Name - (Typ Parameter);
- Das Deklarieren einer Funktion bedeutet, ihren Prototyp anzugeben
- Definiert durch:
  Funktionskopf = Prototyp ohne Semikolon und -Rumpf
- Prototyp:

```
float StrahlenSatz (float A, float B, float C);     // oder
float StrahlenSatz (float, float, float);
/* die Namen der Variablen müssen nicht angegeben werden */
```

- Definition:

```
float StrahlenSatz (float D, float E, float F) {
  Anweisungen;
  return Wert; }
```

- Aufruf der Funktion: `Z = StrahlenSatz(G, H, I) + K;`
- Der Aufruf einer Funktion ist ihre Benutzung. Der Rückgabewert ist der Wert, der für die Funktion in der aufrufenden Anweisung eingesetzt wird.
- Aufruf der Funktion:

```
Y = StrahlenSatz() - M; // Aufruf ohne Übergabe von Parametern
```

Ende