Grundlagen wissenschaftlichen Arbeitens

Jens KNOOP

An Overview of C++

Autor: Ali CICEK
Matr. Nummer: 0327617
Kennzahl: 534

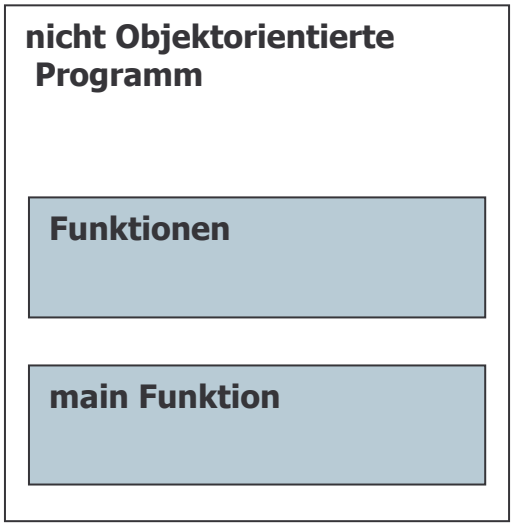
Kurzfassung:

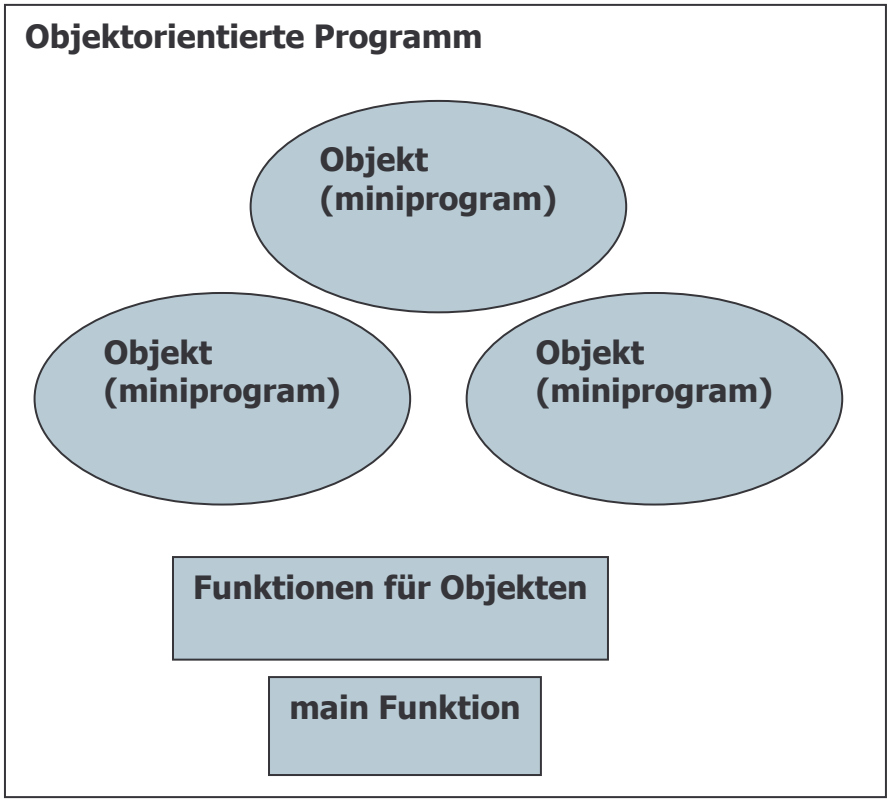
Die Programmiersprache C++ wurde von 1979 bis 1982 von Bjarne STROUSTRUP entwickelt. Die erste Erweiterung von C++ ist im Jahre 1979 in New Jersey begonnen worden. Sie ist eine Erweiterung der Programmiersprache C und besteht aus der Programmiersprache C mit kleinen ausnahmen.

Erste Version von Sprache ist unter dem Namen „C with Classes" benutzt worden. „C with Classes" ist in 1983 als „C++" geändert worden. In 1985, in 1990 und in1994 wurde C++ Versionen weiterentwickelt.

Allgemeine Eigenschaften von C++:

- C++ ist ein Object Oriented Programme.





- bessere Modularisierung des Codes.
- höhere Wartbarkeit und Wiederverwendbarkeit der Einzelmodule.
- höhere Flexibilität des Programmes.

OOP teilt das Programm in mehrere Teile auf, welche jeweils den Code und die Daten enthalten.

- C++ ist eine „Multiparadigmen-Sprache". Das heißt, daß man nicht auf ein bestimmtes Programmierparadigma, sondern jeweils das Paradigma auswählen kann, das für das jeweilige Problem am geeignetsten ist. Sie lässt dem Programmierer sehr viele Freiheiten.

- C++ unterstützt Objektorientierte Programmierung, Prozedurale Programmierung, Modulare Programmierung, Strukturierte Programmierung, Programmierung mit selbstdefinierten Datentypen und Generische Programmierung.

Ein einfaches Beispiel einen Einblick in die Programmiersprache C++:

Das ***Hello World!*** Programm

```
#include <iostream>

using namespace std;

int main ()

{

    cout << "Hello World!";

    return 0;

}
```

Am Anfang des Programs steht im C Programm <*stdio.h*> aber denken Sie daran hier <*iostream*> einzugeben. Sie ersparen sich in dem Fall auch die Verlängerung .h, weil <*iostream*> eine moderne Datei ist.

Die Datei <*iostream*> wird durch den Befehl *#include* eingebunden. In dieser Datei ist die Deklaration der Funktionen zur Ein- und Ausgabebehandlung abgelegt. Ohne diesen *include*-Befehl wäre die Ausgabeanweisung *cout* << nicht verwendbar.

Der Befehl *#include* muss am linken Rand der zu kompilierenden Datei stehen, da dieser Befehl sonst nicht ausgeführt wird.

Namespaces benötigt man, um große Programme zu organisieren.

Jedes C++ Programm muss eine Funktion namens *main*( ) enthalten. Die *main* ( ) Funktion ist deshalb so wichtig, weil mit dieser Funktion die Ausführung des Programms beginnt.

Zuerst setzt man den Funtionskopf auf. Wie dieser auszusehen hat, ist mehr oder weniger vorgegeben. Hier verwenden wir *integer.*

Bevor Sie ein Element aus einer Bibliothek verwenden können, müssen Sie das Element beim Compiler bekannt machen- im Fachjargon sagt man „deklarieren". Hier verwenden wir den Datentyp int.

C++ verlangt nämlich, dass die main( )Funktion mit einer return-Anweisung beendet wird, die einen Ergebniswert zurückliefert.

Mittels cout << werden Ausgaben vorgenommen, und << Operator ist ein Ausgabeoperator.

Elementare Datentypen

Datentyp	Bedeutung	Große
char	Zeichen	8 Bit
int	Integer	16 Bit
short	Integer	8 Bit
long	Integer	32 Bit
float	Fließkomma einfache Genauigkeit	32 Bit
double	Fließkomma doppelte Genauigkeit	64 Bit

if Bedingung

```
if (Bedingung)
{
  Anweisung(en);
}
```

„Wenn die Bedingung erfüllt ist, dann (und nur dann)führe die Anweisungen aus."

Bei Ausführung des Programms wird zuerst die Bedingung ausgewertet. Die Bedingung ist dabei nichts anders als ein Vergleich. Liefert der Vergleich als Ergebnis *true* (wahr), ist die Bedingung erfüllt und der zur *if* –Bedingung gehörende Anweisungsblock wird ausgeführt; andernfalls wird das Programm mit der nächsten Anweisung hinter dem Anweisungsblock fortgesetzt.

Auf die *if*-Bedingung kann eine einzelne Anweisung oder ein Anweisungsblock folgen.

Eine einzelne Anweisung muss immer mit einem Semikolen beendet werden. Wenn ein Anweisungsblock ausgeführt werden soll, muss dieser in geschweifte Klammern eingeschlossen werden.

if– else Bedingung

```
if (Bedingung)
{
   Anweisung1;
}
else
{
   Anweisung2;
}
```

„Wenn die Bedingung nicht erfüllt ist, dann führe die else Alternative."

if–else Beispiel

Wenn man mehr als zwei Fälle unterscheiden möchte, kann man if-else Bedingung verwenden. Zuerst lesen Sie die Note der Prüfung in eine *int* – Variable Note ein. Dann verzweigen Sie wie :

```
int note;

if (note > 50) {
        cout <<"positives Zeugnis"<< endl;
}
else if (note >= 45) {
        cout <<"2. Chance"<< endl;
}
else {
        cout <<"negatives Zeugnis"<< endl;
}
```

Wenn die Note großer als 50 ist, wird das Ergebnis pozitives Zeugnis geworden.
Wenn die Note großer gleich 45 ist, wird das Ergebnis 2. Chance geworden.
Wenn die Note kleiner gleich 45 ist, wird das Ergebnis negatives Zeugnis geworden.

Die *switch* Verzweigung

Will man die Programmausführung in Abhängigkeit von Wert einer Variablen mehrfach aufspalten, bietet sich dazu eine *switch* –Konstruktion an.

```
switch (Ausdruck)
{
        case Konstante1:Anweisungen ;
                        break ;
        case Konstante2:Anweisungen ;
                        break ;
        case Konstante3:Anweisungen ;
                        break ;
        case Konstante4:Anweisungen ;
                        break ;
        default:  Anweisung ;
}
```

Bei *switch* – Anweisung wird die Bedingung ausgewertet und mit den Konstanten verglichen. Wird eine Konstante_i gefunden, die mit dem Wert der Bedingung übereinstimmt, so wird die entsprechende Anweisung_i ausgeführt. Wird keine solche Konstante gefunden, wird die *default* – Anweisung ausgeführt.

switch Beispiel

```
switch (x)
{
 case 1: cout << "x is 1";
         break;
 case 2: cout << "x is 2";
         break;
```

```
    case 3: cout << "x is 3";
            break;
    default: cout << "value of x unknown";
}
```

Die *for* Schleife

Schleifen dienen dazu, einen Anweisungsblock mehrfach hintereinander auszuführen. Sicherlich macht es keinen Sinn, einer Variablen x-mal hintereinander den gleichen Wert zuzuweisen oder x-mal nacheinander die gleiche Ausgabe auf den Bildschirm zu schreiben.

```
for(Initialisierung; Bedingung; Veränderung)
{
    Anweisung(en);
}
```

Initialisierung : Code, der beim ersten Eintritt in die Schleife ausgeführt wird.
Bedingung: Solange die Bedingung erfüllt ist, wird der zur *for* – Scleife gehörende Anweisungsblock ausgeführt; andernfalls wird die Schliefe verlassen.
Veränderung: Die Anweisung dieses Teils wird nach jedem Schleifendurchgang und vor der neuerlichen Prüfung der Schleifenbedingung ausgeführt.

Die drei Teile *Initialisierung, Bedingung* und *Veränderung* bilden zusammen den so genannten *Schleifenkopf*. Sie enthalten Code, der festlegt, wie ort die Schleife ausgeführt wird.

for Beispiel

```
#include <iostream>
using namespace std;
int main ()
{
int n;
  for (n=10; n>0; n--) {
      cout << n << ", ";
  }
  cout << "FIRE!";
  return 0;
}
```

Die Ausgabe:

10, 9, 8, 7, 6, 5, 4, 3, 2, 1, FIRE!

Der Code beginnt mit der Deklaration der Schleifenvariablen, die hier n genannt wurde. Bei Eintritt in die Schleife muss der Schleifenvariablen ein Anfangswert zugewiesen werden. Dies geschieht in der Anweisung : n=10;

Dahinter ist die Bedingung formuliert, die festlegt, wie lange die Schleife ausgeführt wird: n>0; .Solange die Bedingung erfüllt ist, das heißt, solange die Schleifenvariable *n* großer als 5 ist, wird die Schleife ausgeführt.

n-- sinkt den Wert in jeder Ausführung 1.

Die *while* Schleife

Bei der while – Schleife sind nicht im Schleifenkopf zusammengefasst, sondern werden über den Code verteilt.

```
Initialisierung;
while (Bedingung)
{
   Anweisung(en)  inklusive Veränderung;
}
```

```
int j=1;
while(j<10)
{
      cout<<j<<" ";
      j++;
}
```

Die Ausgabe:

1 2 3 4 5 6 7 8 9

Was auf diese Weise zustande kommt: Da j zunächst den Wert 1 besitzt, ist die Schleifenbedingung erfüllt (TRUE) und die Anweisungen der Schleife werden

ausgeführt. Wenn die Schleifenbedingung erfüllt (FALSE) und da i nun nicht mehr kleiner als 10 ist.

Continue (Schleifen Durchgang)

Die continue – Anweisung wird nur selten verwendet. Sie veranlasst den sofortigen Beginn der nächsten Schleifeniteration der innersten einschließenden for-, while- .

Für die ersten 5 Schleifendurchgänge wird der Anweisungsblock ganz normal durchgeführt. Zu beginn des 6.Durchgangs ergibt die if – Bedingung aber true und die continue- Anweisung unter der if – Bedingung wird ausgeführt. Die continue – Anweisung sorgt dafür, dass die nachfolgenden Anweisungen im Schleifenblock nicht mehr ausgeführt werden. Stattdessen wird die Schleife mit der letzten Anweisung im Schleifenkopf (n--) und dem Eintritt in den nächsten Schleifendurchgang fortgesetzt.

continue Beispiel

```
#include <iostream>
using namespace std;
int main ()
{
    for (int n=10; n>0; n--) {
        if (n==5) continue;
        cout << n << ", ";
    }
    cout << "FIRE!";
    return 0;
}
```

Die Ausgabe:

10, 9, 8, 7, 6, 4, 3, 2, 1, FIRE!

break (Schleife abbrechen)

Die *break* – Anweisung wird grundsätzlich genauso eingesetzt wie die *continue*- Anweisung. Sie bricht aber nicht nur den aktuellen Schleifendurchgang, sondern gleich die ganze Schleife ab.

Die break – Anweisung wird auch in der switch – Verzweigung verwendet.

Die break – Anweisung wird meist dann verwendet, wenn Sie Aufgaben dieser Form zu lösen haben:

„Führe die folgenden Anweisungen immer wieder aus, bis ein bestimmtes Ereignis eintritt."

break Beispiel

```
#include <iostream>
using namespace std;
int main ()
{
    int n;
    for (n=10; n>0; n--)
    {
        cout << n << ", ";
        if (n==3)
        {
          cout << "countdown aborted!";
          break;
        }
    }
return 0;
}
```

Die Ausgabe:

10, 9, 8, 7, 6, 5, 4, 3, countdown aborted!

Vor- und Nachteile

+ Man kann hocheffizienten Codes erzeugen.
+ Man kann mit C++ auch hochabstrakt programmieren.
+ Flexibilität und Ausdruckstärke sind sehr hoch.
+ C++ ist geeignet für die große Projekt.
+ C++ ist kompatible mit C. (breite Codebasis zur verfügung)
- C++ ist nicht unabhängig von dem verwendeten Betriebsystem

Weiter führende Literaturen:

Objektorientierte Programmierung mit C++
Volker Hillmann
ISBN: 3-87791-558-2

Einführung in die Programmiersprache C++
Falko Bause – Wolfgang Tölle
ISBN: 3-528-04689-9

Das Grundlagen Buch C++
Gerhard Williams
ISBN: 3-8158-1437-5

In C++ denken
Bruce Eckel
ISBN: 3-8272-9506-8