1. Einführung

1.1.C++ und Java

C++ stellt heute, zu Beginn des 21. Jahrhunderts, die wichtigste Programmiersprache für die Entwicklung leistungsfähiger Anwendungen dar. So ist C++ beispielsweise die am meisten verwendete Programmiersprache für die Entwicklung professioneller Anwendungen für Microsoft Windows.

Als Gründe für die Beliebtheit von C++ können die hohe Ausführungsgeschwindigkeit, große Flexibilität und einfache Bewältigung komplexer Aufgabensituationen genannt werden. Auch die Kompatibilität zur Programmiersprache C spielt eine entscheidende Rolle.

Seit dem Aufkommen der Programmiersprache Java Mitte der 90er Jahre versucht Java, C++ den Rang abzulaufen. Der große Vorteil von Java-Programmen ist die hohe Portabilität. Während Java-Programme einmal erstellt unter allen Betriebssystemen funktionieren, werden C++-Programme für ein Zielsystem entwickelt: So läuft ein C++-Programm entweder auf einem Microsoft-Windows-PC mit Intel-Chip oder aber auf einem Apple-Macintosh mit PowerPC-Prozessor.

In den letzten Jahren ist der Aspekt der Portabilität immer wichtiger geworden. Java bietet hier die ideale Plattform an, um portable Anwendungen zu erstellen - Programme, die überall laufen. Der Werbespruch der Firma Sun Microsystems, die Java erfunden hat, lautet auch "Write Once - Run Anywhere" (also einmal programmieren, überall laufen lassen). Während Java gegenüber C++ eindeutig den Vorteil einer größeren Portabilität besitzt, sind C++-Programme leistungsfähiger. So kann man davon ausgehen, dass ein C++-Programm ganz grob gesagt zehnmal schneller als eine Java-Anwendung läuft.

1.2.C und C++

Unter C++ versteht man erst einmal die Syntax und Semantik einer Programmiersprache. Das heißt, C++ definiert eine Reihe von Schlüsselwörtern und legt fest, welche Bedeutung die einzelnen Schlüsselwörter besitzen. Dies ist die eigentliche Definition einer Programmiersprache.

C++ als Programmiersprache wurde vor langer Zeit von *Bjarne Stroustrup* basierend auf der Programmiersprache C entwickelt und ist schon lange de facto standardisiert. Das heißt, Sie können Ihr C++-Programm normalerweise mit jedem beliebigen C++-Compiler übersetzen - egal, von welchem Hersteller der Compiler stammt, egal, unter welchem Betriebssystem er läuft.

Letzteres Kriterium bedeutet jedoch gleichzeitig, dass in C++ Programme geschrieben werden können, die lediglich auf dem kleinsten gemeinsamen Nenner aller Betriebssysteme basieren. So können Sie beispielsweise in C++ zwar Daten auf den Bildschirm ausgeben, dies erfolgt jedoch ausschließlich Zeichen-orientiert. Grafische Anwendungen können rein mit der

Programmiersprache C++ nicht erstellt werden, da es Betriebssyteme gibt, die keine grafischen Oberflächen zur Verfügung stellen. Der kleinste gemeinsame Nenner ist also die Textausgabe - mehr können Sie daher auch in C++ erstmal nicht machen.

Seit 1997/98 gibt es einen offiziellen C++ Standard. Dieser Standard bezieht sich zum einen auf die Syntax und Semantik der Programmiersprache C++. Dies ist der eher uninteressantere Teil des Standards, weil hier eigentlich nur die bis dato bereits weit verbreitete und allgemeingültige Syntax von C++ von offizieller Seite bestätigt wurde. Es gibt zwar ein paar besondere Neuheiten zum bisherigen de-facto-Standard, aber diese Erweiterungen halten sich in Grenzen und sich leicht überschaubar.

Der viel wichtigere Teil des Standards ist die C++ Standard Library. In dieser Library sind nützliche Funktionen, Klassen und Objekte zusammengefasst und standardisiert worden, um diese Hilfsmittel allen C++-Programmierern in einheitlicher Form zur Verfügung zu stellen. Es handelt sich also bei diesem Standard um eine Art Werkzeugkasten, auf den jeder C++-Programmierer in seinen Programmen zugreifen kann.

1.3.Compiler und Linker

Um ein C++-Programm zu schreiben, wird der Quellcode einfach in eine Text-Datei (genauer ASCII-Datei) gespeichert. Während der Mensch aber Quellcode schreiben und lesen kann, versteht der Computer nur Maschinencode. Das heißt, der Quellcode muss in Maschinencode übersetzt werden, damit das Programm vom Computer überhaupt ausgeführt werden kann.

Eine Übersetzung von Quellcode in Maschinencode findet in C++ in zwei Schritten statt: Zuerst wird der Quellcode durch den Compiler in Objektcode übersetzt, dann wird der Objektcode durch den Linker in eine ausführbare Datei umgewandelt.

Im ersten Schritt übersetzt der Compiler den von Ihnen geschriebenen C++-Quellcode direkt in den äquivalenten Maschinencode. Der Compiler geht Schritt für Schritt durch den Quellcode und übersetzt beispielsweise bestimmte Anweisungen in C++ durch fest definierten Maschinencode. Die Datei, die vom Compiler nach getaner Arbeit erzeugt wird, heißt Objektdatei. Während C++-Quellcode normalerweise in Dateien mit der Endung .cpp gespeichert wird, erhalten Objektdateien standardmäßig die Endung .o oder .obj.

Sie greifen aber normalerweise in ihren Programmen auf Hilfsmittel, die Sie nicht selber programmiert haben. Diese sind normalerweise Bibliotheken. Eine Bibliothek ist nichts anderes als eine Datei, in der lauter Funktionen definiert sind. Eine Bibliothek liegt hierbei nicht als Quellcode vor, sondern als Maschinencode.

Wenn Sie also eine Funktion aus einer Bibliothek aufrufen, kann der Compiler gar nicht den dazugehörigen Quellcode übersetzen, weil Sie ja gar nicht diese Funktion programmiert haben.

Der Linker merkt nun, dass zu der besagten Funktion noch der Maschinencode fehlt, und kopiert den entsprechenden Maschinencode nun einfach aus der Bibliothek in die Objektdatei hinein. Hat der Linker den Maschinencode für alle Funktionen kopiert, die ebenfalls nicht von Ihnen programmiert wurden und in anderen Bibliotheken liegen, erhalten Sie eine ausführbare Datei. Diese Datei ist Ihr fertiges Programm in Maschinencode. Im Betriebssystem Windows erkennen Sie derartige Dateien an der Endung .exe, unter Unix ist das eXecute-Flag gesetzt. Der Computer kann nun Ihr C++-Programm ausführen.

1.4.Das erste Programm – etwas interessanteres als Hello World!

Nun sehen wir uns ein kleines Programm an und versuchen, es zu verstehen und eventuell auch zu verändern.

Beispiel 1

```
#include <iostream>

void main() {
   int alter;

   // Ausgabe der Begruessung
   std::cout << "Hallo! Wie alt bist du denn?" << "\n";

   // Eingabe des Alters
   std::cin >> alter;

   // Ausgabe der Reaktion
   if (alter < 10 ) {
         std::cout << "Was machst du ueberhaupt in einem C++ Kurs? Gehörst du
nicht in den Kindergarten zu der Zeit?!" << "\n";
         return;
   }

   if (alter > 40 ) {
         std::cout << "Du alter Sack! Du wirst ja sowieso nichts kapieren, das
hier ist nur für junge, aufgeschlossene, intelligente Leute..." << "\n";
         return;
   }

   std::cout << "Willkommen im coolsten C++ Kurs!" << "\n";
   return;
}
```

Frage. Was macht das Programm? Bei welchen Eingaben wird es mit welchen Ausgaben reagieren?

Einzelne Teile des Programms:

- *#include* Anweisung
- *void main()*
- *std::cout und std::cin*

- *if () {}*
- *return*
- *int alter*
- Kommentare

Frage. Wann wird der Text **"Willkommen im coolsten C++ Kurs"** ausgegeben? Warum? Warum nicht immer (es steht kein if davor)?

Mit dem Programm experimentieren – Compilieren, Linken, Ausführen, Testen.

1.5.Ein- und Ausgabe mit der Standard-Bibliothek

Die einfachste Ein- und Ausgabe mit C++ ist die der Standard Library. Dafür benutzt man, wie im Beispiel oben, std::cout und std::cin. Andere I/O Operationen und Möglichkeiten werden wir später im Kurs sehen.

Wenn man etwas auf dem Standard Output ausgeben möchte, muss man cout benutzen. Hier ein paar Beispiele dafür:

Beispiel 2

```
#include <iostream>

void main() {

   int number = 13;

   std::cout << "Hallo!";        //String-Ausgabe (Text)
   std::cout << "\n\n\n";        //Spezielle Zeichen (New Line)
   std::cout << 32;              //Konstanten
   std::cout << "\n";
   std::cout << number;          //Variablen
   std::cout << "\n";
   std::cout << number * 2 + 1; //Ausdruck mit Variablen und Konstanten
   std::cout << "\n";
   std::cout << "Das ist eine Kombination aus allem " << number << " " << number*2
<< "\n";

   return;
}
```

1.6.Elementare Typen, Variablen und Konstanten

Warum brauchen wir unterschiedliche Typen?

Intern im Computer werden alle Informationen als Bitfolge dargestellt. Auf einem Computer entspricht das Wort "DACH" dem Bitmuster 01001000010000110100000101000100. Dieses Bitmuster entspricht, wenn es als Fließkommazahl (Floating Point) interpretiert wird, der Zahl 199941. Als ganze Zahl (Integer) entspricht es 1212367172. Daher muss der Compiler immer wissen, welchen Typ wir benutzen, damit er die Zahl in die richtige Bitfolge übersetzen kann.

Im Moment brauchen wir nicht so viele Typen zu kennen. Wir brauchen nur ein paar, so dass wir weiterarbeiten können und kurz experimentieren können. Die wichtigsten (die meistgenutzten) Typen von Variablen sind

- char
- int
- float

Hier ein paar Beispiele für die Definition der Variablen und Konstanten mit diesen Typen:

Beispiel 3

```
#include <iostream>

#include <iostream>

void main() {

   int num_1 = 13;
   const int num_2 = 24;

   float fl_1 = 1.2f;
   const float fl_2 = 3.14f;

   char ch_1 = 'a';
   const char ch_2 = 'z';

   std::cout << num_1;
   std::cout << "\n";
   std::cout << num_2;
   std::cout << "\n";
   std::cout << fl_1;
   std::cout << "\n";
   std::cout << fl_2;
   std::cout << "\n";
   std::cout << ch_1;
   std::cout << "\n";
   std::cout << ch_2;

   return;
}
```

1.7.Elementare Funktionen

Hier haben wir nur eine Funktion gesehen – main. Ein ausführbares Programm in C++ muss immer diese Funktion beinhalten – genau sie wird ausgeführt. Die Definition einer Funktion sieht folgendermassen aus:

<rückgabetyp> name (<parametertyp1> parameter1, ...) {

//hier passiert etwas...

}

1.8.Aufgaben

1.8.1. Verändern Sie das Programm beispiel1_1.cpp so, dass es auf andere Werte des Alters reagiert. Fügen Sie eine Anfrage über den Namen und dem Geschlecht zu. Verändern Sie und erweitern Sie die Reaktionen mit Hilfe von verschachtelten if-s.

1.8.2. Definieren Sie verschiedene Variablen und Konstanten der Typen char, int und float. Experimentieren Sie damit. Versuchen Sie, in Ausdrücken Variablen und Konstanten unterschiedlicher Typen zu benutzen! Was passiert?

char + char

char + float

int + float

char + int

1.8.3. Programmieren Sie einen Taschenrechner mit den Operationen Addition, Subtraktion, Multiplikation, Division, Modulo-Operation usw. Benutzen Sie dafür die Ein- und Ausgabe von Standard Library (cout und cin). Das Programm soll zuerst das eine Operand einlesen, dann die Operation und dann das zweite Operand. Es soll das Ergebnis berechnen und warten, bis der Benutzer eine Taste drückt und dann enden.

1.8.4. Schreiben Sie ein Programm, dass den Volumen von verschiedenen Körpern berechnet. Das Programm muss mit einem Menu anfangen, dass die unterschiedlichen Körpern auflistet. Davon muss der Benutzer einen Körper auswählen und dann die Massen eingeben.

1.8.5. Programmieren Sie Tic-Tac-Toe mit 9 Feldern. Das Programm soll die Zustände der Felder (0 – für frei, 1 – für Computer und 2 – für Spieler) in 9 verschiedenen Variablen speichern. Nach jeder Eingabe des Spielers soll das Programm das aktuelle Spielfeld anzeigen.