C / C++

Andreas Schweiner
Matrikelnummer: 0425776
andreas_schweiner@hotmail.com

Einleitung:

In den folgenden Kapiteln werden die beiden Programmiersprachen C und C++ beschrieben. C und C++ sind objektorientierte, allgemein verwendbare Programmiersprachen. Neben der geschichtlichen Entwicklung, den Anwendungsgebieten und der Syntax werden vor allem Besonderheiten an Hand von einem kurzen Beispiel erklärt.

Geschichtliche Entwicklung und Motivation:

Mitte der 60er kreierte Martin Richards die Sprache BCPL, welche für die Entwicklung verschiedener Betriebssysteme verwendet wurde (z.B. OS6 Operating System). Ken Thompson wollte im Jahre 1969 in den Bell Laboratories am MIT für die DEC PDP-7 eine Systemprogrammiersprache (für Unix) entwickeln. Er wandelte sie von BCPL ab und taufte sie B. Dennis M. Ritchie begann 1971 die neue Sprache B zu erweitern, sie wurde C genannt. 1973 war C weit genug entwickelt und seine Compiler stark genug, um
damit einen neuen Unix-Kernel für die PDP-11 zu schreiben. Für andere Rechner wurden ebenfalls C-Compiler gebaut und Bibliotheken entwickelt. Die Veröffentlichung von *'The C Programming Language'* (Kerningham und Ritchie) erfolgte im Jahre 1978. C und die dazugehörige Standardbibliothek wurde 1982 durch das ANSI X3J11-Komitee standardisiert.

Im Jahre 1979 wollte Bjarne Stroustrup impulsgesteuerte Simulationen schreiben. Simula67 war dafür eigentlich ideal, benutzte er allerdings aus Effiziensgründen nicht. Somit entwickelte er '*C with classes*', eine
Programmiersprache, die C um das Klassenkonzept aus Simula67 bereicherte. Sie wurde von Stroustrup und seinen Kollegen weiterentwickelt, ohne dabei die Implementierung (des Übersetzers) durch irgendeine
Sprache zu beachten. Ziel war es, das Programmieren komfortabler zu gestalten. Das Resultat war die neue Sprache C++. 1985 veröffentlichte Stroustrup '*The C++ Programming Language*'. Da C++ immer komplexer wurde und stetig an Popularität gewann, war es erforderlich die Sprache zu standardisieren. Es fand durch die Anregung von Hewlett Packard 1990 das erste ANSI X3J16 Treffen statt. 1998 wurde der internationale Standard durch ISO/IEC festgelegt: '*Programming Language - C++*'.

Charakterisierung:

Es gibt in C keine komplexe Datenstrukturen (Listen, Mengen, o.ä.), sondern nur Objekte, mit denen der Computer selbst umgeht (Zeichen, Zahlen, Zeiger). C ist eine maschinennahe Sprache mit wenigen Schnittstellen zum Betriebssystem. Anweisungen für den Zugriff auf solche (z.B. Ein- und Ausgabe, Dateizugriff), werden durch Funktionen (z.B. printf(), fopen()) aus mitgelieferten Bibliotheken realisiert. Das hat zur Folge, dass der Kern von C sehr klein ist und C somit portabel und effizient.

C++ ist eine Erweiterung von C und beinhaltet diese bis auf wenige Ausnahmen als Teilsprache. Daher ist die Modularisierung (z.B. Trennung von Deklarationen und Definitionen in verschiedene Dateien) aus C erhalten und weitergeführt (namespaces). Zusätzlich zum Klassenkonzept und der Vererbung aus der Sprache

Simula67 wurden weitere Konzepte realisiert: Generik (*Templates*, inspiriert durch Ada, Clu), Ausnahmebehandlung (exception handling: Ada, Clu, ML). Das Konzept des Polymorphismus (virtuelle Funktionen) und der Mehrfach-Vererbung wurden eigenständig bei der Benutzung von C++ umgesetzt und realisiert. Genau wie C besitzt die Sprache C++ keine komplexen Datenstrukturen, jedoch wird mit dem Übersetzer eine mächtige Bibliothek geliefert. Teil der Bibliothek ist die *Standard Template Library* (STL), in der so genannte Container zur Speicherung und effizienten Suche zur Verfügung stehen.

Syntax und Semantik:

Die Entwicklung eines Programms lässt sich in vier Schritte aufteilen:

1. Erstellen des Quelltextes mittels ASCII-Editor:
   _ *main.c* enthält das Hauptprogramm
   _ *<file>.c* enthalten Funktionen und Variablen
   _ *<file>$.h*, die so genannten Header-Dateien, enthalten Deklarationen der für die Schnittstellen nötigen Funktionen und Objekte aus den entsprechenden *<file>.c* Dateien

2. Die Dateien (ohne Header-Dateien) werden mit einem C-Compiler in Objektdateien übersetzt und

3. durch den Linker wird ein ausführbares Programm erzeugt. Namensreferenzen im Objektcode werden

4. evtl. unter Hinzunahme von Bibliotheken aufgelöst

5. Falls kein Name spezifiziert wurde, heißt das Programm *a.out*

Ein C - Programm beginnt üblicherweise mit Deklarationen von benutzten Funktionen und Variablen (Evtl. durch Einfügen von Header-Dateien). Funktionen können stattdessen auch direkt definiert werden. Jedes Programm muss eine Funktion main() besitzen, die beim Programmstart als Erste aufgerufen wird. Die Anweisungen von Funktionen folgen in geschweiften Klammern, beginnend mit den Vereinbarungen. Nach jeder Anweisung folgt ein Semikolon. Der Funktion printf() wird eine konstante Zeichenkette (genauer: Array von konstanten Zeichen) übergeben. '\n' ist **ein** Zeichen, der Zeilenumbruch.

Die Syntax und Semantik von C++ ist genau wie in C. Es gibt aber noch viele zusätzliche Klassen, wie z.B. string, die jedoch Teil der Bibliotheken und nicht der eigentlichen Programmiersprache sind. Eine Ausnahme bilden die Referenzen. Werte können mit echtem '*call-by-refernce*' übergeben werden:
In C++ ist es möglich eigene Datentypen als Klassen zu definieren. In C würden dafür Strukturen und Funktionen, welche diese Strukturen benutzten, definiert werden. In C++ steht das Objekt im Vordergrund.

Kontrollstrukturen:

Wie auch in anderen Sprachen gibt es auch in C/C++ Kontrollstrukturen die den Ablauf von Programmen steuern. In C/C++ gibt es z.B. Verzweigungen mit „if", „else if" oder „else" Aufrufen. Ebenfalls gibt es „while" oder „for" – Schleifen und „case" Verzweigungen.

Anwendungsgebiete von C++:

C++ ist eine allgemein verwendbare Programmiersprache
Das hauptsächliche Anwendungsgebiet ist die Systemprogrammierung im weitesten Sinne. Ein neues Betriebssystem, welches vor kurzem auf den Markt kam, unterstützt die Programmierung von Anwendungen indem es Klassen von C++ mit Dokumentation mitliefert. Dieses Betriebssystem heißt BeOS. Ein weiterer Einsatz von C++ liegt auch in der Spieleprogrammierung.

Besonders zeitkritische Spiele wie 3D-Shooter (zum Beispiel Half Life, Far Cry, Doom III) profitieren von der hohen Geschwindigkeit von C++.

Anwendungsbeispiele und –gebiete:

Das wohl erste Programm welches man schreibt wenn man eine neue Programmiersprache erlernen möchte ist sicherlich das „Hello World", ob in C, Java oder anderen Sprachen.

1. Inhalt der ASCII-Datei main.c:

```
#include <stdio.h>
main() {
printf("Hello World!\n");
}
```

2. Übersetzt wird das Programm z.B. mit: *gcc main.c*
3. Der Linker wird bei beim Aufruf des *gcc*-Compilers automatisch aufgerufen
4. Der Aufruf der ausführbaren Datei *a.out* liefert: *Hello World!*

Der Aufwand des Erlernens von *C++* ist besonders für diejenigen angebracht, die schon wissen, dass sie sich für längere Zeit mit dem Programmieren beschäftigen wollen und dabei eher Systemprogramme als Anwendungsprogramme entwickeln wollen.

Auch systemnahe Anwendungsprogramme mit einer grafischen Benutzeroberfläche können in *C++* geschrieben werden. Doch da *C++* keine eigene Schnittstelle für graphische Benutzeroberflächen definiert (wie z.B. *Java*), sind solche *C++* -Programme weniger portabel.

Schließlich eignet sich *C++* besonders zur Implementation von Systemprogrammen, wie etwa der Prozessverwaltung eines Betriebssystems, der Implementierung eines Internetprotokolls, oder der Programmierung von Interpretern und Laufzeitsystemen für andere Programmiersprachen. So gesehen ist es auch verständlich, dass *C++* beispielsweise keine Speicherverwaltung enthält, denn *C++* ist ja dafür gedacht, solche Systeme erst zu implementieren. Um aber eine Speicherverwaltung oder ein Internetprotokoll mit *C++* zu implementieren, ist natürlich auch Fachwissen des Anwendungsgebietes erforderlich, so dass *C++* -Programmierer oft gute allgemeine Kenntnisse entsprechender Gebiete der Informatik benötigen.

Vor- und Nachteile der Sprachen:

Vorteile von C++ gegenüber C:

- Ein-/Ausgabeoperationen sind mit Objekten realisiert, der fehlerhafte Umgang mit printf() und scanf(), insbesondere die falsche Verwendung der Formatspezifizierer, entfällt.
- Das dynamische Anlegen von Elementen und Arrays ist wesentlich einfacher, sicherer mit C++ und datentypbezogener.
- Konstanten müssen nicht mehr mit Präprozessorbefehlen erzeugt werden, es gibt das Schlüsselwort const dafür. Solche Konstanten können dann auch für Arraydimensionen verwendet werden.
- Die freie Plazierung von Variablendefinitionen wird unterstützt.
- Ausnahmebehandlung ist bei Fehlern möglich und muss nicht selbst entwickelt werden.

Vorteile von C++:

- objektorientierte Programmiersprache
- leichte Übertragfähigkeit in andere Betriebssysteme
- Kürze, wenig Befehle, aber ausreichend (Knappheit dieser)
- Schnelligkeit
- Übertragbarkeit von C einfach zu realisieren, notwenig für Erweiterungen
- gutes Verständnis für Leser (übersichtlich)

Nachteile von C++

Keine Standardisierung (wie in Java) von grafischer Oberflächen, Fenster und von Grafikoperationen in C++! Man kann in C++-Programmen alle/viele OOP-Möglichkeiten ignorieren. Die wenigen echten Erweiterungen und Verallgemeinerungen von C++ sind die mehrfache Vererbung, static-Elementfunktionen und rein virtuelle Funktionen.

Zusammenfassung:

Zusammenfassend kann man sagen dass C bzw. C++ Sprachen sind die auch in der heutigen Zeit noch sehr viel Anklang finden und gerne verwendet werden. Zwar ist sind es zwei aufgrund der maschinennahen Programmierung zwei „schnelle“ Programmiersprachen, jedoch leidet darunter die Benutzerfreundlichkeit. Im Gegensatz zu Java oder anderen Sprachen ist das schreiben eines GUI doch um einiges schwieriger. Trotz alledem war C++ ein Meilenstein in der objektorientierten Programmierung.

Literaturliste:

Bjarne's FAQ
http://www.hpl.hp.com/personal/
http://dict.org
http://lexikon-definition.de
und noch einige weitere Websites zum Thema C/C++