Hochschule Hof
University of Applied Sciences

Skript zur Vorlesung

Effizientes Programmieren mit C/C++

(für Verwaltungsinformatik)

WS 2022/23

Dipl. - Inf. (FH) Stefan Müller
Hochschule Hof

Inhaltsverzeichnis

Inhalt

1 Die Entwicklung von C nach C++

Die Programmiersprache C entstand Anfang der siebziger Jahre im Zusammenhang mit dem Betriebssystem Unix in den AT&T Bell Laboratories. Eine wesentliche Rolle spielte dabei, Ende der siebziger Jahre die Entwicklung des Betriebssystem Unix (Version 7), dessen Kernel von Dennis Ritchie mit der Programmiersprache C anstatt wie bisher üblich in Assembler programmiert wurde. Aus diesen Erfahrungen heraus entstand das 1978 erschienene Buch " Brian W. Kernighan und Dennis M. Ritchie, „The C Programming Language“, Prentice Hall, Englewood Cliffs N.Y. (1978)", das lange Zeit als Quasi-Standard (sog. K&R-Standard) für die Programmiersprache C galt. Die in diesem Buch gemachten Festlegungen bezüglich Syntax und Semantik der Programmiersprache C stellten den kleinsten gemeinsamen Nenner für effektive C-Programme dar.

Erst im Jahre 1988 hat das ANSI-Komitee (*ANSI – American National Standards Institute*) den ANSI-C Sprachstandard für die Programmiersprache C veröffentlicht. C-Compiler sollten einem klar definierten ANSI-Standard entsprechen. Da die Softwarehersteller gerne proprietäre Erweiterungen in Ihren Produkten verwenden, um sich und ihr Produkt von anderen Herstellern herauszuheben. Die ANSI-C Kompatibilität kann optional durch Compileroptionen oder ein „Häkchen“ in den Optionseinstellungen der integrieten Entwicklungsumgebungen erreicht werden.

Aufgrund dieses Entstehungsprozesses besitzt die Programmiersprache C folgende Vorteile:

- C ist eine kompakte Sprache. Sie ist eng in der Implementierung an Assembler angelehnt und bietet daher eine hohe Performance der ausführbaren Programme.
- In C geschriebene Programme sind effizient bezüglich ihrer geringen Programmgröße.
- C ist sehr flexibel, d.h. C lässt dem Programmierer einen breiten Spielraum. C wurde für das strukturierte Programmieren konzipiert.
- C ist im Vergleich zu anderen Sprachen sehr gut standardisiert und deshalb sind Quellcodedateien auf Zielsysteme, die einen ANSI-C Compiler anbieten, portierbar.
- C-Compiler existieren für alle wesentlichen Rechnerplattformen.
- C ist Bestandteil der MAC-OSx (clang) und Unix-Programmierumgebung (gcc/g++)
- C wird häufig im Umfeld von IoT verwendet, steht somit auch auf SBC’s zur Verfügung (Rasperry & Co.)

In einer ersten Version wurde C++ „C with Classes“ genannt.Diese Version wurde in den 80er Jahren von Bjarne Stroustrup aus C weiterentwickelt. C++ ist als die nächste Generation der Programmiersprache C, gewissermaßen als eine Obermenge zu verstehen. Dabei wurde C vorallem um das Konzept der Objekt-Orientierung erweitert.

Bjarne Stroustrup begann 1979 mit seiner Arbeit an “C with Classes”, nachdem er bei einer großen Simulationsaufgabe, die in Simula entwickelt worden war, Probleme hatte.

Er fügte zunächst folgende c++-Sprach-Konstrukte hinzu:
- Klassen
- Konstruktoren und Destruktoren
- Vererbung (noch ohne Polymorphismus)
- Typüberprüfung

Die Implementierung von “C mit Klassen” bestand in Form von Präprozessor-Anweisungen, die Makros genannt werden. Sie werden durch einen Präprozessor, der beim Umwandlungsprozess ***vor*** dem Compiler ausgeführt wird, aufgelöst. Der Präprozessor erzeugt aus dem “C mit Klassen”-Sourcecode durch Textersetzung einen Sourcecode in C-Syntax, der dann anschließend fehlerfrei durch den C-Compiler in Objektcode übersetzt werden kann.

Ab 1982 führte Bjarne Stroustrup die Entwicklung von “C mit Klassen” weiter, welche 1990 in einem ersten Standard von C++ (V3.0) mündete. In C++ umfasste in diese Version folgende C++-Erweiterungen:

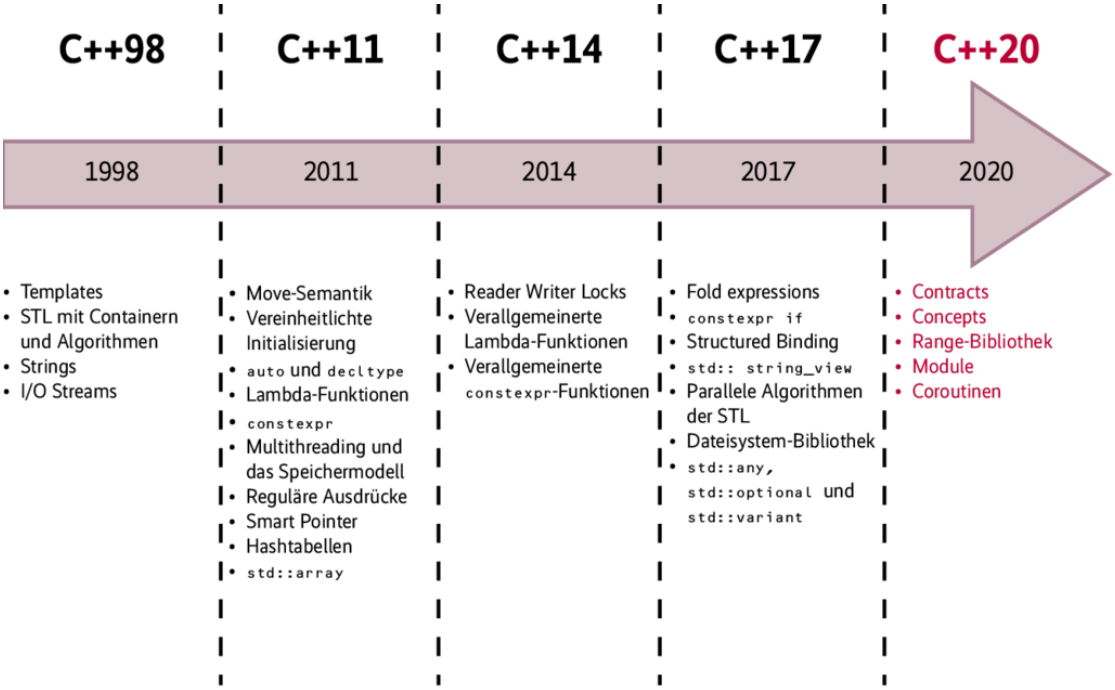
- virtuelle Funktionen
- Überladen von Funktionen und Operatoren
- Referenzen
- Speichermanagement
- Templates

1991 fand das erste Treffen der ISO Workgroup 21 statt. 1998 wurde dann ein “vorläufig endgültiger” Standard von der Arbeitsgruppe verabschiedet. Im wesentlich wurde noch folgende Funktionalität hinzugefügt:

- Ausnahmebehandlung
- Namensräume
- und ein Typinformationssystem zur Laufzeit des Programms

Dieser ANSI/ISO-Standard ist (mit umfangreichen Erweiterungen) auch heute noch gültig und wird vielseitig eingesetzt. Beispielsweise wurde das Betriebssystem UNIX auf C++ umgeschrieben.

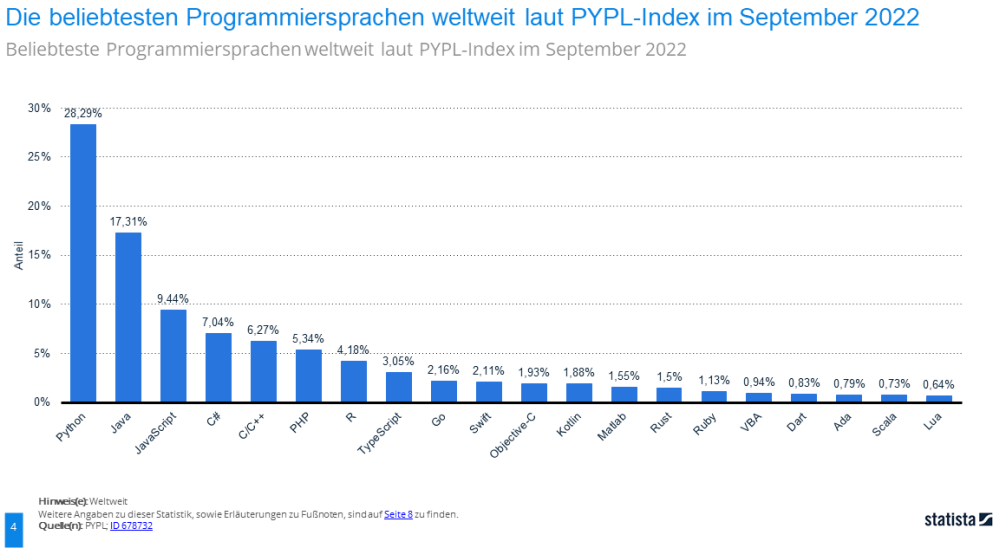
Zum Zeitpunkt im Jahre 2022 hat es bisher folgende Standards gegeben:



Quelle: Stand 2022, Zur Heise Magazin Webseite

Bjarne Stroustrup ist für die enge Anlehnung von C++ an C oft kritisiert worden, aber der Erfolg der Programmiersprache C++ gibt ihm im Nachhinein wohl recht.

Nach wie vor ist C/C++ weit verbreitet und eine der gängigsten objektorientierten Programmiersprachen.



Quelle: Statista, Stand 2022 zur Webseite

Verwendet man C/C++ - so wie wir - als Lehrsprache, so sollten unbedingt die Regeln für strukturiertes Programmieren bekannt sein und auch beachtet werden. Wir sollten also immer im Auge behalten, lesbare, effektive und wartbare Programme (nicht nur funktionierende Programme) zu schreiben.

Da man C primär als Teilmenge von C++ betrachten kann, reduziert sich die Beschreibung von C im Wesentlichen auf die folgenden zwei Aspekte:

- Welche Anteile von C sind keine Teilmenge von C++?
- Welche C++-Erweiterungen fehlen in C und wie werden Sie durch C++-Features ersetzt?

1.1 Vom Quellcode zur ausführbaren Datei

Ein C/C++-Programm wird in folgenden Schritten aus einer Quelle in ein ausführbares Programm umgewandelt:

- Erzeugen eines Quellprogramms in einem Editor
  Es wird eine Textdatei erzeugt und editiert. In dieser Textdatei wird das Quellprogramm (*Source Code*) gespeichert. Im Zusammenhang mit der Programmentwicklung in C ist es üblich als Datei-Endung für die Quelldatei ".c" zu wählen. Im Verlauf der Vorlesung werden wir so genannte Header-Dateien kennenlernen. Sie enthalten keine vollständigen Programme sondern nur Programmbausteine und enden üblicherweise auf ".h". Für C++-Programme wählt man üblicherweise die Datei-Endungen *.cpp (Windows) bzw. *.cc (unter Linux). Entsprechend enden auf Header-Dateien auf *.hpp (Windows) oder *.hh (unter Linux).

- Ein Präcompiler nimmt Textersetzungen vor
  Bevor der Sourcecode vollständig übersetzt werden kann, werden vordefinierte Schlüsselwörter, die mit dem Zeichen „#“ eingeleiet werden, von einem sog. Präcompiler durch C-Code ersetzt. Das Paradebeispiel hierfür ist die Anweisung #include <stdio.h>. Die angegebene Headerdatei wird vom Präprozessor gelesen und direkt in den Quellcode eingefügt.

- Übersetzen des Quellprogramms
  Nach er Ergänzung der Quelldatei durch den Präcompiler wird diese von einem Compiler übersetzt. Ein C/C++-Compiler ist ein Programm, das aus der Quelldatei eine Folge von Maschinenbefehlen erzeugt, die vom jeweiligen Prozessor direkt ausgeführt werden können. (Eine Zwischenstufe mit Byte-Code, der auf einer virtuellen Maschine ausgeführt wird, gibt es nicht.) Ein Compiler erzeugt also Binärcode für jeweils nur eine Prozessorfamilie (AMD, ARM oder Intel). Der vom Compiler erzeugte Binärcode wird in einer Datei abgespeichert, welche in der Regel die Extension ".obj" oder ".so" hat. Solch eine "Objekt"-Datei enthält Maschinenbefehle, welche vom einem Texteditor nicht mehr interpretiert werden können. Findet ein Compiler eine Stelle im Quellcode, die nicht den Regeln der Programmiersprache entspricht (Syntaxfehler), so erzeugt er lediglich eine entsprechende Fehlermeldung aber keine „Objekt“-Datei.

- Binden des ausführbaren Programms
  Die nach einer fehlerfreien Übersetzung des Compilers erzeugte Objekt-Datei ist noch nicht "lauffähig". Fast jedes Programm verwendet Unterprogramme, z.B. aus der so genannten Standardbibliothek. Der Objektdatei fehlt noch die Information, wo sich die Einsprungadressen der externen Unterprogramme befinden. Ein Linker verknüpft die in unterschiedlichen Dateien liegenden Programmteile zu einem ausführbaren Programm. Für diese Programme ist die Erweiterung ".com" oder ".exe" üblich. Unter Unix ist keine Erweiterung notwendig, da die Information im Header der Datei gespeichert ist. Es werden aus Gründen der Übersichtlichkeit aber manchmal Dateiendungen wie „.out“ oder „.o“ verwendet.

Editor, Präcompiler, Compiler und Linker können völlig eigenständige Programme sein, welche der Programmierer explizit der Reihe nach aufruft. In den letzten Jahren hat sich jedoch das Konzept der Integrierten Entwicklungsumgebung (*IDE – Integrated Development Environment*) durchgesetzt, bei dem alle Aufgaben in einem Programmrahmen integriert zur Verfügung stehen. Dies erleichtert den Bedienungskomfort für den Programmierer. Trotzdem sollte der Programmierer immer wissen in welcher Phase der Umwandlung er sich gerade befindet, um die Gründe für eine Fehlersituation zuordnen zu können.