Vergleich der Programmiersprachen

Inhalt:

- Hardwarebezug
- Entwicklung und Entstehung
- Einsatzgebiete
- Eigenschaften
- Vergleich von Quellcode

Programmiersprachen

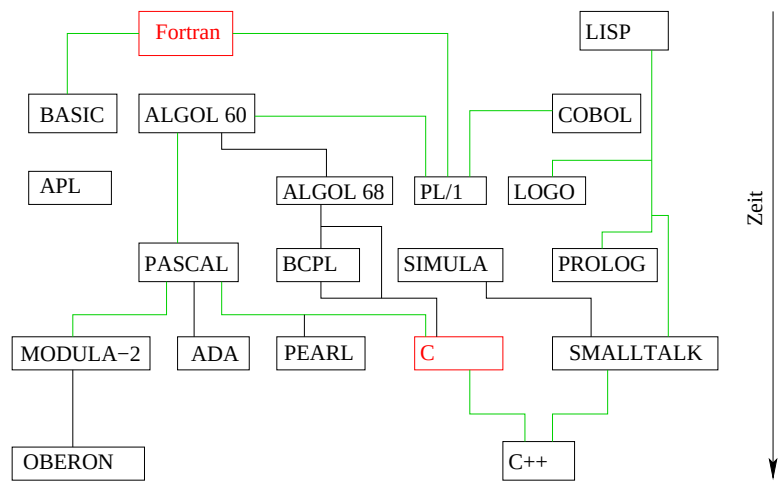
- Verbindung zwischen Programmierer und Computer
- Computer versteht nur Maschinensprache
- Quelltext wird durch Compiler oder Interpreter in Maschinensprache übersetzt
- Verschiedene Arten von Programmiersprachen sind:
  - Maschinensprache
  - Assemblersprache
  - imperative Hochsprachen (z.B. BASIC, **Fortran**, **C**)
  - deklarative Hochsprachen (z.B. Haskell, SQL, XSLT)

Definiton von Programmiersprachen auf *stupidedia.org*:
„Programmiersprachen sind wirre, unstrukturierte Gebilde von Befehlen, die den Computer veranlassen, in einer besonderen Art und Weise abzustürzen“

Hardwarenähe

- Hardwarenahe Hochsprachen
  - eine Anweisung entspricht wenigen Assemblerbefehlen
  - kaum Einschränkungen
  - direktes Ansprechen der Hardware
  - Hierzu gehört **C**
- Weniger hardwarenahe Hochsprachen
  - eine Anweisung kann vielen Assemblerbefehlen entsprechen
  - kürzerer Quelltext
  - leichter zu verstehen
  - Hierzu gehört **Fortran**

Entwicklung der Sprachen



Entstehung von Fortran

- 1957 nach Idee von John W. Backus entstanden
  - Fortran war die erste Hochsprache
- Motivation:
  - kurze, verständliche Quelltexte
  - optimiert für wissenschaftlichen Anwendungen
    → FORmula TRANslator
- 1966 erste Normierung durch ANSI: FORTRAN 66
- 2008 erscheint bisher letzte ISO-Norm: Fortran 2008

Zitat von J. W. Backus: „*Much of my work has come from being lazy. I didn't like writing programs, [...] I started work on a programming system to make it easyer to write programs.*“

Entstehung von C

- 1971-1973 entwickelt von Dennis Ritchie
- Motivation:
  - optimiert um das Betriebssystem UNIX zu schreiben
- 1978 erscheint "The C Programming Language“ von Kernighan und Ritchie
  - Einführung der Input/Output-Standardbibliothek
  - Erweiterung um einige Sprachbestandteile
- 1989 erlässt das ANSI eine Norm für C: **ANSI-C**
- 1999 erscheint bisher letzte ISO-Norm: C99

Einsatzgebiete

Einsatzgebiete von C

- Systemprogrammierung
  - Gerätetreiber
  - Betriebssysteme (Windows, Linux, UNIX)
- Compiler (z.B. GCC)
- Interpreter (z.B. Java Virtual Machine)
- Programmbibliotheken
- Microcontroller
- Anwendungsprogramme

Einsatzgebiete von Fortran

- wissenschaftliche Berechnungen, z.B.:
  - numerische Wettervorhersage
  - numerische Strömungsmechanik
  - numerische Mathematik
  - Hochleistungsrechnen
- Benchmark Supercomputer

Eigenschaften von Fortran

- Großer Sprachumfang
- Datentyp für komplexe Zahlen: `COMPLEX`
- Potenzoperator: `**`
- Matrizenoperatoren
- Dynamische Speicherverwaltung:
  `INTEGER,ALLOCATABLE::b(:,:,:)`
- Sehr schnell: Quellcode gut optimierbar wegen Einschränkungen (z.B. können Iterationsvariablen nicht manipuliert werden)

Eigenschaften von C

- geringer Sprachumfang
  - zusätzliche Funktionen in Standardbibliotheken
- C Ausdrücke können leicht in Maschinencode übersetzt werden
  - leicht portierbar
- Direkte Speicherverwaltung
- Direktes Ansprechen der Hardware
- Kombinierte Operatoren (z.B. += -= *=)
- Anweisungen werden mit Semikolon abgeschlossen

Wechsel der Programmiersprache

- Viele Konzepte von Fortran und C sind gleich
  - imperative Sprachen
  - Aufbau von Schleifen, Verzweigungen...
- Erlernte Programmierparadigmen sind sprachunabhängig
  - strukutrierte Programmierung
  - modulare Programmierung

Umstieg zwischen C und Fortran ist relativ leicht!

Fallunterscheidung I

in C

```
if (var <= 7)
   var2 = 3;
else if (var != 9)
   var2 = 4;
...
else
   var2 = 17;
```

in Fortran

```
IF (var <= 7) THEN
   var2 = 3
ELSEIF (var /= 9) THEN
   var2 = 4
...
ELSE
   var2 = 17
ENDIF
```

Die zweite Bedingung wird nur geprüft, wenn die vorherige nicht zutrifft usw.

Fallunterscheidung II

in C

```
switch (var)
{
   case 1:
      var2 = 2;
      break;
   case 5:
      var2 = 42;
      break;
...
   default:
      var2 = 0;
}
```

in Fortran

```
SELECT CASE (var)

   CASE (1)
      var2 = 2

   CASE (5)
      var2 = 42

...
   CASE DEFAULT
      var2 = 0
END SELECT
```

Schleifen I

in C	in Fortran

`for (i=1; i<=10; i++)`
`{`
`  var2 = var2 + i;`
`}`
	`DO i=1, 10, 1`
	
	`  var2 = var2 + i`
	`ENDDO`

Zählschleifen durchlaufen Schleifenkörper so häufig wie im Kopf festgelegt, unabhängig vom Inhalt des Schleifenkörpers

Schleifen II

in C	in Fortran

`i = 1; var = 0;`
`while (var != 32)`
`{`
`  var = 8 * i;`
`  i++;`
`}`
	`i = 1; var = 0`
	`DO WHILE (var /= 32)`
	
	`  var = 8 * i`
	`  i = i + 1`
	`ENDDO`

Anzahl der Durchläufe hängt vom Schleifenkörper ab. Endlosschleifen können entstehen, wenn die Bedingung niemals falsch wird.

Schleifen III

in C	in Fortran

`i = 1;`
`do`
`{`
`  var = 8 * i;`
`  i++;`
`}`
`while (var != 32);`
	`i = 1`
	`DO`
	
	`  var = 8 * i`
	`  i = i + 1`
	
	`IF (var == 32) EXIT`
	`ENDDO`

Anzahl der Durchläufe hängt vom Schleifenkörper ab. Endlosschleifen können entstehen, wenn die Bedingung niemals falsch (C) / wahr (Fortran) wird.

Ein- und Ausgabe

in C	in Fortran
`printf (''Hallo %s\n'', var);`	`PRINT *,''Hallo '', var`
`fgets (buffer, 50, stdin);`	
`sscanf (buffer, ''%i'', &var2);`	`READ *, var2`
`fprintf (datei, ''%i\n'', var2);`	`WRITE (unt,fmt,ios) var2`
`fscanf (datei, ''%i'', &var2);`	`READ (unt,fmt,ios) var2`

`printf, sscanf, fprintf` und `fscanf` sind nicht Bestandteil von C. Es sind Funktionen der Input/Output-Standardbibliothek, welche in die meisten Programme eingebunden werden muss
`PRINT, READ` und `WRITE` sind integrative Bestandteile von Fortran