Gliederung

1 Syntax, Semantik und Pragmatik von Sprachen

2 Programmiersprachen als Abstraktion der Maschinensprache

3 Sprachklassen für höhere Programmiersprachen

4 Historische Entwicklung der Programmiersprachen

Sprachvergleiche

Zum Vergleich von Programmiersprachen verwendet man am besten denselben Algorithmus, z.B.

- „Hello World!“, `www.roesler-ac.de/wolfram/hello.htm` oder
- „99 bottles of beer“, `www.99-bottles-of-beer.net` .

Wir benutzen den TPK-Algorithmus von Trabb Pardo und Knuth, der typische Sprachelemente von imperativen Programmiersprachen enthält:

- mathematische Standardfunktionen,
- Ein- und Ausgabe,
- bedingte Anweisungen,
- Schleifen,
- Unterprogramme sowie
- Felder und indizierte Variablen.

Er liest eine Folge von 11 Gleitpunktzahlen ein, kehrt die Reihenfolge um, berechnet für jede Zahl einen Funktionswert und gibt ihn aus, falls er nicht zu groß ist.

TPK-Algorithmus in Pascal

```
program TPK (input, output);
var i: integer; y: real;
    a: array [0..10] of real;

    function f (t: real) : real;
    begin
        f := sqrt(abs(t)) + 5*t*t*t
    end;

begin
for i:= 0 to 10 do read(a[i]);
for i:= 10 downto 0 do
    begin
    y := f(a[i]);
    if y > 400 then writeln(i, 'TOO LARGE')
    else write(i, y)
    end
end.
```

Plankalkül

Mitte der 1940er Jahre entwickelte der Erfinder des ersten Computers, Konrad Zuse, den Plankalkül, eine –damals nicht implementierte– Programmiersprache, die schon viele Konzepte imperativer Sprachen vorwegnahm.

Da die Sprache aber erst in den 1970er Jahren bekannt wurde, konnte sie die Entwicklung der Programmiersprachen nicht beeinflussen.

Short Code

Für den von John P. E. Eckert und J. W. Mauchly entwickelten UNIVAC (*universal automatic computer*) schrieb W. F. Schmitt von 1950–52 den algebraischen Interpretierer Short Code.

Der Interpretierer arbeitete die codierte Darstellung –von rechts nach links– ab;
die Übersetzung in die codierte Darstellung erfolgte per Hand.

Short Code ermöglichte

- Konstanten im Dezimalsystem,
- Gleitpunktzahlen,
- symbolische Marken für Programmadressen.

TPK-Algorithmus in Short Code

Short Code:

```
    i = 10
0:  y = (√ abs t) + 5 cube t

    y 400 if ≤ to 1
    i print,'TOO LARGE' print-return
    0 0 if = to 2
1:  i print, y print-return
2:  T0 U0 shift
    i = i-1
    0 i if ≤ to 0
    stop
```

codierte Darstellung:

```
00   00 00 00 W0 03 Z2
01   T0 02 07 Z5 11 T0
02   00 Y0 03 09 20 06
03   00 00 00 Y0 Z3 41
04   00 00 Z4 59 W0 58
05   00 00 00 Z0 Z0 72
06   00 00 Y0 59 W0 58
07   00 00 00 T0 U0 99
08   00 W0 03 W0 01 Z1
09   00 00 00 Z0 W0 40
10   00 00 00 00 ZZ 08
```

TPK-Algorithmus in Short Code

Code-Operator-Zuordnung:

01	$-$	06	abs value	1n	(n+2)te Potenz	59	print und return
02	)	07	$+$	2n	(n+2)te Wurzel	7n	if $=$ to n
03	$=$	08	pause	4n	if $\leq$ to n	99	zykl. Shift des Speichers
04	/	09	(	58	print und tab	00	no operation

$\times$ hat keinen Operator-Code; die Operation wird implizit vorgenommen.

Variable: Speicherbelegung: `i = W0, t = T0, y = Y0`.
Eingaben kommen nacheinander in die elf Speicherzellen `U0, T9, ..., T0`.

Konstanten:

```
Z0 = 000000000000
Z1 = 010000000051    (1.0, floating decimal form)
Z2 = 010000000052    (10.0)
Z3 = 040000000053    (400.0)
Z4 =     TOO LARGE
Z5 = 050000000051    (5.0)
```

Marken: 0: = Zeile *01*, 1: = Zeile *06*, 2: = Zeile *07* (der Codierung)

FORTRAN

FORTRAN (*formula translating system*) wurde 1954 unter Leitung von John Backus (IBM) für die nummerische Programmierung entwickelt.

Ziel war ein effizienter Maschinencode
– vergleichbar mit dem Code eines guten Assemblerprogrammierers.

Der Compiler wurde 1957 ausgeliefert.

FORTRAN führte ein:

- verzweigende `IF`-Anweisungen ($<$, $=$, $>$ Null),
- Schleifen (`DO`) mit Endmarkierung und Laufvariable sowie
- Felder (`DIMENSION`) ein, auf deren Elemente über –zur Laufzeit berechnete– Indizes zugegriffen werden konnte,
- Kommentare.

Ein `FORMAT`-Interpretierer ermöglicht formatierte Ein- und Ausgabe.

TPK-Algorithmus in FORTRAN I

```
C       THE TPK-ALGORITHM IN FORTRAN I
        FUNF(T) = SQRTF(ABSF(T))+5.0*T**3
        DIMENSION A(11)
    1   FORMAT (6F12.4)
        READ 1,A
        DO 10 J = 1,11
        I = 11-J
        Y = FUNF(A(I+1))
        IF (400.0 - Y) 4,8,8
    4   PRINT 5,I
    5   FORMAT (I10, 10H TOO LARGE)
        GO TO 10
    8   PRINT 9,I,Y
    9   FORMAT (I10,F12.7)
    10  CONTINUE
        STOP
```

FORTRAN

FORTRAN-Programme sind Lochkarten-orientiert; die Bedeutung eines Zeichens variiert mit der Spalte.

Leerzeichen werden ignoriert, sind also kein Trennsymbol.

Variablen brauchen nicht deklariert zu werden; der Anfangsbuchstabe des Namens beeinflusst dann den Typ.

FORTRAN wurde oft weiterentwickelt (FORTRAN II, FORTRAN IV, FORTRAN 66, FORTRAN 77, Fortran90, Fortran95, Fortran2003, Fortran2008) und durch die Unterstützung von IBM sehr verbreitet.

Der sehr effiziente Maschinencode und die große Zahl von Unterprogrammpaketen für nummerische Anwendungen in FORTRAN erhält die Sprache am Leben.

FORTRAN

Da Leerzeichen in FORTRAN einfach überlesen wurden und keine Namen oder Schlüsselwörter begrenzten, war manchmal eine weite Vorschau nötig:

Erst nach dem Lesen des Kommas in der Anweisung

```
DO 10 J = 1,11
```

wird klar, dass `DO` ein Schlüsselwort ist und es nicht um eine Wertzuweisung an die (undeklarierte) Variable `DO10J` geht.

COBOL

Die Sprache COBOL (*common business-oriented language*) wurde 1959 unter Leitung von Grace Hopper vom Verteidigungsministerium der USA zusammen mit Computerherstellern und Anwendern für kaufmännische Anwendungen entwickelt.

COBOL führte Record-Strukturen für Daten ein.

Die Sprache enthält umfangreiche Formatierungsmöglichkeiten für Daten, fällt aber durch Geschwätzigkeit und ein fehlendes Prozedurkonzept auf.

COBOL war zeitweise die am meisten benutzte Programmiersprache der Welt. Dies lag auch an der Kompatibilität von COBOL-Compilern durch frühe Standardisierung der Sprache.

TPK-Algorithmus in COBOL

```
IDENTIFICATION DIVISION.
PROGRAM-ID. TPK-Algorithmus.
AUTHOR. mak.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CONSOLE IS CRT,
    DECIMAL-POINT IS COMMA.
INPUT-OUTPUT SECTION.

DATA DIVISION.
FILE SECTION.
WORKING-STORAGE SECTION.
77 i PIC S99.
01 y USAGE IS COMPUTATIONAL-1.
01 a-hilf.
    02 a USAGE IS COMPUTATIONAL-1 OCCURS 11 TIMES.
```

TPK-Algorithmus in COBOL

```
LINKAGE SECTION.

PROCEDURE DIVISION.
    PERFORM VARYING i FROM 1 BY 1 UNTIL i > 11
        ACCEPT a(i)
    END-PERFORM.
    PERFORM VARYING i FROM 11 BY -1 UNTIL i < 1
        PERFORM BERECHNUNG
            IF y > 400 THEN
                DISPLAY i
                DISPLAY "TOO LARGE"
            ELSE
                DISPLAY i
                DISPLAY y
            END-IF
        END-PERFORM.
        STOP RUN.
```

TPK-Algorithmus in COBOL

```
BERECHNUNG.
    IF a(i) IS NEGATIVE THEN
        COMPUTE y = (-a(i)) ** 0,5 + 5*a(i)*a(i)*a(i)
    ELSE
        COMPUTE y = a(i) ** 0,5 + 5*a(i)*a(i)*a(i)
    END-IF.
```

Algol 60

Zur gleichen Zeit entstand die Sprache Algol 60 (*algorithmic language*), die von einem europäisch-amerikanischen Komitee definiert wurde.

Die Syntax von Algol 60 wurde durch eine kontextfreie Grammatik (in **Backus-Naur-Form**) formal definiert.

Algol 60 führte erstmals ein:

- Blockstruktur,
- rekursive Prozeduren,
- explizite Deklaration von Variablen,
- reservierte Wörter,
- dynamische Felder (mit variabler Feldgröße) und
- eine (Lochkarten-)formatfreie Eingabe.

Fehlende Anweisungen für Ein- und Ausgabe erzeugten aber Inkompatibilitäten zwischen den Implementierungen.

TPK-Algorithmus in Algol 60

```
begin
    integer i; real y; real array a[0:10];

    real procedure f(t); real t; value t;
        f := sqrt(abs(t))+5*t↑3;

    for i := 0 step 1 until 10 do read(a[i]);
    for i := 10 step -1 until 0 do
        begin
            y := f(a[i]);
            if y > 400 then write(i,'too large')
                else write(i,y)
        end
end
```

Algol 60

Algol 60 ist der Prototyp einer imperativen Programmiersprache, d.h. sie verwendet Variablen und stellt Anweisungen –wie die Wertzuweisung– bereit, um die Werte von Variablen zu verändern.

Algol 60 wurde Referenzsprache zur Veröffentlichung von Algorithmen; wirtschaftlichen Erfolg hatte sie nicht.

Aber sie hatte einen gewaltigen Einfluss auf nachfolgende imperative Programmiersprachen wie Pascal, C, Modula-2, Ada und Java.