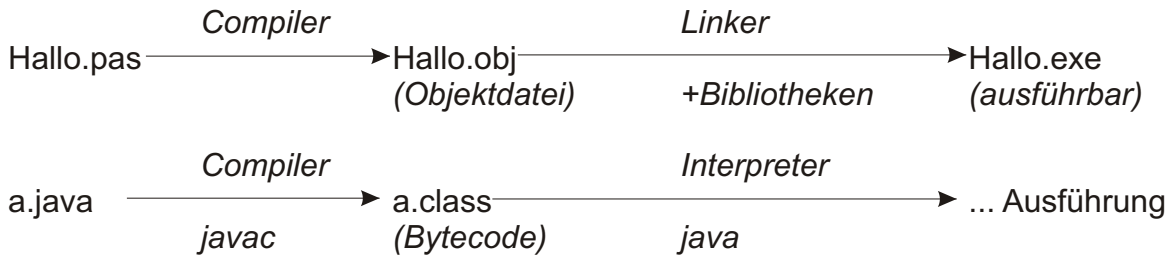


C, C++	Interpreter	Compiler + Linker
	Skripte PHP Basic sh, bash Smalltalk	Pascal, Delphi C, C++ Fortran, Algol 60

Der Linker wandelt in Maschinensprache um und führt dann aus.



C -> Ken Thomson, Brian W. Kerningham, Denis (M.) Ritchie

Prozedural: Programme/Daten getrennt

Objektorientiert: Alles gekapselt

```

program abc
  integer i, j  eingebaute Typen
  i = 12
  call up(i)
end
  
```

```

class c
{
  private int i, j;
  public up(1) {...}
}
  
```

```

subroutine up(k)
  integer k
  print *.k
end
  
```

Veraltet: Kerningham & Ritchie: Programmieren mit C, 1. Auflage -> K&R-C

=> ANSI-C -> ISO 1989;90 - ISO-C 89
- ISO-C 90

=> K&R ... 2. Auflage = Nachschlagewerk für C

1.) Linux

2.) keine IDE, sondern Editor + Compiler + (ggf.) Linker + make

Boolean gibt es nicht

68HC11 Atmal AVR ...

```
int g;

int up (int i)
{
    int l;
    l = 10;
}

int main()
{
    up(5);
}
```

char	1 Byte	?	?
unsigned char	1 Byte	0	255
signed char	1 Byte	-128	127
short	2 Byte	-32768	32767
unsigned short	2 Byte	0	65536
signed short	2 Byte	-32768	32767
int	2..4	...	
unsigned int			
signed int			
long	4 (8)	- 2 Miard.	2 Miard.
unsigned long	4 (8)	0	~ 4 Miard.
signed long	4 (8)	- 2 Miard.	2 Miard.
long long	8	?	?
unsigned long long	8	0	?
signed long long	8	?	?

```
unsigned short n1, n2;
    n1 = 4000;
    n2 = 5000
short n_diff = n1 - n2;
```

```
char c;
    c = 'A'; // evtl. ASCII-Code 65 = 'A'
    c = c + 2; // evtl. ASCII-Code 67 = 'C'
    c--;
    c++;
    c = 'C' - 2;
    'C' - 'A'
    'A' - 'C' // -2
                // oder 254
```

ISO-C99, (C++)

uint8_t
int8_t
uint16_t
int16_t
uint32_t
int32_t
uint64_t
int64_t

```
int8_t c;  
  c = 'm';  
  c = 12-13; // -1
```

Gleitkommazahlen (floating point)

Festkommazahlen z.B. 1.99 Euro

- Datenbanken
- COBOL

float

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

```
float f;  
  f = 31415E5; // 314150.000  
                31415      Mantisse  
                5         Exponent (105)  
                +/-       Vorzeichen
```

IEEE 754	simple precision	32
	double precision	64
	extended precision	80

float ca. 6-7 Stellen dezimal, -1.E38 .. 1E-38, 0.0, + 1E-38 .. 1E38

double ca. 15-16 Stellen dezimal, -1E308 .. - 1E-308, 0.0, 1E-308 .. 1E308

ISO C91, C++

```
long double             float a = 3.0;  
                       float b = 17.0;  
FPU                     if ( a+b == 20) {...}
```

0..1 dezimal

```
float a = 1.0; b = 3.7;  
float c;  
c = a + b;
```

Konstanten

```
i = 1;                                int-Konstante          long l;
                                      1 = 10;                int-Konstante
x = 13 + 7;                            1 = 101;              long-Konstante

x = 'A'; // C: int-Konstante           1 = 10n;              unsigned-Konstante
          (Wert 65)
          // C++: char-Konstante      1 = 10u;
          (65)

int x = 'A' + 'B' + 'C' + 'D' + 'E';
      //C: 65 + 66 + 67 + 68 + 69 = 335

      //C: 65 + 66 + 67 + 68 + 69 = 80
```

```
+ .3f
4.3
.5
1.
1E5
1.23E-7
1.000E-25
```

```
enum // C
{
    rot;
    gelb;
    gruen;
};

enum wieschlau // C++
{
    schlau = 13;
    doof;
    ganzdoof = 20;
}

int i = rot;          //rot = 0, i = 0;
    i = gelb;        // i = 1;

wieschlau i = schlau;
            i = doof;
```

Variablennamen

```
int 123;          (int $123xy;)
int x123;
int 1E5;
```

a-z A-Z _ (\$) (0-9)

```
#include <stdio.h>
...
printf ( ... ;

printf ( "Hallo!" );
printf ("i ist etwa %d groß", 15+19 );
printf ("i ist %d, j ist %d, k ist %d ", i, j, k);
```

%d sagt aus, dass der Wert als Dezimalzahl dargestellt werden soll.

```
printf ("i ist %x", i);
printf ("i ist %o", i);
```

%x ist hexadezimal,
%o ist oktal,
%u ist unsigned dezimal,
%l wie oben, aber long,
%f double oder float,
%e double oder float mit Exponentialanzeige,
%g double oder float,
%ld lang, dezimal,
%_8ld z.B. _____1
%08ld 00000001
%-8ld 1_____

```
unsigned int x = 4 Mia;
printf (" x = %d", x);
```

```
printf ("Hallo\n");           Hallo
printf ("%d\n", 12);         12
```