declarative
  functional
  dataflow
  logic, constraint-based
imperative
  von Neumann
  object-oriented
  scripting

Lisp/Scheme, ML, Haskell
Id, Val
Prolog, spreadsheets, SQL
C, Ada, Fortran, ...
Smalltalk, Eiffel, Java, ...
Perl, Python, PHP, ...
int gcd(int a, int b) { // C
    while (a != b) {
        if (a > b) a = a - b;
        else b = b - a;
    }
    return a;
}

let rec gcd a b = (* OCaml *)
    if a = b then a
    else if a > b then gcd b (a - b)
    else gcd a (b - a)

gcd(A,B,G) :- A = B, G = A. % Prolog
gcd(A,B,G) :- A > B, C is A-B, gcd(C,B,G).
gcd(A,B,G) :- B > A, C is B-A, gcd(C,A,G).
pushl %ebp
movl %esp, %ebp  # ) reserve space for local variables
subl $16, %esp  # /
call getint  # read
movl %eax, -8(%ebp)  # store i
call getint  # read
movl %eax, -12(%ebp)  # store j
movl -8(%ebp), %edi  # load i
movl -12(%ebp), %ebx  # load j
cmpl %ebx, %edi  # compare
je D  # jump if i == j
movl -8(%ebp), %edi  # load i
movl -12(%ebp), %ebx  # load j
cmpl %ebx, %edi  # compare
jle B  # jump if i < j
movl -8(%ebp), %edi  # load i
movl -12(%ebp), %ebx  # load j
subl %ebx, %edi  # i = i - j
movl %edi, -8(%ebp)  # store i
jmp C
B: movl -12(%ebp), %edi  # load j
movl -8(%ebp), %ebx  # load i
subl %ebx, %edi  # j = j - i
movl %edi, -12(%ebp)  # store j
C: jmp A
D: movl -8(%ebp), %ebx  # load i
push %ebx  # push i (pass to putint)
call putint  # write
addl $4, %esp  # pop i
leave  # deallocate space for local variables
mov $0, %eax  # exit status for program
ret  # return to operating system
Source program → Compiler → Target program → Output

Input → Target program → Output
Source program

Input

Interpreter

Output
Fortran program

Incomplete machine language

Library routines

Linker

Machine language program
Source program → Compiler → Assembly language → Assembler → Machine language
Source program

Preprocessor

Modified source program

Compiler

Assembly language
Source program

Compiler 1

Alternative source program (e.g., in C)

Compiler 2

Assembly language
Pascal to machine language compiler, in Pascal

Pascal to P-code compiler, in P-code

Pascal to machine language compiler, in P-code

Pascal to machine language compiler, in machine language
Java program

Java compiler

Java byte code

JIT compiler

Bytecode interpreter

Input

Output

Machine language

Input

Output