Wstęp do programowania komputerów

Ostatnia aktualizacja: 15 grudnia 2025

Programowanie to wszechstronny proces prowadzący od problemu obliczeniowego do jego rozwiązania w postaci programu.

Celem programowania jest odnalezienie sekwencji instrukcji, które w sposób automatyczny wykonują pewne zadanie.

Programowanie proceduralne to paradygmat programowania zalecający dzielenie kodu na procedury, czyli fragmenty wykonujące ściśle określone operacje.

pl.wikipedia.org

Przegląd zagadnień

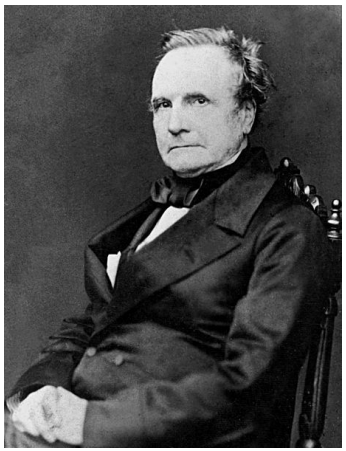
- Podstawy programowania w oparciu o język C
- *Case study* - przykłady programów, demonstracje
  Problem $\rightarrow$ Algorytm $\rightarrow$ Program $\rightarrow$ Rozwiązanie
- Paradygmat programowania proceduralnego,
  modularna struktura, algorytmy w postaci uniwersalnych funkcji
- Reprezentacja danych w komputerze:
  typy proste, złożone, wskaźniki, struktury dynamiczne, ...
- Elementy inżynierii oprogramowania: projektowanie, implementacja, wykrywanie błędów, testowanie, narzędzia
- Laboratorium: język C, rekomendowane środowisko Visual Studio C++
- Zaliczenie wykładu: pozytywna ocena z laboratorium + TEST zaliczeniowy

Jak uczyć się programowania?

- pytaj
- czytaj
- podglądaj
- programuj, programuj, programuj, ...

- Brian W. Kernighan, Dennis M. Ritchie, *Język ANSI C*, WNT, Warszawa, 2000.
- David Griffiths, Dawn Griffiths, *Rusz głową! C.*, Helion, Gliwice, 2013.
- D. Harel, *Rzecz o istocie informatyki. Algorytmika*, WNT, Warszawa, 1992.
- Maciej M. Sysło, *Algorytmy*, WSiP, Warszawa, 2002.

Charles Babbage (1791–1871)



1822 Projekt maszyny różnicowej.

1837 (Parowa) **maszyna analityczna** - sterowanie sekwencyjne, pętle, odgałęzienia, projekt niedokończony

Augusta Ada Lovelace (1815-1852)



1842 *Note G*, algorytm wyznaczania liczb Bernoullego. **Pierwszy program komputerowy.**

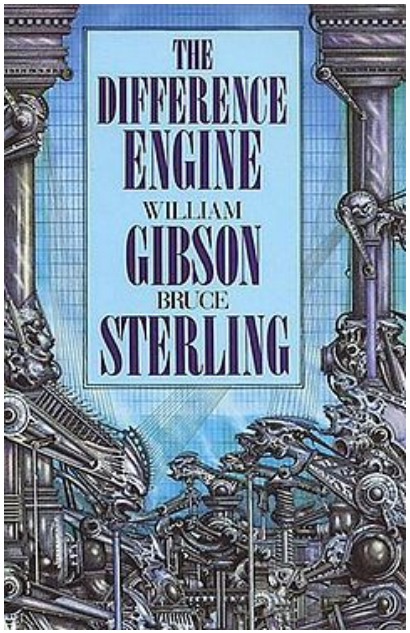
1979 Język ADA.

1849 Difference Engine No. 2, dokładność 31 cyfr, wydruk na wyjściu

2002 Realizacja projektu rekonstrukcji maszyny różnicowej, ☞Computer History Museum.

2011 Początek (10 letniego) projektu rekonstrukcji maszyny analitycznej, ☞Plan 28.





http://www.computerhistory.org/

Generacje komputerów

0 mechaniczne, przekaźnikowe (do 1945)

Z3 (Berlin, 1941), Harvard Mark 1 (USA, 1944), GAM-1 (Warszawa, 1950), PARK (AGH, 1957)

1 lampy elektronowe (1945-59)

ABC Atanasoff-Berry Computer (USA, 1942), COLOSSUS (UK, 1943), ENIAC (USA, 1945),

XYZ (Warszawa, 1957/1958)

2 tranzystory i pamięci ferrytowe (1959-64)

PDP-1 (USA, 1960), ZAM-41(Polska, 1961), Odra 1204 (Polska, 1967)

3 układy scalone o małej skali integracji SSI (1965-70)

IBM 360 (1965), Odra 1305 (Polska, 1973)

4 układy o wysokiej skali integracji LSI i VLSI, mikroprocesory

mikroprocesor Intel 4004 z częstotliwością taktowania 0,1 MHz (1971), IBM 5150 PC (1981)

5 komputery przyszłości: kwantowe, optyczne, biologiczne ?



Komputery generacji 0

Mechaniczne i przekaźnikowe

Konrad Zuse

1936 Mechaniczny **Z1**,
liczby zmiennopozycyjne.

1941 ☞Przekaźnikowy **Z3**,
pierwszy działający
programowalny komputer
5.3Hz,
64 słowa 22 bitowe (176 B).



Inne komputery 0 generacji

1939-44 Harvard Mark 1 Howarda Aikena (IBM ASCC)

1950 GAM-1, Państwowy Instytut Matematyczny w Warszawie

1957 PARK (Programowany Automat Rachunków Krakowianowych), AGH

Język Plankalkül 1943

Komputery 1 generacji (1945-59)

Lampy próżniowe

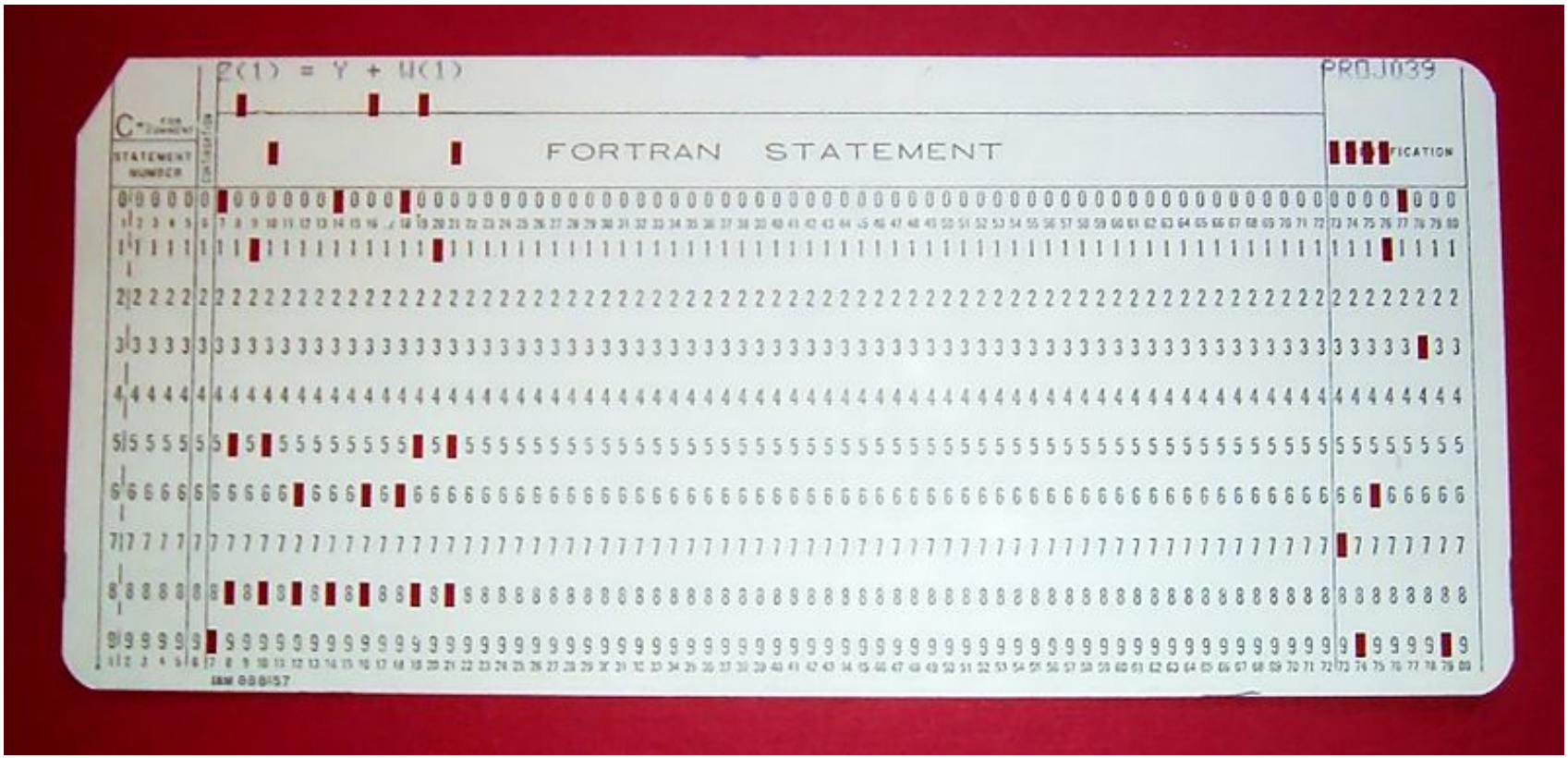
- Zastosowanie do obliczeń numerycznych (łamanie szyfrów, balistyka)
- Wejście: karty dziurkowane, taśmy papierowe
- Wyjście: wydruk, dalekopis, lampy
- Pamięć: dane przechowywane na dyskach magnetycznych, rtęciowe linie opóźniające
- Program: głównie język maszynowy

1947-50 pierwsze assemblery (EDSAC)
1952 pierwszy kompilator A-0 (UNIVAC I)
1954-57 Język **Fortran** (John Backus, IBM)
pierwszy język wysokiego poziomu,
który znalazł powszechne zastosowanie



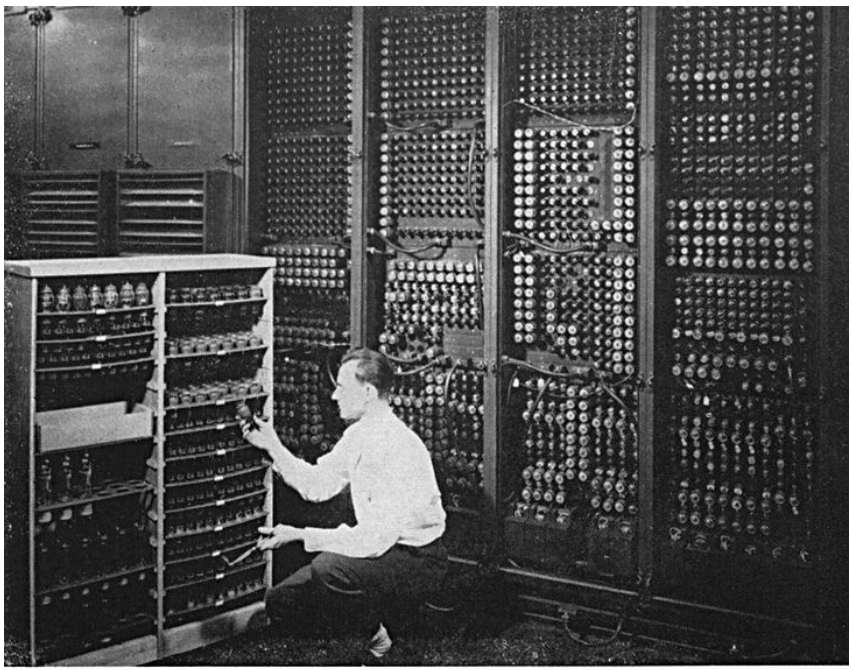
Karta perforowana

80 kolumn, 10 wierszy numerycznych + 2 wiersze strefowe (IBM, 1928)



ENIAC (1946)

Elektroniczne urządzenie numeryczne całkujące i liczące



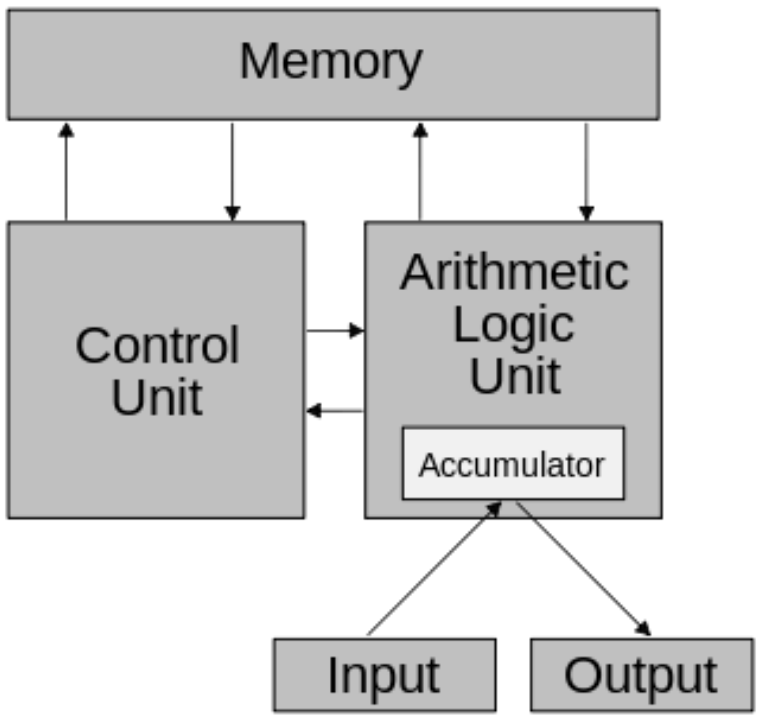
Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

18 000 lamp,
30 ton, 170 m^2, moc 160kW,
5000 operacji dodawania / sec.
system dziesiętny,
ręczne programowanie przez ustawianie 6K przełączników i wtykanie kabli

Współczesna koncepcja komputera

John von Neumann, 1945

Pamięć używana zarówno do przechowywania danych jak i samego programu, każda komórka pamięci ma unikatowy identyfikator (adres).





1949 EDVAC, Electronic Discrete Variable Computer, współpracuje już z dyskami magnetycznymi.

1960 PDP-1, pierwszy dostępny w sprzedaży minikomputer z monitorem i klawiaturą.



Pierwsza gra wideo „Spacewar!” (Steve Russel), pierwszy edytor tekstu, interaktywny debugger, komputerowa muzyka.

1970 Minikomputer K-202
(potencjalnie) wydajniejszy od IBM 5150 PC (1981 r.)
Opracowany i skonstruowany przez inż. Jacka Karpińskiego.



16 bitów
adresowanie stronicowe do 8MB
(konkurencja max. 64kB)
modularność
wielodostępowość

1 mln. operacji/s.

1981 IBM 5150 PC



procesor Intel 8088 (4.77 MHz)
64 kB pamięci ROM
do 640 kB pamięci RAM
brak dysku twardego (taśmy na kasetach, późniejsze modele dyskietki 5,25 cala)
karta CGA (kolor) lub MGA (monochromatyczna),
system operacyjny MS-DOS,

dźwięk z PC speakera

Rozwój języków programowania

- kod maszynowy, assembler
- lata 50-te, języki wysokiego poziomu

  Fortran (1955), Lisp (1955), COBOL (1959)
- lata 60-te, rozwój języków specjalistycznych

  Simula I (1960, el. obiektowości), Lisp, COBOL

  Pierwsze próby stworzenia języków ogólnych

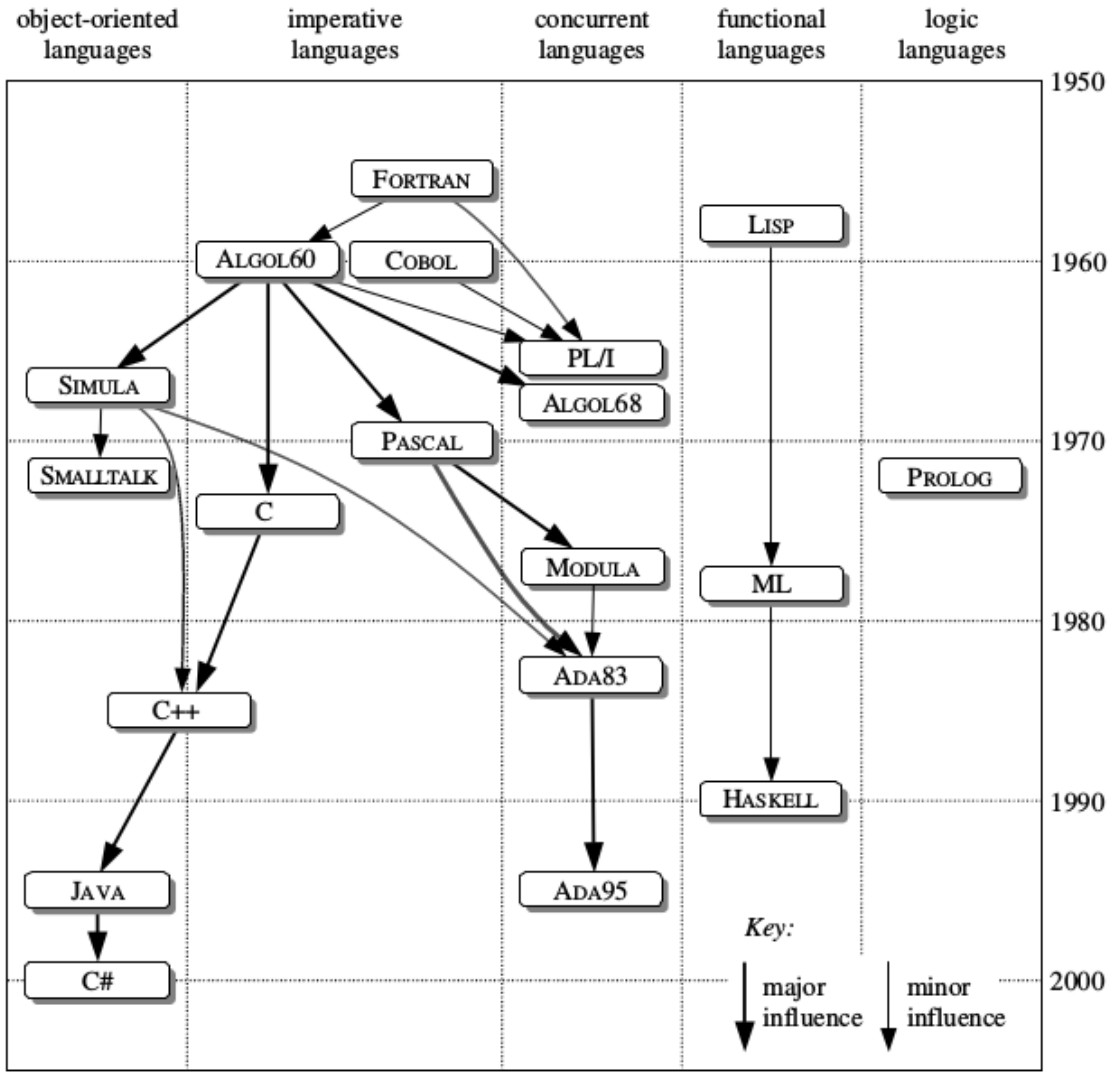
  Algol (58/60), PL/1 (1964).
- lata 70-te, początek pojedynku: Pascal vs. C

  Zalążki obiektowości: Smalltalk (1972)
- lata 80-te, Dominują: C, Pascal, Basic

  Powstają: C++ (1980), Matlab (1984)
- lata 90-te, era internetu, programowanie obiektowe

  Java (1996), Python (1991), PHP (1995), JS (1995), .NET (C#, 2001)

Źródło: ☞ History and Evolution of Programming Languages



Źródło: David A. Watt, "Programming Language Design Concepts"

Głowne typy języków programowania

- **imperatywne**: program określa kolejność instrukcji do wykonania, zmieniając stan programu krok po kroku. Kod określa jak coś zrobić (C, Pascal)
- **obiektowe**: łączą dane i działania (metody) w obiektach. Ułatwiają modelowanie złożonych systemów przez enkapsulację i dziedziczenie (Java, C++, C#)
- **równoległe** (*concurrent*): umożliwiają wyrażanie i zarządzanie współbieżnością, czyli wykonywaniem wielu procesów lub wątków jednocześnie (Go, Erlang)
- **funkcyjne**: opierają się na funkcjach matematycznych, unikają zmiany stanu. Programy opisują co ma być zrobione, a nie jak (Haskell, Lisp)
- **logiczne**: bazują na logice formalnej, definiując fakty i reguły, a język sam dochodzi do wniosków poprzez dedukcję (Prolog).

Języki programowania

- kod maszynowy, języki symboliczne
- wysokiego i niskiego poziomu
- paradygmaty programowania: proceduralne, strukturalne, obiektowe, funkcyjne, logiczne, uniwersalne, ...
- ☞ Lista 2500 języków komputerowych, Bill Kinnersley
  ☞ Lista języków na Wikipedii
  ☞ Porównanie języków

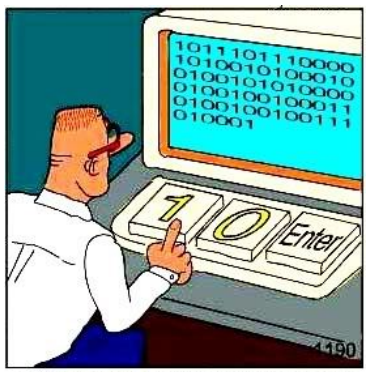


Pieter Bruegel (starszy), 1563

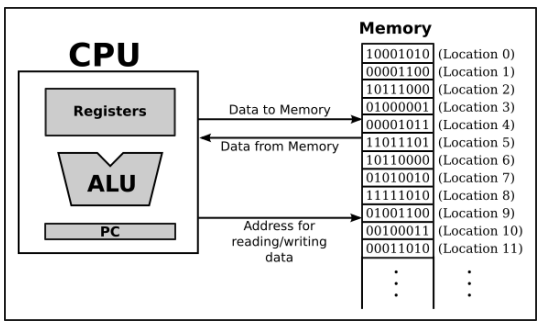
Kod maszynowy

ciąg instrukcji w postaci binarnej wykonywanych bezpośrednio przez procesor

- rozkazy procesora i dane w postaci słów bitowych pobierane są z pamięci do rejestrów procesora
- procesor wykonuje rozkazy, zmieniając stan rejestrów i pamięci
- kod maszynowy nie jest przenośny, każdy procesor ma swój specyficzny zestaw instrukcji
- tworzenie programów bezpośrednio w kodzie maszynowym jest trudne



REAL Programmers code in BINARY.



Język assemblera

zastępuje rozkazy maszynowe
tzw. **mnemonikami**, zrozumiałymi przez człowieka słowami określającymi konkretną czynność procesora.

```
push   rbp
mov    rbp,rsp
mov
DWORD PTR [rbp-0x4],0x0
jmp    11 <main+0x11>
add
DWORD PTR [rbp-0x4],0x1
cmp
DWORD PTR [rbp-0x4],0x9
jle    d <main+0xd>
pop    rbp
ret
```

- **Assembler** - program tłumaczący język asemblera na kod maszynowy (assemblacja)
- **Deasembler** - program tłumaczący kod maszynowy na jezyk asemblera

Języki wysokiego poziomu

- nie są bezpośrednio wykonywane przez procesor, przez co pozwalają uniezależnić program od platformy sprzętowej i systemowej

```c
#include<stdio.h>

int main()
{
    puts("Witaj swiecie!");
    return 0;
}
```

- muszą być skompilowane lub interpretowane
- przypominają jezyk naturalny, składnia i instrukcje mają za zadanie maksymalizować zrozumienie kodu programu przez człowieka
- pozwalają skupić się na logice zadania
- „kara za abstrakcję": kod niskiego poziomu zazwyczaj będzie bardziej efektywny od kodu wyższego poziomu,
- jedna instrukcja w języku wysokiego poziomu może odpowiadać wielu instrukcjom w kodzie maszynowym

Kompilator

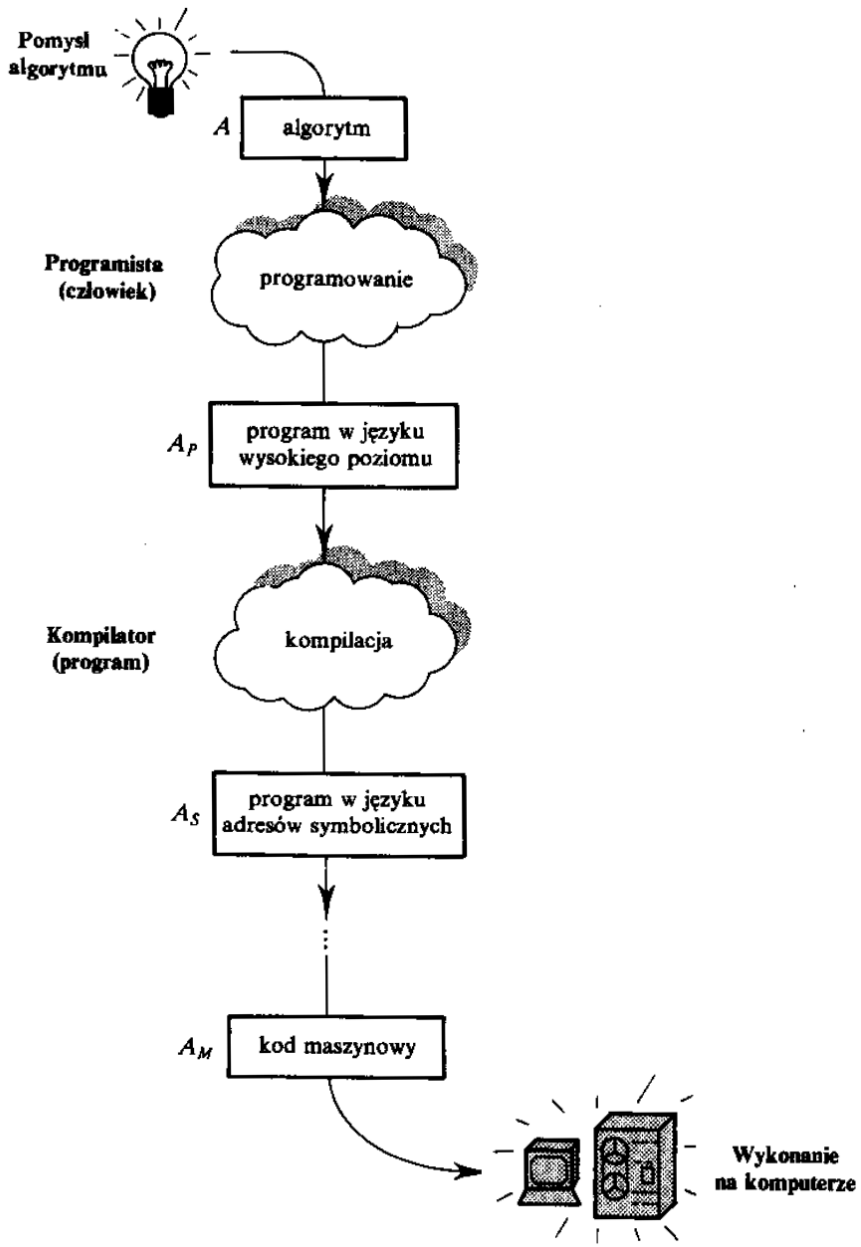
program tłumaczący kod napisany w jednym języku na równoważny kod w innym języku. Przykłady:

- kod źródłowy → kod maszynowy (C, C++, Pascal, Fortran)
- kod źródłowy → *byte code*, kod pośredni, interpretowany lub kompilowany przez maszynę wirtualną (Java, .Net)

Interpreter

odczytuje, analizuje i uruchamia instrukcje zawarte w kodzie źródłowym (brak procesu kompilacji).
Języki skryptowe: Bash, Perl, Python

Od pomysłu do programu



D. Harel, [3]

Wersja minimalistyczna: *edytor tekstu* + *kompilator*

- **GCC** (GNU Compiler Collection)
  Linux, MacOS, Windows (Cygwin, MinGW)
  `cc hello.c -o hello`
- **Clang** (LLVM), Linux, MacOS, Windows (Cygwin, MinGW)
  `clang hello.c -o hello`
- **Microsoft Visual C++** (MSVC), Windows
  `cl hello.c /out:hello.exe`

IDE, Integrated Development Environment
kompilator, edytor, deasembler, debugger, ...

Windows	**Linux**	**Mac OS**
MS Visual Studio	KDevelop, Anjuta	Xcode
Visual Studio Code, CodeBlocks		