rozdział 1:
PROGRAMOWANIE

Historia programowania w pigułce

Podstawy programowania

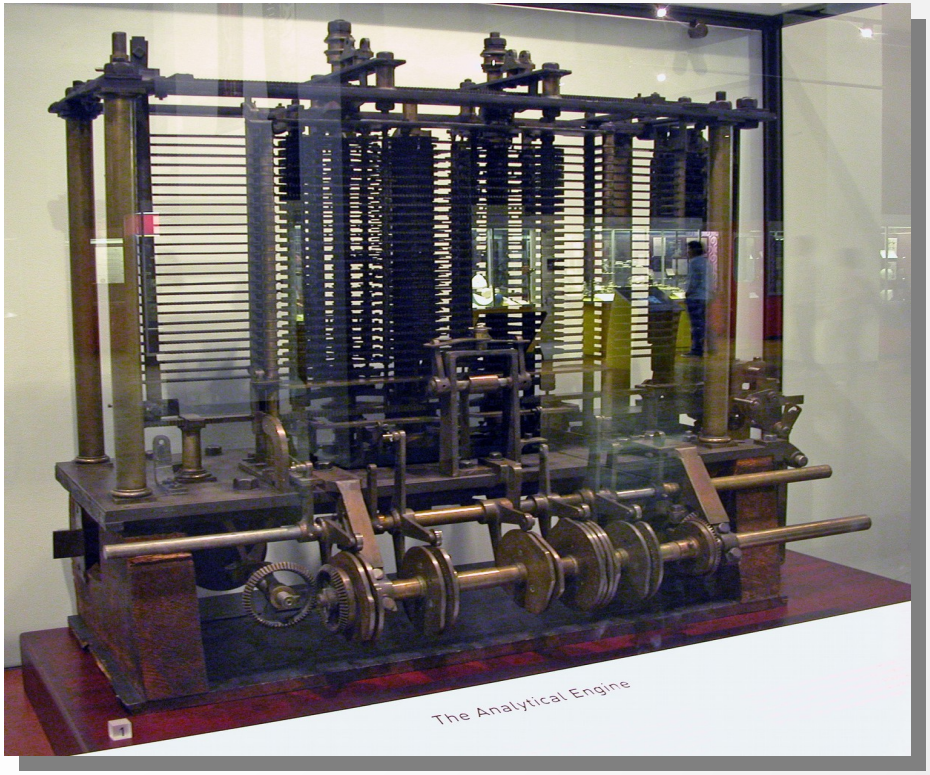
Charles Babbage
1791-1871

- matematyk
- astronom
- mechanik
- konstruktor
- maszyna różnicowa (porzucona)
- maszyna analityczna (nigdy nie zrealizowana)

Podstawy programowania

maszyna analityczna Babbage'a

- napęd: parowy
- nośnik danych: karta perforowana
- pamięć (*store*)
- procesor (*mill*)
- potrafiła(by) realizować:
  - instrukcje warunkowe
  - pętle
- była(by) pierwszym programowalnym urządzeniem w historii ludzkości
- od roku 2011 grupa entuzjastów rekonstruuje maszynę Babbage'a pod patranotem Science Museum w Londynie



Podstawy programowania

Ada Lovelace Byron
1815-1852

- córka Lorda Byrona
- matematyczka
- pierwsza programistka (sic!)
- patronka języka programowania *Ada*
- niecnie wykorzystana przez firmę Microsoft

Podstawy programowania

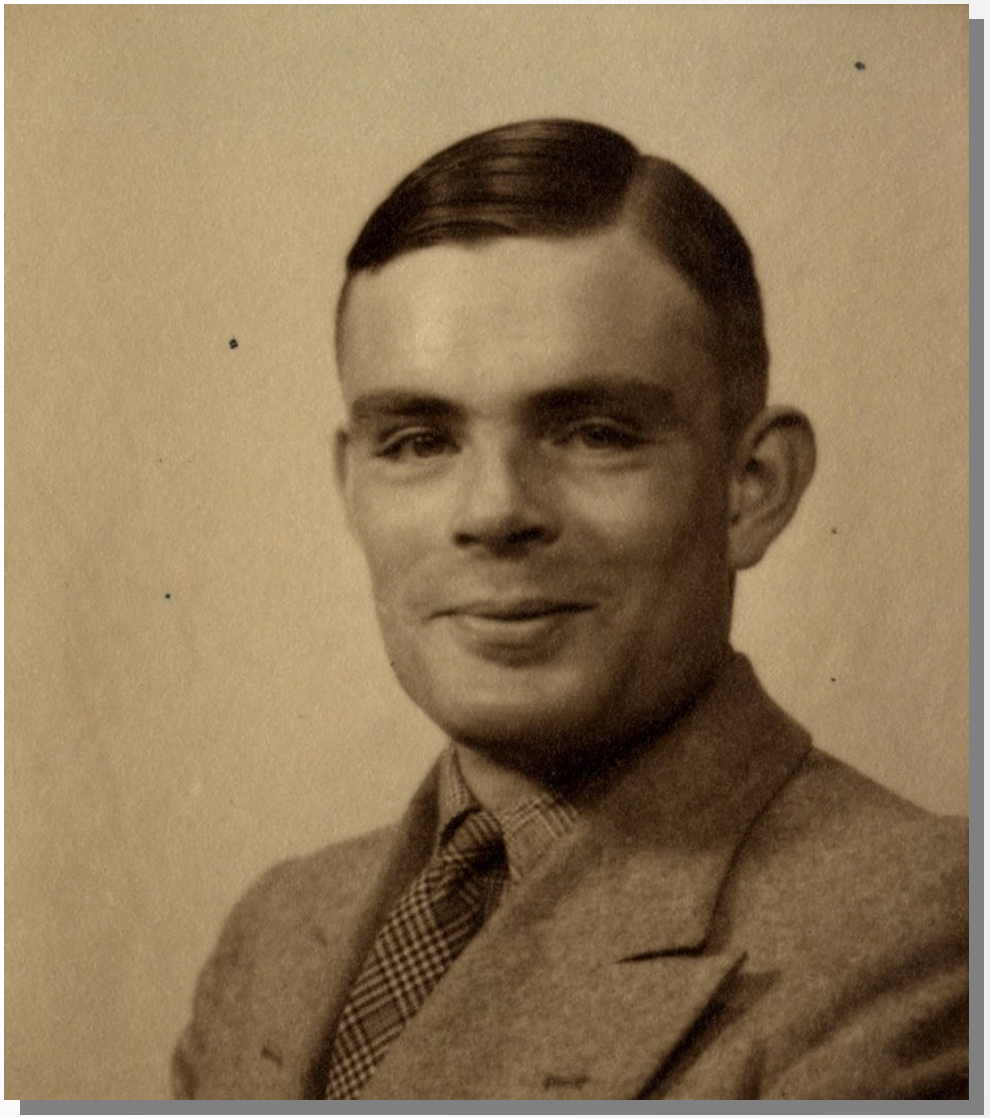
Herman Hollerith
1860 - 1929

- wynalazca, inżynier
- maszyna dla spisu powszechnego 1890
- opatentował kartę perforowaną
- 1896 – założył Tabulating Machine Company
- 1924 – TMC przekształca się w International Business Machines

Podstawy programowania

Alan Turing
1912 - 1954

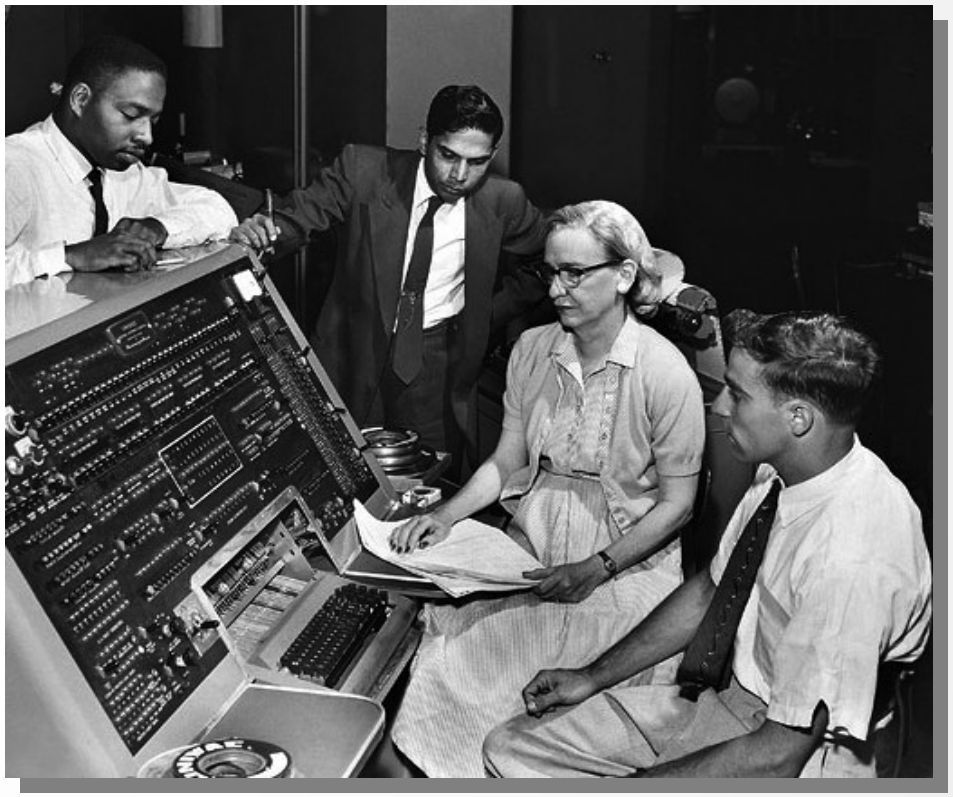
- syn indyjskich emigrantów
- matematyk
- „maszyna Turinga” i algorytm
- projekt komputera w oparciu o prace Babbage'a
- „bomba Turinga” i Bletchley
- „test Turinga”



Podstawy programowania

Grace Hopper
1906-1992

- oficer, a później kontradmirał US Navy
- współautorka pierwszych języków programowania
- wprowadziła nowe znaczenie słowa „bug“ jako błędu w kodzie komputerowym



Podstawy programowania

Dennis Ritchie
1941-2011

- współtwórca systemu operacyjnego Unix
- twórca języka programowania „C“

Podstawy programowania

Richard Stallman
1953-

- twórca ruchu GNU i Fundacji Wolnego Oprogramowania
- współautor wielu fundamentalnych programów narzędziowych
- postać niesłychanie kontrowersyjna

Podstawy programowania

Linus Torvalds
1969-

- twórca pierwszego jądra Linux
- koordynator rozwoju jądra Linux

Podstawy programowania

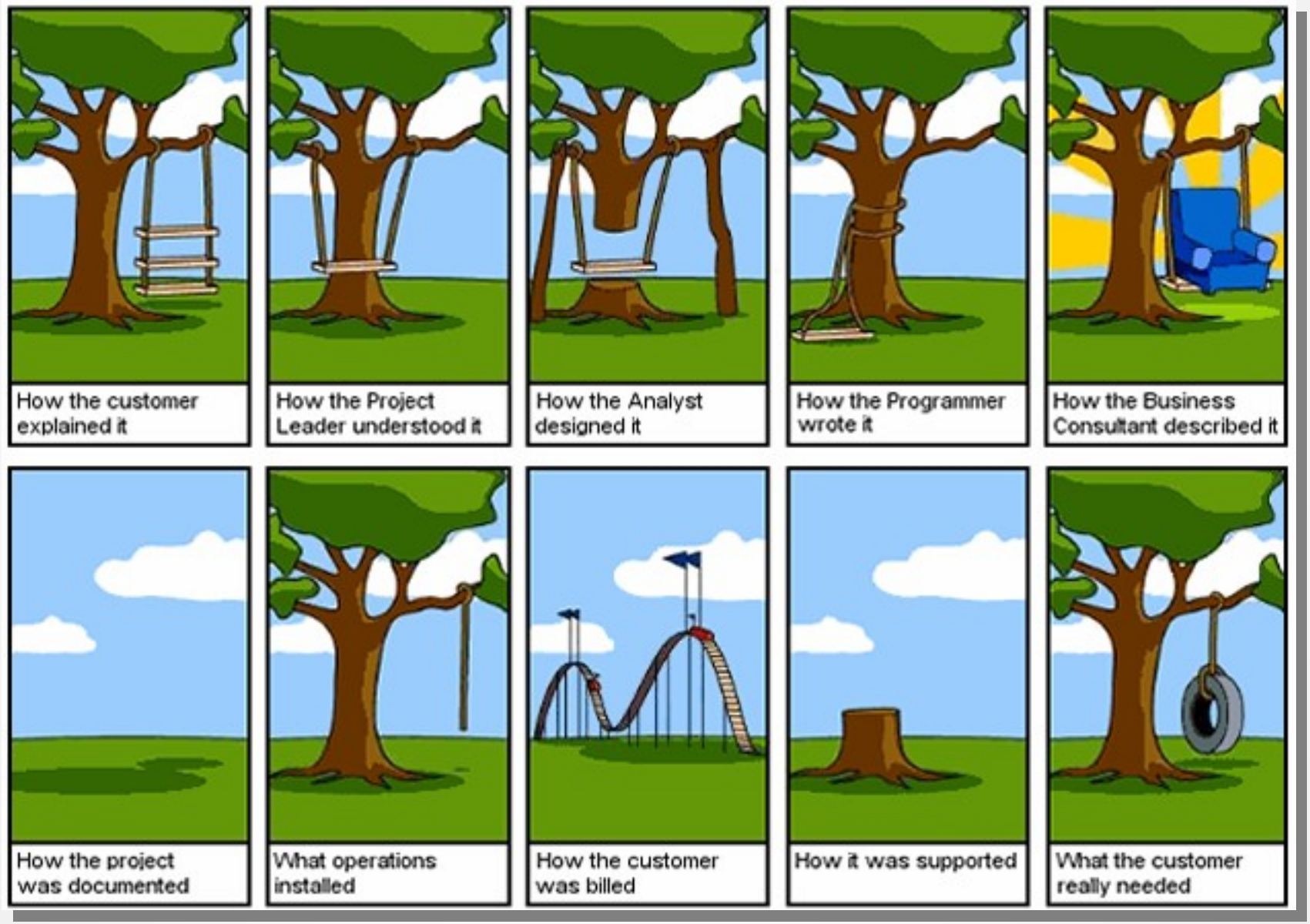
Inna twarz Linusa…

Programowanie

to nie tylko profesja – to także styl życia

Niektórzy twierdzą, że również diagnoza… (psychiatryczna).

Podstawy programowania



Podstawy programowania

Zadajmy dramatyczne pytanie:

Co to jest język?

Podstawy programowania

Język to środek utrwalania i wyrażania myśli.

Pora więc zadać pytanie:

Co to jest język programowania?

Podstawy programowania

Język programowania to język służący sformalizonej reprezentacji algorytmów.

Podstawy programowania

Z czego składa się każdy język?

- **alfabet** → zbiór symboli dopuszczalnych w danym języku

(np. alfabet łaciński w języku angielskim, cyrylica w rosyjskim, kropki i kreski w języku telegraficznym, etc)

Podstawy programowania

- **leksyka** (słownik) → zbiór takich złożeń znaków z alfabetu, który w danym języku na przypisane znaczenie

(np. słowo „*spacer*“ jest poprawne leksykalnie w języku angielskim i polskim, chociaż oznacza coś innego, a słowo „*ndzjkqwiskj*“ nie jest poprawne leksykalnie w żadnym języku)

Podstawy programowania

- **syntaktyka** (składnia) ⟶ zbiór takich złożeń elementów słownika, które są formalnie poprawne w danym języku

(np. polskie wyrażenie „*Programista zjadł bułkę*“ jest poprawne składniowo, a „*Programistą zjedli bułko*“ nie)

Podstawy programowania

- **semantyka** → zbiór wyrażeń poprawnych składniowo , które mają przypisane znaczenie (innymi słowy, mają sens)

(np. polskie wyrażenie „*Programista zjadł bułkę*“ jest poprawne semantycznie, a „*Bułka zjadła programistę*“ niekoniecznie)

Co to jest program?

Podstawy programowania

Program to napis w języku programowania.

Podstawy programowania

Jak czyta się program?

- zwyczajowo program czyta się i wykonuje:
  - od **lewej** do **prawej**
  - od **góry** do **dołu**
- czyli:
  - to, co jest po prawej, wykona się po tym, co jest po lewej
  - to, co jest niżej, wykona się po tym, co jest wyżej
  - uwaga – są **wyjątki**!

Podstawy programowania

Program, aby być poprawnym, musi:

- używać znaków ze swojego alfabetu
- spełniać wymagania leksyki języka
- zawierać konstrukcje poprawne składniowo
- być w zgodzie z wymaganiami semantyki języka
- robić to, co zamierzono

Podstawy programowania

Programista otrzymał zadanie napisania programu, który mnoży liczby **a** i **b**, a wynik przechowuje w **c**.

Programista napisał:

```
c = a + b
```

Powyższy program **jest** poprawny, bo:

- używa właściwego alfabetu
- nie zawiera błędów leksykalnych, składniowych i semantycznych

Powyższy program **nie jest** poprawny, bo:

- nie robi tego, co założono

Nie do końca precyzyjna, ale obrazowa

klasyfikacja języków programowania

Podstawy programowania

- niskiego poziomu
  - maszynowe
  - assemblery
- wysokiego poziomu
  - imperatywne
    - proceduralne
    - obiektowe
    - ...
  - nieimperatywne
    - logiczne
    - funkcyjne (funkcjonalne)
    - ...

Podstawy programowania

języki maszynowe

- ciągi bitów, interpretowane przez sprzęt (procesor) jako instrukcje do wykonania
- niemal kompletnie nieczytelne dla człowieka
- w praktyce każda rodzina procesorów ma inny język maszynowy
- alfabetem języka maszynowego jest tzn. **lista rozkazów**

Podstawy programowania

assembler

- inny sposób zapisu języka maszynowego, posługujący się alfabetem zrozumiałym dla ludzi
- instrukcje maszynowe są w nim zapisywane jako pojedyncze słowa bądź skróty
- w assemblerze zapisuje się szczególnie krytyczne fragmenty kodu np. w systemach operacyjnych
- programowanie w assemblerze wymaga dokładnej wiedzy o konstrukcji procesora

Podstawy programowania

przykład fragmentu programu w assemblerze (bardzo akademicki)

```
mov   ax,[a]
add   ax,[b]
mov   [c],ax
```

Podstawy programowania

języki imperatywne

- od łacińskiego *imperativus* - rozkazujący
- zawierają ciągi precyzyjne ciągi instrukcji, które wykonywane są przez komputer w określonej i przewidywalnej kolejności
- przytłaczająca większość obecnie używanych języków programowania to języki imperatywne

Podstawy programowania

języki nieimperatywne

- zamiast kolejno wykonywanych instrukcji zawierają luźno powiązane *recepty* na rozwiązywanie problemów elementarnych
- wykonanie takiego programu polega na dopasowywaniu tych recept do problemu i szukania ścieżki, która doprowadzi do rozwiązania

Prolog – logiczny język nieimperatywny

```
mezczyzna(adam).
mezczyzna(bogdan).
kobieta(anna).
kobieta(marta).

malzenstwo(adam,anna).
maz(X) :- mezczyzna(X), malzenstwo(X,_).
zona(X) :- kobieta(X), malzenstwo(_,X).
dziecko(adam,anna,marta).
dziecko(adam,anna,bogdan).
rodzenstwo(X,Y) :- dziecko(A,B,X), dziecko(A,B,Y), X \== Y.
```

Podstawy programowania

wracamy do języków imperatywnych

- **komputer** → organicznie zdolny do wykonywania wyłącznie kodu maszynowego
- **człowiek** → organicznie niechętny do pisania programów w kodzie maszynowym, za to chętnie używający licznych języków wysokiego poziomu
- jak wyjść z tego impasu?

Podstawy programowania

rozwiązanie:

- **przetłumaczyć** (oczywiście automatycznie) program z języka wysokiego poziomu na kod maszynowy
- jeżeli takie tłumaczenie będzie **szybkie** i **wygodne**, będziemy mogli zapomnieć o programowaniu w językach maszynowych (niestety, nie zawsze będzie to możliwe...)

Podstawy programowania

tłumaczenie takie obejmuje kolejno

- analizę leksykalna
- analizę składniową
- analizę semantyczną
- i w końcu finalny przekład na język maszynowy

- każda z powyższych faz może zakończyć się wykryciem **błędu** i przerwaniem tłumaczenia

Podstawy programowania

dygresja lingwistyczna:

- polskie słowo *tłumacz* ma dwa angielskie odpowiedniki:
  - *translator*
  - *interpreter*

Podstawy programowania

translator

- **tłumacz**, który tłumaczy dzieła z jednego języka na inny, robiąc to w swoim tempie, przy wygodnym biurku, w domowym zaciszu, popijając wyborną kawę i w swoim tempie (limitowanym zapewne umową z wydawcą)
- możemy powiedzieć, że *translator* tłumaczy **off-line**

Podstawy programowania

interpreter

- **tłumacz**, który tłumaczy wypowiedzi z jednych języków na inne, robiąc to na żywo, **symultanicznie**, w obecności autorów tych wypowiedzi i działając pod presją czasu, bez zbędnej zwłoki
- możemy powiedzieć, że *interpreter* tłumaczy **on-line**

Podstawy programowania

w świecie komputerów istnieje dokładnie taki sam podział – program zapisany w języku wysokiego poziomu **może** zostać wykonany przez komputer dwiema drogami:

- **translacji**
- **interpretacji**

Podstawy programowania

translacja (zwana też **kompilacją**):

- plik zawierający program w języku wysokiego poziomu tłumaczy się na kod maszynowy i wynik tłumaczenia umieszcza się w innym pliku
- taki przetłumaczony plik można następnie uruchamiać dowolnie wiele razy
- program wykonujący takie tłumaczenie nazywa się **kompilatorem**
- kompilator dąży do odnalezienia w źródle **wszystkich** błędów i jeśli ich nie ma, dokonuje przekładu

Podstawy programowania

zauważ!

- **kompilator** jest potrzebny, aby przetłumaczyć program, ale nie jest już potrzebny, aby go wykonać

Podstawy programowania

interpretacja:

- plik zawierający program w języku wysokiego poziomu jest tłumaczony (interpretowany) zawsze wtedy, kiedy jest uruchamiany
- wynik tłumaczenia nie jest nigdzie zapamiętywany
- interpreter zatrzymuje się na pierwszym znalezionym błędzie i odmawia dalszej pracy

Podstawy programowania

zauważ!

- **interpreter** jest potrzebny zawsze wtedy, gdy chcemy wykonać program, ponieważ w tym przypadku tłumaczenie i wykonanie kodu są ze sobą związane nierozerwalnie

Podstawy programowania

ważne!

- języki programowania są z założenia projektowane tak, aby napisane w nich programy były albo kompilowane, albo interpretowane

Podstawy programowania

języki kompilowane:

- C/C++
- C#
- Java
- Pascal
- Fortran

- i wiele, wiele innych

Podstawy programowania

języki interpretowane:

- Perl
- **Python**
- JavaScript
- Ruby
- Basic

- i wiele, wiele innych

Podstawy programowania

kompilacja:

- **zalety**
  - uzyskany przetłumaczony kod jest z reguły bardzo szybki i zwięzły
- **wady**
  - kompilacja może być czasochłonna, a więc czas od napisania programu do chwili uruchomienia może być znaczny

Podstawy programowania

interpretacja:

- **zalety**
  - program można uruchomić niezwłocznie po jego napisaniu
- **wady**
  - kod interpretowany nigdy nie będzie tak wydajny jak skompilowany

Podstawy programowania

co to jest kod źródłowy (*ang. source code*)**?**

- tekst programu zapisany w pliku
- przeznaczony do kompilacji albo intrepretacji
- nazwa takiego pliku jest w zasadzie **dowolna**, ale przyjęte jest, aby **rozszerzenie** wskazywało na **język**, który został użyty do napisania programu

Podstawy programowania

konwencje nazewnicze:

- **.c** język C
- **.cpp, .cxx** język C++
- **.cs** język C#
- **.pl** język Perl
- **.py, .pyw** język Python

- i wiele, wiele innych

Podstawy programowania

czym pisać kod źródłowy?

- **nie wolno** go pisać w edytorach, które potrafią formatować tekst - czyli takie narzędzia jak *MSWord* czy *LibreOffice Writer* są kategorycznie **wykluczone**!
- **należy** go pisać w edytorach, które obrabiają **surowy tekst**, bez jakichkolwiek dekoracji – czyli *Notatnik* z MSWindows jest jak najbardziej OK

Podstawy programowania

czym pisać kod źródłowy?

- istnieją specjalizowane **edytory dla programistów**, zawierające szereg pożytecznych funkcjonalności,np. **Notepad++** (wolne oprogramowania)
- istnieją specjalizowane **środowiska** (tzw. IDE – Integrated Development Enviroment), które oprócz edytora mogą zawierać dodatkowe narzędzia, np. zintegrowany z edytorem kompilator, np. **MSVisualStudio** czy **Eclipse**