

Podstawy Programowania

Wykład XII

Języki programowania

Robert Muszyński

Katedra Cybernetyki i Robotyki, PWr

Zagadnienia: generacje języków programowania, kod maszynowy, assembler, drzewo genealogiczne języków wysokiego poziomu, języki: imperatywne, aplikatywne, deklaratywne, symboliczne, obiektowe; środowiska programistyczne, języki idealnie nieproceduralne, generatory aplikacji, inteligentne systemy wiedzy.

Copyright © 2007–2025 Robert Muszyński

Niniejszy dokument zawiera materiały do wykładu na temat podstaw programowania w językach wysokiego poziomu. Jest on udostępniony pod warunkiem wykorzystania wyłącznie do własnych, prywatnych potrzeb i może być kopiowany wyłącznie w całości, razem ze stroną tytułową.

– Skład FoilT_EX –

Generacje języków programowania

- Pierwsza generacja — kod maszynowy
- Druga generacja — assembly
- Trzecia generacja — języki wysokiego poziomu
 - ★ języki imperatywne (proceduralne)
 - ★ języki aplikatywne (funkcjonalne)
 - ★ języki deklaratywne (regułowe)
 - ★ języki symboliczne
 - ★ języki obiektowe
- ;-) Generacja trzy i pół — środowiska programistyczne
- Czwarta generacja — języki idealnie nieproceduralne, generatory aplikacji
- Piąta generacja — inteligentne systemy wiedzy

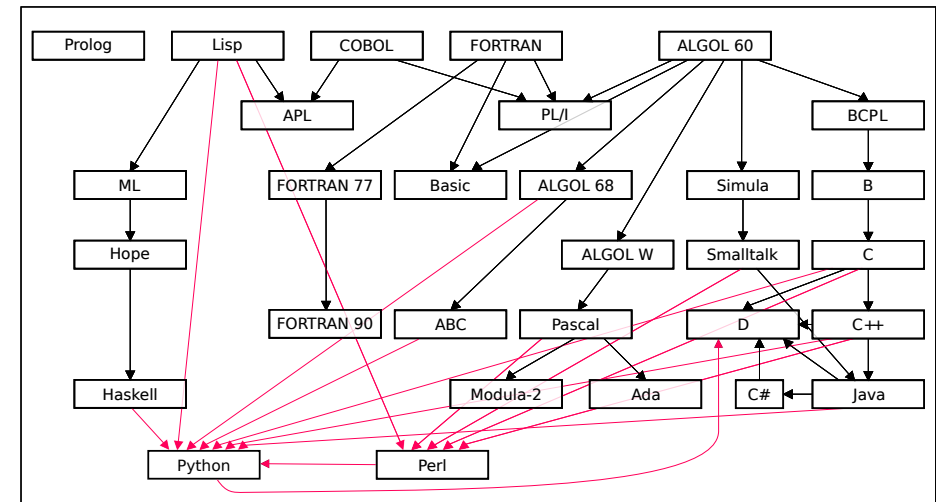
Przykłady języków wysokiego poziomu

- ★ języki imperatywne (proceduralne) — Fortran, ALGOL, COBOL, C
- ★ języki aplikatywne (funkcjonalne) — Lisp, Scheme, Haskell
- ★ języki deklaratywne (regułowe) — Prolog, CLIPS, SQL
- ★ języki symboliczne — Lisp, Prolog, Mathematica, Maple
- ★ języki obiektowe — Smalltalk, Ada95, Lisp, C++, C#, Java

Środowisko programistyczne

Język programowania wysokiego poziomu + wbudowane funkcje dla systemów informacyjnych (obsługa ekranu, bazy danych itp.) + interfejs graficzny — buildery i wizardy

Drzewo genealogiczne języków wysokiego poziomu (domniemane)



Co więc wybrać

Wyjątki z listy 50 najbardziej popularnych języków programowania (www.tiobe.com/tiobe-index/):

- 15 Asembler (1949) (był 9, 16, 8)
- 12 Fortran (1957)
- 42 Lisp (1958)
- ~~14~~ BASIC (1964) (Visual Basic .NET (2001) 8 miejsce)
- 28 Prolog (1972)
- 2 C (1972) (11,44% rynku, spadek 4,81%, było 16% rynku, przyrost 3,82%)
- 3 C++ (1985) (9,96% rynku, spadek 2,95%, było 13% rynku, przyrost 4,62%)
- 30 Perl (1987)
- 1 Python (1990) (13,97% rynku, spadek 2,39%, było 16% rynku, przyrost 2,78%)
- 4 Java (1995) (7,87% rynku, spadek 4,34%, (było 12% rynku, przyrost 1,55% w 2020 miejsce 1: 15%, przyrost 0,88%)

Wniosek:

Nowe języki powstają ustawicznie i koegzystują ze starszymi

Jak wybrać

Obserwacja:

Typowy programista, inżynier oprogramowania (architekt:) z kilkuletnim doświadczeniem programuje w kilkunastu językach

Zamiast pytać:

Jaki język programowania jest obecnie najpopularniejszy?

warto zapytać:

Który język programowania jest obecnie niedoceniany i pozwala robić rzeczy, których inni programiści nie są w stanie zrobić, tym samym dając mi unikalne umiejętności?

Popularne kryteria wyboru języka programowania do nauki:

- Wysokość uposażenia
- Popularność – ogrom różnorodnych ofert pracy
- Wzrostowa tendencja popularności
- Łatwość nauczenia, komfort pracy

Języki, które „dobrze rokują”:

- Rust (rekomendacja AiR!)
- TypeScript (nadzbiór JavaScript)
- Python (≥ 3.0)
- Swift (iOS)
- Kotlin (Android)
- Go (Golang – by Google)
- R (danologia – data science)
- Dart (w SDK Flutter, aplikacje mobilne)
- Julia (obliczenia naukowe)
- Zig??? (następca C)
- plus biblioteki!

Warto zapamiętać także, że:

Tworzone aplikacje działają w jakimś środowisku sprzętowym, zazwyczaj pod nadzorem jakiegoś systemu operacyjnego, więc...

Rust, czyli co?

- 2006 projekt, 2009 opieka Mozilli, 2011 „samokompilacja”, 2012 wersja alfa, 2015 wersja 1.0, 2021 Fundacja Rust (Mozilla, AWS, Google, Microsoft, Huawei), przyjęty przez Amazon, Discord, Dropbox, Meta
- Nacisk na bezpieczeństwo
- Dobrze zoptymalizowany kod wynikowy
- Start dużo łatwiejszy niż w C++
- Czytelność komunikatów o błędach
- Duża szansa działania według założeń przy „warunkach brzegowych”
- Pierwszy język inny niż C i assembler wspierany w rozwoju jądra Linuksa
- Wycieki pamięci wyeliminowane z definicji
- Także wyścigi w programach wielowątkowych
- Świetne „oprzyrządowanie” (builder, menedżer pakietów/projeków)
- Krytyka za blok `unsafe`

Popularność języków według dziedzin

- Oprogramowanie niskiego poziomu, sprzętowe, jądro systemu: **C, assembler, Rust**
- Systemy wbudowane: **C/C++, Embedded C, Python, Java, Ada, assembler, Rust, Lua, Verilog**
- Automatyka przemysłowa: **PLC Ladder, STL, SCL** (Siemens, Allen-Bradley, Mitsubishi Electric, Beckhoff), **SCADA, DCS** → **Industrial IoT (Python, Java, C/C++ z protokołami komunikacji), MATLAB, LabVIEW** (język G)
- Robotyka: **C/C++, Python, Java, C#, Matlab, assembler, HDLs (FPGA), Lisp, Rust**, języki programowania robotów: **RAPID** (ABB), **KRL** (Kuka), **Karel** (Fanuc), **AS** (Kawasaki), **INFORM** (Yaskawa), **VAL3** (Staubli), **URScript** (Universal Robots)
- Aplikacje www: **JavaScript, TypeScript, Elm, Scala, Python, C#, PHP, Go, Ruby, Dart, Flutter, Rust, Cordova, Svelte**
- Aplikacje mobilne: **HTML, Swift, Kotlin, C/C++, C#, Java, JavaScript, Python, Dart, Ruby, Rust**
- Gry: **C/C++, Java, C#, Unity, TypeScript, Visual Basic/.NET, Rust, Swift, Lua, Python, Lisp, Perl, Smalltalk**
- Chmury obliczeniowe: **SQL, XML, Python, R, Clojure, Haskell, Scala, OCaml, JavaScript, F#, Common Lisp, Scheme**
- Uczenie maszynowe: **Python, R, Julia, Java, Lisp, JavaScript, C++**
- Blockchain: **C++, Java, Python, Simplicity, Solidity, Serpent, LLL**
- Roboty Boston Dynamics: **C, C++, Python z ROsem**
- Robot Sophia: **Java, C#, C++, OSGi, CogChar, Prolog, AIML, ChatScript, SingularityNET**

Biblioteki, zestawy narzędzi (SDK/API), środowiska (IDE), platformy (frameworks), „buildery” i inne „ekosystemy”

- Systemy wbudowane: **STL, ETL, CMSIS, HAL libs., Unity, FreeRTOS, PlatformIO, Zephyr, Mbed, LVGL, Qt For MCU, Slint, Embassy, Android, Java Embedded Framework**
- Robotyka: **ROS (Robotic Op. Syst.), ROS Industrial, ACADO Toolkit, Robotic Systems Toolbox, Robotics Toolbox for MATLAB, SUNDIALS, NVIDIA Isaac SIM, Gazebo, RoboDK, Webots, Unity, AWS RoboMaker, v-rep, Microsoft Robotics Developer Studio, LabVIEW Robotics Module, ROS-Based Tools like RViz, ROS-Mobile, MoveIt Studio**
- Aplikacje www: **FastAPI, GraphQL, .Net, Node.js, Express, Next.js, Nuxt.js, React.js, Angular, Prisma, Laravel, Symfony, WordPress, Redis, RabbitMQ, Apache kafka**
- Interfejs użytkownika (HMI): **Unreal Engine, Unity, Qt, Kanzi, Android Automotive, QNX**
- Gry: **SDL, SFML, Allegro, AllegroGL, CanLib, Crystal Space**
- Python: **Django, Web2Py, Flask, Bottle, CherryPy, FastAPI, Pyramid, Tornado, PyQt, Tkinter, Kivy, PySide, PySimpleGUI, scikit-learn, TensorFlow, PyTorch, Keras, NumPy, SciPy, Pandas, Matplotlib, Pytest, Unittest, nose2, asyncio, Aiohttp**
- JavaScript: **jQuery, React, AngularJS, Bootstrap, Aurelia, Mocha, Gatsby**
- Ruby: **Ruby on Rails, Sinatra, Hanami, Django**
- Narz. do aut. zarz. proc. komp. programu: **GNU make, CMake, nmake (Windows), MPW Make (macOS), Bazel, Meson, Maven, Rust Cargo, Ninja**
- kontenery: **Docker, Kubernetes**
- I jeszcze skrypty / mini narzędzia :)

W podsumowaniu

- Co bym wybrał, gdybym zaczynał jeszcze raz?
 - ★ C++: – niewiele wybacza, ale przez to bardzo dużo uczy
 - samo mięsko, samo gęste (ale z wielkimi możliwościami wiąże się wielka odpowiedzialność)
 - ★ Świeżym studentom AiRu najbardziej polecałbym Rusta, potem Pythona
 - ★ Python to super język-klej z automatycznie generującą się dokumentacją
 - ★ Javascript/Typescript – prędzej czy później warto się zapoznać
 - ★ No i jeszcze skrypty / mini narzędzia, w bashu lub Pythonie (takie trytytki)
 - ★ No i polecam też jak najprędzej przyswoić sobie Dockera
 - ★ No i nauczyć się pisać na klawiaturze bez patrzenia na nią
 - ★ Tak czy inaczej warto być miłym :)
- Jak wyglądają trendy i rynek pracy?
 - ★ Coroczna ankieta Stack Overflow
 - ★ Raport Just Join IT
 - ★ Oferty pracy Just Join IT

Kod maszynowy i asemblery

Dodanie dwóch liczb na MC68HC705J1A (Motorola, 8-bitowy)

Kod maszynowy

Komorka Pamięci	Zawartosc
0300	A6
0301	02
0302	AB
0303	02
0304	CC
0305	03
0306	04
07FE	03
07FF	00

To samo w S-rekordach

```
S10C0300A602AB02CC0304C8
S10507FE0300F2
S9030000FC
```

Asembler

```
org $0300
DODAJ lda #2 ;zaladuj do akumulatora 2
add #2 ;dodaj do akumulatora 2
PETLA jmp PETLA ;skocz do PETLA
org $07FE
dw DODAJ ;adres startu po resecie
```

Po przetworzeniu

```
0300 1 org $0300
0300 A602 2 DODAJ lda #2
0302 AB02 3 add #2
0304 CC0304 4 PETLA jmp PETLA
07FE 5 org $07FE
07FE 0300 6 dw DODAJ
Symbol Table
PETLA 0304
DODAJ 0300
```

Języki wysokiego poziomu

- język imperatywny — C

```
int dodaj()
{ int a = 2;
  int b = 2;
  return(a + b);}
int l = dodaj();
```

- język aplikatywny — Lisp

```
(defun dodaj() ;aplikatywnie
  ((lambda (x y) (+ x y)) 2 2))
(defun dodaj() ;imperatywnie
  (let (x y) (setq x 2) (setq y 2) (+ x y)))
(dodaj)
```

- język deklaratywny — Prolog

```
dodaj(X) :- X is 2 + 2.
dodaj(X).
```

- język symboliczny — Mathematica

```
dodaj := Module[{a = 2, b = 2}, a + b];
l = dodaj;
```

- język obiektowy — C++

```
class CDodaj
{ public:
  int a;
  int b;
  CDodaj()
  { a = 2;
    b = 2;};
  int Dodaj()
  { return(a + b);}
};
/*statycznie*/
Cdodaj d;
int l = d.Dodaj();
/*dynamicznie*/
Cdodaj *d = new CDodaj;
int l = d->Dodaj();
```

Języki czwartej generacji

Elementy składowe języka czwartej generacji:

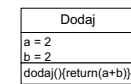
- fizyczny słownik danych
- formater ekranu
- generator raportów
- język zapytań
- specyfikator dialogu
- generator kodu
- język wysokiego poziomu

Języki czwartej generacji:

- CASE — Computer Aided Software Engineering
- Oracle — Komercyjny pakiet zawierający:
 - ★ maszynę relacyjnej bazy danych;
 - ★ interfejs bazy danych
 - ★ narzędzia do integracji baz danych z językami wysokiego poziomu;
 - ★ kreator formularzy;
 - ★ kreator raportów;
 - ★ kreator reprezentacji graficznych;
 - ★ narzędzia do wyszukiwania
- UML — Unified Modeling Language

UML

Dodanie dwóch liczb



Symulacja systemu robotycznego

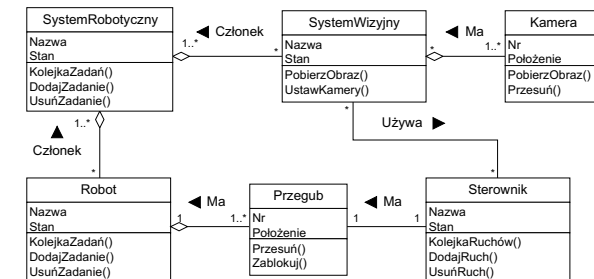


Diagram klas

Do tego należy utworzyć:

- diagram obiektów
- diagram implementacji
- diagram rozmieszczenia
- diagram zachowań
- diagram sekwencji
- diagram interakcji

Symulacja systemu robotycznego cd.

- C++

```
class CRobot
{ private:
  int id
  public:
  char      Nazwa[42];
  CStan     Stan;
  CPrzegub *Przeguby;
  CRezolwer *Rezolwery;
  CKolejka  *Kolejka = NULL;

  CRobot()
  { Przeguby=new CPrzegub[5]; /*...*/
    Inicjuj(); /*...*/;

    int KolejkaZadan(){/* definicja */;
    int DodajZadanie(/*...*/){/* def. */;};
    int UsunZadanie(/*...*/){/* def. */;};};

  CRobot *Robot1 = new CRobot;
  try{Robot1.DodajZadanie(/*...*/);}
  catch(CException blad){/* obsluga */}
```

- C

```
struct SRobot
{ int id
  char      Nazwa[42];
  SStan     Stan;
  SPrzegub  *Przeguby;
  SRezolwer *Rezolwery;
  SKolejka  *Kolejka = NULL;};

int DodajRobota(SRobot *Robot)
{ Robot = malloc(sizeof(SRobot));
  Robot->Przeguby=malloc(5*sizeof(SPrzegub));
  Inicjuj(); /*...*/;
int UsunRobota(SRobot *Robot){/*...*/;
int KolejkaZadan(/*...*/){/* def. */;};
int DodajZadanie(/*...*/){/* def. */;};
int UsunZadanie(/*...*/){/* def. */;};};

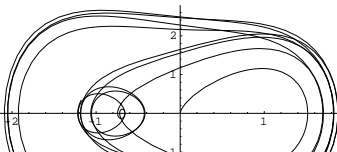
SRobot *Robot1;
DodajRobota(Robot1);
if blad = DodajZadanie(/*...*/) then
/* obsluga */
```

Symulacja systemu robotycznego cd.

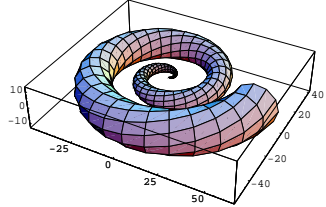
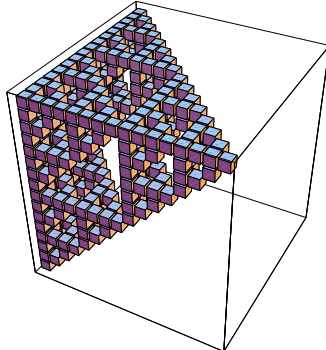
- Asembler i kod maszynowy

```
98: float CRobot::UstalKrok(float Krok, int Kp)
99: {
00402828  push    ebp
00402829  mov     ebp,esp
0040282B  sub     esp,0Ch
0040282E  mov     dword ptr [ebp-8],ecx
100: float delta = 0.001;
00402831  mov     dword ptr [delta],3A83126Fh
101: switch(Kp){
00402838  mov     eax,dword ptr [Kp]
0040283B  mov     dword ptr [ebp-0Ch],eax
0040283E  cmp     dword ptr [ebp-0Ch],0
00402842  je      CRobot::UstalKrok(0x0040284c)+24h
00402844  cmp     dword ptr [ebp-0Ch],1
00402848  je      CRobot::UstalKrok(0x00402857)+2Fh
0040284A  jmp     CRobot::UstalKrok(0x00402862)+3Ah
102: case 0: Krok += delta;
0040284C  fld     dword ptr [Krok]
0040284F  fadd   dword ptr [delta]
00402852  fstp   dword ptr [Krok]
103: break;
00402855  jmp     CRobot::UstalKrok(0x00402869)+41h
104: case 1: Krok -= delta;
00402857  fld     dword ptr [Krok] ...
```

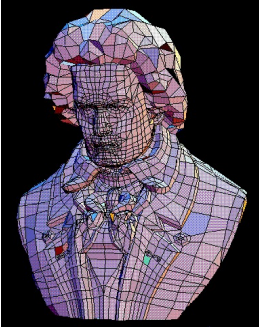
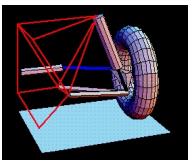
Języki symboliczne — Mathematica

<code>Sqrt[4]</code> 2	<code>Integrate[Sqrt[x] Sqrt[a + x], x]</code>
<code>Sqrt[-4]</code> 2i	$\sqrt{a+x} \left(\frac{a\sqrt{x}}{4} + \frac{x^{3/2}}{2} \right) - \frac{1}{4}a^2 \log[\sqrt{x} + \sqrt{a+x}]$
<code>Sqrt[a]</code> \sqrt{a}	$\int \sqrt{x}\sqrt{a+x} dx$
<code>Sqrt[a]^2 + a + 3b + 5b</code> 2a + 8b	$\sqrt{a+x} \left(\frac{a\sqrt{x}}{4} + \frac{x^{3/2}}{2} \right) - \frac{1}{4}a^2 \log[\sqrt{x} + \sqrt{a+x}]$
<code>Solve[x^2 + 5x + 4 == 0, x]</code> {x → -4}, {x → -1}	<code>NDSolve[{x''[t] + x[t]^3 == sin[t], x[0] == x'[0] == 0}, x, {t, 0, 50}]</code> {x → InterpolatingFunction[{{0., 50.}}, <>]}
<code>Solve[ax^2 + bx + c == 0, x]</code> {x → $\frac{-b - \sqrt{b^2 - 4ac}}{2a}$ }, {x → $\frac{-b + \sqrt{b^2 - 4ac}}{2a}$ }	<code>ParametricPlot[Evaluate[{x[t], x'[t]}/.%, {t, 0, 50}];</code>
<code>Solve[Sqrt[x] + a == 2x, x]</code> {x → $\frac{1}{8}(1 + 4a - \sqrt{1 + 8a})$ }, {x → $\frac{1}{8}(1 + 4a + \sqrt{1 + 8a})$ }	

Języki symboliczne — Mathematica

```
ParametricPlot3D[{u cos[u](4 + cos[v + u]), u sin[u](4 + cos[v + u]), u sin[v + u]}, {u, 0, 4π}, {v, 0, 2π}, PlotPoints → {60, 12}];
Show[Graphics3D[Flatten[Table[If[Mod[Multinomial[x, y, z], 2] == 1, Cuboid[1.2{x, y, -z}], {}], {x, 0, 15}, {y, 0, 15}, {z, 0, 15}]]]]]
```

Inne języki

- języki opisu strony — html, $\text{T}_{\text{E}}\text{X}/\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$
- języki manipulowania tekstem — sed, awk
- meta-języki — Lex/Flex, Yacc/Bison
- języki „specjalistyczne/dedykowane” — HDLs, Ladder Diagram, SCADA, RAPID, Karel
- APL — A Programming Language

```

▽ FIB N
[1] A ← 1 1
[2] → 2 × N > 0 A ← A, +/- 2 ↑ A ▽

```

Komputery kwantowe

- Python — z odpowiednimi bibliotekami (np. QuTiP)
- Qiskit — SDK dla Pythona (opracowany przez IBM dla komputerów IBM Quantum)
- Ocean — SDK dla Pythona (opracowany przez D-Wave dla swoich rozwiązań)
- Q# — microsoftowy QDK (integruje się z .NET jak C# czy F#)
- Cirq — framework dla komputerów NISQ (noisy intermediate scale quantum, opracowany przez Google Quantum AI dla swoich komputerów)

Zagadnienia pokrewne

- Klasyfikacja stylów programowania
 - * programowanie transformacyjne
 - * programowanie reaktywne
 - * styl imperatywny bez i z procedurami
 - * styl imperatywny z modułami/pakietami
 - * styl obiektowy
 - * styl aplikacyjny
 - * styl programowania sterowanego danymi
- Techniki i metody tworzenia systemów informacyjnych
- Planowanie i zarządzanie systemami informacyjnymi
- Ocena systemów informacyjnych

Więcej w Paul Beynon-Davies, „Inżynieria systemów informacyjnych”, WNT 2004.

Podsumowanie

• Zagadnienia podstawowe

1. Wskaż podstawowe różnice w procesie programowania w językach niskiego i wysokiego poziomu.
2. Z jakiego języka wywodzi się język C?
3. Jaką podstawową zaletę posiada język C++ w porównaniu z językiem C?
4. Czym się wyróżniają języki regułowe?
5. Czym jest środowisko programistyczne i jakie są jego podstawowe elementy?
6. Czym się różni środowisko programistyczne od języka programowania?
7. Jak klasyfikuje się style programowania? Czym one się różnią?
8. Czy styl programowania jednoznacznie wynika z rodzaju użytego języka programowania? A jaki wpływ na styl ma wybrany kompilator języka (środowisko programistyczne)?

• Zagadnienia rozszerzające

1. Jakie style programowania można stosować w języku C?
2. Jakie rodzaje diagramów definiowane są w języku UML? Do czego służą?
3. Podaj jakie są dziedziny zastosowań każdego z języków wysokiego poziomu wymienionych na 2 stronie prezentacji.