Informatyka 1

Wykład XII

Języki programowania

Robert Muszyński
ZPCiR ICT PWr

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

– Skład FoilTEX –

Generacje języków programowania

- Pierwsza generacja — kod maszynowy
- Druga generacja — asemblery
- 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

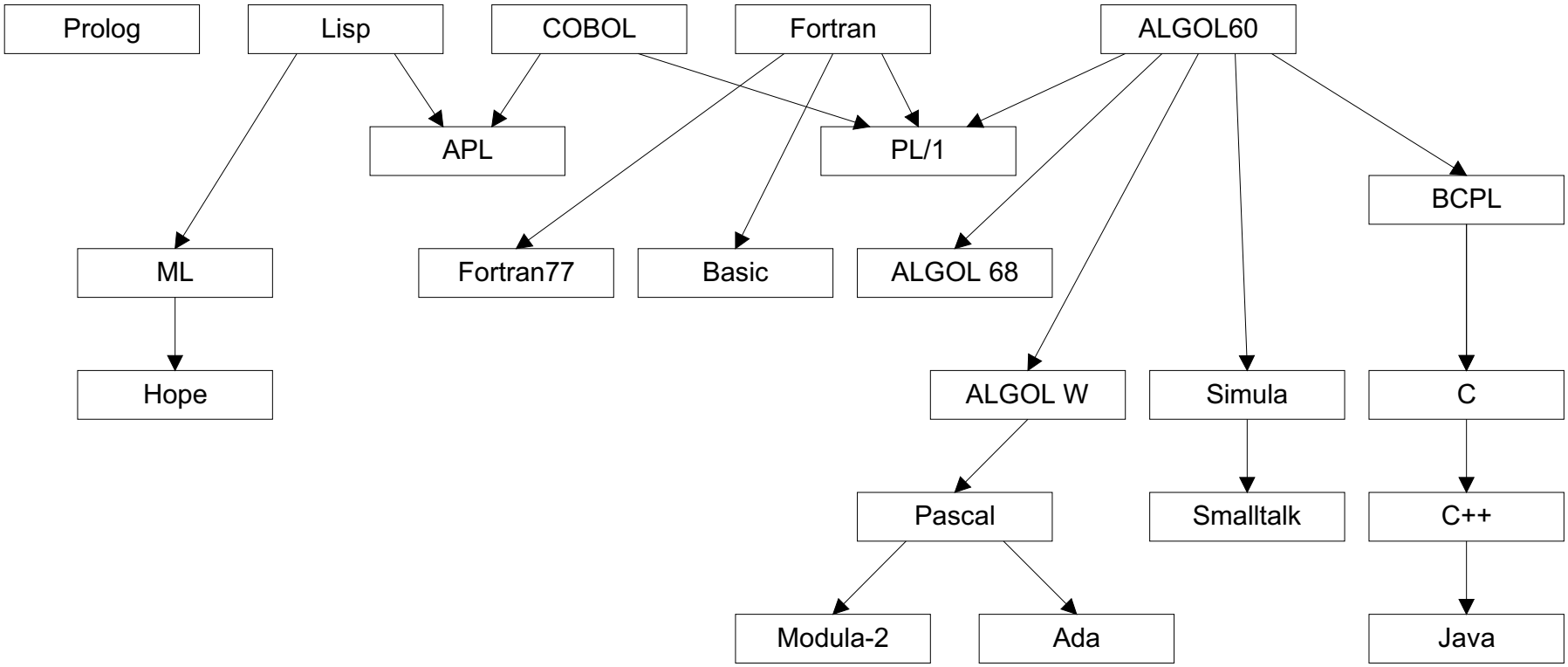
Przykłady języków wysokiego poziomu

- ⋆ języki imperatywne (proceduralne) — `Fortran, COBOL, C`
- ⋆ języki aplikatywne (funkcjonalne) — `Lisp`, `Scheme`
- ⋆ języki deklaratywne (regułowe) — `Prolog, CLIPS`
- ⋆ języki symboliczne — `Lisp, Prolog, Mathematica, Maple`
- ⋆ języki obiektowe — `Smalltalk, Ada95, Lisp, 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)



Kod maszynowy i asemblery

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

Kod maszynowy

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

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
```

To samo w S-rekordach

```
S10C0300A602AB02CC0304C8
S10507FE0300F2
S9030000FC
```

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:

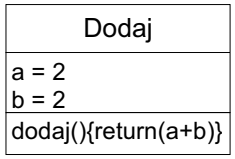
- fizyczny słownik danych
- formater ekranu
- generator raportów
- język zapytań
- specyfikator dialogu
- generator kodu
- język wysokiego poziomu

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

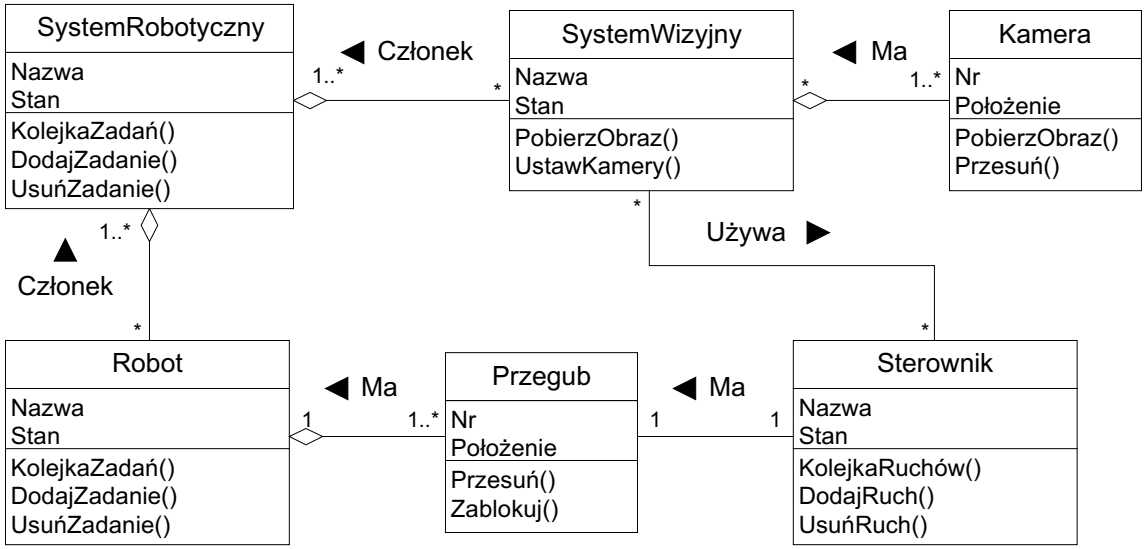


Diagram klas

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

```
Sqrt[4]
```
$$2$$

```
Sqrt[-4]
```
$$2i$$

```
Sqrt[a]
```
$$\sqrt{a}$$

```
Sqrt[a]^2 + a + 3b + 5b
```
$$2a + 8b$$

```
Solve[x^2 + 5x + 4 == 0, x]
```
$$\{\{x \to -4\}, \{x \to -1\}\}$$

```
Solve[ax^2 + bx + c == 0, x]
```
$$\left\{\left\{x \to \frac{-b-\sqrt{b^2-4ac}}{2a}\right\}, \left\{x \to \frac{-b+\sqrt{b^2-4ac}}{2a}\right\}\right\}$$

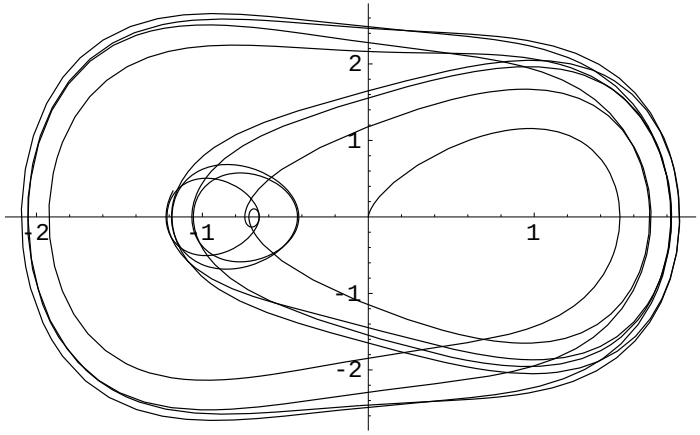
```
Solve[Sqrt[x] + a == 2x, x]
```
$$\left\{\left\{x \to \frac{1}{8}\left(1 + 4a - \sqrt{1+8a}\right)\right\}, \left\{x \to \frac{1}{8}\left(1 + 4a + \sqrt{1+8a}\right)\right\}\right\}$$

```
Integrate[Sqrt[x] Sqrt[a + x], x]
```
$$\sqrt{a+x}\left(\frac{a\sqrt{x}}{4} + \frac{x^{3/2}}{2}\right) - \frac{1}{4}a^2 \log\left[\sqrt{x} + \sqrt{a+x}\right]$$

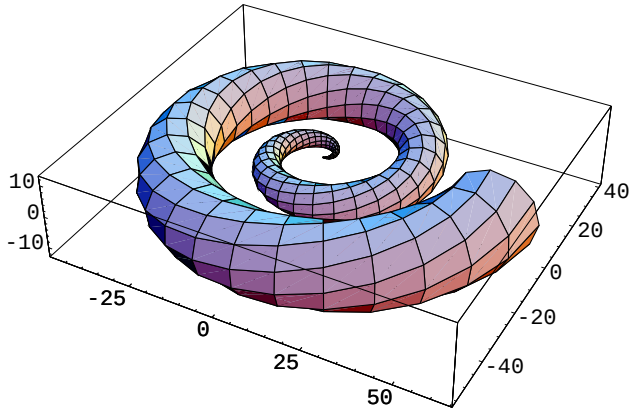
$$\int \sqrt{x}\sqrt{a+x}dx$$
$$\sqrt{a+x}\left(\frac{a\sqrt{x}}{4} + \frac{x^{3/2}}{2}\right) - \frac{1}{4}a^2 \log\left[\sqrt{x} + \sqrt{a+x}\right]$$

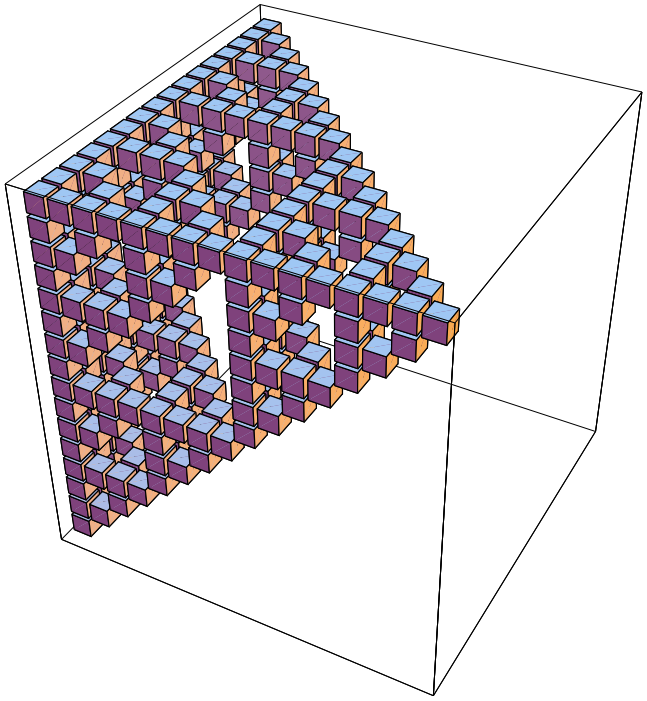
```
NDSolve[{x''[t]+x[t]^3 == Sin[t], x[0] == x'[0] == 0},
x, {t, 0, 50}]
```
$$\{\{x \to \text{InterpolatingFunction}[\{\{0., 50.\}\}, <>]\}\}$$

```
ParametricPlot[Evaluate[{x[t], x'[t]}/.%], {t, 0, 50}];
```



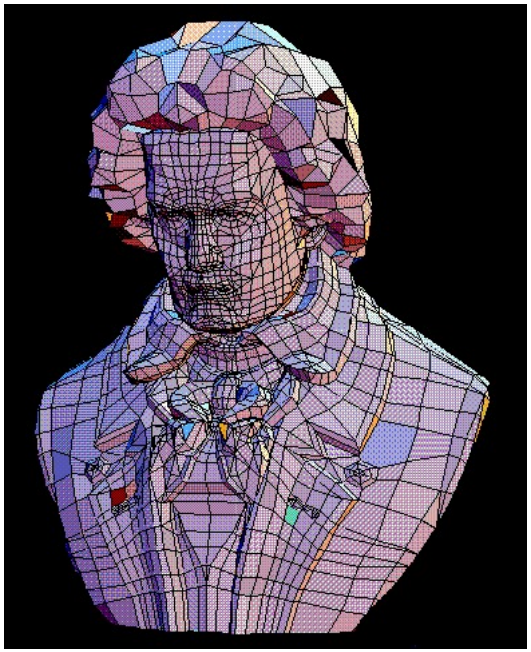
Języki symboliczne — Mathematica

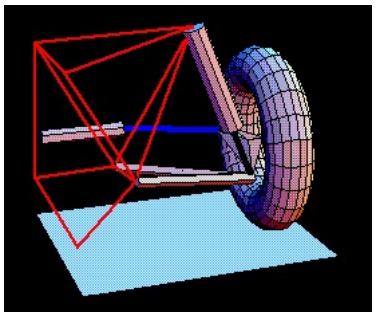




```
ParametricPlot3D[{u Cos[u](4 + Cos[v + u]), u Sin[u](4 + Cos[v +
u]), u Sin[v + u]}, {u, 0, 4π}, {v, 0, 2π}, PlotPoints → {60, 12}];
```

```
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`, TEX/LATEX
- języki manipulowania tekstem — `sed`, `awk`
- meta-języki — `Lex/Flex`, `Yacc/Bison`
- `APL` — A Programming Language

```
∇  FIB N
[1]  A←1 1
[2]  →2×N>⍴A← A, +/¯2↑A∇
```

Zagadnienia pokrewne

- Klasyfikacja styli programowania
  - ⋆ programowanie transformacyjne
  - ⋆ programowanie reaktywne
  - ⋆ styl imperatywny bez i z procedurami
  - ⋆ styl imperatywny z modułami/pakietami
  - ⋆ styl obiektowy
  - ⋆ styl aplikatywny
  - ⋆ 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 1999.