FAKULTA
INFORMATIKY
Masarykova univerzita

PV109: Historie a vývojové trendy ve VT

Od strojového kódu k programovacím jazykům

Luděk Matyska a Eva Hladká

podzim 2019

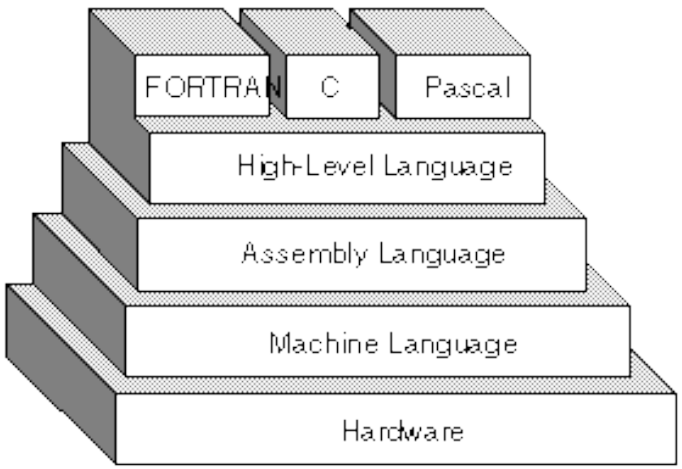
Programovací jazyky v historickém vývoji

Strojově závislé

- Strojový kód
- Jazyky relativních adres
- Jazyky symbolických adres – assemblery
- Autokódy

Strojově nezávislé

- Zaměřené na určitý typ úloh
  - Fortran – vědeckotechnické výpočty
  - Cobol – hromadné zpracování dat
  - Pascal – výukové programy
  - Simula, Simscript – modelování diskrétních systémů
  - Eiffel – objektově orientované
- Univerzální
  - Algol 60, PL/I, Algol 68, Ada



Zdroj: `http://webopedia.com`

Procedurální vs. deklarativní jazyky

- Procedurální
  - důraz na popis postupu výpočtu
  - how to do it
- Deklarativní
  - důraz na popis řešeného problému
  - what to do
  - LISP, Prolog, databázové jazyky apod.

Strojový kód

- Program se skládá z *instrukcí*
- Instrukci tvoří
  - kód operace – *co* se má provést
  - jedna či více adres – *s jakými daty*
  - absolutní adresy
- Instrukce zapsány v dvojkové nebo šestnáctkové soustavě
- Každý *typ* počítače zpracovává vlastní sadu instrukcí
- *Rodiny počítačů* sdílejí stejnou množinu instrukcí

```
00 00 00 00 00 90 BE 07 - D8 07 F2 07 F2 07 F2 07
F2 07 F2 07 F2 07 F2 07 - 0C D8 00 00 00 00 C0 00
D5 01 2E 8F 06 3C D8 0E - 2E FF 36 3C D8 2E FF 36
3E D8 2E FF 36 40 D8 55 - CB FA FC 2B F6 8E DE 2E
89 2E 2C 07 2E 8C 16 2E - 07 8E D6 BC 00 07 2E 89
16 28 07 2E 89 16 2A 07 - 1E 2E 8E 1E AA 07 89 3E
42 0F A3 4E 0F 89 1E 50 - 0F 0E 1F C7 06 D8 00 00
10 EB 13 AC 84 C0 74 0D - 3C 24 74 09 B4 0E BB 07
00 CD 10 EB EE C3 9C 53 - 33 DB 33 C0 50 9D 9C 58
```

Assemblery

- Stojí nad strojovým kódem – symboliský zápis, vyšší srozumitelnost a snazší programování než strojový kód
- Vázány na konkrétní počítač (rodinu počítačů)
- Tři typy
  - Jazyky relativních adres
    - Numerická nebo symbolická jména instrukcí
    - Adresy jsou počítány relativně vůči bázové adrese
    - Před spuštěním se transformují na adresy reálné
  - Jazyky symbolických adres
    - Symbolické adresy, mohou být absolutní nebo relativní
    - Relokace adres při zavedení programu
    - Nejrozšířenější jazyky nižší úrovně
  - Autokódy
    - Již nebyly omezeny pravidlem jeden příkaz = jedna řádka
    - „Instrukce“ autokódu mohou být přeloženy do více instrukcí počítače
    - Mezistupeň jazyků nižší a vyšší úrovně

Příklad – fragment programu v assembleru

```
ExcessOfMemory label near
    mov bx, di
    add bx, dx
    mov word ptr _heapbase@ + 2, bx
    mov word ptr _brklvl@ + 2, bx
    mov ax, _psp@
    sub bx, ax          ; BX = Number of paragraphs to keep
    mov es, ax          ; ES = Program Segment Prefix address
    mov ah, 04Ah
    push di             ; preserve DI
    int 021h            ; this call clobbers SI,DI,BP !!!!!!
    pop di              ; restore DI
```

Jazyky vyšší úrovně

- Vyšší úroveň abstrakce
- Prvním jazykem vyšší úrovně byl *Plankalkül* K. Zuseho (pouze specifikace)
- Nejsou vázány na konkrétní architekturu
  - Ne vždy realizováno do důsledků
- Interpretované vs. překládané jazyky
- Překlad do strojového kódu nebo mezikódu (např. Java bytecode)

Fortran – FORmula TRANslator (1954)

- První jazyk, nezávislý na konkrétním počítači
  - Určen pro numerické a vědecké aplikace
  - Tvar zápisu odvozen z formátu děrného štítku (odstraněno teprve koncem 80. let)
  - Procedury a funkce (nerekurzivní), mnoho vestavěných funkci
  - Ve starších verzích absence většiny obvyklých řídicích struktur, velmi omezené možnosti práce s texty
  - Velmi efektivní přeložený kód – pro vědeckotechnické výpočty prakticky bez konkurence
- Zaměřen na separátní překlad jednotlivých modulů – snadná tvorba knihoven
  - Existují mimořádně rozsáhlé knihovny programů

Fortran – vývojové etapy

- Fortran (1954)
- Fortran I (1956), II (1957), III (1958)
- Fortran IV (1962), základ pro oficiální normu ANSI Fortran (1966)
- Fortran 77 – pozdě, ale přece zavedl strukturu IF–THEN–ELSE–ENDIF, proměnné typu CHARACTER, příkazy pro práci se soubory aj.
- Fortran 90 – mnoho dalších rozšíření (práce s polem jako celkem, prvky OOP), celková modernizace jazyka při zachování zpětné kompatibility
  - Fortran 95
- Fortran 2003 – podpora objektového programování, abstraktní datové typy; ukazatele na procedury; asynchronní I/O
  - Fortran 2008 (2010)

Fortran 77 – příklad programu

```
      CHARACTER RECORD*9,VERSE*9,PEN6*6,PEN(6)
      EQUIVALENCE (PEN6,PEN)
      VERSE=CHAR(0)//'GR 90.06'
      OPEN(8,FILE=' ',FORM='BINARY')
      READ(8,END=200) RECORD
      IF(RECORD.NE.VERSE) THEN
      PRINT *,'**ERROR: MISSING "PLOTS" OR INVALID FILE'
      GOTO 300
      END IF
      XMIN= 1.0E9
      YMIN= 1.0E9
      XMAX=-1.0E9
      YMAX=-1.0E9
      DO 110 I=1,6
110   PEN(I)=' '
      CALL SIZSUB(XMIN,YMIN,XMAX,YMAX,PEN,'H',NUM)
       PRINT 1000,NUM,XMIN,XMAX,MIN(32.00/(XMAX-XMIN),
     * 28.70/(YMAX-YMIN)),
     * YMIN,YMAX,MIN(22.00/(XMAX-XMIN),17.00/(YMAX-
YMIN))
1000  FORMAT(12X,'**** GRAPH LIMITS ****'/
     * ' Number of graphs in the metafile:',i3/
     * ' XMIN =',F9.3,' CM',10X,'XMAX =',F9.3,' CM',10X,
     * 'LP_FACTOR =',F10.5/' YMIN =',F9.3,' CM',10X,
     * 'YMAX =',F9.3,' CM',10X,'COLFACTOR =',F10.5)
       IF(PEN6.NE.' ') PRINT 1001,PEN
1001  FORMAT(/' 1 2 3 4 5 6 7'/
     *' PENS USED : *',6A3)
       GOTO 400
200   PRINT '(''****ERROR: EMPTY GRAPHIC FILE'')'
300   PRINT '(''****SIZE_GR ABORTED'')'
400   END
```

Cobol – Common Business Oriented Language (1959)

- Určen pro hromadné zpracování dat
- Procedurální část programu vypadá převážně jako psaná v běžné angličtině
- První jazyk, který zavedl datové struktury
- V novějších verzích zabudováno třídění, zjednodušená tvorba sestav, přístup k databázím aj.
  - Obdobně jako Fortran celá vývojová řada (OO COBOL 2002)
- Program sestává ze 4 částí:
  `IDENTIFICATION DIVISION` – program a operační systém
  `ENVIRONMENT DIVISION` – používané soubory; počítač a operační systém
  `DATA DIVISION` – popis dat
  `PROCEDURE DIVISION` – algoritmus zpracování (program)

Cobol – příklad programu

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.     HELLOWORLD.
000300
000400*
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400     DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500     DISPLAY "Hello world!" LINE 15 POSITION 10.
100600     STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800     EXIT.
```

Algol – Algorithmic Language (1958)

- První pokus o plně univerzální jazyk
  - Zavedl některé netradiční znaky (operátor násobení $\times$, celočíselného dělení $\div$, umocňování $\uparrow$, negace $\neg$, dekadický exponent a_{10})
  - Klíčová slova se podtrhují nebo tisknou **polotučně**
- Mnoho revolučních prvků (v dnešních jazycích už považovaných za samozřejmé): podmíněné výrazy a příkazy, příkaz **for**, přepínač, rekurzivní procedury, volání parametrů jménem nebo hodnotou, apod.
- Starší verze neřešily vstup a výstup
- Minimální možnosti práce s texty ve starších verzích
- Přesný formální popis syntaxe (Backusova normální forma)

Vývojové etapy

- Algol 58
  - Pouze specifikace jazyka
- Algol 60
  - Snad nejvýznamnější jazyk všech dob, ovlivnil na desetiletí ostatní jazyky
  - I samotný překladač představoval průlom
- Algol 68
  - Nový pokus o revoluci v programovacích jazycích
    - Zavedl ”krátké”a ”dlouhé”datové typy (int, short int, real, long real, long long real ...),
    - Zavedl ukazatele, tj. odkazy či reference na proměnné (ref int ...), l-hodnoty,
    - Umožnil paralelní větve programu, struktury a uniony, uživatelem deklarované operátory aj.
    - Přesná formální definice syntaxe a částečně i sémantiky.
  - Kritizován jako příliš složitý

Algol 60 – Násobení úsporně uložené symetrické matice vektorem

```
procedure mulsym(A,b,n);
      value n;  integer n;  real array A,b;
 comment mulsym násobí matici invertovanou procedurou symin
         na symetrickou matici A~řádu n vektorem b.
         Výsledek přepíše prvních n prvků pole A;
 begin integer i,k,l,p;
      l:=n;
      for i:=1  step 1  until n  do
         begin A[i]:=0; p:=l+i;
            for k:=i step -1 until 1 do
               begin
                  A[i]:=A[i]+A[p]×b[k];
                  p:=p-n-1+k;
               end;
             if i<n  then
                for k:=i+1  step 1  until n  do
                   A[i]:=A[i]+A[l+k]×b[k];
               l:=l+n-i;
          end
 end multsym;
```

LISP – List Processing (1958)

- Funkcionální jazyk, syntaxe silně ovlivněna Lambda kalkulem
- Určen pro zpracování seznamů
  - Ty považovány za vhodný zápis pro umělou inteligenci (LISP samotný ale nebyl původně pro umělou inteligenci navržen)
- Není rozdíl mezi daty a programem
  - Právě to jej učinilo vhodným pro UI
  - Průkopník mnoha nových programovacích technik
- Program tvořen s-výrazy
  - `(f a b c)` – funkce `f` aplikovaná na argumenty `a`, `b`, `c`
- LISP je mimořádně nepřehledný, je proto extrémně důležité strukturu programu zvýrazňovat vhodnou grafickou úpravou.

LISP – program „guess my number”

```
(defparameter *small* 1)
(defparameter *big* 100)

(defun guess-my-number ()
     (ash (+ *small* *big*) -1))

(defun smaller ()
     (setf *big* (1- (guess-my-number)))
     (guess-my-number))

(defun bigger ()
     (setf *small* (1+ (guess-my-number)))
     (guess-my-number))

(defun start-over ()
   (defparameter *small* 1)
   (defparameter *big* 100)
   (guess-my-number))
```

Basic – Beginners All-purpose Symbolic Instruction Code (1964)

- Vyvinutý na Dartmouth College
- Původní verze velmi primitivní, kritizován za pomalost a osvojování špatných návyků
- Později nejvýznamnější jazyk na nejmenších počítačích (na stolních kalkulátorech a domácích počítačích často jediný dostupný jazyk, integrovaný přímo do hardwaru)
  - názvy proměnných pouze jedno písmeno nebo jedno písmeno a jedna číslice
  - pouze 2 datové typy: číslo (reálné) a řetězec
- Každý příkaz začíná pořadovým číslem, které lze použít jako návěští
- První slovo příkazu charakterizuje, co příkaz dělá
- Později podstatně rozšířené dialekty (např. Visual Basic).

Basic – příklad programu

```
10  REM PROLOZENI BODU PRIMKOU Y=A+BX
20  REM METODOU NEJMENSICH CTVERCU
30  PRINT "KOLIK BODU BUDE ZADANO (3-100)?"
40  INPUT N
45  IF N<3 THEN 30
46  IF N>100 THEN 30
50  REM POCATECNI STAV PROMENNYCH
52  LET P1=0
54  LET P2=0
56  LET P3=0
58  LET P4=0
60  REM CTENI SOURADNIC A~SUMACE
70  FOR I=1 TO N
80    PRINT I;". DVOJICE X,Y: ";
90    INPUT X,Y
```

```
100   LET P1=P1+X
110   LET P2=P2+Y
120   LET P3=P3+X*Y
130   LET P4=P4+X^2
140 NEXT
I~150 REM VYPOCET REGRESNICH KOEFICIENTU A, B
160 LET B=(P3-P1*P2/N)/(P4-P1^2/N)
170 LET A=(P2-P1*B)/N
180 IF(B<0 THEN 186
182   LET Z\$ ="-"
184   GOTO 190
186 LET Z\$ ="+"
190 PRINT
200 PRINT "ROVNICE PRIMKY JE Y="A;Z\$ ;ABS(B);"X"
210 PRINT
```

PL/I – Programming Language One (1964)

- Univerzální programovací jazyk vyvinutý v IBM
  - Snaha spojit Fortran a COBOL
  - Původně NPL (také National Physical Paboratory), přejmenován na PL/I
- PL/I úzce propojena s OS/360
  - Dialekty pro různé počítače (kvůli propojení s příslušnými operačními systémy)
- Operační systém Multics byl napsán v PL/I
- Až příliš univerzální, což vedlo k nepřehledné syntaxi i sémantice:
  - E. Dijkstra: Používat PL/I musí být jako pilotovat letadlo pomocí 7000 tlačítek, přepínačů a pák v kokpitu. Absolutně nechápu, jak bychom mohli intelektuálně zvládnout naše rozrůstající se programy, když už sám programovací jazyk – náš hlavní nástroj! – se pro svou baroknost vymkl z naší intelektuální kontroly.

PL/I – Deklarované cíle

- Co je intuitivně jasné a jednoznačné, není zakázáno
- Pokud je to možné, neuchylovat se ke strojovému kódu
- Strojově nezávislý (přesto je na něm patrno, že vznikl pro IBM/360)
- Modularita
- Péče o začátečníky
  - Stačí znalost malé podmnožiny jazyka pro úspěšné psaní programů

PL/I – příklad podprogramu

```
DECLARE ABS GENERIC
    (ABS1 ENTRY (FIXED),
     ABS2 ENTRY (FLOAT REAL),
     ABS3 ENTRY (COMPLEX));
ABS1: ABS2: PROCEDURE (X); DECLARE Y;
    Y=X;
    IF Y LT 0 THEN Y=-Y; RETURN(Y);
ABS3: ENTRY (X);
    RETURN (SQRT(REAL(X)**2 + IMAG(X)**2));
END ABS;
```

APL – A Programming Language (1964)

- Univerzální programovací jazyk
- Velmi „hutný“ zápis (až 10krát stručnější než v běžných jazycích)
- První verze již v roce 1962
- Implementace na IBM 360/370, CDC, UNIVAC, ruském BESM aj.
- Používá se dodnes, překladače a speciální fonty jsou na Webu

APL

- Speciální sortiment znaků (nutnost používat zvláštní terminál či font nebo nahrazovat speciální znaky náhradními kombinacemi znaků ASCII)
- Program na první pohled (nezasvěcenému) zcela nesrozumitelný
- Velmi mnoho operátorů, většina jako unární i binární (obvykle s různými významy: ÷x znamená 1/x, x÷y znamená x/y; *x znamená e^x, x*y znamená x^y)
- Operátory mají stejnou prioritu a provádí se **odzadu**, pořadí však lze změnit závorkami
- Bohatá podpora práce s vektory a maticemi (např. A⌹B znamená vynásob matici A inverzí matice B)

APL – příklad

```
[6]    L←(L⍳':')↓L←,L          ⍝ drop To:
[7]    L←LJUST VTOM',',L       ⍝ mat with one entry per row
[8]    S←¯1++/∧\L≠'('          ⍝ length of address
[9]    X←0⌈⌈/S
[10]   L←SΦ(-(⍴L)+0,X)↑L       ⍝ align the (names)
[11]   A←((1↑⍴L),X)↑L          ⍝ address
[12]   N←0 1↓DLTB(0,X)↓L       ⍝ names)
[13]   N←,'α',N
[14]   N[(N='_')/⍳⍴N]←' '      ⍝ change _ to blank
[15]   N←0 ¯1↓RJUST VTOM N     ⍝ names
[16]   S←+/∧\' '≠ΦN            ⍝ length of last word in name
```

Pascal (1970)

- Snadný k naučení, zvlášť vhodný pro výuku, méně pro praktické programování
- Autorem je Niklaus Wirth, jazyk je pojmenován na počest B. Pascala
- Silná typová kontrola, většina chyb se zachytí už při kompilaci
- Program se musí kompilovat vcelku – žádná modularita
- Vyjadřovací možnosti tak malé, že většina implementací jazyk rozšiřuje (neportabilně, např. unit v překladačích fy Borland)
- Některé nedostatky byly odstraněny v revidované verzi jazyka (např. zavedení konformantních polí).
- Novější překladače jazyk podstatně rozšířily – za cenu vážného omezení přenositelnosti programů.

Smalltalk (1970)

- Plně objektově orientovaný jazyk
  - Silně typovaný, dynamický a interpretovaný
- Vznikl v Xerox PARC
- Nástroj pro programování v grafickém prostředí
- Specifická syntaxe (inspirace obrácenou polskou notací BPN)
- Několik variant označených podle roku vzniku (72, 74, 76, 78, 80)
- Výrazně ovlivnil nové (C++, Java) i aktualizované verze starších (Cobol, LISP) programovacích jazyků

Prolog – Programation en Logique (1970)

- Deklarativní logický jazyk, logika prvního řádu
- Autory jsou Alain Colmeraurer a Phillipe Roussel z univerzity Aix-Marseille
- Sloužil pro usnadnění komunikace s počítačem v přirozeném jazyce
  - Využíván zejména v oblasti umělé inteligence a v počítačové lingvistice
  - Jednoduchá syntax, založen na predikátové logice prvního řádu
  - Hornovy klauzule, unifikace, rekurze a backtracking
- Základ logického programování
  - Japonská 5. generace počítačů
- Řada verzí, včetně podpory OOP

Prolog – Problém Hanoiských věží

```
% move(N,X,Y,Z) - move N disks from peg X to peg Y, with peg Z being the
%                  auxilliary peg
%         Transfer the first n-1 discs to some other peg X
%         Move the last disc on X to Y
%         Transfer the n-1 discs from X to peg Y

     move(1,X,Y,_) :-
         write('Move top disk from '),
         write(X),
         write(' to '),
         write(Y),
         nl.
     move(N,X,Y,Z) :-
         N>1,
         M is N-1,
         move(M,X,Z,Y),
         move(1,X,Y,_),
         move(M,Z,Y,X).
```

Zdroj: http://www.cs.toronto.edu/~hojjat/384w09/simple-prolog-examples.html

Jazyk C (1972)

- Vyvinut v Bell Laboratories – D. Ritchie a K. Thompson
- Spojuje prvky jazyků vysoké a nízké úrovně
- Univerzální, zvlášť vhodný pro systémové programování
- Těsná návaznost na operační systém, zpočátku silně svázán s Unixem
- Snadno se v něm udělá chyba, která neporušuje syntax a není proto zachycena překladačem
- Velmi efektivní generovaný kód, bohatá nabídka operátorů
  - Fortran vs. C
- Vývojové etapy
  - Kernighan-Ritchie (K-R) C
  - ANSI C (postupně několik verzí)
  - ISO/IEC C (C99)

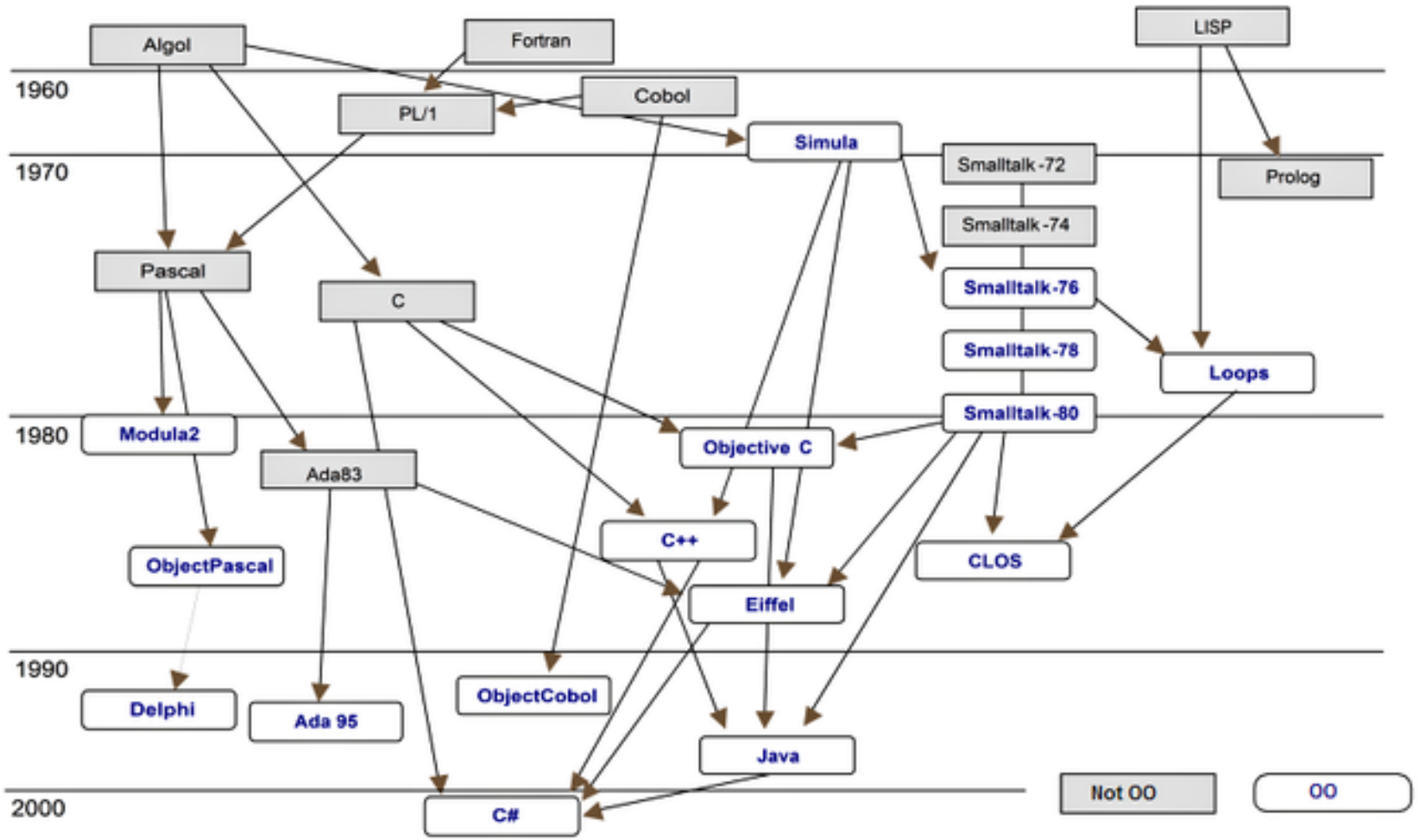
Ada (1980)

- Vyvinut na zakázku ministerstav obrany (DoD) USA
  - Pojmenován na počet Ady Augusty Byronové hraběnky Lovelace
  - Původně tuto ambici měl Algol 68
  - Prosazen jako jediný jazyk pro vývoj nových systémů (bezpečné programování)
- Univerzální jazyk pro všechny typy aplikací včetně řízení v reálném čase
  - Strukturovaný, staticky typovaný a objektově orientovaný jazyk
  - Zavedl nebo převzal některé neobvyklé moderní prvky (zpracování výjimek, generické funkce)
  - Podpora „bezpečného“ paralelismu a explicitní podpora souběžnosti
- Velmi dokonalý, ale málo rozšířený

Ada – příklad programu

```
restricted(MATH_LIB, TEXT_IO)
procedure QUADRATIC_EQUATION is
    use TEXT_IO;
   A, B, C, D: FLOAT;
begin
   GET(A); GET(B); GET(C);
   D:=B**2-4.0*A*C;
    if D<0.0 then
       PUT("COMPLEX ROOTS");
    else
        declare
            use MATH_LIB; -- tam je funkce SQRT
            begin
               PUT("REAL ROOTS: ");
               PUT(B-SQRT(D))/(2.0*A));
               PUT(B+SQRT(D))/(2.0*A));
               PUT(NEWLINE);
            end;
    end if;
end QUADRATIC_EQUATION;
```

Historie programovacích jazyků



Zdroj: `http://iwi.uni-hannover.de`

Další odkazy na historii programovacích jazyků

- http://oreilly.com/news/graphics/prog_lang_poster.pdf
- http://www.scriptol.com/programming/history.php
- http://www.levenez.com/lang/