Co to jest: paradygmat programowania

- Paradygmat - zbiór założeń i ustaleń przyjętych jako podstawa pewnej teorii.
- Παράδειγμα – wzorzec, przykład
- Paradygmat programowania – zestaw założeń, konstrukcji językowych, mechanizmów przyjętych dla danej klasy języków programowania

3

Podstawowe paradygmaty

- Imperatywny (Algol, Fortran, Pascal, C)
- Obiektowy (Smalltalk, Java, C++, C#,, Ada)
- Funkcyjny (Scheme, Haskell, ML, Scala)
- Logiczny (Prolog)
- Programowanie współbieżne
- Programowanie równoległe
- Programowanie strukturalne
- Programowanie sterowane zdarzeniam

4

Obszary zastosowań języków programowania

- Obliczenia naukowe i techniczne (Fortran, Algol 60; proste struktury danych - tablice, działania na liczbach rzeczywistych, podprogramy)
- Aplikacje biznesowe (Cobol; skomplikowane zestawienia i raporty, później jednak C++, Java i techn. internetowe)
- Sztuczna inteligencja, rozumowania logiczne (LISP, Prolog; działania na symbolach, wnioskowanie)
- Programowanie dla www (php, java, javascript; różnorodność struktur danych, niezależność od platformy sprzętowo-programowej, bezpieczeństwo)
- Programowanie systemowe (C, PL/I; dostęp do cech niskiego poziomu)

5

(oczekiwane) Cechy języków programowania

- Czytelność (readability) - ułatwia pracę zespołową, znajdowanie błędów, pielęgnację kodu
- Łatwość kodowania (writability) – relatywnie niewielki zbiór symboli i konstrukcji (łatwy do opanowania) , jednoznaczny sposób kodowania i budowania konstrukcji języka
- Niezawodność (reliability) – odporność na błędy, minimalizacja skutków ubocznych, mechanizmy obsługi błędów i wyjątków

6

Czytelność

- Ogólna prostota
  - Relatywnie niewielki zbiór cech i konstrukcji
  - Jednoznaczność konstrukcji – NIE: Wiele sposobów opisu tego samego: count=count+1, count+=1, count++, ++count
  - Jednoznaczność operacji

```
int a = 5, b=7;
suma = a+b;
```

```
string imie, nazwisko, famName;
Int wiek = 17;
famName = imie+' '+ nazwisko+' '+wiek;
```

7

Czytelność cd

- Ortogonalność
  - Stosunkowo niewielka liczba konstrukcji podstawowych może być organizowana na niewiele sposobów,
  - według zawsze tych samych reguł

TAK: w C, Javie elementem tablicy może być inna tablica albo struktura (rekord), elementem rekordu może być również tablica
NIE: funkcja w C nie może zwrócić tablicy

Co lepsze?

Nawiasy i wcięcia dla bloków

```
x=szukane; dol=0; gora=x+1;
while(gora-dolj > 1) {
  srodek =(dol+gora)/2;
 if (srodek*srodek <= x) {
  dol = srodek; else { gora = srodek;
printf(" sqrt(%i) =%i\n", x,dol)
}
.. }
```

słowa kluczowe – przegadane?

```
x=2;
For  x =1 to N  do begin
 If  N div x != 0 then
   writeln   …
 end if
end
end
```

8

Czytelność cd

- Typy danych
  - Odpowiedni zbiór predefiniowanych typów danych, a może bez .wskazywania typów?
- Cechy składniowe
  - Zasady budowania identyfikatorów
  - Słowa kluczowe i zasady budowy wyrażeń
  - Samoobjaśniające/samodokumentujące się instrukcje i słowa kluczowe

9

Łatwość kodowania (writability)

- Ortogonalność - dużo specyficznych konstrukcji – trudniej je opanować "w piśmie", mało konstrukcji + jednoznaczność reguł – łatwiej pisać program
- Wsparcie dla abstrakcji
  - Abstrakcja procesu - możliwość definiowania złożonych operacji i używania ich bez pamiętania szczegółów definicji np. podprogramy
  - złożone (abstrakcyjne) struktury danych np. drzewo binarne (odwołujemy się do węzłów drzewa vs. węzły reprezentowane w tablicy) , ogólnie obiekt
- Siła wyrazu
  - Odpowiedni zbiór konstrukcji
  - Odpowiednia liczba predefiniowanych operatorów i funkcji

10

Niezawodność języków programowania

Program jest niezawodny wówczas, gdy wykonuje swoje zadania w każdej sytuacji. Sprzyja temu:

- Kontrola typów: może być wykonana na etapie kompilacji lub na etapie wykonania – co jest bardziej kosztowne, może być bardziej lub mniej restrykcyjna, mogą występować (jawne, bądź niejawne) konwersje typów
- Obsługa wyjątków: Java, C# mają specjalne konstrukcje, „starsze" języki ich nie miały

11

Program imperatywny

```
program zagadka;
  var   s, i, N integer;
begin
  read(N);
  s=1;
  for i = 1 to N
     s = s*i;
  write(s);
end;
```

•Wykonanie każdej instrukcji zmienia stan programu (pamięci, rejestrów, znaczników procesora)
•Takie widzenie programu jest ściśle powiązane z budową komputera w architekturze von Neumanna, w którym poszczególne instrukcje kodu maszynowego zmieniają stan komputera.
•Zmienne są abstrakcjami komórek pamięci, a instrukcja podstawienia polega na zmianie wartości komórki pamięci symbolizowanej przez zmienną.
•Zmienne mogą być różnych typów (jawnie bądź niejawnie deklarowanych)
•Instrukcje mogą być zagnieżdżane (strukturalność)

•modularyzacja

12

Program obiektowy

```
public class Para {
      int a, b;
      public Para(int x, int y) {
         this.a = x;
         this.b = y;
      }
      public int dodaj() {
         return a+b;
      }
      public int mnoz() {
         return a*b;
      }
      public static void main(String[] args) {
         Para p = new Para (2,5);

   System.out.println("mnozenie:"+p.mnoz()
   +" dodawanie: "+p.dodaj());}

   }
```

•Abstrakcja
•Program działa na obiektach, które są elementami pewnej klasy i które mogą się ze sobą komunikować.
•Dla obiektów danej klasy definiujemy ich cechy (zmienne) i zachowania (procedury).
•Obiekty są powoływane do życia za pomocą odpowiednich poleceń.
•Obiekty są hermetyzowane czyli „pokazują światu" tylko potrzebne cechy i zachowania,
•Dziedziczenie
•Wymiana komunikatów

13

Program funkcyjny

```
(DEFINE (fun n)
  (IF (<= n 0)
  1
  (* n (fun (- n 1) ) ) ) )
```

Zapis wyrażeń:
(operator arg1 arg2)
(+ a b) zamiast a+b

- Program opisuje, CO należy obliczyć. Często stosując rekurencję.
- Nie ma stanu maszyny
- Nie ma instrukcji
- Nie ma zmiennych i brak efektów ubocznych
- Nie ma pętli, często stosowana rekurencja
- Konstruowanie programu polega na składaniu (w matematycznym sensie) funkcji z istotnym udziałem rekurencji

14

Program w paradygmacie logicznym

-- fakty
rodzic(kasia, robert).
rodzic(tomek, robert).
rodzic(stan, tomek).
rodzic(basia, tomek).
kobieta(kasia).
kobieta(basia).
mezczyzna(tomek).
mezczyzna(stan).

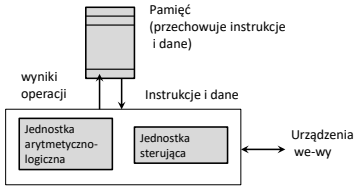
-- reguły
matka(X,Y) :- rodzic(X,Y), kobieta(X).
siostra(X,Y) :- kobieta(X),
matka(Z,X), matka(Z,Y).
przodek(X,Y) :- rodzic(X,Z), rodzic(Z, Y).

-- stwierdzenie „do udowodnienia"
? –matka(kasia, X)

- Wykonanie programu to próba udowodnienia celu w oparciu o podane przesłanki np. kobieta(basia) true – czy basia jest kobietą matka(basia, X) – czyją matką jest basia
- Nie podajemy instrukcji
- Opisujemy fakty i reguły
- Pytamy czy nasze stwierdzenia są prawdziwe

15

Architektura von Neumanna

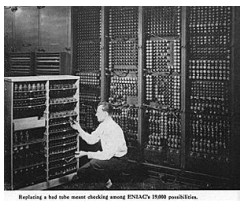


Powtarzaj:
Pobierz instrukcję wskazaną przez licznik programu
Ustaw licznik na kolejną instrukcję
Dekoduj instrukcję
Wykonaj instrukcję
end

Dlatego centralnym pojęciem języków imperatywnych jest zmienna jako abstrakcja komórki pamięci

16

Historia rozwoju języków programowania

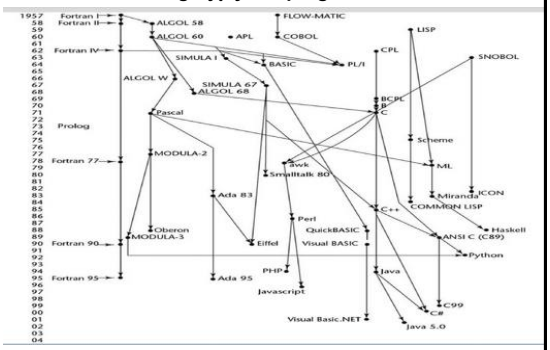


Kiedy powstał pierwszy komputer?
- Na początku … nie było żadnego języka , albo był język rozkazów maszynowych
- Potem wymyślono assembler

- Języki wysokiego poziomu Fortran (1957) ost (F 1995)
- Algol (1958)
- Lata 1960 – 70 programowanie strukturalne (eliminacja skoków go to)
- Początki obiektowości – Simula 67, Smalltalk, dojrzałość C++ (lata 198*)

17

Genealogia języków programowania



18

Implementacja języków

Programy w dowolnym języku, w dowolnym paradygmacie są finalnie wykonywane na architekturze von Neumanna.

Kompilacja : analiza leksykalna → analiza składniowa → kod wstępny → optymalizacja kodu → kod maszynowy – szybkie wykonanie ale …

Interpretacja: kod realizowany instrukcja po instrukcji – wolne wykonanie ale ..

Implementacje hybrydowe (np. java, .net): kompilacja do kodu pośredniego → interpretacja instrukcji kodu pośredniego na maszynie wirtualnej

Programy mają strukturę modularną, kod programisty korzysta z różnego rodzaju bibliotek (C, Fortran), gotowych klas (Java). Po kompilacji następuje więc często **konsolidacja** (linkage). Ostatecznie pewne elementy finalnego programu mogą być określone już na etapie kompilacji, inne dopiero na etapie wykonania.

Znane Wam przykłady języków: kompilowanych, interpretowanych, realizowanych hybrydowo ???

19

Jak opisujemy języki programowania

- **Składnia**:
  - jakie symbole są dopuszczalne
  - jak zapisuje się wyrażenia
  - jak zapisuje się instrukcje sterujące (jakie są w języku) przykłady??
  - Jak (czy?) zapisuje się deklaracje

- Składnię dość łatwo jest opisać formalnie używając np. notacji BNF (Backus-Naur Form)

Np.. typ ::= char | int | float | double // | znaczy LUB

instr_warunkowa ::= if wyr_logiczne then instr [else instr] // [elt opcjonalny

lista_arg ::= arg { "," arg } // { element może być wielokrotnie powtórzony

- **Semantyka**: znaczenie przypisywane symbolom i zapisom operacji – tu trudno o formalne metody opisu

20