Programski jeziki

Od problema do programa

Od problema do programa

- Problem
- Algoritem (postopek reševanja problema)
- Zapis v programskem jeziku – uporaba ukazov, ki jih znamo izvesti
- Prevajanje v obliko, ki jo razume procesor (izvajalec ukazov)
- Izvajanje

- Ali rešitev ustreza problemu?

Od problema do programa

- opredelitev problema
  - določitev tega, kaj vemo - katere podatke poznamo,
  - in tega, kaj želimo dobiti - rezultat;
- načrtovanje postopka rešitve - algoritem;
- zapis postopka rešitve v programskem jeziku;
- Prevajanje v obliko, ki jo razume procesor
- izvršitev programa na računalniku;
- preverjanje programa (preverjanje pravilnosti rešitve)

Zakaj programski jeziki?

- Oseba A govori LE jezik *blable*, oseba B pa LE jezik *blublu*. Kako naj se A sporazume z B?
- Lahko se A nauči jezika *blublu* in se potem pogovarjata v *blublujščini*.
- Procesor "govori" le strojni jezik: 0000110101 in ne kaže nobene želje, da bi se naučil "naš" jezik
- Torej se moramo mi naučiti strojni jezik
- Programiranje v strojnem jeziku: zapleteno, polno napak, specifično za vsak procesor
- Zapleten jezik: pogovor omogoči posebej izučen stokovnjak – prevajalec
- Ta jezik *blable* prevede v jezik *blublu*

Zakaj programski jeziki?

- Strokovnjakov, ki bi znali strojni jezik, je (razmeroma) malo, želja po "ukazovanju" procesorju (pogovarjanju z njim) (zelo) veliko
- Avtomatsko prevajanje – prevajanje iz "našega" jezika v strojni jezik naj opravi program
- Kaj je naš jezik?
  - Posebni programski jezik
  - Govorjeni jeziki

Zakaj programski jeziki?

- Zakaj pa bi potrebovali poseben programski jezik?
- Navodila napišimo v pogovornem jeziku – program pa naj jih prevede v strojni jezik
- Sestaviti tak prevajalnik: Izjemno kompleksna (trenutno praktično nemogoča) naloga – naravni jeziki so "preveč ohlapni", da bi bilo avtomatsko prevajanje mogoče.

- Ste že videli dva slavista, ki bi se strinjala glede tega, kako kaj napisati?

Zakaj programski jeziki?

- Vmesni člen: programski jeziki
- Nam se je enostavneje izražati v njih (pisati programe – zapise algoritmov)
- Dovolj "natančni" (stroga sintaksa), zato mogoče sestaviti avtomatske prevajalnike

Programski jeziki

- Služijo za zapis algoritma
- Najrazličnejši
- Morajo omogočati avtomatsko pretvorbo v strojni jezik (jezik, ki ga razume procesor)

Programski jeziki

- Ogromno jezikov:
  - pascal, basic, cobol, Smalltalk, C#, ada, ...
  - Različne zvrsti
    - Generacije jezikov
    - Objektni (predmetni) jeziki, funkcijski jeziki, ...
- Programski jezik Java
- Prevajalniki
  - Avtomatska pretvorba iz zapisa v jezik procesorja
  - Programi
    - Podatki: izvorna koda
    - Rezultati: prevedena koda
  - Kot za pripravo besedil obstajajo različni urejevalniki, tudi tu obstajajo različni prevajalniki
    - JAVAC, JBUILDER, VisualCafe, IBM Visual Age for Java, ...
  - Vsi zapis v programskem jeziku Java prevedejo v obliko, ki jo razume procesor (ni čisto res, a ...)

Primer

- Preberi stranico kvadrata in ga nariši
- Algoritem:
  - Preberi število in si ga zapomni kot celo število v spremenljivko a
  - Nariši pravokotnik s stranicama a in a
- Včasih bo potrebno korake algoritma še razgraditi, če posameznega koraka ni moč neposredno zapisati v programskem jeziku
  - JAVA: 1.korak:
    - podatek preberemo kot niz
    - Pretvorimo niz v število
  - pascal: 2.korak
    - Sam jezik ne pozna ukazov za risanje!

Program v jeziku Java – različica 1

```
import java.awt.*;
import java.applet.*;
import javax.swing.JOptionPane;

public class KvadratBeri extends Applet
/*  Narisali bomo kvadrat s stranico a */
{  public void paint (Graphics g)
   {  int a; // Stranica kvadrata
      String str; // stranica prebrana kot niz

      str = JOptionPane.showInputDialog("Vnesi stranico kvadrata");
              // V str preberemo stranico kvadrata
      a = Integer.parseInt(str); // iz str naredimo celo število
                                 // in ga shranimo v a
      // Narisemo kvadrat
      g.drawRect(10, 10, a, a);
   }
}
```


Program v jeziku Java - različica 2

```
import java.awt.*;
import javax.swing.*;

class Kvadrat_Okno extends JFrame  /*  Narisali bomo kvadrat s stranico a */
{  public void paint (Graphics g)
   {  int a; // Stranica kvadrata
      String str; // stranica prebrana kot niz

      str = JOptionPane.showInputDialog("Vnesi stranico kvadrata");
              // V str preberemo stranico kvadrata
      a = Integer.parseInt(str); // iz str naredimo celo število
                                 // in ga shranimo v a
      // Narisemo kvadrat
      g.translate(getInsets().left, getInsets().top); // da se "znebimo" zg. vrstic
       g.setColor(Color.black); // risemo s crno barvo
      g.drawRect(10, 10, a, a);
   }
}

public class KvadratOkno {
   public static void main(String[] s){
       Kvadrat_Okno o = new Kvadrat_Okno();
       o.resize(200, 200); // velikost okna
       o.show(); // prikaz
   }
}
```

Kako?

- Priprava izvorne datoteke (source code)
  - TextPad, NotePad – `KvadratBeri.java`
- Prevajanje (compile)
  - V DOS oknu se postavimo v imenik, kjer je datoteka `KvadratBeri.java`
  - `JAVAC KvadratBeri.java`
- Izvedemo program
  - `Priprava ustrezne datoteke s HTML,kjer kličemo ta program(ček)`
  - `APPLETVIEWER kvadrat.htm`

Narišimo trikotnik

- podatki: velikost trikotnika (število vrstic)
- rezultat: narisan polni trikotnik, sestavljen iz zvezdic (za n=3)

```
*
***
*****
```

Postopek

- izpišemo prvo vrstico,
- izpišemo drugo vrstico, …
- izpisujemo i-to vrstico
  - izpišemo ustrezno presledkov
    - v 1. vrstici n - 1, v 2. vrstici n - 2, …
    - v i-ti vrstici n - i
  - izpišemo ustrezno število *
    - v 1. vrstici 1, v 2. vrstici 3, v 3. vrstici 5, …
    - v i-ti vrstici 2 * i - 1

Program v jeziku C

```
/* trikot. c */
#include <stdio.h>

int main(void) {

  int i, j, velikost;

  printf("\n\n Velikost trikotnika: "); scanf("%d",&velikost);
  printf("\n\n\n");

  for (i = 1; i <= velikost; i = i + 1) {   /* izpis i - te vrstice */
    for (j = 1; j <= velikost - i; j = j + 1) printf(" "); /* presledki */
    for (j = 1; j <= 2 * i - 1; j = j + 1) printf("*"); /* zvezdice */
    printf("\n"); /* v novo vrsto */
  }

  return 0;
}
```

Program v pascalu

```
program trikot;

var
  i, j, velikost: integer;
begin
  writeln; writeln;
  write("Velikost trikotnika: "); read(velikost);
  writeln; writeln; writeln;


  for i := 1 to velikost do begin  {  izpis i - te vrstice }
    for j := 1 to velikost - i do write(" "); { presledki }
    for j := 1 to 2 * i - 1 do write("*"); { zvezdice }
    writeln { v novo vrsto }
  end
end.
```


Program v jeziku Java

```
import javax.swing.*;

public class Trikot {
   public static void main(String[] args) {
     int i, j, velikost;
     String vel_s;

      // Preberimo velikost

      vel_s = JOptionPane.showInputDialog("Velikost trikotnika:");
      velikost = Integer.parseInt(vel_s);

      System.out.print("\n\n\n");

      for (i = 1; i <= velikost; i = i + 1) {   // izpis i-te vrstice
        for (j = 1; j <= velikost - i; j = j + 1)
           System.out.print(" "); /* presledki */
        for (j = 1; j <= 2 * i - 1; j = j + 1)
           System.out.print("*"); /* zvezdice */
        System.out.println(""); /* v novo vrsto */
      }
    }
}
```

Sintaksa

- Pravila, kako mora biti sestavljen program
- stroga pravila omogočajo avtomatično prevajanje
- prevajalnik odkrije sintaktične napake
- Napake v sintaksi:
  - javi prevajalnik
  - zgled

Semantika

- Sintaktično pravilen, a drugače napačen program

```
public class Narobe_semantika
{  // Sintaktično pravilen, a semantično napačen program
   public static void main(String[] g){
    System.out.println("Vsota stevil 2 + 3 = " + 2 * 3);
   }
}
```

```
public class Narobe_semantika2
{  // Sintaktično pravilen, a semantično napačen program
   public static void main(String[] g) {
     System.out.println("Vsota stevil 2 + 3 = " + 2 + 3);
   }
}
```

Semantika

- Napake v semantiki:
  - razumevanje problema
  - Tehnike priprave programov
    - Strukturirano programiranje
      - Problem razgrajujemo na zaključene podprobleme, ki jih razgrajujemo naprej
    - Ekstremno programiranje
      - Najprej pripravimo testne primere in pričakovane odgovore
    - ...
  - Preverjanje, preverjanje, preverjanje
    - Ne moremo preveriti, ali program dela prav, lahko pa ugotovimo, da ne dela prav