Programiranje – ZAKAJ in KAJ?

Več v

- http://uranic.tsckr.si/VI%C5%A0JA%20%C5%A0OLA/Programiranje%20I/Programiranje%20%201%2
- http://uranic.tsckr.si/VI%C5%A0JA%20%C5%A0OLA/Programiranje%20I/C%23_UVOD.pdf
- http://uranic.tsckr.si/VI%C5%A0JA%20%C5%A0OLA/Programiranje%20I/Programiranje%20I%20-%20VI%C5%A0JA%20%C20INFORMATIKA.doc

Zmotne predpostavke

- Danes je učenje programiranja povsem odveč in potrata časa.
- Za vse, kar želimo narediti z računalnikom, so na voljo ustrezna orodja.
- Programiranje je potrebno le kot zelo specialistično znanje skupinice strokovnjakov, ki pišejo programe, ki jih potem običajni uporabniki uporabljamo.

- **Znanje programiranja je odveč**

Izbira programskega jezika

- Za učenje osnov
  - Precej nepomembna
  - Stvar osebnega okusa, okolja, dostopnosti, mode, ...
- Za "pravo" programiranje
  - Razvojno okolje
  - Razvojna orodja
  - Vrsta problema
- Skupni gradniki, enostaven prehod iz enega v drugi jezik

Osnovni gradniki

- Konstante
  - števila, znaki, nizi, logične vrednosti
- Spremenljivke
  - int, float, double, char, string, bool, …
- Prireditev, izrazi
- Branje in izpisovanje
- Vejitev – pogojni stavek
- *Zanke*

- *Funkcije, metode, podprogrami, ...*
- *Objekti in objektno programiranje*
- *Sestavljene podatkovne strukture*

Od problema do programa

- Problem
- Algoritem (postopek reševanja problema)
  - Psevdokoda
  - Diagram poteka
- Zapis v enem od programskih jezikov – uporaba ukazov, ki jih znamo izvesti
- Prevajanje v obliko, ki jo razume procesor (izvajalec ukazov)
- Izvajanje

- Ali rešitev ustreza problemu?

Programski jeziki

- Programskih jezikov je veliko
  - Pascal, basic, cobol, C++, C#, …
  - Različne zvrsti
    - Generacije jezikov
    - Objektni (predmetni) jeziku, funkcijski jeziki
- Programski jezik C#
- Prevajalniki
  - Avtomatska pretvorba iz zapisa v jezik procesorja
  - Programi
    - Podatki – izvorna koda
    - Rezultati: prevedena koda
  - Tako kot za pisanje besedil obstajajo različni urejevalniki, tudi tu obstajajo različni prevajalniki: ti zapis v programskem jeziku prevedejo v obliko, ki jo razume procesor

Algoritem: kaj in zakaj

- Algoritem je postopek, ki nam korak za korakom pove, kako rešiti dani problem in nas pripelje do rešitve
- Za dani problem v splošnem obstaja več algoritmov, ki določijo postopek, s katerim rešimo problem
- Obstaja npr. veliko algoritmov za izračun produkta dveh števil:
  - Tabela poštevanke (primerno le za majhna števila)
  - Pisno množenje
  - Množenje z uporabo logaritmov
  - Uporaba računala
  - Uporaba postopkov vgrajenih v računalnik
  - ...

Algoritem

- Algoritem je navodilo, kako opraviti določen postopek
- Z njim povemo, KAJ moramo storiti in KAKO to storiti
- Algoritem mora biti končen (število korakov končno), vsak korak pa mora biti neka znana operacija
- Značilnosti algoritma
  - Ima podatke
  - Vrne rezultat (število, risba na zaslonu, izdelek, …)
  - Je natančno določen
  - Se vedno konča
  - Mogoče ga je izvesti

Algoritem - zapis

- Zapis algoritma
  - tekstovna oblika (PSEUDOKODA)

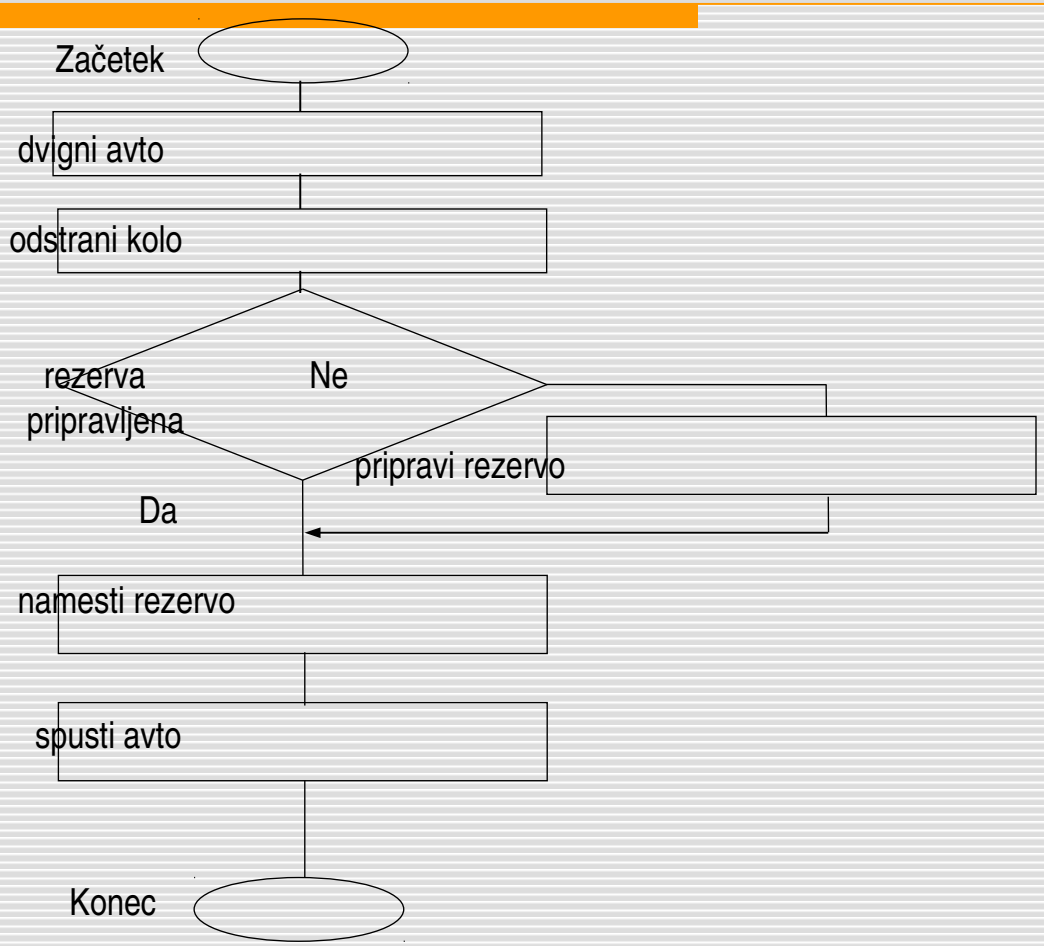
Napišimo algoritem za menjavo kolesa pri avtomobilu :

```
Začetek
    dvigni avto
    odstrani kolo
    če rezervno kolo ni pripravljeno, ga pripravi
    namesti rezervno kolo
    spusti avto
Konec
```

Algoritem - zapis

- Zapis algoritma
  - grafična oblika - *DIAGRAM POTEKA*, STRU*KTOGRAMI*
    - začetni oz. končni blok : stoji na začetku in koncu vsakega diagrama poteka (grafični znak - elipsa)
    - b) vhodno izhodni blok : uporabimo ga takrat, ko v algoritem vnašamo podatke oz. kadar algoritem izpisuje rezultate. (grafični znak - romboid)
    - c) Prireditveni blok : v tem bloku se izvajajo različne operacije . Prireditev zapišemo podobno kot matematične formule, le da namesto enačaja napišemo puščico (grafični znak - pravokotnk).
    - č) odločitveni blok : v njem se nahaja nek pogoj, na osnovi katerega se diagram poteka razveji (grafični znak - za 90 zasukan romb)
    - V diagramu poteka so bloki povezani s črtami, ki nam obenem kažejo tudi smer izvajanja diagrama poteka .

Diagram poteka - primer



Vprašanja

- Kako zasnovati algoritem (metode, strategije)
- Kako preveriti algoritem (dokaz pravilnosti)
- Kako analizirati algoritem: prostorska in časovna zahtevnost
- Kako izraziti algoritem (komu je namenjen, enoličnost, kaj so osnovna navodila, komentarji)

Učenje programskih jezikov

- Učenje izražanja določenega algoritma
- Zasnova algoritma: enostavna, enostavni problemi, ...
- Učenje zapisovanja algoritma v izbran programski jezik

Programski jezik

- Sintaksa
  - Kako je sestavljen jezik, “slovnična” pravila
  - *stavek, imeti napak polno*.
  - Naravni jeziki: dokaj ohlapna sintaksa, z leti spreminjajoča se
- Semantika
  - Kaj sintaktično pravilen stavek pomeni
  - Sintaktično pravilni stavki lahko povejo nesmisel.
    - Vsota števil 2 in 3 je 7.
    - Danes je padlo pol metra snega.
    - Konj ima zeleno nabrušen rep.

Sintaksa jezika

- Pravila, kako mora biti sestavljen program
- Stroga pravila omogočajo avtomatično prevajanje
- Sintaktične napake odkrije in nam jih sporoči prevajalnik
- Napake v sintaksi:
  - javi prevajalnik
  - zgled

Semantika

- Sintaktično pravilen, a drugače (pomensko) napačen program

```
// Sintaktično pravilen, a semantično napačen program
public static void Main()
{
   Console.WriteLine("Vsota stevil 2 + 3 = " + 2 * 3);
}
```

```
// Sintaktično pravilen, a semantično napačen program
 public static void Main(string[] g)
 {
    Console.WriteLine("Vsota stevil 2 + 3 = " + 23);
 }
```

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