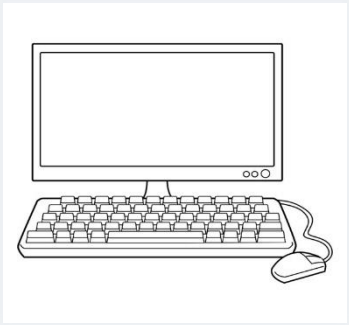
OSNOVE PROGRAMIRANJA

RAČUNALNIK

- stroj, ki nam omogoča pisanje besedila, iskanje informacij po internetu, igranje iger, predvajanje filmov, klepetanje s prijatelji
- olajša opravila, ki smo jih prej opravljali brez njegove pomoči

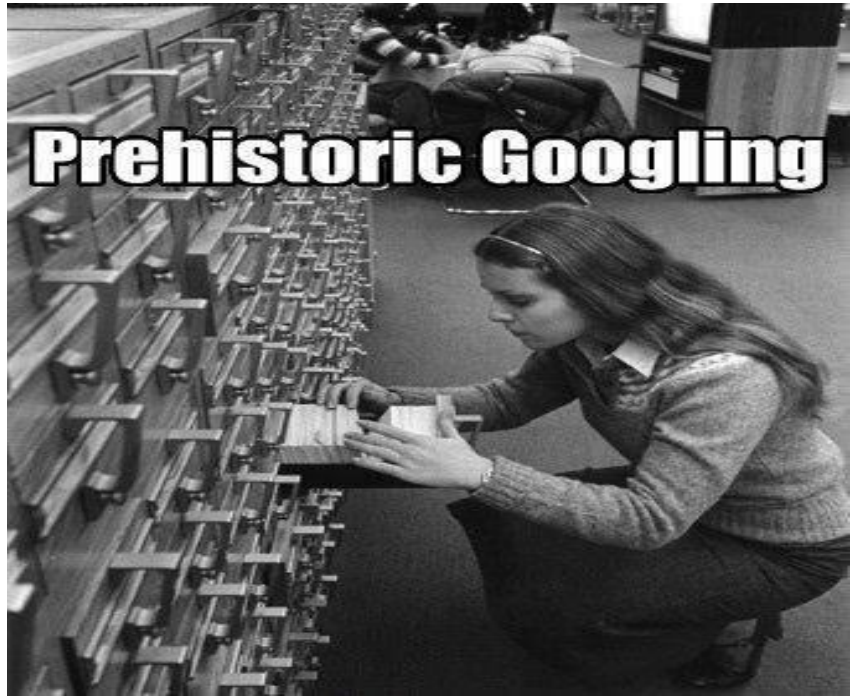


PISALNI STROJ

- če so se prevečkrat zmotili, so morali besedilo natipkati še enkrat
- če so hoteli že napisano besedilo uporabiti še v kakšnem drugem besedilu, so ga morali ponovno natipkati

ISKANJE INFORMACIJ

- v knjižnicah - več dni ali celo mesecev
- neuspešno, če knjižnica ni imela knjig



SPLETNI BRSKALNIKI

- v manj kot sekundi urejen seznam spletnih povezav, ki ustrezajo našim zahtevam
- v nekaj minutah najdemo ustrezno informacijo

RAČUNALNIŠKI PROGRAM

- opravil, v prejšnjem odstavku, ne bi mogli izvesti, če ne bi imeli na računalniku ustreznih programov
- računalnik - stroj, ki izvaja **zaporedje ukazov** (računalniški program), ki mu poda človek
- računalniški program - preslikava rešitve problema (npr. kako pisati, urejati in shraniti besedilo) v ukaze, ki jih zna izvajati računalnik
- računalniški program

ideja → podrobno razdelati → zapisati v enem izmed programskih jezikov

ALGORITEM

- seznam navodil za izvedbo opravila
- definiramo zaporedje akcij (ali postopkov) nad podatki, da dosežemo želen rezultat
- za računalnika - napisati dovolj natančno in nedvoumno, da ga bo lahko rešil z ukazi, ki jih ima na voljo
- predvideti vse možne situacije (npr. za prejšnji primer bi morali predvideti možnost, da je stol v drugem prostoru, do katerega lahko pridemo po stopnicah)
- algoritem mora torej enolično opredeliti, končno zaporedje akcij, s katerimi dosežemo želeni cilj

ALGORITEM

Poglejmo primer algoritma, s katerim opišemo opravilo, ki je vezano na pripravo in zaužitje kosila:

1. Umij si roke.
2. Skuhaj kosilo.
3. Serviraj kosilo.
4. Pojej kosilo.
5. Pomij posodo.

ALGORITEM

Zgoraj opisane korake je potrebno opisati podrobneje.

1. Pripravi posodo za kuhanje zelenjavne juhe.
2. Iz hladilnika vzami zelenjavo.
3. Umij zelenjavo.
4. Stresi zelenjavo v posodo.
5. Dolij vodo v posodo, tako da bo zelenjava plavala v vodi.
6. Prižgi ploščo na štedilniku. Nastavi jo na vrednost 9.
7. Daj posodo na štedilnik in kuhaj juho 20 minut. Pri tem pazi, da ti juha ne povre, zato sproti po potrebi dolivaj vodo.
8. Ko poteče 20 minut, izključi ploščo na štedilniku.

ALGORITEM

- njihov opis se ponavadi razlikuje v odvisnosti od tega, na katerih področjih življenja uporabljamo ta algoritem
- ustno (algoritem izrazimo z besedami)
- pisno v obliki psevdokoda (množica korakov, zapisanih v mešanici naravnega jezika in matematične notacije)

VHOD: dve števili

IZHOD: vrednost večjega izmed dveh števil

ALGORITEM:

```
vnesi števili
    if (prvo število > drugo število)
      izpiši prvo število
    else
      izpiši drugo število
```

ALGORITEM

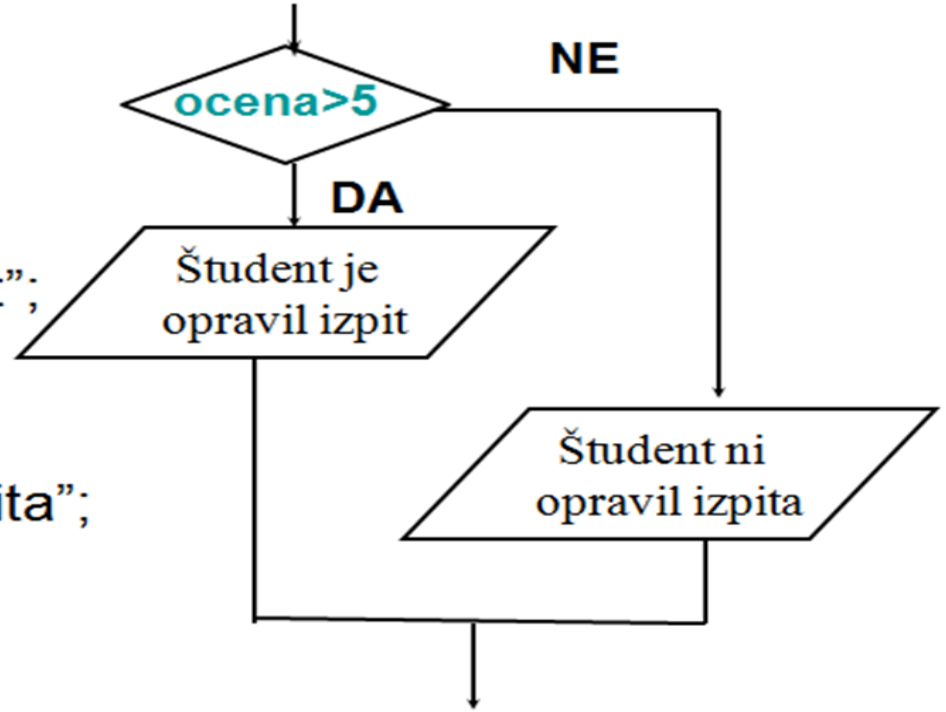
- tabelarično – algoritem je predstavljen z eno ali več tabel

Tir 1	Tir 2	Zapornice
vlaka ni	vlaka ni	odprte
vlak prihaja	vlaka ni	zaprte
vlaka ni	vlak prihaja	zaprte
vlak prihaja	vlak prihaja	zaprte

ALGORITEM

- z diagrami poteka (flowchart) – v obliki diagrama z akcijskimi pravokotniki (ali drugimi simboli), ki so povezani s puščicami
- puščice nakazujejo zaporedje akcij, ki naj se izvedejo

```
if ( ocena > 5 )
 cout << "Študent je
          opravil izpit";
else
 cout << "Študent ni
          opravil  izpita";
```



LASTNOSTI ALGORITMOV

Razumljivost: razumljiv in enostaven za uporabo

Nedvoumnost: vsi koraki natančno in nedvoumno definirani, obstaja en sam način interpretacije posameznega koraka

Determinističnost: vedno moramo dobiti enak rezultat ob istih vhodnih podatkih

Končnost: mora se izvesti v končnem številu korakov in mora uporabljati vnaprej definirano končno množico vhodnih, vmesnih in končnih spremenljivk

Dobra strukturiranost: narejen po predlogah, ki jih je enostavno razložiti, so razumljive in jih lahko testiramo/spreminjamo

LASTNOSTI ALGORITMOV

Učinkovitost: čim manj ukazov/manj pomnilniškega prostora

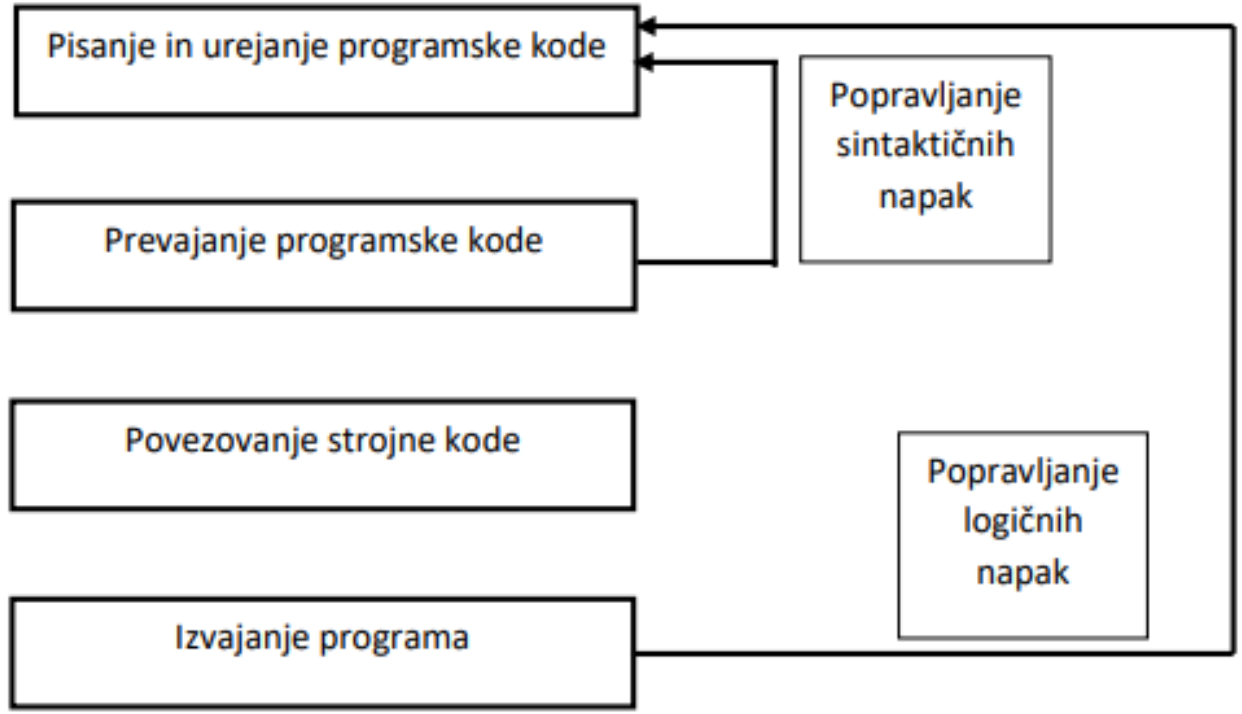
Robustnost: predvidi vse možnosti, ki se lahko pojavijo pri vnosu vhoda in njegovi izvedbi (npr. če vnašamo kot vhodni podatek celo število v območju med 1 in 10, mora algoritem predvideti možnost, da bo uporabnik vnesel podatek, ki ni ustrezen (npr. niz znakov, realno število itd.)).

Splošnost: lahko rešeujemo več podobnih problemov (npr. algoritem s katerim opišemo urejanje celih števil po naraščajočem vrstnem redu lahko uporabimo tudi za urejanje nizov znakov).

Ekonomičnost: zmanjšan čas izvajanja opravil in stroške, v primerjavi s stroški preden smo izdelali računalniški program

ORODJA ZA POMOČ PRI RAZVOJU PROGRAMOV

- urejevalnik besedila (npr. Notepad)
- urejevalnik razvojnega okolja, ki nam programsko kodo ustrezno obarva in nas že sproti opozarja na sintaktične/slovnične napake



ORODJA ZA POMOČ PRI RAZVOJU PROGRAMOV

- pri pisanju se lahko zatipkamo, ali pa ne poznamo dovolj sintakse/slovnice jezika
- pri prevajanju pride do sintaktičnih napak
- napake odpravljamo tako dolgo, dokler program ni sintaktično povsem pravilen
- program prevaja **prevajalnik ali tolmač**
- **prevajalnik (angl. compiler**) prevede celoten program v strojno kodo, medtem ko tolmač sintaktično preveri vsako vrstico in jo izvede
- ko smo program povezali, ga lahko zaženemo - to nalogo opravlja nalagalnik **(angl. loader)**
- pri izvedbi programa vpisujemo vhodne podatke in preverjamo izhodne rezultate

ORODJA ZA POMOČ PRI RAZVOJU PROGRAMOV

- moramo narediti ustrezen testni načrt
- testnem načrtu definiramo množico vhodnih podatkov ter množico izhodnih rezultatov, če vpišemo te vhodne podatke
- program zaganjamo tako dolgo, dokler ne izvedemo vseh testnih scenarijev
- v določenih primerih je odkrivanje nepravilnega delovanja programa bolj zahtevno, saj ne moremo na enostaven način ugotoviti kje so logične napake
- v teh primerih uporabimo **razhroščevalnik (angl. debugger)**
- ta nam omogoča, da izvajamo program po korakih - v vsakem koraku lahko preverimo vrednosti spremenljivk in objektov, tako lahko lažje ugotavljamo, na katerem mestu smo naredili logično napako