

## STRUKTURA PROGRAMA

1. Konkretno probleme na nivoju programa lahko rešimo s različnimi programskimi prijemi, vendar je določen način zapisa boljši od drugih – strukturiran program.

Prednosti:

1. Program napišemo lažje in hitreje
2. Testiranje je preprosto, pa tudi pravilnost lažje dokazljiva
3. Če so potrebne spremembe jih je lažje izvesti
4. Program je hitrejši
5. Program je bistveno krajši, vsebuje manj spremenljivk in potroši manj pomnilniškega prostora.

Če program ni pregleden, se lahko v lastnem programu izgubimo.

## FAZE V RAZVOJU PROGRAMA

Naloga programerja je, da napiše program, ki reši dani problem. Navadno poznamo opis problema v pogovornem jeziku, ki ga moramo prevesti v jezik razumljiv računalniku. Ta proces razdelimo v 4. faze:

1. razjasnitev problema in zahtev
2. razvoj postopka rešitve problema
3. specifikiranje podatkovnih struktur
4. pisanje programa

### 1. Razjasnitev problema in zahtev

Problem in zahteve morajo biti precizno definirani, ker je ta faza bistvena pri razvoju programa. Ker programerju problem največkrat zastavi uporabnik, ki ne pozna delovanja in zmožnosti računalnika, je prav na tem nivoju največ težav. Zelo hitro pride do nesporazumov pri definiciji problema, kar ima za posledico, da programer napiše program, ki reši problem drugačen, kot mu je bil zastavljen. Zato je zelo pomembno, da so vse zahteve pri reševanju problema natančno definirane. Zelo važnih je nekaj ključnih vprašanj, ki jih je treba brezpogojno razjasniti.

#### 1. Vnos podatkov

Kakšna je oblika podatkov in kakšno je zaporedje pri vnašanju. Kako testiramo konec vnosa podatkov in v kakšnem intervalu ležijo podatki. Ali se da dobiti vzorčni primerek.

#### 2. Izpis rezultata

Kakšna je vsebina, oblika in zaporedje prikaza izpisanih rezultatov.

#### 3. Napake

Do kakšne vrste napak lahko pride in kakšno naj bo opozorilo.

ZGLED – Kako razjasnimo problem:

Problem je dan v splošni obliki.

Napiši program, ki izračuna vsoto podanega zaporedja števil.

1. Ali so podatki, ki jih vnašate v eni vrstici ali pa je vsak podatek v svoji vrstici.
2. Kakšno je število podatkov in kako podatke zaključujemo.
3. Kakšen tip in kako so veliki podatki (realna, cela števila,...)
4. Kako se izpiše rezultat (oblika in format zapisa)

5. Kakšen naj bo komentar izpisa
6. Kako naj program reagira na podatke, ki niso ustreznega tipa in velikosti.
7. Kako naj program reagira, če je vnesen podatek prevelik.

## 2. Razvoj postopka rešitve problema

Faza razvoja postopka rešitve je najkreativnejši in najustvarjalnejši del programerjevega posla. Pri razvoju postopka naj programer pozabi na računalnik in poiskuje pravilno zastavljeni problem rešiti na papirju ali miselno.

Metoda zaporednih izboljšav pride v tej fazi do izraza.

Zgled: reševanje enačbe

Program naj reši linearno enačbo z eno neznanko. Enačba naj bo prečitana iz vhodne datoteke, kjer je zapisana v eni sami vrstici. Vsa števila v vhodni enačbi naj bodo realna, brez decimalnega dela. Izpis na zaslonu pa naj bo v nasledni obliki:

VHOD

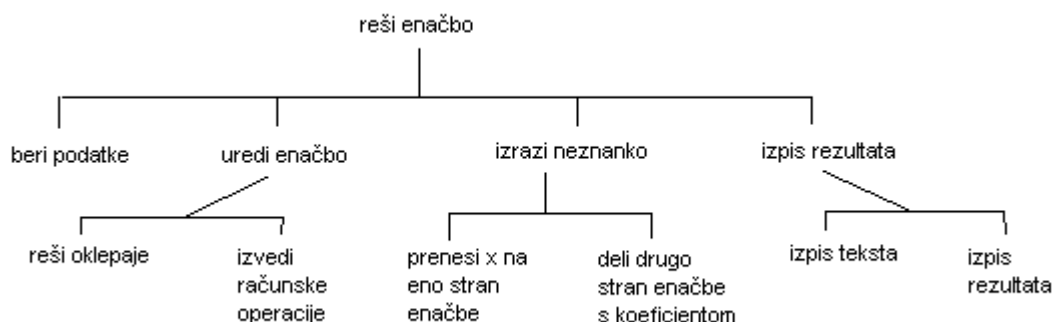
$$x - 2 = 6$$

$$2 * (x + 3) = 7$$

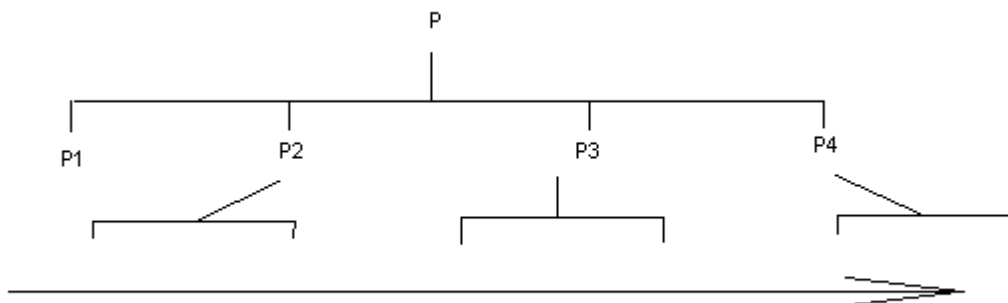
IZHOD

$$x = 8$$

$$x = 0,5$$



drevo razvoja:



Rešitev teče vedno iz leve proti desni strani.

### 3. Specificiranje podatkovnih struktur

Pri izvoru podatkovnih struktur je potrebno vedeti s kakšnim tipom podatkov bomo operirali. Najbolje je, da specificiramo podatke čim pozneje, ker si tako prihranimo precej popravljanja. V pascalu moremo vse spremenljivke tudi deklarirati. Koristno je, da izberemo taka imena spremenljivk, ki s svojim imenom nakažejo, kaj spremenljivka predstavlja.

Zgled:

Preprost informacijski sistem bolnišnice vsebuje podatke o posameznih pacientih, kot so ime, priimek, datum sprejema v bolnišnico, številka sobe in številka zdravnika, ki je pacienta napotil v bolnišnico.

Vsi pacienti (1000x)

- en pacient
  - ime pacienta (20x znak)
  - datum sprejema
    - dan (cela števila 1..31)
    - mesec (cela števila 1..12)
    - leto (cela števila 1995..2080)
  - številka sobe (cela števila 1..100)
  - šifra zdravnika (cela števila 0..9999)

ena izmed možnosti konverzije:

program norišnica;

const postelje = 1000;

type

```
    ime = array[1..20] of char;
    datum = record
        dan:1..31;
        mesec:1..12;
        leto:1995..2080;
    end;
    soba = 1..100;
    zdravnik=0..9999;
    pacient=record
        ime_priimek:ime;
        datum_sprejema:datum;
        st_sobe:soba;
        sifra_zdravnika:zdravnik;
    end;
```

var

```
pacienti:array[1..postelja] of pacient;
```

### 4. Pisanje programa

1. Program mora biti ustrezno dokumentiran. Parametri podprogramov naj bodo dokumentirani in komentirani v podprogramih.

Komentarje pišemo kot nadaljevanje ustrezne vrstice in jih nadaljujemo tako, da so vrstice istega stavka poravnane.

Osnovna dokumentacija:

1. opis problema, ki ga program rešuje
2. avtor

3. datum prvega prevajanja programa
4. kratek opis bistvenih lastnosti delovanja programa
5. vhodni in izhodni podatki in uporabljene datoteke

2. program piši tako, da brez sprememb deluje na bilokaterem računalniku, ki ima ustrezen prevajalnik.

Noben segment naj ne presega pol strani izpisa (mišljeno je pri izpisih podatkov, da naj izpisi vsebujejo tudi izpise mogočih tipk, ki jih uporabnik pritisne, naslov in ne samo podatkov).

1. Vsak stavek naj začne v novi vrstici
2. krmilni stavki repeat-until in begin..end naj zasedajo svojo vrstico s ustreznim komentarjem
3. imena spremenljivk, konstant, funkcij in procedur (hungarian notation!)
4. rezervirane besede program, function, procedure naj se začnejo na skrajno levem robu.
5. Vsak stavek med begin in end, repeat .. until mora biti poravnan.
6. Pred in za operatorji (+,-,\*,/,...) naj bo eno mesto prazno.
7. Vsi parametri v programu, ki imajo ves čas izvajanja konstantno vrednost, naj bodo deklarirani kot konstanta
8. globalne spremenljivke naj bodo skrbno izbrane
9. stavek GOTO je nezaželen

Izpelji drevo razvoja za naslednji problem:

Program naj prebere vrstico sestavljeno iz besed, ločenih s praznimi mesti in izpisuje obrnjene besede. Če ima beseda več kot 10 znakov(črk), naj jo program razbije na več besed.

MANJKA SLIKA!!

### **Napotki za dobro programiranje**

1. ne bodi paničen
2. popolnoma definiraj problem
3. takoj začni z dokumentiranjem
4. najprej premisli, nato napiši
5. začni od zgoraj navzdol (problem postopoma razgrajuj)
6. program piši tako, da je zgrajen iz logičnih enot
7. uporablaj podprograme
8. goto stavek ni zaželen
9. uporabi imena, ki nekaj pomenijo
10. piši učinkovite komentarje
11. konstante naj bojo konstante
12. piši čitljivo in pazi na obliko
13. sintaksa naj bo pravilna
14. program preizkusi ročno
15. svoje napake popravljal sam
16. program daj v pregled še komu drugemu
17. ne boj se začeti znova od začetka

Nadalni razvoj postopkov pri razvoju programov

1. Problem razgradimo na več majših vase zaključenih podproblemov.
2. Vsak stavek v opisu postopka postopoma nadomeščamo z zaporedjem preciznejših stavkov.
3. Stavek, ki pove KAJ je treba napraviti preoblikujemo v stavke, ki povedo KAKO je treba napraviti.

4. Stavke, s katerimi opisujemo algoritem-postopek prevedemo v stavke programskega jezika
5. Stavke v programskem jeziku združimo v program in tako opis problema prevedemo v programskem jeziku.

**Prednost metode razgrajevanja od vrha navzdol je v tem, da se v začetku osredotočimo na nivo, kjer je jasno, kaj je treba narediti in sistematično napredujemo proti nivoju, kjer povemo kako je treba narediti.**

Zgled: urediti množico števil (urejanje)

Izberemo metodo največjega elementa – urejanje z izbiranjem

Bistvo metode je v tem, da tabelo števil pregledujemo toliko časa, dokler ne najdemo največjega števila, ki ga potem zamenjamo s številom na koncu neurejene tabele. Postopek ponavljamo od začetka do konca neurejene tabele. Novo največje število iz neurejene tabele zopet zamenjamo s zadnjim številom neurejene tabele. Tako se neurejena tabela števil krajša, urejena pa daljša. Ob koncu urejanja so števila urejena po naraščajočem vrstnem redu.

<b>2</b>	<b>12</b>	<b>4</b>	<b>8</b>	<b>5</b>
2	5	4	8	<b>12</b>
2	5	4	<b>8</b>	<b>12</b>
2	4	<b>5</b>	<b>8</b>	<b>12</b>
2	<b>4</b>	<b>5</b>	<b>8</b>	<b>12</b>

Problem razgradimo na zaporedje podproblemov:

- > uredi tabelo :
  - > a) preberi števila v tabelo
  - > b) uredi tabelo
  - > c) izpiši urejeno tabelo

b.1.) dokler je neurejena tabela ponavljaj

b.2.) poišči največje število (v neurejenem delu tabele)

b.3.) zamenjaj največje število z zadnjim številom v neurejenem delu tabele.

```

max := a[i];
p := 1;
for j := 2 to n do
  if a[j] > max then
    begin
      p := j;
      max := a[j];
    end;

```