

## DEFINIRANI PODPROGRAMI - FUNKCIJE in PROCEDURE

Poznamo:

Standardne funkcije (so zapisani podprogrami, ki imajo svoje ime in so že napisani v programskem jeziku).

Definirane funkcije (so podprogrami, ki jih programer sam izdelava za lažje delo)

Vsak podprogram ima vhode in izhode.

Vmes podprogram podatke obdeluje.

Podprogram prikažemo na naslednji način:

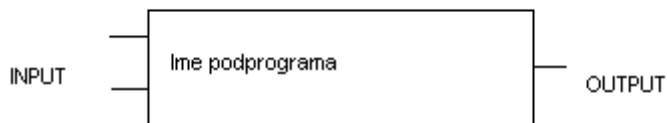


Podprograme povezujemo v knjižnice. Standardni podprogrami so shranjeni v standardnih knjižnicah (primer stdio.h, math.h,...). Programer lahko tudi sam izdelava module. Pisanje podprogramov nas vodi v modularno programiranje – ponovna uporaba že izdelanih podprogramov.

Kaj pridobimo s definiranimi podprogrami in moduli:

- kot že omenjeno so več uporabni
- razmišljanje se poenostavi (problem razgradimo na enostavne dele in izdelamo podprograme, ki jih povezujemo v celoto)
- hitreje do cilja (lažji problemi se hitreje rešijo, z ponovno uporabo podprogramov,...)

Funkcijski ki vračajo vrednost

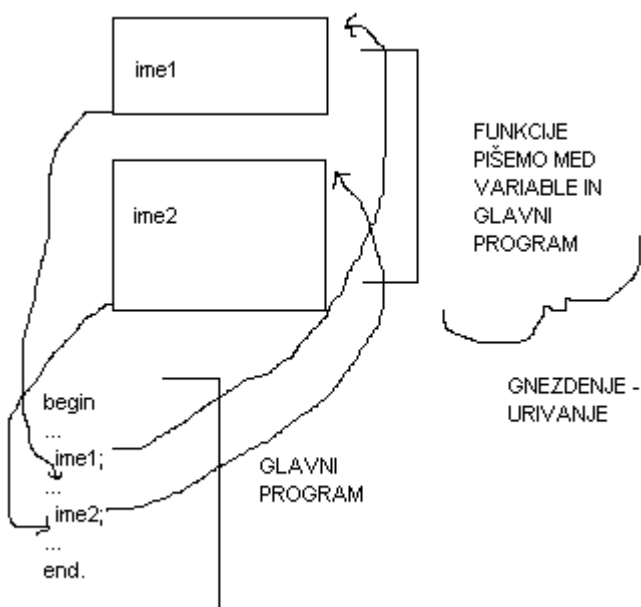


program ...;

var

...

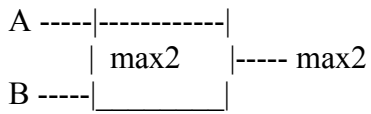
- spremenljivke na tem nivoju so globalne



Gnezdenje funkcij (funkcija v funkciji) v programskem jeziku C je prepovedano!

Zgled:

iskanje maksimalnega števila dveh števil. Imamo vhod A in B, ter izhod max2.



```
float max2(float a, float b)
```

- takemu načinu deklariranja pravimo vrednostno deklariranje  
- FORMALNI PARAMETRI – te spremenljivke so aktivne le v delovanji funkcije

```
{  
    if (a > b)  
        return a;  
    else  
        return b;  
}
```

```
#include <stdio.h>  
#include <stdlib.h>
```

```
float prvo, drugo, rez; // globalne spremenljivke
```

```
float max2(float a, float b)  
{  
    if (a > b)  
        return a;  
    else  
        return b;  
}
```

```
int main()  
{  
    printf("Vnesi prvo stevilo: "); scanf("%f",&prvo);  
    printf("Vnesi drugo stevilo: "); scanf("%f",&drugo);  
    rez = max2(prvo,drugo);  
    printf("Najvecje stevilo je %0.2f\n",rez);  
    return 0;  
}
```

Primer: Izdelaj program, ki prebere tri realna števila in v podprogramu najdete največje realno število.

```
#include <stdio.h>  
#include <stdlib.h>
```

```
float prvo, drugo, tretje, max; // globalne spremenljivke  
float max3(float a, float b, float c)  
{  
    if (a > b)  
        max = a;  
    else  
        max = b;  
    if (c > max)
```

```

max = c;
return max;
}

int main(void)
{
    printf("Največje števimo med tremi je %0.2f\n",max3(123.3,124.2,3456.23));
    return 0;
}

```

### Funkcije, ki ne vračajo vrednosti - procedure

Zgled:

```

void vrstica(int n)
{
    int i;
    for (i=1;i<=n;i++)
        printf("-");
}

```

procedure so podobne funkcijam. Razlika med funkcijami in procedurami je v tem, da funkcije so navadno krajše in vračajo natanko en rezultat (ni mogoče izdelati funkcije, ki ne vrača rezultata ali pa da ima več kot en rezultat), medtem ko procedure navadno nimajo nobenega rezultata. Vračanje rezultatov iz procedure je mogoče le preko vhodnih spremenljivk. Namreč poznamo dva tipa vnosa spremenljivk v proceduro (pa tudi v funkcijo):

- po vrednosti in
- po referenci.

Pri vnosu spremenljivk po vrednosti se dejanske vrednosti priredijo lokalnim spremenljivkam v proceduri (pri tem, se vrednost zapisana v lokalno spremenljivko ob zaključku procedure izgubi). Medtem pri vnosu spremenljivk po referenci, se spremenljivke ne shranijo v lokalne kopije, ampak se uporabi globalna spremenljivka in zaradi tega, se vrednost, ki se zapiše v spremenljivko tudi ohrani po zaključku procedure. S tem omogočimo, da se vrednosti tudi po zaključku procedure ohranijo – kot izhodni podatki.

Primer procedure z spremenljivkami deklariranimi po vrednosti:

```

void izdelaj(int a)
{
    printf("%d\n",a);
    a = 5;
}

```

v tem primeru lahko pokličemo proceduro ali z konstantno vrednostjo ali z spremenljivko. Vrednost se pri izvedbi tega ukaza prepíše v lokalno spremenljivko z imenom a in se po izdeku podprograma vrednost tudi izgubi.

A := 5;

Klica izdelaj(1); kot tudi izdelaj(a); so možna.

Primer procedure z spremenljivkami deklariranimi po referenci:

```

void izdelaj(int *a)
{

```

```
printf("%d\n",*a);  
*a = 5;  
}
```

v tem primeru lahko pokličemo proceduro samo s spremenljivko. Namreč če pokličemo s konstanto se vrednost ne more zapisati v nobeno pomnilniško lokacijo (napako vrne že prevajalnik). Vrednost spremenljivke a v podprogramu se ohrani tudi po izvedbi podprograma. Tako je spremenljivka lahko vhodni podatek kot tudi izhodni podatek.

A := 5;

Možno je proceduro klicati samo z izdelaj(&a); Klic izdelaj(1); je napačen in že prevajalnik javi napako.

Naloge:

- izdelaj podprogram, ki izpiše največje število med dvema vhodnima podatkom
- izdelaj podprogram, ki izpiše največje število med tremi vhodnimi podatki
- izdelaj podprogram, ki zamenja vrednost med dvema podatki
- izdelaj podprogram, ki zamenja vrednost med tremi podatki (a = b, b = c; c = a);
- izdelaj podprogram, ki zamenja vrednosti med tremi podatki (a = c, b = a, c = b);
- izdelaj podprogram, ki ima za vhod dve števili, kateri predstavljata interval iz katerega podprogram vrne naključno generirano število.
- izdelaj podprogram, ki izpiše naključno malo ali veliko črko