# PROGRAMIRANJE 1
# UVOD U PROGRAMSKI JEZIK C

Staša Vujičić

v2

# Algoritmi

# FORMIRANJE ALGORITMA

- Konstruisati efikasan algoritam znači dati skup preciznih uputstava kako doći do rešenja zadatog problema
- Algoritmi se mogu opisivati:
  - pseudo jezikom
  - prirodnim jezikom
  - dijagramom toka.

# OPIS ALGORITAMA PRIRODNIM JEZIKOM

- Opisati prirodnim jezikom detaljno, precizno i nedvosmisleno korake pri rešavanju problema, vodeći računa o redosledu operacija koje se izvršavaju.

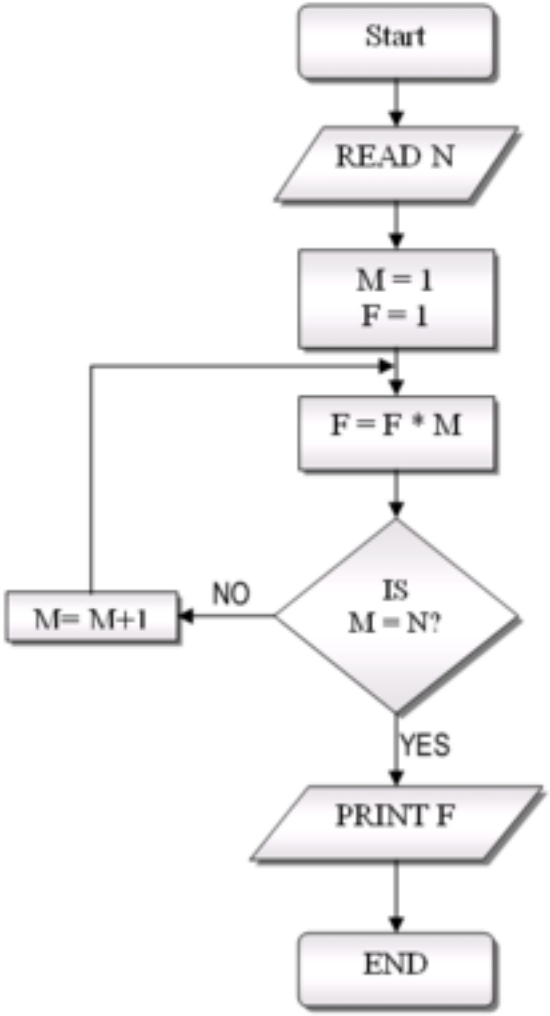
# OPIS ALGORITAMA PSEUDO JEZIKOM

- Pseudo jezik je neformalna kombinacija prirodnog jezika i nekog programskoj jezika.
- Pri upotrebi pseudo jezika mora se voditi računa da se jezičke konstrukcije koriste uvek na isti način i da budu praćene objašnjenjima (ako je potebno).

# OPISIVANJE ALGORITAMA DIJAGRAMOM TOKA

- Za ovaj oblik opisa koriste se grafički simboli čiji je opis propisan ISO standardom. Tekst koji opisuje obradu se zapisuje unutar grafičkih simbola. Tok rada algoritma se opisuje linijama koje povezuju grafičke simbole koji reprezentuju obradu.

# Primer dijagrama toka



# Programski jezik C

# LEKSIČKE KONVENCIJE

- U upotrebi su velika i mala slova, cifre i specijalni simboli iz ASCII skupa
- Programski jezik C razlikuje velika i mala slova!
- Komentari se navode između sekvenci /* i */. Mogu se prostirati u više linija. Ne mogu biti ugnježdeni.
- **int x, X;** /*To su dve razlicite promenljive!!!*/

# TOKENI

- Postoji šest vrsta tokena:
  - identifikatori
  - ključne reči
  - operatori
  - separatori
  - stringovi i
  - konstante.

- Tokeni se razdvajaju belinama, tabulatorima i novim redovima.

# IDENTIFIKATORI

- **Identifikatori** se sastoje iz slova, cifara i znaka _ pri čemu prvi karakter nije cifra.
- Koriste se za imena promenljivih, tipova, funkcija, itd.

# KLJUČNE REČI

- **Ključne reči** su rezervisane reči koje imaju posebnu ulogu, i ne mogu se koristiti kao identifikatori.
- Ključne reči se koriste za:
  - definisanje jezičkih konstrukcija (*if*, *while*, *for*),
  - imena tipova (*int*, *float*, *char*), itd.

# PROGRAM U C-U

- Izvorni program (source code) u C-u je običan tekstualni fajl, kreiran u bilo kom editoru teksta (npr. Notepad).
- Program u C-u sastoji se iz definicija funkcija.
- Funkcija koja mora da postoji u svakom programu je **main()**. Izvršavanje programa se svodi na izvršavanje tela ove funkcije.
- **Telo funkcije** se navodi iza zaglavlja funkcije main(), između vitičastih zagrada { i }.

- U telu funkcije se na početku navode deklaracije pomenljivih, nakon čega sledi proizvoljan niz naredbi .

```
main() /* zaglavlje funkcije main */
{  /* ovde pocinje telo */
   /* deklaracije promenljivih */
   /* naredbe */
} /* kraj tela */
```

# PRVI PRIMERI U C-U

- *Napisati program koji na standardnom izlazu štampa "Zdravo, svete!".*
- **#include <stdio.h>**

  **main()**

  **{**

  **printf("Zdravo, svete!\n");**

  **}**
- Izlaz iz programa:

  Zdravo, svete!

- *Šta je izlaz iz sledećeg programa?*
- **#include <stdio.h>**

```
main()
{
        printf("Zdravo, ");
        printf("svete!");
        printf("\n");
}
```

# PROMENLJIVE. DEKLARACIJA I INICIJALIZACIJA

- Sve promenljive u C-u se moraju deklarisati. Time se za promenljivu u memoriji rezerviše potreban prostor, a ostatak programa postaje svestan postojanja promenljive i njenog tipa.
- Deklaracija se sastoji iz imena tipa za kojim slede imena promenljivih (identifikatori) koje se deklarišu, i koja su razdvojene zarezima.
- Deklaracija se završava simbolom ';'.
- Svako ime promenljive u deklaraciji može biti praćeno inicijalizatorom koji se sastoji iz karaktera '=' za kojim sledi inicijalna vrednost.

- Dakle, da bi se promenljiva mogla upotrebljavati u programu ona se mora na početku programa deklarisati!
- Prilikom deklaracije može se izvršiti i početna inicijalizacija.

```
int broj; /* Deklaracija celog broja */
int vrednost=5; /* Deklaracija i inicijalizacija celog broja */
```

- Postoji i kvalifikator **const** koji može biti dodeljen deklaraciji bilo koje promenljive da bi označio da se ona neće menjati

```
const double e=2.71828182845905 ;
```

- *Uvođenje promenljivih u program.*
- ```
  #include <stdio.h>
  main()
  {
  /* Deklaracija vise promenljivih istog tipa */
        int rez,pom1,pom2;
        pom1=20;
        pom2=15;
        rez=pom1-pom2;
  /* Ispisivanje rezultata */
        printf("Rezultat je %d - %d = %d\n", pom1,
  pom2, rez);
  }
  ```
  ```

- Izlaz iz programa:

  Rezultat je 20-15=5

# OSNOVNI TIPOVI PODATAKA U C-U

| Tip | Opis | Uobičajena veličina |
|---|---|---|
| char | Mali ceo broj (ASCII kod karaktera) | 1 bajt |
| int | Ceo broj | 4 bajta |
| float | Realan broj | 4 bajta |
| double | Realan broj | 8 bajtova |

# FUNKCIJE ULAZA I IZLAZA

- Ulaz i izlaz ostvaruju se posredstvom funkcija koje su definisane u standardnoj biblioteci **stdio.h**.
- Ove funkcije su obične C funkcije, koje se služe direktno servisima operativnog sistema prilikom svog rada.

- Za korišćenje ovih funkcija neophodno je uključiti zaglavlje **stdio.h** navođenjem direktive **#include<stdio.h>** pre definicije funkcije main().
- Ovo zaglavlje je obilan tekstualni fajl u kome su navedene deklaracije funkcija ulaza i izlaza.
- Direktiva **#include** na mestu poziva uključuje kompletan sadržaj fajla koji je naveden, čime funkcije i podaci deklarisani u njemu postaju dostupni funkciji main().

# FUNKCIJA PRINTF()

- Ovom funkcijom se ispisuje poruka zadata format stringom na standardni izlaz.
- Eventualni konverzioni specifikatori se zamenjuju vrednostima izraza koji u tom slučaju slede nakon format stringa, kao argumenti funkcije printf(), razdvojeni zarezima i u onom poretku u kome su odgovarajući konverzioni specifikatori navedeni.
- Tipovi izraza moraju biti u skladu sa tipovima koje određuju konverzioni specifikatori.

# KONVERZIONI SPECIFIKATORI

| Specifikator | Tip | Napomena |
|---|---|---|
| %d | int | Dekadni ispis |
| %f | float, double | Ispis bez eksponenta |
| %Lf | long double | Ispis bez eksponenta |
| %hd | short | Dekadni ispis |
| %ld | long | Dekadni ispis |
| %c | char | Ispis karaktera |

- Funkcija printf je bibliotečka funkcija koja prikazuje izlazne podatke u određenom formatu.
- Primer korišćenja funkcije printf je:

```
printf("%d\t%d\n", broj1, broj2);
```

- Prvi argument ove funkcije je uvek između ” ” i određuje format u kome će se podaci ispisati na izlaz. Ova funkcija vraća kao vrednosti broj upisanih znakova na izlazu.

- Sekvenca **\n** u okviru prvog argumenta funkcije printf je C oznaka za prelazak u novi red, **\t** je oznaka za tabulator, dok **%d** označava da će na tom mestu biti ispisana celobrojna vrednost argumenta koji je sa njim u paru. Svaka % konstrukcija je u paru sa odgovarajućim argumentom koji sledi.
- **%%** koristi se za ispis znaka %
- **** koristi se za ispis znaka \
- **\”** koristi se za ispis znaka ”

- Postoji mogućnost da se precizira i širina polja u kome će se ispisati odgovarajuće vrednosti.
- Na primer, koristimo **%3c** za štampanje karaktera na tri pozicije poravnato zdesna.
- Koristimo **%3d** za štampanje broja na tri pozicije ili **%6d** za štampanje broja na 6 pozicija.

- Važi:
- **%f** – štampaj kao realan broj
- **%6f** – štampaj kao realan broj širok najviše 6 znakova
- **%.2f** – štampaj kao realan broj sa dve decimale
- **%6.2f** – štampaj kao realan broj širok najviše 6 znakova pri čemu su 2 iza decimalne tačke.
- Da bi se izvršilo levo poravnanje, između % i odgovarajućeg karaktera dodaje se znak -.

# PRIMER - PRINTF()

```
#include <stdio.h>
main()
{
printf("Slova:\n%3c\n%5c\n", 'z' , 'Z');
}
```

- Izlaz iz programa:

```
Slova:
  z
    Z
```

# FUNKCIJA SCANF()

- Ovom funkcijom se učitavaju podaci sa standardnog ulaza.
- Prvi argument je format string u kome se navode konverzioni specifikatori kojima se definiše tip podatka koji se očekuje.
- Nakon format stringa slede adrese promenljivih, razdvojene zarezima, u koje treba upisati vrednosti učitane sa ulaza. Adresa promenljive **a** navodi se sa **&a**.

- Adrese se navode u onom poretku u kom su odgovarajući konverzioni specifikatori navedeni u format stringu.
- Tipovi promenljivih moraju biti u skladu sa tipovima koje određuju konverzioni specifikatori.

# KONVERZIONI SPECIFIKATORI

| Specifikator | Tip | Napomena |
|---|---|---|
| %d | int | Opciono označeni dekadni broj |
| %f | float | Realan broj sa opcionim eksponentom |
| %lf | double | Realan broj sa opcionim eksponentom |
| %Lf | long double | Realan broj sa opcionim eksponentom |
| %hd | short | Opciono označeni dekadni broj |
| %ld | long | Opciono označeni dekadni broj |
| %c | char | Karakter iz ASCII skupa |

- Na primer,

```
scanf("%d %d", &broj1, &broj2);
```

- Ova funkcija čita sa ulaza dva cela broja i smešta ih na adresu promenljivih broj1 i broj2, redom.
- Kao rezultat, ova funkcija vraća broj uspešno dodeljenih ulaznih vrednosti.
- Naredni poziv funkcije scanf nastavlja čitanje neposredno iza poslednjeg znaka koji je već pročitan.

# PRIMER - SCANF()

- *Program prikazuje unos celog broja koristeći funkciju scanf("%d", &x).*
- 
```
#include <stdio.h>
main()
{
        int x;
        printf("Unesi ceo broj : ");
        /* Obratiti paznju na znak &
        pre imena promenljive u funkciji scanf. */
        scanf("%d",&x);
        /* U funkciji printf nije
        potrebno stavljati &. */
        printf("Uneli ste broj %d\n", x);
}
```

# KONTROLA TOKA - IF

- **if (izraz)**

  **naredba1**

  **else**

  **naredba2**

- Naredba može biti prosta naredba a može biti i složena naredba (blok) koja se dobije kada se više prostih naredbi grupišu navođenjem vitičastih zagrada.

# PRIMER - IF

- *Program ilustruje if i ispisuje ukoliko je uneti ceo broj negativan.*
- **#include <stdio.h>**

```
main()
 {
 int b;
 printf("Unesi ceo broj:");
 scanf("%d", &b);
 if (b < 0)
 printf("Broj je negativan\n"); //prosta naredba

 }
```

- Ulaz:

  Unesi ceo broj:-5

  Izlaz:

  Broj je negativan
- Ulaz:

  Unesi ceo broj:5

  Izlaz:

- Else se odnosi na prvi neuparen if. Ako želimo drugačije moramo da navedemo vitičaste zagrade.
- **if (izraz) //prvo if**

  **if (izraz1) naredba1 //drugo if**

  **else naredba2**
- Ovo else se odnosi na drugo if a ne na prvo if!

- **if (izraz)**

  **{**

  **if (izraz1) naredba1**

  **}**

  **else naredba2**
- Tek sada se else odnosi na prvo if!!!

# ELSE-IF

- if (izraz1)
  iskaz1
  else if (izraz2)
  iskaz2
  else if (izraz3)
  iskaz3
  else if (izraz4)
  iskaz4
  else iskaz

# PRIMER - ELSE-IF

```
if (a<5)
        printf("A je manje od 5\n");
else if (a==5)
        printf("A je jednako 5\n");
else if (a>10)
        printf("A je vece od 10\n");
else if (a==10)
        printf("A je jednako 10\n");
else printf("A je vece od pet i manje od 10\n");
```

# PRIMER

- *Program ispituje znak broja.*
- ```
  #include <stdio.h>
  int main()
  {
        int b;
        printf("Unesi ceo broj : ");
        scanf("%d", &b);
        if (b < 0)
              printf("Broj je negativan\n");
        else if (b == 0)
              printf("Broj je nula\n");
        else
              printf("Broj je pozitivan\n");
  }
  ```
  ```

- Ulaz:

  Unesi ceo broj:-5

  Izlaz:

  Broj je negativan
- Ulaz:

  Unesi ceo broj:5

  Izlaz:

  Broj je pozitivan

# POGREŠAN PROGRAM SA DODELOM = UMESTO POREĐENJA ==.

```
#include <stdio.h>
int main()
{
        int b;
        printf("Unesi ceo broj : ");
        scanf("%d", &b);
/* Obratiti paznju na = umesto == Analizirati rad programa*/
        if (b = 0)
                printf("Broj je nula\n");
        else if (b < 0)
                printf("Broj je negativan\n");
        else
                printf("Broj je pozitivan\n");
}
```

- Ulaz:

  Unesi ceo broj:-5

  Izlaz:

  Broj je pozitivan
- **Voditi računa o tome da je = operator dodele a == je operator poređenja na jednakost.**
- **Ne mešati ta dva operatora!**

# WHILE PETLJA

- Uslov u zagradi se testira i ako je ispunjen telo petlje (naredba) se izvršava. Zatim se uslov ponovo testira i ako je ispunjen ponovo se izvršava telo petlje. I tako sve dok uslov ne postane neispunjen. Tada se izlazi iz petlje i nastavlja sa prvom sledećom naredbom u programu.
- **while (uslov)**

  **telo petlje**

- Voditi računa o tome da li je naredba koja čini telo petlje prosta ili složena!
- Ako nema vitičastih zagrada onda se prva naredba iza while(uslov) tretira kao telo while petlje.
- Na primer, u sledećem fragmentu koda

```
while (i<j)
        i=2*i;
        j++;
```

samo naredba i=2*i; se ponavlja u okviru while petlje dok se naredba j++; izvršava tek nakon izlaženja iz while petlje.

# PRIMER - WHILE PETLJA

```
#include <stdio.h>
int main()
{
        int x;
        x = 1;
        while (x<10)
        {
                printf("x = %d\n",x);
                x++; /* x++ je isto kao i x=x+1 */
        }
}
```

- Izlaz:

  x = 1

  x = 2

  x = 3

  x = 4

  x = 5

  x = 6

  x = 7

  x = 8

  x = 9