# Tema 4

## Argumenti komandne linije i funkcije

# Argumenti komandne linije

```c
#include <stdio.h>

int main(int argc, char **argv) {
    int i;

    printf("\nBroj argumenata je: %d\n", argc);

    printf("\nKonkretni argumenti su:\n");
    for(i = 0;i < argc;i++) {
        puts(argv[i]);
    }

    return 0;
}
```

Primer ispisa programa:

```
./a.out -o -x argument3 jedan dva tri

Broj argumenata je: 7

Konkretni argumenti su:
./a.out
-o
-x
argument3
jedan
dva
tri
```

- Prosleđivanje vrednosti programu prilikom pokretanja
- Omogućuje upravljanje programom spolja (primer `gcc`, `gedit` itd.)

# Zadatak 1

Napisati program koji prima argument komandne linije koji se sastoje od crtice i jednog slova i jednu reč. U programu zahtevati od korisnika da, takođe, unese jednu reč i na osnovu zadate opcije uraditi sledeće:

- `-s`, spojiti uneti string i string argumenta komandne linije sa razmakom između njih i ispisati rezultujući string
- `-p`, uporediti uneti string i string argumenta komandne linije i ispisati da li su isti po sadržaju
- `-n`, ispisati da li se string argumenta komandne linije nalazi u unetom stringu

Primer poziva programa:

```
./a.out -p test
```

# Funkcije

- Dekompozicija u manje, lakše rešive probleme
- Program čine razumljivijim
  - Uvođenje još jednog nivoa apstrakcije
  - Poznato je šta funkcija radi bez detalja na koji se to način postiže
- Generalizacija često korišćenih delova koda
  - Nije potrebno iznova pisati isti kod

# Primer 1

Jednostavna funkcija koja sabira dva broja

```c
#include <stdio.h>

int zbir(int a, int b) {
    int c;
    c = a + b;

    return c;
}

int main() {
    int z;

    z = zbir(3, 5);
    printf("%d\n", z);

    return 0;
}
```

# Primer 1 - alternativna verzija

Jednostavna funkcija koja sabira dva broja pomoću korišćenja zaglavlja funkcije

```c
#include <stdio.h>

int zbir(int, int);

int main() {
    printf("%d\n", zbir(3, 5));

    return 0;
}

int zbir(int a, int b) {
    return a + b;
}
```

- Obratiti takođe pažnju na izbačene suvišne promenljive iz prošlog primera

# Karakteristike funkcija

- Povratna vrednost
- Naziv
- Formalni parametri
- Telo funkcije

```
int test(int x, int y) {
    int x_square = x * x;
    int y_square = y * y;

    return x_square * y_square;
}
```

- Prepoznati navedene karakteristike u funkciji `test`

# Parametri funkcije

- Lokalna promenljiva, vidljiva samo u telu funkcije
- Argumenti i parametri funkcije (konkretni i formalni parametri)
- Vrednost dobija prilikom poziva funkcije (vrednost prosleđenog argumenta)
- Prestaje da važi čim se funkcija završi
- Napomene:
  - Lokalna promenljiva koja predstavlja parametar je potpuno različita od promenljive čija je vrednost prosleđena kao argument funkcije
  - Promena lokalne promenljive koja predstavlja parametar ne utiče na promenljivu čija je vrednost prosleđena kao argument funkcije

# Primer 2

Prenos parametra po vrednosti

```c
#include <stdio.h>

void foo(int i) {
    i = 3;
}

int main() {
    int i = 5;

    foo(i);
    printf("%d\n", i);

    return 0;
}
```

# Primer 3

Prenos parametra po referenci

```c
#include <stdio.h>

void foo(int *pi) {
    *pi = 3;
}

int main() {
    int i = 5;

    foo(&i);
    printf("%d\n", i);

    return 0;
}
```

- Koja će se vrednost ištampati u prvom, a koja u drugom slučaju?

# Naredba return

- Omogućava funkciji da vrati vrednost
- Tip povratne vrednosti mora odgovarati specificiranom povratnom tipu funkcije
- Funkcija se završava nailaskom na `return` naredbu
- U slučaju postojanja više `return` naredbi u okviru funkcije, ona će se završiti nailaskom na prvu od njih
- Ukoliko je povratni tip funkcije `void` ona ne vraća vrednost, već se može koristiti samo prazna `return` naredba koja će označiti završetak funkcije

# Primer 4

Funkcije za unos i ispis niza

```c
#include <stdio.h>

#define MAX_SIZE 30

void dodavanje(int *, int *);
void ispis(int *, int);

int main() {
    int a[MAX_SIZE], n;

    dodavanje(a, &n);
    ispis(a, n);

    return 0;
}
```

```
void dodavanje(int *a, int *n) {
    int i;

    do {
        printf("Unesite broj clanova niza: ");
        scanf("%d", n);
    } while(*n <= 0 || *n > MAX_SIZE);

    for(i = 0;i < *n;i++) {
        printf("a[%d] = ", i);
        scanf("%d", &a[i]);
    }
}
```

```
void ispis(int *a, int n) {
    int i;

    printf("[");
    for(i = 0;i < n;i++) {
        if(i > 0) {
            printf(", ");
        }
        printf("%d", a[i]);
    }
    printf("]\n");
}
```

# Zadatak 1

Proširiti `Primer 4` sledećim funkcijama

- `suma_elemenata`, čija je povratna vrednost suma elemenata niza
- `srednja_vrednost`, čija je povratna vrednost srednja vrednost elemenata niza
- `minimum`, čija je povratna vrednost minimum niza
- `maksimum`, čija je povratna vrednost maksimum niza

Proširiti `main` funkciju pozivima svake od navedenih funkcija.

# Zadatak 2

Dati su prirodni brojevi `n`, `m` (`n <= m`). Napisati program koji određuje koji od brojeva od `n` do `m` predstavljaju prestupne godine. Godina je prestupna ukoliko je zadovoljen sledeći skup uslova:

1. broj godine je deljiv sa četiri, i
2. važi jedno od sledećih pravila:
   - broj godine nije deljiv sa `100`
   - broj godine je deljiv sa `400`

# Zadatak 3

Napisati program kojim se štampaju svi trocifreni brojevi (ako ih ima) koji su jednaki sumi faktorijela svojih cifara.

# Zadatak 4

Napisati program koji učitava prirodan broj `n`, a zatim koristeći funkciju `prost` štampa sve proste brojeve manje od datog broja `n`.

# Zadatak 5

Napisati program koji učitava paran prirodan broj n veći od 2, a zatim koristeći funkciju prost proverava hipotezu Goldbaha za dati broj n. Prema hipotezi, svaki paran broj veći od dva može se predstaviti zbirom dva prosta broja.

# Zadatak 6

Napisati program koji korisniku dozvoljava da igra popularnu igru pogađanja, takozvanu "Igru vešala". Korisnik pogađa zadatu reč (koja je zadata u kodu programa) tako što pogađa slova u zadatoj reči. Svaki put kada slovo postoji, ono će se otkriti u ispisu (neotkriveno slovo je _). Kada slovo ne postoji u reči, nacrtaće se deo čiča-gliše (redom, `|`, `0`, `/`, `|`, `\`, `/` i `\`). Korisnik može da promaši slovo ukupno 7 puta. Kada korisnik pogodi reč, ispisati na terminalu: "`Cestitke, rec je pogodjena!`", dok u slučaju maksimalnog broja grešaka: "`Vise srece drugi put`". U oba slučaja ispisati zadatu reč. Podeliti rešenje programa u funkcije.

Na sledeća dva slajda nalazi se primer rada programa u trenutku kada je korisnik iskoristio sve šanse da pogodi reč.

```
Rec: _ _ _ _ _ t _ _ _ _ a
Unesite slovo: o

Preostalo zivota: 1
____
|   |
|   O
|  /|\
|  /
|
```

```
Rec: o _ o _ _ t _ _ _ _ a
Unesite slovo: k

Preostalo zivota: 0
____
|   |
|   O
|  /|\
|  / \
|

Zadata rec: onomatopeja
Vise srece drugi put!
```