# C/C++
# PROGRAMSKI JEZIK C

## Organizacija programa

# Lokalne promenljive

- Promenljiva deklarisana u telu funkcije je lokalna promenljiva funkcije

```
int sum_digits(int n) {
    int sum=0;
    while(n>0) {
        sum+=n%10;
        n/=10;
    }
}
```

  - Ovako deklarisanu promenljivu zovemo automatska lokalna promenljiva

# Automatske lokalne promenljive

- Automatsku lokalnu promenljivu karakteriše sledeće:
  - Automatska trajnost pridruženog memorijskog prostora (automatic storage duartion)
    - Automatsko alociranje memorijskog prostora za datu promenljivu u trenutku poziva funkcije u kojoj je deklarisana
    - Alocirani memorijski prostor se dalji automatski oslobađa kada se prekine izvršavanje funkcije
    - Ponovni poziv iste funkcije ne garantuje da će se isti memorijski prostor alocirati za datu lokalnu promenljivu (vrednost promenljive iz prethodnog poziva je nedostupna)
  - Blok oblast važenja (block scope)
    - Pod oblašću važenja promenljive podrazumeva se deo programskog teksta u kome se promenljivoj moze pristupiti (u kome je promenljiva vidljiva)
      - Lokalna promenljiva funkcije ima blok oblast važenja (vidljiva je samo u bloku u kome je deklarisana) i to od mesta deklaracije do kraja bloka u kome je deklarisana
      - Na ovaj način ista imena se mogu bezbedno koristiti za promenljive u različitim funkcijama
      - U standardu C99 deklaracije promenljiih se ne moraju navoditi pre ostalih naredbi bloka tako da oblast važenja promenljive moze biti značajno mala
- Parametri funkcije imaju iste osobine kao automatske lokalne promenljive
  - Jedina azlika je u tome što se one automatski inicijalizuju vrednostima odgovarajućih argumenata pri pozivu funkcije

# Statičke lokalne promenljive

- Ako se u deklaraciji promenljive navede ključna reč static
  - promenljiva ima statičku trajnost memorijskog prostora (static storage duration)
  - Statičkoj promenljivoj je pridružen uvek isti memorijski prostor tokom izvršavanja programa

```
void f(oid) {
    static int i;  /* staticka lokalna promenljiva */
    …
}
```

- U svakom narednom pozivu funkcije vrednost statičke promenljive iz prethodnog poziva se ne gubi
- Statička lokalna promenljiva ima blok oblast važenja kao i automatska lokalna promenljiva

# Eksterne promenljive

- Jedan način razmene informacija izmedju funkcije i ostalih delova programa je korišćenje parametara
- Sa druge strane ovo se moze izvesti  korišćenjem eksternih promenljiih
  - Eksterna (globalna) promenljiva je promenljiva deklarisana van tela funkcije i karakteriše je sledeće
  - Eksterne promenljive kao i statičke lokalne promenljive su pridružene uvek istom memorijskom prostoru tokom izvršavanja programa (statička trajnost memorijskog prostora)
  - Datotečna oblast važenja
    - Eksterna promenljiva je vidljiva od mesta na kome je deklarisana pa do kraja datoteke u kojoj je deklaracija izvršena
    - Ovo praktično znači da njenu vrednost može koristiti odnosno menjati bilo koja funkcija koja sledi posle njene deklaracije

# Eksterne promenljive

- Eksterne promenljive se najčešće koriste kada
  - više funkcija koristi isti podatak
  - manji broj funkcija koristi dosta podataka
- Mane su sledeće
  - Ako se promeni globalna promenljiva (npr. njen tip) odgovarajuće izmene treba izvršiti u svim funkcijama koje koriste datu globalnu promenljivu
  - Komplikovanije otkrivanje grešaka u programu
    - bilo koja funkcija je mogla da postavi nekorektnu vrednost globalne promenljive
  - Otežano korišćenje funkcije koja koristi globalne promenljive u drugim programima

# Blok

```
{
   deklaracije
   naredba
}
```

- Blok je niz naredbi unutar vitičastih zagrada
  - Telo funkcije je blok
  - Blok naredba se tretira kao jedna naredba i može biti elemenat nekog drugog bloka (blokovi se mogu ugnježdavati)
  - Blok može imati i deklaracije koje se navode pre naredbi
- Promenljiva deklarisana u bloku je automatska lokalna promenljia
  - Ima automatsku trajnost memorijskog prostora
    - Memoijski prostor se alocira ulaskom u blok u kome je promenljiva deklarisana, a oslobadja se izlaskom iz bloka u kome je deklarisana
  - Ima blok oblast važenja

# Blok

```
If (i>j) {
    int temp=i;
    i=j;
    j=temp;
}
```

promenljiva temp je uvedena u trenutku kada je bila potrebna

- Prednost korišćenja blokova se ogleda u sledećem:
  - Nema potrebe sve promenljive koje će se koristiti deklarisati na početku bloka koji je telo funkcije, ako one imaju privremeni karakter
  - Redukuje se mogućnost konflikta imena
    - Ime temp se može koristiti u istoj funkciji u druge svrhe
    - Novouvedena promenljiva temp u datom bloku privremeno skriva ostale objekte sa istim imenom

# Oblast važenja

```
int i;
void f(int i) {
    i=1;
}
void g(void) {
    int i=2;
    if(i>0) {
        int i;
        i=3;
    }
    i=4;
}
void h(void) {
    i=5;
}
```

- U C programu neki identifikator može imati razlicita značenja u pojedinim delovima programa
  - Pravila oblasti važenja omogućavaju određivanje značenja pojedinog identifikatora u nekom delu programa
  - Poslednja deklaracija nekog imena privremeno skriva ranije značenje nekog imena u pripadajucem bloku
  - Na kraju bloka staro značenje se ponovo aktivira

# Organizacija C programa

- Elementi C programa
  - Preprocesorske directive
  - Definicje tipova
  - Deklaracije eksternih promenljivih
  - Prototipovi funkcija
  - Definicije funkcija
- Kako ih ukomponovati?
  - Ograničenja su sledeća
    - preprocesorska direktiva nema efekta sve do linije u kojoj se nalazi
    - ime tipa se ne može koristiti pre nego što se definise
    - promenljiva se ne može kosristiti pre nego što se deklariše
    - funkcija se može koristiti i pre nego što se definiše ili deklariše, ali se ne preporučuje (standard C99 to i zahteva)

# Organizacija C programa

- U slučaju da se C program smešta u jednu izvornu datoteku
  - Preporučuje se sledeća organizacija

#include direktive
#define direktive
Definicije tipova
Deklaracije eksternih promenljivih
Prototipovi funkcija sem main funkcije
Definicja main funkcije
Definicije ostalih funkcija

# Zadaci za vežbu

1. Izračunavanje vrednosti izraza zadatog u inverznoj poljskoj notaciji
   i. operand ide na stek
   ii. operator izvlači iz steka operande izračunava se vrednost i pakuje na vrh steka
   iii. rezultat je na vrhu steka
2. Provera da li je izraz ispravno zagrađen (, [, {, ), ], }
   i. ako je otorena zagrada upisuje se na stek
   ii. ako je neka zatvorena zagrada
      - mora se upariti sa odgovarajućom na vrhu steka
   iii. ako je neki drugi karakter ignoriše se