27. POKAZIVAČI

Pokazivač je promenljiva koja sadrži memorijsku adresu promenljive, objekta ili funkcije. To je izvedeni tip podatka koji se intenzivno koristi u programskom jeziku C, vrlo često kada je to jedini način rešavanja problema, a ponekad i zbog toga što pokazivači vode kompaktnijem i efikasnijem kodu od bilo kog drugog načina pisanja koda. Pokazivači se mogu iskoristiti da bi se postigla jasnoća i jednostavnost koda.

Veza između pokazivača i memorijskog polja - Tipičan računar ima memorijsko polje koje je niz adresiranih memorijskih ćelija kojima se može manipulisati pojedinačno i grupno. Pokazivač je skup memorijskih ćelija (često dve ili četiri) koja sadrži adresu polja.

Deklaracija pokazivača obuhvata **upotrebu operatora posrednog pristupa *.** Unarni operator * je indirektni ili dereferentni operator. Kad se primeni na pokazivač, on pristupa objektu na koji pokazivač pokazuje.

Opšti oblik:

```
tip *ime_pokazivaca;
```

gde je:

- Komponenta `tip` specificira tip podatka na koji se pokazuje,
- Komponenta `ime_pokazivaca` je identifikator koji identifikuje pokazivačku promenljivu.

Primer pokazivača na celobrojnu promenljivu:

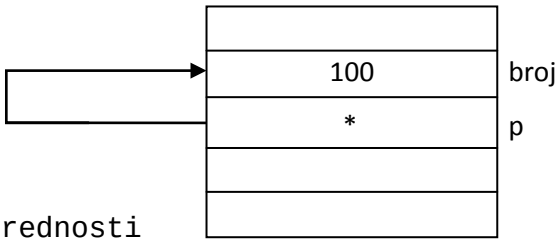
```
int broj; //ceo broj
int *p; //pokazivac na ceo broj
```

Druga deklaracija kaže da je `p` pokazivač na objekat tipa `int` i promenljiva `p` nema definisanu vrednost. Da bi se pokazivačka promenljiva uspešno koristila mora da sadrži ispravnu memorijsku adresu. **Dodelu vrednosti promenljive mora da obavi programer uz pomoć adresnog operatora &**. Sledeća naredba uzima adresu promenljive `broj` i dodeljuje je pokazivačkoj promenljivoj `p`:

```
p=&broj;
```

Pošto je `p` pokazivač na objekat tipa `int`, mora sadržati adresu celog broja, koju daje izraz `&broj`. Posle ove dodele vrednosti, p efektivno pokazuje na broj. Ilustracija primera je data na slici:

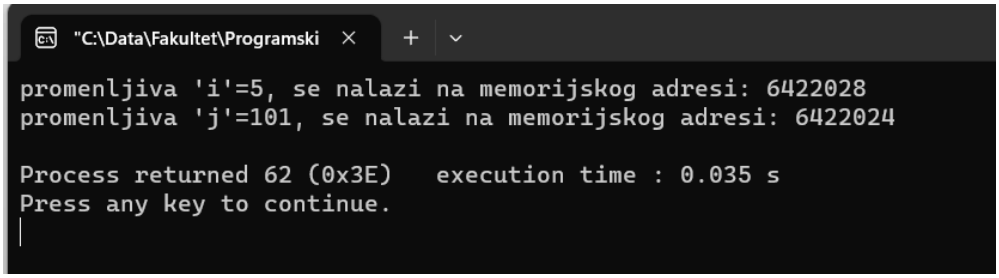
```
int broj;
int *p;

p=&broj;

*p=100;
//posredna dodela vrednosti
broj=100;
//neposredna dodela vrednosti
```



Primer:

```
void main()
{
int i,j;
int *p1,*p2;
i = 5;
j = 101;
p1 = &i;
printf("promenljiva 'i'=%d, se nalazi na memorijskog adresi: %d\n",
i, &i);
p2 = &j;
printf("promenljiva 'j'=%d, se nalazi na memorijskog adresi: %d\n",
j, &j);
}
```

Rezultat izvršavanja:



Nad pokazivačkim promenljivama dozvoljeno je primeniti aritmetičke operatore u izrazima koji uključuju sledeće operacije:

- dodavanje celog broja na pokazivač;
- oduzimanje celog broja od pokazivača;
- traženje razlike dva pokazivača;
- ispitivanje jednakosti dva pokazivača.

Za operacije koje uključuju razliku i poređenje pokazivača, pokazivači moraju da se odnose na isti tip objekta.

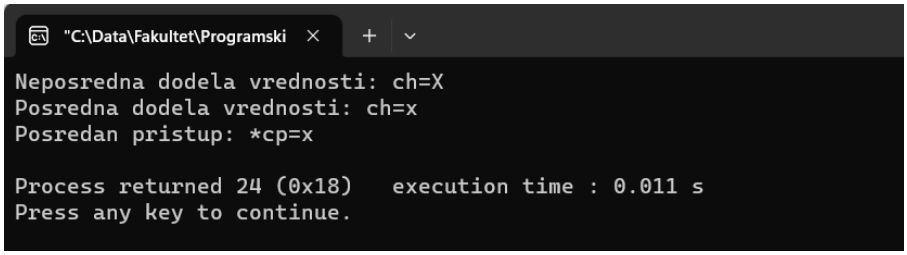
Nije dozvoljeno množenje, sabiranje, deljenje pokazivačke promenljive.

Veličina pokazivača srazmerna je tipu objekta na koji pokazivač pokazuje.

Primer programa sa pokazivačima koji koristi znakove:

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
	char ch;
	char *cp;
	cp=&ch;
	ch='X'; //neposredna dodela vrednosti
	printf("Neposredna dodela vrednosti: ch=%c\n", ch);
	*cp='x'; //posredna dodela vrednosti
	printf("Posredna dodela vrednosti: ch=%c\n", ch);
	printf("Posredan pristup: *cp=%c\n", *cp);
}
```

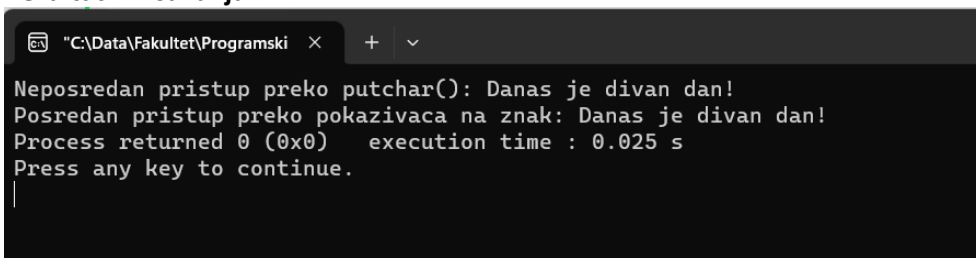
Rezultat izvršavanja:



Primer programa sa pokazivačima koji koristi znakovne nizove:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void main()
{
	int i;
	char *cp;
	char zniz[]="Danas je divan dan!";
	printf("Neposredan pristup preko putchar(): ");
	for (i=0; zniz[i]!='\0';++i)
		putchar(zniz[i]);
	printf("\nPosredan pristup preko pokazivaca na znak: ");
	for (cp=zniz; *cp!='\0';++cp)
		putchar(*cp);
}
```

Rezultat izvršavanja:

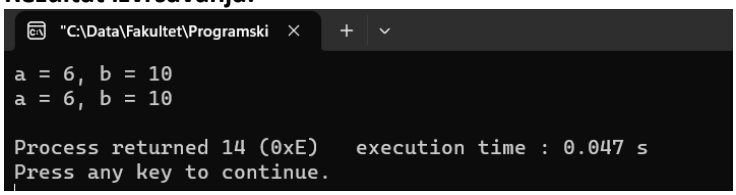


Funkcije, argumenti funkcija i pokazivači - Već je u ranijem tekstu materijala napomenuto da C jezik prenosi argumente funkcija po vrednosti, tj. ne omogućuje direktnu promenu vrednosti argumenata od strane pozvane funkcije.

Primer - funkcija za razmenu vrednosti x, y (zameni) i njen poziv zameni(a, b) neće razmeniti vrednosti argumenata a i b (zbog prenosa argumenata po vrednosti) :

```
#include <stdio.h>
void zameni(int x, int y)
{
	int temp;
	temp =x;
	x = y;
	y = temp;
}
void main()
{
	int a = 6, b = 10;
	printf("a = %d, b = %d\n", a, b);
	zameni (a, b);
	printf("a = %d, b = %d\n", a, b);
}
```

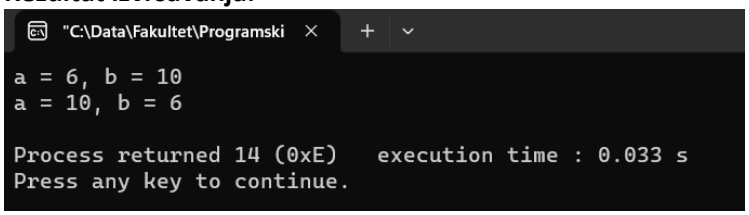
Rezultat izvršavanja:



Željeni rezultat zamene vrednosti promenljivama, u primeru, se postiže prenosom pokazivača na argumente umesto samih argumenata, tj. pozivom funkcije zameni(&a, &b) definisane sledećom definicijom funkcije (parametri su pokazivači na celobrojne promenljive):

```
#include <stdio.h>
void zameni(int *px, int *py)
{
	int temp;
	temp = *px;
	*px = *py;
	*py = temp;
}
void main()
{
	int a = 6, b = 10;
	printf("a = %d, b = %d\n", a, b);
	zameni (&a, &b);
	printf("a = %d, b = %d\n", a, b);
}
```

Rezultat izvršavanja:



28. MAKROI I FUNKCIJE SA PROMENLJIVIM BROJEM ARGUMENATA

Makro je deo koda u programu koji je zamenjen vrednošću. Makro je definisan direktivom #define. Kad god kompajler naiđe na ime makroa, on to ime zamenjuje definicijom makroa.

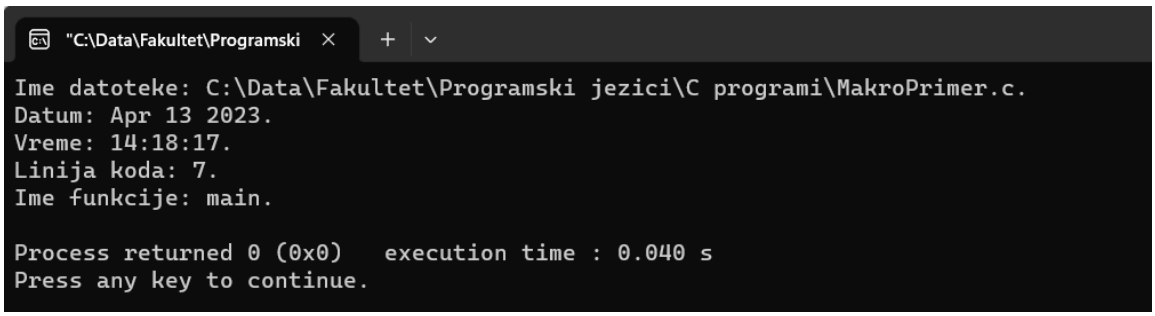
Primer:
```
#include <stdio.h>
#define MESEC 12
#define ISPLATA 2
void main()
{
	int i;
	float zarada[MESEC][ISPLATA][3];
	float ukupnazarada[MESEC];
	…
	for(i = 0; i < MESEC; i++)
	{
	…
	}
…
}
```

Predefinisani makroi – U C jeziku, standardom su definisani određeni predefinisani makroi. Neki od njih su:

- DATE - Datum preprocesiranja
- TIME - Vreme preprocesiranja
- FILE - Ime datoteke s izvornim kodom
- LINE - Trenutna linija koda
- func - Ime funkcije.

Primer programa sa makroima:
```
#include <stdio.h>
int main(void)
{
	printf("Ime datoteke: %s.\n",__FILE__);
	printf("Datum: %s.\n",__DATE__);
	printf("Vreme: %s.\n",__TIME__);
	printf("Linija koda: %d.\n",__LINE__);
	printf("Ime funkcije: %s.\n",__func__);
	return 0;
}
```

Rezultat izvršavanja:



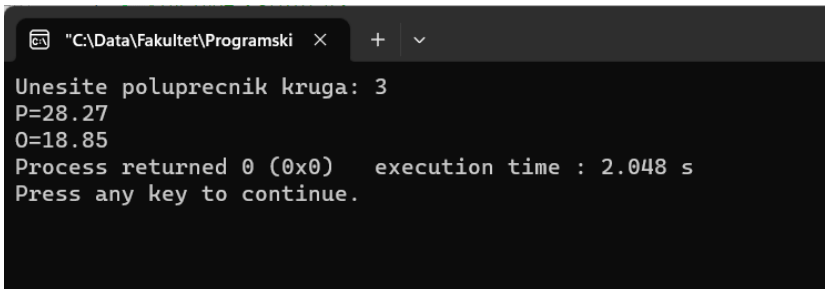
Makroi se često koriste za ispis poruka o greškama, kao u sledećem primeru:

If (x!=y) printf("Greska: linija %d, datoteka %s\n",__LINE__, __FILE__);

Primer programa za izračunavanje površine kruga:

```
#include <stdio.h>
#define PI 3.1415
int main()
{
    float r, p, o;
    printf("Unesite poluprecnik kruga: ");
    scanf("%f", &r);
    // upotreba makroa PI
    p = PI*r*r;
    o = PI*2*r;
    printf("P=%.2f",p);
    printf("\nO=%.2f",o);
    return 0;
}
```

Rezultat izvršavanja:



Funkcije sa promenljivim brojem argumenata

Delimična lista parametara mora biti završena elipsisom, zarezom praćenom tri tačke (, ...), da bi se naznačilo da u funkciji može biti i više argumenata koji joj se prosleđuju, ali se o njima ne daje više informacija. Provera tipa se ne vrši na takvim argumentima. Najmanje jedan parametar mora da prethodi zapisu elipsise, a elipsis mora da bude poslednja oznaka na listi parametara. Bez zapisa elipsis, ponašanje funkcije je nedefinisano, ako prima parametre pored onih koji su deklarisani na listi parametara.

Da biste pozvali funkciju sa promenljivim brojem argumenata, jednostavno treba navesti bilo koji broj argumenata u pozivu funkcije. Primer je funkcija printf() iz C stdio biblioteke. Poziv funkcije mora da sadrži jedan argument za svako ime tipa deklarisano u listi parametara ili listi tipova argumenata.

Heder STDARG.H datoteka sadrži makroe u ANSI stilu za pristup argumentima funkcija koje uzimaju promenljiv broj argumenata.

Primer deklaracije funkcije koja poziva promenljivi broj argumenata:

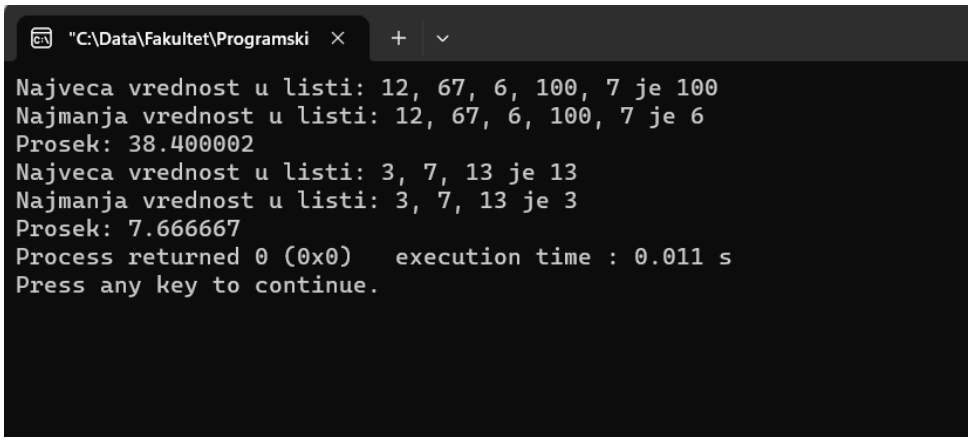
```
int average(int first, ...);
```

Primer upotrebe funkcija koje imaju promenljivi broj argumenata – za izračunavanje proseka elemenata liste celih brojeva, minimalnog i maksimalnog elementa:

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

int max(int brarg, ...)
{
  int i;
  int max, a;
  va_list ap;
  va_start(ap, brarg);
  max = va_arg(ap, int);
  for(i = 2; i <= brarg; i++)
    {
    if((a = va_arg(ap, int))>max)
      max = a;
    }
  va_end(ap);
  return max;
}

int min(int brarg, ...)
{
  int i;
  int min, a;
  va_list ap;
  va_start(ap, brarg);
  min = va_arg(ap, int);
  for(i = 2; i <= brarg; i++)
    {
    if((a = va_arg(ap, int))< min)
      min = a;
    }
  va_end(ap);
  return min;
}

float prosek(int brarg, ...)
{
  int i, a, suma;
  float prosek;
  va_list ap;
  va_start(ap, brarg);
  suma = 0;
  for(i = 1; i <= brarg; i++)
    {
    a = va_arg(ap, int);
    suma = suma + a;
    }
  va_end(ap);
  prosek=(float)suma/brarg;
  return prosek;
}

int main()
{
   int brelem = 5;
   printf("Najveca vrednost u listi: 12, 67, 6, 100, 7 je %d", max(brelem,
12, 67, 6, 100, 7));
   printf("\nNajmanja vrednost u listi: 12, 67, 6, 100, 7 je %d",
min(brelem, 12, 67, 6, 100, 7));
   printf("\nProsek: %f", prosek(brelem, 12, 67, 6, 100, 7));
   brelem = 3;
```

```
   printf("\nNajveca vrednost u listi: 3, 7, 13 je %d", max(brelem, 3, 7,
13));
    printf("\nNajmanja vrednost u listi: 3, 7, 13 je %d", min(brelem, 3, 7,
13));
   printf("\nProsek: %f", prosek(brelem, 3, 7, 13));
   return 0;
}
```

Rezultat izvršavanja:



29. DEFINICIJA DATOTEKE, PODELA DATOTEKA

Datoteka je osnovna logička jedinica za zapis podataka preko jedinica masovne memorije (npr. magnetne trake, čvrsti - hard diskovi, CD i DVD diskovi, USB memorijski drajv). Za unos i izlaz velikog obima podataka u/iz programa se ne koristi ulaz sa tastature već iz datoteke.

Podela datoteka prema organizaciji datoteke:

- sekvencijalne – na mediju za datoteku podaci su fizički smešteni jedan iza drugog (slogovi) u redosledu u kojem je datoteka kreirana i u istom redosledu se mogu obrađivati.
- indeksne - osim slogova sa podacima, postoje indeksi koji omogućavaju brži pristup pojedinom slogu iz datoteke.
- relativne – sadrže ključeve koji zahtevaju konvertovanje podatka ključa sloga u fizičku adresu u memoriji.

U C jeziku postoje samo sekvencijalne datoteke koje su po strukturi niz bajtova, bez podele na zapise (koristi se i termin tok).

Prema načinu smeštanja podataka, postoje dve vrste datoteka:

- tekstualne datoteke,
- binarne datoteke.

Tekstualne datoteke sastoje se od niza znakova koji je znakovima za prelazak u novi red ('\n') podeljen u redove. Podela tekstualnih datoteka u redove je logička, a ne fizička organizacija. Tekstualne datoteke su samo dugački niz znakova. Sadržaj čine podaci koji odgovaraju karakterima nekog kodnog sistema: ASCII, EBCDIC, UTF-8, UTF-16.

Binarne datoteke sastoje se od niza bajtova čiji je sadržaj verna slika načina predstavljanja podataka u memoriji. Sadržaj čine podaci predstavljeni u binarnom obliku.

Pri radu sa datotekama postoje sledeće radnje:

- Kreiranje datoteke
- Otvaranje datoteke,
- Pristup datoteci (čitanje, pisanje, pozicioniranje),
- Ispitivanje statusa datoteke,
- Zatvaranje datoteke.

Otvaranje datoteke služi za uspostavljanje veze sa fizičkim fajlom datoteke. Tom prilikom se u memoriji kreira određena struktura podataka koja omogućava efikasan pristup datoteci u toku rada. Ta struktura sadrži određene podatke o trenutnom stanju i statusu datoteke. **Pristupom datoteci** realizuje se prenos podataka u datoteku ili iz datoteke sa ili bez konverzije u toku prenosa. **Ispitivanjem statusa datoteke** mogu da se dobiju informacije o eventualnim greškama koje su nastale u toku prenosa podataka. **Zatvaranje datoteke** je završna radnja, kojom se raskida veza sa fizičkom datotekom. Tom prilikom poništavaju se strukture podataka stvorene prilikom otvaranja datoteke.

Navedene radnje mogu se ostvariti samo pod određenim uslovima:

- Funkcije za rad sa datotekama tretiraju datoteku kao sekvencijalni niz karaktera.
- Datotekama se pristupa pomoću pokazivača na sistemsku strukturu u oznaci **FILE**. Struktura **FILE** je definisana u datoteci zaglavlja "**stdio.h"**, a sadrži članove koji opisuju tekuće stanje datoteke.
- U datoteci zaglavlja su definisane dve simboličke konstante:

  - Konstanta **EOF** (End of file) koja definiše oznaku kraja datoteke i prema inicijalizaciji u datoteci zaglavlja ima vrednost **-1**.
  - Konstanta **NULL** koja označava neuspešno izvršavanje nekih funkcija za upravljanje datotekama čija je vrednost **0** definisana u datoteci zaglavlja "**stdio.h"**.
- Standardizovane funkcije koriste memorijski bafer, odakle čitaju ili upisuju podatke. Kad se bafer napuni njegov sadržaj se prebacuje u datoteku. Kad se bafer isprazni, njegov sadržaj se popunjava iz datoteke. Korišćenjem memorijskog bafera se smanjuje broj pristupa datoteci. Standardizovane funkcije za rad sa datotekama su: fopen(), fgetc(), fputc(), fprintf(), fscanf(), fgets(), fputs(), feof(), fclose(), ftell(), fseek(), rewind(), fread() i fwrite().

Za svaku datoteku koja se koristi u programu mora da postoji pokazivač na podatak tipa **FILE**. Neka je to pokazivač pod nazivom **datoteka**, pri tome deklaracija je:

```
FILE *datoteka;
```

Na osnovu deklaracije pokazivača **datoteka** saopštava se kompajleru da je **datoteka** pokazivač koji može da sadrži adresu podatka tipa **FILE**.

Biblioteka "**Stdio.h"** sadrži sledeće funkcije za rad sa datotekama:

`fopen()`	Otvaranje datoteke
`fclose()`	Zatvaranje otvorene datoteke
`getw()`	Čitanje celog broja iz datoteke
`putw()`	Upisivanje celog broja u datoteku
`fgetc()`	Čitanje karaktera iz datoteke
`putc()`	Upisivanje karaktera u datoteku
`fputc()`	Upisivanje karaktera u datoteku
`fgets()`	Čitanje stringa iz datoteke, cele linije teksta
`fputs()`	Upisivanje stringa u datoteku
`feof()`	Pronalazak kraja datoteke
`fgetchar()`	Čitanje karaktera sa tastature
`fprintf()`	Upisivanje formatiranog teksta u datoteku
`fscanf()`	Čitanje formatiranog teksta iz datoteke
`fputchar()`	Ispisivanje karaktera sa tastature
`fseek()`	Pomeranje pokazivača datoteke na poziciju
`ftell()`	Trenutna pozicija pokazivača na datoteku
`rewind()`	Pomeranje pokazivača datoteke na početak

`sprint()`	Ispisivanje formatiranog izlaza u string
`sscanf()`	Reads formatted input from a string
`remove()`	Brisanje datoteke
`fflush()`	Brisanje sadržaja memorijskog bafera za rad sa datotekom
`SEEK_SET`	Pomeranje pokazivača datoteke na početak
`SEEK_CUR`	Pomeranje pokazivača datoteke na poziciju
`SEEK_END`	Pomeranje pokazivača datoteke na kraj