Uvod u programiranje

- predavanja -

listopad 2020.

Osnove programskog jezika C

- 1. dio -

Temeljni elementi jezika C

Struktura C programa

Struktura C programa

- C program se sastoji od deklaracija i definicija funkcija (imenovanih blokova), deklaracija i definicija varijabli i direktiva pretprocesoru
  - razlika između pojmova *deklaracija* i *definicija* bit će objašnjena kasnije. Za sada će se koristiti samo pojam *definicija*.
- složena naredba ili blok (imenovani ili neimenovani) može obuhvaćati deklaracije i definicije varijabli, ostale naredbe (*statement*) i neimenovane blokove
- svaka naredba mora završavati znakom ;
  - terminator: oznaka da na tom mjestu naredba završava (i može se, ako treba, početi pisati sljedeća)
  - blok NE završava znakom ; tj. iza znaka } ne stavlja se ;

Primjer

```
#include <stdio.h>                                  direktiva pretprocesoru

int main(void) {                                    imenovani blok (funkcija)
   int n, rez;                                      definicija varijabli
   scanf("%d", &n);                                 naredba (statement)

   // izracunaj apsolutnu vrijednost
   if (n < 0) {                                     početak neimenovanog bloka
      rez = -1 * n;
   }                                                kraj neimenovanog bloka
   else {                                           početak neimenovanog bloka
      rez = n;
   }                                                kraj neimenovanog bloka

   printf("Ulaz: %d Rezultat: %d", n, rez);
   return 0;
}
```

C je jezik slobodnog formata

- standard ne propisuje stil pisanja
  - mjesto početka naredbe u retku je proizvoljno, umetnute praznine nemaju specijalno značenje
  - dopušteno je napisati više naredbi u istom retku ili jednu naredbu u više redaka

```
...
int n, rez; scanf("%d", &n);
...
printf("Ulaz: %d Rezultat: %d"
     , n
     , rez
     );
...
```

- međutim, poželjno je uredno pisanje, odnosno umetanje praznina i praznih redova na odgovarajućim mjestima

Primjer

- Što nije u redu s ovim programom?

```
#include <stdio.h>
int main(

void
) { int n
, rez
; scanf(
"%d", &n);  // izracunaj apsolutnu vrijednost
                                    if ( n < 0 )
{rez = -1 * n
;}
else                                          {rez =
n; }printf("Ulaz: %d Rezultat: %d", n
                , rez); return 0;}
```

Temeljni elementi jezika C

Ključne riječi

Uporaba velikih i malih slova

Ključne riječi

- ključne riječi su predefinirani identifikatori koji za prevodioca imaju posebno značenje. ISO/IEC 9899:2011 (C11) propisuje sljedeće 44 ključne riječi:

`auto`	`extern`	`short`	`while`
`break`	`float`	`signed`	`_Alignas`
`case`	`for`	`sizeof`	`_Alignof`
`char`	`goto`	`static`	`_Atomic`
`const`	`if`	`struct`	`_Bool`
`continue`	`inline`	`switch`	`_Complex`
`default`	`int`	`typedef`	`_Generic`
`do`	`long`	`union`	`_Imaginary`
`double`	`register`	`unsigned`	`_Noreturn`
`else`	`restrict`	`void`	`_Static_assert`
`enum`	`return`	`volatile`	`_Thread_local`

Uporaba velikih i malih slova

- C prevodilac razlikuje velika i mala slova
  - za imena varijabli, ključne riječi i ostale identifikatore mora se koristiti propisani oblik slova (veliko/malo)

```
#Include <STDIO.h>
INT Main(Void) {
   Return 0;
}
```

- svaka riječ u prethodnom programu je napisana neispravno. Prevodilac u tom programu neće moći prepoznati:
  - pretprocesorsku naredbu include
  - datoteku stdio.h
  - ključne riječi int, void, return
  - funkciju main

Temeljni elementi jezika C

Komentari

Komentari

- komentari nemaju utjecaj na izvršavanje programa. Mogu se ugraditi na dva načina, na bilo kojem mjestu u izvornom kôdu:
  - komentar koji započinje s dvije kose crte proteže se do kraja retka

```
// podaci o studentu
int godRod;     // godina rodjenja
int godUpis;    // godina upisa na FER
```

  - komentar koji započinje dvoznakom /* i završava dvoznakom */ može se protezati i kroz više redaka
  - komentari ovog oblika ne smiju se ugnježđivati

```
/* funkcija izracunava najveci zajednicki djelitelj za
   zadane cijele pozitivne brojeve m i n
*/
int najveciDjelitelj(int m, int n) {
   ...
```

Temeljni elementi jezika C

Funkcija *main*

Glavna funkcija (*main*)

```
int main(void) {
   ...
   return 0;
}
```

- glavna funkcija predstavlja mjesto na kojem počinje izvršavanje C programa
  - svaki program mora sadržavati točno jednu funkciju *main*
  - *int* ispred *main* znači da funkcija u pozivajući program (u ovom slučaju operacijskom sustavu) vraća cijeli broj (*integer*). S time povezana naredba *return* u pozivajući program vraća cijeli broj
    - za sada: operacijskom sustavu uvijek vratiti cijeli broj nula, kao što je prikazano u primjeru
  - *void* znači da funkcija *main* ne prima niti jedan argument
  - početak i kraj bloka naredbi, koji predstavlja tijelo funkcije, označeni su vitičastim zagradama *{* i *}*

Temeljni elementi jezika C

Varijable i konstante

Varijable

- općenito: promjenljiv podatak (lat. *variabilis* - promjenljiv)
- u programiranju: prostor u memoriji računala, unaprijed zadane i nepromjenjive veličine, kojem je pri definiciji dodijeljeno ime i tip i čiji se sadržaj može mijenjati, npr. naredbom pridruživanja ili učitavanjem vrijednosti s tipkovnice.

```
...
int main(void) {
   int m, n;
   ...
   ...
   n = 128;
   scanf("%d", &m);
   ...
   m = 1000;
   scanf("%d", &n);
   ...
```

sadržaj varijabli ovog trenutka nije siguran, varijable sadrže "garbage value", "smeće". Kažemo: varijable još nisu inicijalizirane (*uninitialized*)

sadržaj varijable n postavljen je pridruživanjem

sadržaj varijable m postavljen je učitavanjem vrijednosti s tipkovnice

sadržaj varijabli može se ponovo promijeniti

Definicija varijable

```
int n, rez;
```

- prethodnom naredbom definirane su dvije cjelobrojne varijable u koje je moguće pohranjivati isključivo cijele brojeve. Kažemo: varijable su tipa int

```
float x, y, z;
float v;
```

- prethodnim naredbama definirane su realne varijable x, y, z i v u koje je moguće pohranjivati isključivo realne brojeve. Kažemo: varijable su tipa float (naziv je izveden iz pojma *floating point*)
- za sada će se koristiti samo tipovi int i float. Ostali podržani tipovi podataka bit će objašnjeni kasnije
- varijabla se može definirati na bilo kojem mjestu u bloku, ali obavezno prije nego se prvi puta koristi

Definicija varijable uz inicijalizaciju

```
int k, m, n;
m = 3;
n = 3;
```

k, m, n ovog trenutka sadrže "smeće"

m sadrži 3

n sadrži 3

k trenutačno sadrži "smeće"

- Naredba za definiciju varijable može sadržavati tzv. *inicijalizator* kojim se već pri definiciji postavlja početna vrijednost varijable
  - inicijalna vrijednost mora se navesti pojedinačno za svaku varijablu koju se želi inicijalizirati

```
...
int main(void) {
   int k, m = 3, n = 5;
   ...
```

m sadrži 3; n sadrži 5

k trenutačno sadrži "smeće"

Imena varijabli

- imena varijabli (i svi drugi identifikatori, npr. imena funkcija) sastavljena su od slova, znamenki i znakova podcrtavanja _
  - ime ne smije započeti:
    - znamenkom
    - s dva znaka podcrtavanja
    - znakom podcrtavanja i velikim slovom
  - ime ne smije biti jednako niti jednoj ključnoj riječi
  - prema konvenciji, za tvorbu imena varijabli prvenstveno se koriste mala slova, uz eventualni dodatak nekoliko velikih slova (vidjeti kasnije tzv. *camelCase*)
  - primjeri neispravnih imena varijabli

```
novi+datum          x1/1          x$          rezultat!
float               int           return      void
__suma              _Produkt      1.suma      1produkt
```

Imena varijabli

- duljina imena je proizvoljna, ali treba voditi računa o sljedećem:
  - kod nekih prevodioca moguća su ograničenja u broju značajnih znakova. Standard zahtijeva samo to da najmanje 31 prvih znakova imena bude značajno. Stoga je moguće da neki prevodioci neće moći međusobno razlikovati sljedeća imena varijabli:
    - prosjecna_ocjena_na_predmetu_upro_2018_godine
    - prosjecna_ocjena_na_predmetu_upro_2019_godine
  - preduga ili prekratka imena smanjuju preglednost ili otežavaju pisanje programa. Korištenje imena koje odražava značenje varijable bitno unapređuju jasnoću programa
    - umjesto predugih (gore) ili prekratkih imena (npr. p1, p2) bolje je:

      upro_prosj_2018, upro_prosj_2019 (*snake_case* oblik) ili

      uproProsj2018, uproProsj2019 (*camelCase* oblik)

Konstante

- slično varijablama, konstante također imaju svoje tipove. Tip konstante ovisi o formi u kojoj je napisana

```
float x, y;
int m, n;
x = 1.f;        vrijednost realne konstante pridružuju se realnoj varijabli
y = 2;          vrijednost cjelobrojne konstante pridružuju se realnoj varijabli
                (prije pridruživanja obavlja se konverzija)
m = 3;          vrijednost cjelobrojne konstante pridružuje se cjelobrojnoj varijabli
n = 3.5f;       vrijednost realne konstante pridružuju se cjelobrojnoj varijabli
                (prije pridruživanja obavlja se konverzija)
```

- zašto se realnim konstantama na kraj dodaje slovo f bit će objašnjeno u predavanjima o tipovima i načinima pohrane podataka

Temeljni elementi jezika C

Direktive pretprocesoru

Direktive pretprocesoru - *#include*

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

- uputa (direktiva) pretprocesoru: u program prije prevođenja uključiti sadržaj datoteke <stdio.h>
  - <stdio.h> sadrži deklaracije i definicije koje su potrebne da bi program na ispravan način mogao koristiti, između ostalog, funkcije printf i scanf (funkcije za čitanje i pisanje)
  - zaključak: na početak svakog programa koji će koristi funkcije scanf ili printf treba ugraditi direktivu #include <stdio.h>

Primjer

- `gcc -E prog1.c > prog1.i`

izvorni kôd **prog1.c**

```
#include <stdio.h>

int main(void) {
   int n, rez;
   scanf("%d", &n);

   // izracunaj apsolutnu vrijednost
   if (n < 0) {
      rez = -1 * n;
   } else {
      rez = n;
   }

   printf("Ulaz: %d Rezultat: %d", n, rez);

...
```

pretprocesirani izvorni kôd **prog1.i**

```
...
__attribute__((__cdecl__))
__attribute__((__nothrow__))
int printf (const char *, ...);
...
__attribute__((__cdecl__))
__attribute__((__nothrow__))
int scanf (const char *, ...);
...

int main(void) {
   int n, rez;
   scanf("%d", &n);

   if (n < 0) {
      rez = -1 * n;
   } else {

...
```

Direktive pretprocesoru - *#define*

```
#define PI 3.14159f
```

- uputa (direktiva) pretprocesoru: tijekom faze pretprocesiranja, svaku pojavu riječi PI u izvornom kodu zamijeniti s 3.14159f
  - time je definirana *simbolička konstanta*
  - prema konvenciji, imena konstanti pišu se velikim slovima

```
#include <stdio.h>
#define PI 3.14159f

int main(void) {
   float r;   // polumjer kruga
   scanf("%f", &r);
   printf("Povrsina kruga: %f\n",
           r * r * PI);
   printf("Opseg kruga: %f",
           2 * r * PI);
   return 0;
}
```

"ispis novog reda", skok u novi red na zaslonu

Ne ovako!

```
float pi;
pi = 3.14159f;
```

Primjer

- `gcc -E prog2.c > prog2.i`

izvorni kôd **prog2.c**

```
#include <stdio.h>
#define PI 3.14159f

int main(void) {
   float r;   // polumjer kruga
   scanf("%f", &r);
   printf("Povrsina kruga: %f\n",
           r * r * PI);
   printf("Opseg kruga: %f",
           2 * r * PI);

   return 0;
}
```

pretprocesirani izvorni kôd **prog2.i**

```
...
__attribute__((__cdecl__))
__attribute__((__nothrow__))
int printf (const char *, ...);
...

int main(void) {
   float r;
   scanf("%f", &r);
   printf("Povrsina kruga: %f\n",
           r * r * 3.14159f);
   printf("Opseg kruga: %f",
           2 * r * 3.14159f);

   return 0;
}
```

Ne ovako!

```
#define DVA 2
...
printf("Opseg kruga: %f", DVA * r * PI);
...
```

Primjer izvršavanja prethodnog programa

```
5↵
Povrsina kruga: 78.539801
Opseg kruga: 31.415920
```

- crvenom bojom prikazuju se znakovi koje je tipkovnicom upisao korisnik
- oznaka ↵ će se koristiti uvijek kada će u primjerima trebati naglasiti da se na nekom mjestu "ispisuje skok u novi red" ili da je pritisnuta tipka Enter (odnosno Return)

Temeljni elementi jezika C

Izrazi

Izrazi

- Izraz (*expression*) je kombinacija operatora, operanada (konstante, varijable, pozivi funkcija, izrazi, ...) i zagrada, koja po evaluaciji daje rezultat
  - izraz pridruživanja
  - aritmetički izraz
  - relacijski izraz
  - logički izraz
- Izrazi se mogu ugrađivati u druge, složenije izraze, koristiti kao argumenti funkcija i dijelovi nekih naredbi

Izrazi

- Primjeri izraza (uz pretpostavku da su a i b cjelobrojne varijable):

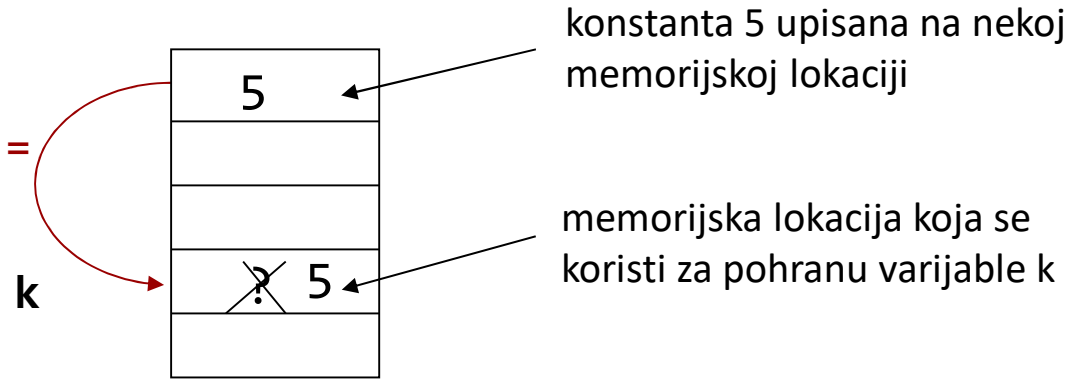
```
256                          aritmetički izraz
a                            aritmetički izraz
a + 11                       aritmetički izraz
(a + 1) * (b - 1)            aritmetički izraz
a = 20                       izraz pridruživanja
b = a + 3                    aritmetički izraz unutar izraza pridruživanja
a <= 10                      relacijski izraz
(a <= 10) && (b == 0)        dva relacijska izraza unutar logičkog izraza
```

- Primjeri korištenja izraza na mjestu argumenta funkcije i dijela naredbe (u ovom slučaju naredbe return)

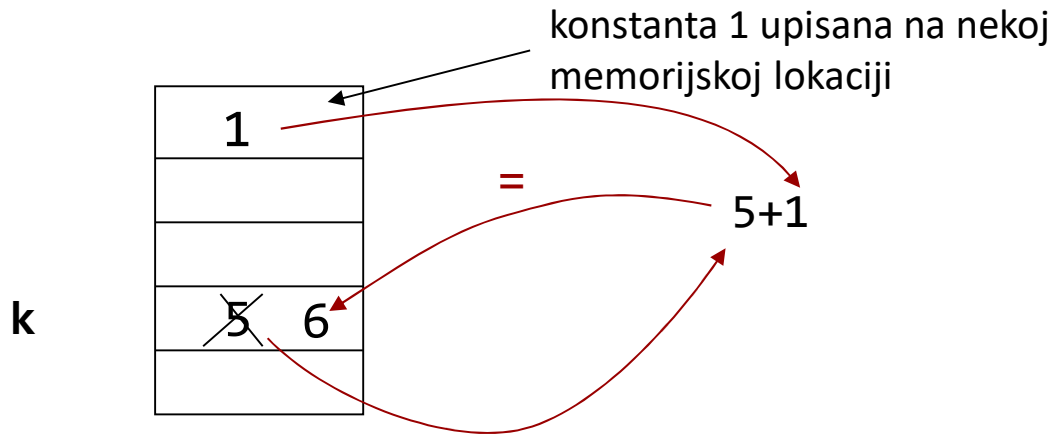
```
printf("%d", (a + 1) * (b - 1));
return a - b;
```

Operator i izraz pridruživanja

- Operator pridruživanja se koristi za pridruživanje vrijednosti
  - simbol u pseudo-kodu :=
  - u C-u =
- Primjeri:
  - `int k;`
    `k = 5;`



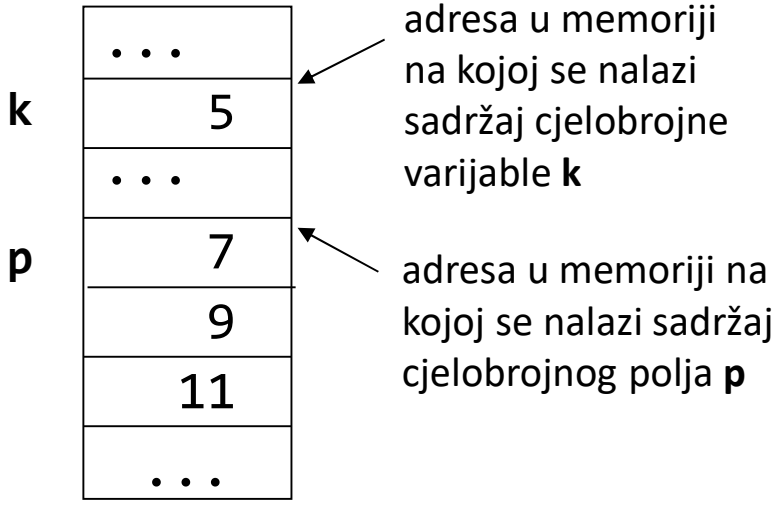
  - `k = k + 1;`



Locator value, lvalue

- *Locator value (Lvalue, L-value, lvalue)* je izraz koji predstavlja (određuje, *designates*) objekt koji se u memoriji nalazi na određenoj adresi (ili "ima određenu adresu")
  - npr. imena varijabli **k** i **p** su *lvalue* jer predstavljaju objekte (u ovom slučaju cjelobrojnu varijablu, odnosno cjelobrojno polje) čije su vrijednosti (*value*) pohranjene na određenim adresama u memoriji

```
...
int k = 5;
int p[3] = {7, 9, 11};
...
```



Modifiable lvalue, non-modifiable lvalue

- *lvalue* koji predstavlja objekt čija se vrijednost može promijeniti naziva se izmjenljivi *lvalue* (*modifiable lvalue*)
  - npr. ime varijable (jednostavnog tipa) je *modifiable lvalue*

```
...
int k = 5;
...
k = 10;
```

vrijednost objekta može se promijeniti pomoću *lvalue* k

  - npr. ime varijable tipa polje je *non-modifiable lvalue*. Sadržaj polja nije moguće promijeniti korištenjem (samo) imena varijable p1

```
...
int p1[3] = {7, 9, 11}, p2[3] = {6, 8, 10};
...
p1 = p2;
```

neispravno: p1 je *non-modifiable lvalue*

Lijeva strana u izrazu pridruživanja

- Lijeva strana (lijevi operand) izraza pridruživanja mora biti *modifiable locator value (modifiable lvalue)*.

```
int m, n, k;
int p[3] = {7, 9, 11};
n = 15 + 3;                 n je modifiable lvalue
m = m + n + 1;              m je modifiable lvalue
k + 1 = m + 1;              neispravno: k + 1 nije lvalue
7 = m;                      neispravno: konstanta 7 nije lvalue
p = 10;                     neispravno: p nije modifiable lvalue
```

Prema (danas zastarjelom) tumačenju iz *Kernighan and Ritchie: The C Programming Language*, pojam *Lvalue* (*left-value*) odnosi se na vrstu izraza koji je dopušteno koristiti na lijevoj strani izraza pridruživanja.

Desna strana u izrazu pridruživanja

- Na desnoj strani izraza pridruživanja može se nalaziti bilo koji izraz (može, ali ne mora biti *lvalue*). Npr. varijabla, konstanta, aritmetički izraz, poziv funkcije, itd.
  - vrijednost izraza (*value of expression*) na desnoj strani izračunava se (evaluira) i postavlja kao nova vrijednost objekta kojeg predstavlja *lvalue* na lijevoj strani izraza pridruživanja

```
int m, n;
n = 15 * 3 - 100;
m = abs(n) / 2;
```

Prema danas zastarjelom, ali u literaturi često korištenom tumačenju, pojam *Rvalue* (*right-value*) odnosi se na vrstu izraza koji je dopušteno koristiti na desnoj strani izraza pridruživanja.

Rezultat izraza pridruživanja

- izraz pridruživanja se prvenstveno koristi za pridruživanje vrijednosti, ali kao i svaki drugi izraz, "po evaluaciji" daje rezultat.
  - taj rezultat najčešće se ne koristi, kao u sljedećem primjeru: aritmetički izraz daje rezultat 30, izrazom pridruživanja 30 se pridružuje varijabli m, a konačni rezultat izraza pridruživanja jest opet vrijednost 30 (vrijednost koja je upravo pridružena). Vrijednost izraza pridruživanja ostala je neiskorištena.

```
int m;
m = 15 * 2;
```

  - u sljedećem primjeru, rezultat izraza pridruživanja će se iskoristiti. Konstanta 5 pridružuje se varijabli m, konačni rezultat izraza pridruživanja je 5. Ta vrijednost se ispisuje na zaslon.

```
int m;
printf("%d", m = 5);
```

Višestruko pridruživanje

- činjenica da izraz pridruživanja "po evaluaciji" daje rezultat može se lijepo iskoristiti kod višestrukog pridruživanja

```
int a, b, c;
...
c = 3;
...
a = b = c * 5;
```

Redoslijed obavljanja: `a = (b = (c * 5));`

  - izračunata je vrijednost c * 5 (u primjeru 15) i pridružena varijabli b.
  - rezultat izraza pridruživanja (15) pridružuje se varijabli a
  - rezultat izraza pridruživanja (15) se nema za što iskoristiti, pa se odbacuje

- Je li sljedeći izraz s višestrukim pridruživanjem ispravan?

```
a = b + 2 = c;
```

Neispravno jer `b + 2` nije lvalue

Prioritet i asocijativnost operatora

- Redoslijed obavljanja operacija u izrazima ovisi o
  - **prioritetu operatora**, ako se radi o operatorima različitog prioriteta

    `a + b * c` `(a + b) * c` ili `a + (b * c)`

    prioritet operatora određuje da se prvo obavlja operacija `b * c`
  - **asocijativnosti operatora**, ako se radi o operatorima jednakog prioriteta

    `a / b * c` `(a / b) * c` ili `a / (b * c)`

    asocijativnost operatora određuje da se prvo obavlja operacija `a / b`

Prioritet ↓ Niži ↑ Viši

Operator	Asocijativnost operatora
* /	L → D
binarne operacije + -	L → D
=	D → L

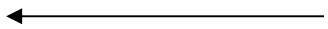
Primjer

- Tijekom evaluiranja izraza

```
a = b = c * 5
```

  treba obaviti nekoliko operacija. Kojim redoslijedom će se operacije obaviti?

- Operator množenja ima viši prioritet od operatora pridruživanja
  - prvo će se obaviti operacija množenja `a = b = (c * 5)`
- Operatori pridruživanja imaju jednaki prioritet. Redoslijed obavljanja određen je asocijativnošću operatora
  - asocijativnost operatora D→L znači da se operandi i operacije grupiraju od desna prema lijevo, dakle



```
(a = (b = c * 5))
```

za operaciju `c * 5` je već prije odlučeno da se obavlja prva, stoga se ovdje ne razmatra.

Primjer

- Cjelobrojne varijable a i b inicijalizirati na vrijednosti 14 i -9. Program treba ispisati njihove vrijednosti, zamijeniti vrijednosti u varijablama te ponovo ispisati vrijednosti varijabli na zaslon. Ispis na zaslonu treba izgledati ovako:

```
a=14, b=-9
a=-9, b=14
```

```
#include <stdio.h>
int main(void) {
   int a = 14, b = -9, pom;
   printf("a=%d, b=%d\n", a, b);
   pom = a;
   a = b;
   b = pom;
   printf("a=%d, b=%d", a, b);
   return 0;
}
```

a	b	pom
14	-9	?
14	-9	?
14	-9	14
-9	-9	14
-9	14	14
-9	14	14

Aritmetički operatori i izrazi

- Ovdje su navedeni samo osnovni aritmetički operatori

Operator	Značenje	Operandi
+	zbrajanje	int, float
-	oduzimanje	int, float
*	množenje	int, float
/	dijeljenje	int, float
%	ostatak cjelobrojnog dijeljenja	int

  - aritmetički operator i pridruženi operandi čine aritmetički izraz
  - operandi mogu biti varijable, konstante i složeniji aritmetički izrazi

Djelovanje operatora na cjelobrojne operande

- uočiti: ako su oba operanda cjelobrojna
  - rezultat je cjelobrojan
  - operacija se obavlja u cjelobrojnoj domeni (naročito važno za operaciju dijeljenja)

```
int a = 11, b = 2;
```

aritmetički izraz	rezultat
a + b	13
a - b	9
a * b	22
a / b	5
a % b	1

Djelovanje operatora na realne operande

- uočiti: ako je barem jedan operand realnog tipa
  - rezultat je realan
  - operacija se obavlja u realnoj domeni (naročito važno za operaciju dijeljenja)
  - operator *modulo* ne smije se koristiti

```
int a = 11;
float b = 2.f;
```

aritmetički izraz	**rezultat**
`a + b`	`13.0`
`a - b`	`9.0`
`a * b`	`22.0`
`a / b`	`5.5`
`a % b`	prevodilac odbija prevesti program

Prioritet aritmetičkih operatora

- operatori množenja, dijeljenja i ostatka cjelobrojnog dijeljenja imaju veći prioritet od operatora zbrajanja i oduzimanja

```
a + b * c ≡ a + (b * c)
b * c + a ≡ (b * c) + a
```

- ako aritmetički operatori imaju jednak prioritet (npr. množenje, dijeljenje i ostatak cjelobrojnog dijeljenja), tada se operacije obavljaju s lijeva na desno

```
a / b * c ≡ (a / b) * c
x / a + b * c + d * e ≡ ((x / a) + (b * c)) + (d * e)
```

- ako pretpostavljeni redoslijed obavljanja operacija treba promijeniti, koristiti okrugle zagrade, npr.

```
(a + b) * c
x / ((a + b) * (c + d) * e)
```

Primjer

- Odrediti veličinu i tip rezultata sljedećih aritmetičkih izraza

```
int m = 11;
float x = 2.f;
m / x                    5.5, float
m / 2                    5, int
m / 2 * x               10.0, float
x * m / 2               11.0, float
x * (m / 2)             10.0, float
m + 1    / x            11.5, float
(m + 1) / x              6.0, float
```