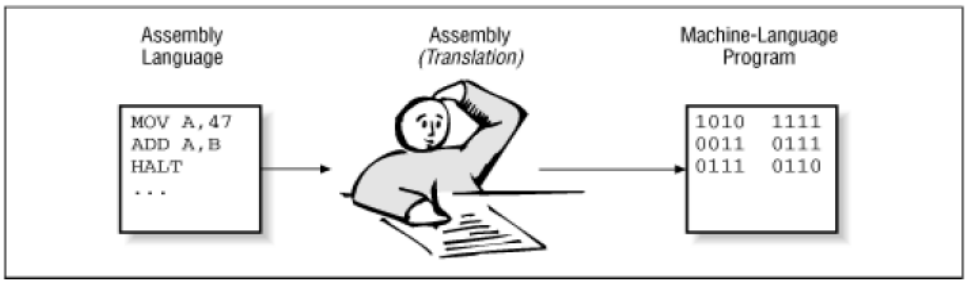
# ВОВЕД

## 1. Општ развој на програмските јазици

Со појавата на првите компјутери се појавува потребата од пишување на програми преку кои ќе функционираат деловите од компјутерот. Во тоа време програмерите програмираа на многу неприроден начин, комбинирајќи низи од 0 и 1, користејќи го т.н. машински јазик.

Подоцна програмерите согледуваат дека од секој блок на 0 и 1 може да се искомбинираат наредби-зборови кои ќе се многу поразбирливи и полесно ќе се применуваат во текот на пишувањето на програмите. Ваквиот програмски јазик е познат како assembler.




Со тек на времето се појавуваат т.н. програмски јазици на високо-ниво, кои му нудат на програмерот множество на инструкции кои се лесно разбирливи, а исто така се доволно прецизни и едноставни за компјутерот да може да ги разбере (овде спаѓаат FORTRAN, COBOL, PASCAL…).

Во 1970 година програмерот, Dennis Ritchie, го креира новиот програмски јазик наречен C. (Името го измислил според тоа што овој нов јазик го надградува програмскиот јазик кој претходно го користел – наречен B).

C бил почетно дизајниран за пишување на оперативни системи. Јазикот бил екстремно едноставен и флексибилен, така да подоцна се користи за пишување на најразлични програми. Поради овие причини јазикот станува најпопуларен програмски јазик во светот.

Идејата за измислувањето на програмскиот јазик C е давањето на слобода на програмерот при организацијата и пишувањето на програмата, односно да го напише кодот на начин кој е разбирлив за него, а и за останатите програмери. По пишувањето на програмата се користи компајлер кој ја преведува програмата во машински код кој е лесно разбирлив за компјутерот.

## 2. Основни елементи на програмскиот јазик C

Секој програмски јазик на високо ниво се состои од множество на резервирани зборови, а комбинацијата од еден или неколку клучни зборови дава наредба од програмскиот јазик.

Множеството на клучни зборови од програмскиот јазик C е следното (32 според ASCII стандардот, 28 според Richie).

```
auto        double      int         struct
break       else        long        switch
case        enum        register    typedef
char        extern      return      union
const       float       short       unsigned
continue    for         signed      void
default     goto        sizeof      volatile
do          if          static      while
```

Програмскиот јазик C е функционален јазик, односно кодот напишан во овој програмски јазик е базиран на функции. Основниот облик на програма напишана во програмскиот јазик C е следниот:

| Програма во C | Програма во Pascal |
|---|---|
| int main()<br>{<br>  deklaracija_na_promenlivi;<br>  programski_naredbi;<br>  return 0;<br>} | Program ime_na_programata;<br>var deklaracija_na_promenlivi;<br>begin<br>  programski_naredbi;<br>end. |

main е функција која означува главна програма. Во овој специјален тип на функција може да се декларираат променливи, нови под-функции, и да се пишува кодот на програмата.
Функциите може да примаат влезни параметри. Ова се обезбедува преку () делот по името на функцијата. Во овој случај празните загради означуваат дека оваа функција не прима влезни параметри.

Типот на резултатот кој го враќа функцијата стои пред името на функцијата. Во овој случај резервираниот збор int означува дека функцијата треба да врати резултат кој претставува целобројна вредност.

Телото на секоја функција започнува со {, а завршува со }. Наредбите кои се употребуваат меѓусебно се одвојуваат со ; - точка и запирка.
За дополнително појаснување на некој програмски сегмент многу често се користат коментарите. Во програмскиот јазик C коментарите се задаваат преку користење на /* … */ конструктот или преку //… конструктот.

```
int main()
{
   deklaracija na promenlivi;
   naredba 1;  /*  komentar  1
   */ naredba 2; /* komentar 2
   */ naredba 3; // komentar 3
   return 0;
}
```

**Пример 1*:** Програма Dobredojdovte na FEIT!

```
#include <stdio.h>
int main()
{
    printf("Dobredojdovte na FEIT!\n");
    /* na ekran ke se ispecati gornata poraka */
    return 0;
}
```

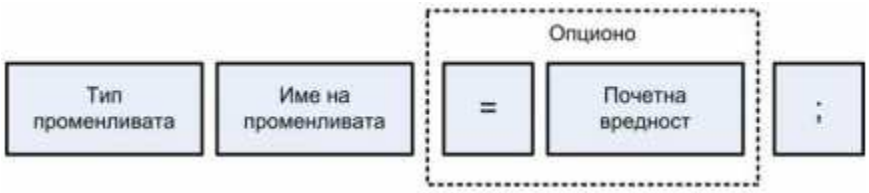
* кратко објаснување на делот include заедно со библиотеката stdio.h кратко објаснување на наредбата printf

Од овде се гледа дека основната верзија на структурата на С програмата може да се прошири на:

| | |
|---|---|
| **INCLUDE** секција | содржи #include изрази за вклучување на надворешни библиотеки, односно да се искористат готови надворешни веќе дефинирани функции |
| **DEFINE** секција | содржи #define изрази за декларирање на константи и податочни типови |
| … | дефинирање на глобални променливи и функции |
| tip **main**() | |
| { | |
| tip promenlivi; | декларација на променливи |
| naredbi; | извршни изрази |
| } | |

## 3. Променливи

Променливите претставуваат места во меморијата каде што ќе се чува некоја вредност. Во С ова место во меморијата се идентификува преку името на променливата. Сите променливи кои ќе се користат во програмата треба претходно да бидат декларирани, а потоа да се користат. Со секое ново сместување на вредност во променливата, старата вредност се брише! Начинот на декларација на променливите е следниот:




Типовите на променливите може да бидат:

| Цели броеви | Знакови | Децимални броеви |
|---|---|---|
| int<br>short<br>long<br>unsigned | char | float<br>double |

При дефинирањето на имињата на променливите треба да се запазат следниве правила, односно имињата на променливите треба да содржат:

- мали букви од a до z;
- големи букви од A до Z;
- цифри од 0 до 9;
- знак за подвлекување _ кој се третира како буква;
- најчесто должината на имињата на променливите е 32 знаци;
- C ги разликува малите и големите букви!

**Пример 2**: Да се напише програма во C која ќе ја пресмета сумата на два броја.

```
int main()
{
   int a, b, c;
   a = 5;
   b = 10;
   c = a + b; // c=15
   return 0;
}
```

Како што се гледа од примерот освен делот за декларирање на променливите, останатиот дел од програмата ги содржи наредбите кои ја извршуваат бараната задача. Бидејќи се работи за математичка операција, користени се оператори за работа со декларираните променливи.

## 4. Константи

Терминот константа значи дека истата не се менува за време на работата на програмата. Секоја променлива се декларира дека припаѓа на одреден тип податоци. Овој тип ја дефинира големината на променливата и како истата може да се користи. Слично, кога се специфицира константа, се дава и типот на константата. Кај променливите типот е очигледен со нивната декларација. Константите меѓутоа не се декларираат. Одредувањето на нивниот тип не е толку едноставно.

Постојат повеќе типови константи во C:

- декадни: -1, 0, 12, 189, …
- октални: 015, 019, 0105, …
- хексадецимални: 0x25, 0xA4C, …
- реални: 3.5F, -2.845F, 1.34e-9, …
- знаковни: ‘a’, ‘b’, ‘c’, ‘\n’, …
- текстуални: “ ”, “Struktuirano programiranje”, …

- Броевите кои содржат „.“ или „e“ се double: 3.5, 1e-7, -1.29e15
- За наместо double да се користат float константи на крајот се додава „F“: 3.5F, 1e-7F
- За long double константи се додава „L“: 1.29e15L, 1e-7L
- Броевите без „.“, „e“ или „F“ се int: 1000, -35
- За long int константи се додава „L“: 9000000L.

Именуваните константи може да се креираат со користење на резервираниот збор **const**. Со користење на const типот на константата експлицитно се дефинира.

```
#include <stdio.h>
main()
{
     const long double pi = 3.141592653590L;
     const int denovi_vo_nedela = 7;
     const nedela = 0; // po default int

     denovi_vo_nedela = 5;     // greska
}
```

По креирањето на константата, таа може да се јави само од десната страна на операторот за доделување вредност.

Именуваните константи може да се креираат и со користење на предпроцесорот и за нив по правило се користат големи букви.

```
#include <stdio.h>

#define PI   3.141592653590L
#define DENOVI_VO_NEDELA 7
#define NEDELA           7
main()
{
     long broj = PI;
     int den = NEDELA;
}
```

Бидејќи предпроцесорот е едитор тој ја изведува операцијата најди и замени. За да се влезе во овој начин на работа се користи наредбата #define. Нејзината синтакса е всушност:

***#define*** *tekst_za_baranje tekst_za_zamena*

Се заменуваат само цели зборови. Низи од знаци во наводници се игнорираат.

## 5. Оператори

Операторите се користат за градење на изрази, при што операциите се изведуваат од лево надесно со што се применува правилото на приоритет на операторите во нивното изведување.

**Аритметички оператори:** Се применуваат над бројни променливи (цели броеви или децимални броеви).

| | |
|---|---|
| **+** | Собирање на два броја |
| **-** | Одземање на два броја |
| ***** | Множење на два броја |
| **/** | Делење на два броја |
| **%** | Остаток при делење на два цели броја |

**Релациони оператори:** Се применуваат над било кои споредливи типови на податоци, а резултатот е цел број 0 или 1. 0 – неточно, 1 – точно.

| | |
|---|---|
| **<** | Помало |
| **<=** | Помало еднакво |

| | |
|---|---|
| > | Поголемо |
| >= | Поголемо еднакво |
| == | Еднакво |
| != | Различно |

**Логички оператори:** Се користат најчесто во комбинација со релационите оператори за формирање на сложени логички изрази, кои повторно враќаат резултат 0 или 1.

| | |
|---|---|
| && | И |
| \|\| | ИЛИ |
| ! | НЕ |

**Пример 3:** Работа со промеливи и доделување на вредност.

```
int main()
{
	int a;
	float p;
	p=1.0/2.0;	/* p=0.5 */
	a=5/2;		/* a=2 */
	p=(1/2)+(1/8);	/* p=0.0 */
	p=3.5/2.8;	/* p=1.25 */
	a=p;		/* a=1 */
	a=a+1;		/* a=2 */
	return 0;
}
```

### *Реални броеви наспрот целобројни вредности*

Операторот за делење е специјален. Постои голема разлика меѓу делењето на целобројни и реални вредности. Ако станува збор за целобројно делење, резултатот е само целиот дел од делењето. Така на пример, 19/10 е 1. Ова значи дека делењето се извршува на различни начини во зависност од контекстот во кој се наоѓа, односно во зависност од типот на операндите со кои ќе работи. Така, ако двата операнди се цели броеви ќе се изврши целобројно делење.

Ако барем дељеникот или делителот е реален број, тогаш се извршува делење на реални броеви и резултатот од ова делење е реален број. Така 19.0/10.0 е 1.9 (резултатот е исти и за 19/10.0 и 19.0/10).

```
int main(void)
{
	int i = 5, j = 4, k;
	double f = 5.0, g = 4.0, h;

	k = i / j;  // celobrojno delenje na dva int rezultat 1
	h = f / g;  // realno delenje rezultat 1.250000
	h = i / j;// celobrojno delenje iako h e realen, rezultatot e 1.000000

	return 0;
}
```

С се снаоѓа со употребата на аритметичките оператори со различни типови на операнди така што операторот ќе одбере каков вид на операција ќе изведе. При тоа комапјлерот се грижи само за типот на операндите. При изведувањето на операцијата не се зема во предвид типот на променливата на која и се доделува резултатот.

## 6. Печатење на податоци (излезни функции)

Како што беше покажано на првиот пример “Dobredojdovte na FEIT!”, се користи функција за испраќање на некој текст кон стандардниот уред за испис на податоци – мониторот.

Бидејќи C не вклучува наредби за влез и излез на податоци, се користи библиотека со функции која претходно мора да се пријави во програмата. Тоа се прави со користење на:

**#include<stdio.h>** stdio – standard input output

На овој начин пред да се искомпајлира програмата, предпроцесорот на C знае дека треба да ги вклучи функциите од библиотеката stdio.h со цел да не се јави грешка при компајлирањето. За печатење се користи функцијата printf:

***int printf(kontrolna_niza,lista_na_promenlivi);***

Контролната низа содржи било каков текст за испис и контролни знаци предводени од % или \. Контролните знаци зависат од видот на променливата чија вредност треба да се испише или од саканата акција што треба да биде превземена.

Во следната табела се прикажани сите контролни низи:

| Контролна низа | Објаснување |
|---|---|
| **%d** | За цели броеви |
| **%i** | За цели броеви |
| **%c** | За знаци |
| **%s** | За низа од знаци |
| **%e** | Реален број во технички формат (e) |
| **%E** | Реален број во технички формат (E) |
| **%f** | Реален број во децимален формат |
| **%g** | Реален број во пократкиот од форматите %e и %F |
| **%G** | Реален број во пократкиот од форматите %E и %F |
| **%u** | Цел број без предзнак |
| **%o** | Октален цел број без предзнак |
| **%x** | Хексадецимален цел број без предзнак (мали букви) |
| **%X** | Хексадецимален цел број без предзнак (големи букви) |
| **%p** | Покажува покажувач |
| **%n** | Бројот на испишани знаци се доделува на аргументот |
| **%%** | Испишување на знакот % |

Во низата на променливи покрај променливи може да има и константи и аритметички изрази.

**Пример 4:** Примена на printf функцијата.

```
#include<stdio.h>
int main()
{
      printf(", brojot na znaci e %d",printf("abcd"));
      return 0;
}
```

Излез: abcd, brojot na znaci e 4

**Пример 5:**

```
#include <stdio.h>
#include <math.h>
main()
{
     float i = 2.0, j = 3.0;
     printf ("%f %f %f %f", i, j, i+j, sqrt(i+j));
}
```

*Излезот на програмата е следниот:*

2.000000 3.000000 5.000000 2.236068

Бројот на конверзии во даден формат треба точно да одговара на бројот на аргументи во функцијата. При тоа, C ова нема да го потврди. Ако се дадени поголем број аргументи, вишокот аргументи се игнорираат. Ако, пак, нема доволен број аргументи, C ќе генерира чудни броеви за аргументите кои недостигаат.

**Пример 6**:

```
#include <stdio.h>
/* Variable for computation results */
int answer;
int main()
{
     answer = 2 + 2;
     printf("The answer is %d\n");
     return 0;
}
```

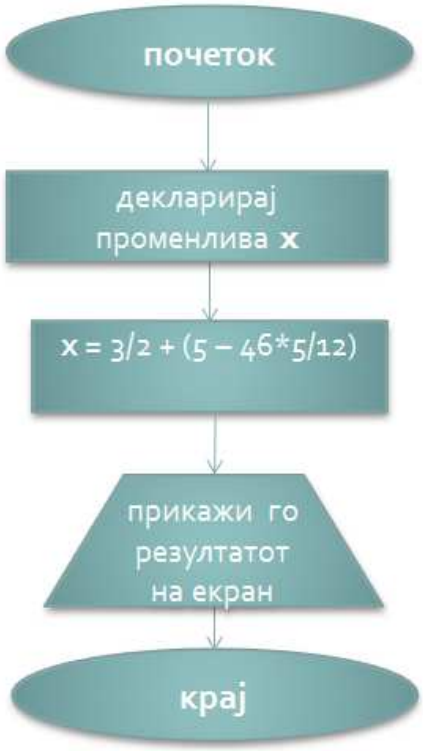
**Пример 7**:

```
#include <stdio.h>
float result; /* Result of the divide */
int main()
{
     result = 7.0 / 22.0;
     printf("The result is %d\n", result);
     return 0;
}
```

### *Задачи (прв дел)*

1. Да се напише програма која ќе ја пресметува вредноста на математичкиот израз:
x = 3/2 + (5 – 46*5/12)

```c
#include <stdio.h>
int main()
{
    float x;
    x = 3/2 + (5-46*5/12);
    printf("Vrednosta na x e %f\n", x);
    return 0;
}
```

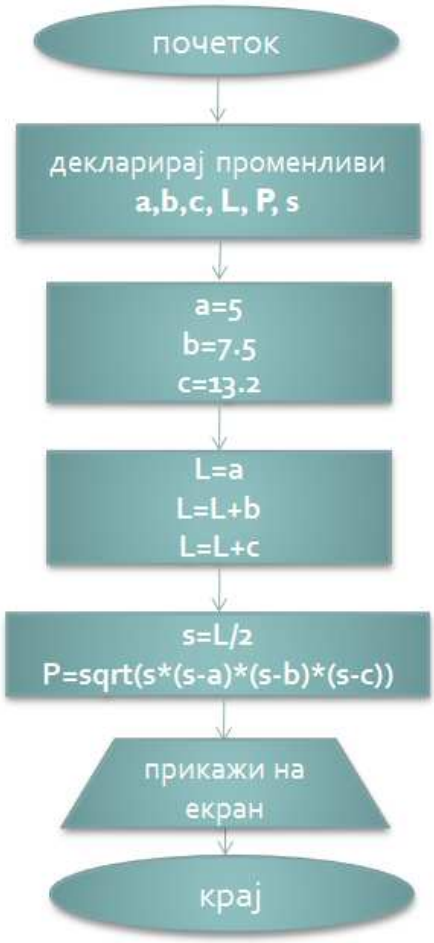



2. Да се напише програма која за различни вредности на x ќе го пресмета и испечати $x^2$.

```c
#include <stdio.h>
int main()
{
      int x, kvadrat;
      x = 25;
      kvadrat = x*x;
      printf("%d na kvadrat e %d.\n", x, kvadrat);
      return 0;
}
```

3. Да се напише програма која за дадени страни на еден триаголник ќе ги испечати периметарот и квадратот од плоштината (нека се работи со a=5, b=7.5, c=13.2).

```c
#include <stdio.h>
int main()
{
   float a=5.0, b=7.5, c=13.2;
   float L, P, s;
   L = a + b + c;
   s = L/2;
   P = s*(s-a)*(s-b)*(s-c);
   printf("Plostinata na kvadrat e: %f\n", P);
   printf("Perimetarot e: %f\n", L);
   return 0;
}
```

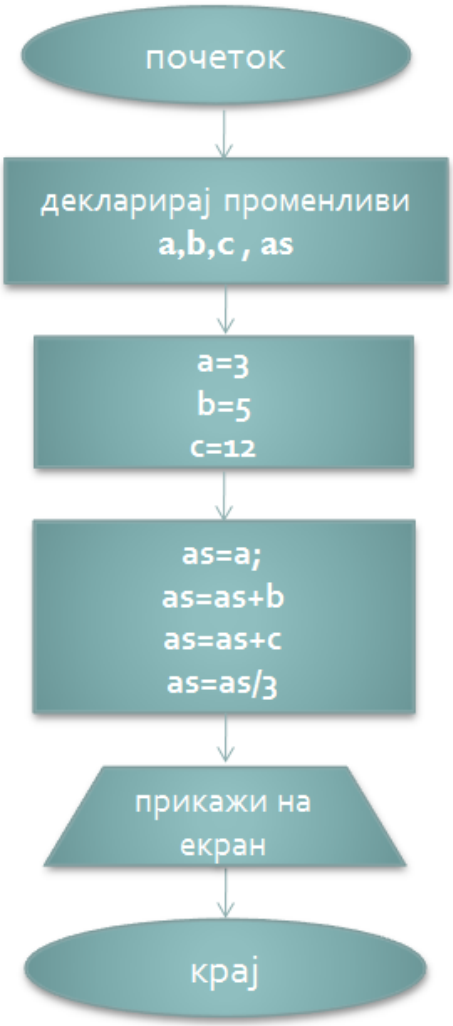



4. Да се напише програма која за дадени страни на еден правоаголник ќе ги испечати неговите плоштина и периметар (пр. a=7, b=10).

```c
#include <stdio.h>
int main()
{
      int a=7, b=10;
      int L, P;
      L = 2*a + 2*b;
      P = a * b;
      printf("Plostinata e: %d\n", P);
      printf("Perimetarot e: %d\n", L);
      return 0;
}
```

5. Да се напише програма за пресметување на аритметичката средина на броевите 3, 5 и 12.

```c
#include <stdio.h>
int main()
{
   int a=3;
   int b=5;
   int c=12;
   float as;
   as = (a + b + c)/3;
   printf("Aritmetickata sredina e: %f\n", as);
   return 0;
}
```




6. Да се напише програма која ќе ги испечати на екран остатоците при делењето на бројот 19 со 2, 3, 5 и 8.

```c
#include <stdio.h>
int main()
{
      int a=19;
      printf("Ostatokot pri delenjeto so 2 e: %d\n",  a%2);
      printf("Ostatokot pri delenjeto so 3 e: %d\n",  a%3);
      printf("Ostatokot pri delenjeto so 5 e: %d\n",  a%5);
      printf("Ostatokot pri delenjeto so 8 e: %d\n",  a%8);
      return 0;
}
```

# 7. Повеќе за операторите

## Аритметички оператори

Аритметичките оператори се користат за претставување на математички изрази. При тоа, + и – може да се користат и на унарен начин:

X = + Y;
X = - Y;

Првата наредба е иста со x=y, додека втората ја множи вредноста на y со -1, а потоа ја доделува на x.

**Оператори за инкрементирање и декрементирање**

Операторот за инкрементирање **(++)** го зголемува операндот за еден, додека операторот за декрементирање **(--)** го намалува операндот за еден. Операндот кој се користи со овие оператори мора да биде една променлива.

Пример: ако на целобројната променлива i и е доделена вредноста 5, изразот ++i ја зголемува вредноста на i за еден, со што i станува 6, изразот --i ја намалува вредноста на i за еден и i станува 4.

Операторите за инкрементирање и декрементирање може да се користат на два различни начини, зависно од тоа дали операторот е запишан пред или по операндот. Ако операторот е пред операндот (++i) тогаш операндот ќе ја промени својата вредност пред да се искористи во понатамошниот тек на наредбата. Ако меѓутоа, операторот е по операндот (i++) тогаш вредноста на операндот се менува откако истата е искористена.

**Пример 1:**

```
i = 1;
a = i++;
printf("a=%d i=%d\n", a, i);  //a=1, i=2
b = ++i;
printf("b=%d i=%d\n", b, i); //b=3, i=3
```

**Пример 2:**

```
i = 1;
printf(" i = %d\n", i);
printf(" i = %d\n", ++i);
printf(" i = %d\n", i);
```

Овие наредби ќе го генерираат следниот излез:

i = 1
i = 2
i = 2

**Пример 3:**

```
i = 1;
printf(" i = %d\n", i);
printf(" i = %d\n", i++);
printf(" i = %d\n", i)
```

Овие наредби ќе го генерираат следниот излез:

i = 1

i = 1

i = 2

## Релациони и логички оператори

Треба да се води сметка дека нема boolean тип на податок во C, наместо него се користи целобројниот int тип. При тоа, вредноста 0 е неточно, додека секоја друга вредност е точно.

Резултатот на логичката И операција (оператор &&) ќе биде точно само ако двата операнди се точни, додека резултатот на логичката операција ИЛИ (оператор ||) ќе биде точно ако било кој од операндите или, пак, двата операнди се точни. Со други зборови, резултатот од логичката операција ИЛИ ќе биде неточно, само ако двата операнди се неточни. Резултатот од логичката операција NE (оператор !) ќе биде спротивното од операндот, односно точно ако операндот има вредност неточно, и неточно ако операндот има вредност точно.

При тоа треба да се води сметка дека резултатот на секој релационен логички израз во C е 0 за неточно или 1 за точно ( и покрај тоа што секоја ненулева вредност ја третира како точно).

**Пример 4:**

```
i = 5; j = 0; k = -1;
l = i && k || j; // l = 1
```

поради истиот приоритет прво се изведува И операцијата, а потоа ИЛИ.

**Пример 5:**

```
#include <stdio.h>
main()
{
	float f = 5.5;
	int i = 7;
	char c = 'w';
	printf("%d\t",(i>=6)&&(c=='w'));		// 1
	printf("%d\t",(i>=6)&&(c==119));		// 1
	printf("%d\t",(f<11)&&(i>100));		// 0
	printf("%d\n",(c != 'p')||((i+f)<=10)); // 1
	// !!! da se upotrebuvaat zagradi
	printf("%d %d\n",(!i == 7), (!(i == 7))); // 0 0
	// slucajno da ne se upotrebi = namesto ==
	printf("%d %d\n",(!i == 0), (!(i = 0)));  // 0 1
}
```

Треба да се води сметка дека евалуацијата на релационите логички изрази е од лево на десно и дека истата не се изведува докрај доколку нема потреба за тоа.

**Пример 6:**

```
i = 5; k = 10;
printf("%d\t%d\n", (i==4)&&(k=5), (i!=4)||(k==5)); // 0 1
```

## Оператор за доделување

Доделувањето на вредност се врши со помош на = операторот. При тоа вредноста од левата страна на = се доделува на променливата од десната страна на =. Употребата на овој оператор е многу флексибилна. Доделената вредност е секогаш достапна за повторна употреба при доделување и при тоа операцијата се изведува секогаш од десно на лево, а има низок приоритет:

```
int i, j, k, l, m, n;
i = j = k = l = m = n = 22;
printf("%i\n", j=93);
```

**Други оператори за доделување**

Постои фамилија на оператори за доделување:

+= -= *= /= %=

Кај секој од нив

***израз1 оператор= израз2***

може да се интерпретира како **(израз1) = (израз1) оператор (израз2)**

Треба да се запамети дека C најпрвин ја евалуира десната страна на =, така a*=b+7 значи a=a*(b+7).

**Пример 7:**

```
#include "stdio.h"
int main(int argc, char* argv[])
{
	int a=5, i=7, j=9, k;
	float f=10.5;

	a += 27;
	f /= 9.2;
	k = i *= j+2;

	printf("%d %f %d %d\n",a, f, i, k);
	return 0;
}
```

Излезот е: 32 1.141304 77 77

# 8. Влез и излез на податоци

### Читање на еден знак – getchar() функција

Со користење на библиотечната функција getchar() може да се внесе еден знак од тастатурата. Оваа функција е дел од стандардната C I/O библиотека. Таа враќа еден знак превземен од стандардниот влез. Функцијата нема аргументи, но мора да биде повикана со празни загради.

**char var=getchar()**

### Печатење на еден знак – putchar() функција

И putchar() функцијата е дел од стандардната I/O библиотека на C. Таа емитува еден знак кон стандардниот излез. Знакот кој се пренесува е претставен како променлива од типот знак – char. Овој знак мора да се даде како аргумент кон функцијата во заградите кои следат по нејзиното име:

**putchar(char var)**

**Пример 8:** Да се напише програма која три прочитани знаци ќе ги прикаже во обратен редослед.

```
#include <stdio.h>
int main()
{
	char a, b, c;
	a = getchar();
	b = getchar();
	c = getchar();
	putchar(c);
	putchar('\t');
	putchar(b);
	putchar('\t');
	putchar(a);
	putchar('\n');
	return 0;
}
```

## Внесување на податоци со scanf() функцијата

Влезните податоци може да се внесат во компјутерот од стандардниот влез со користење на библиотечната функција scanf(). Во општ случај оваа функција се запишува како:

**scanf(Controlna niza od znaci, arg1, arg2, … , argn)**

Контролната низа од знаци е всушност низа од знаци (стринг) кој ја содржи потребната информација за форматирање, а arg1, arg2, ..., argn се аргументите кои ги претставуваат индивидуалните податоци. Аргументите претставуваат покажувачи кои ја даваат мемориската адреса каде ќе се смести податокот кој ќе се прочита.
Контролната низа од знаци е изградена од индивидуални групи на знаци со една група на знаци за секој податок кој ќе се чита. Секоја група на знаци мора да започне со знакот %. Во нејзината наједноставна форма една група знаци се состои од % по кој следи знак за конверзија кој го дава типот на соодветниот мемориски елемент каде ќе се запише податокот.

| Знак за конверзија | Објаснување |
|---|---|
| **c** | Податочниот елемент е еден знак |
| **d** | Податочниот елемент е цел број |
| **f** | Податочниот елемент е реална вредност |
| **h** | Податочниот елемент е краток цел број |
| **l** | Податочниот елемент е декаден, хексадекаден или октален број |
| **o** | Податочниот елемент е октален цел број |
| **s** | Податочниот елемент е низа од знаци по која следи празно место, нов ред или табулатор |
| **u** | Податочниот елемент е декаден цел број без знак |
| **h** | Податочниот елемент е хексадекаден цел број |

**Пример 9:**

```
#include <stdio.h>
main()
{
	char del;
	int delbroj;
	float cena;
	...
	scanf("%c%d%f",&del,&delbroj,&cena);
	...
}
```

Првата група знаци покажува дека првиот аргумент е знак, втората дека вториот аргумент е децимален цел број, додека третата дека третиот аргумент е реален број. Со помош на scanf() функцијата вредностите на трите променливи del, delbroj и cena можат да се прочитаат од стандардниот влез кога ќе се изврши програмата. Влезот за програмата може да е следниот:

P 12345 570.34

Или

P
12345
570.34

**Пример 10:**

```
#include <stdio.h>
main()
{
      int a, b, c;
      ...
      scanf("%3d %3d %3d",&a,&b,&c);
}
```

Ако влезот е:
 1 2 3
Тогаш на трите променливи ќе им бидат доделени следните вредности:
 a=1, b=2, c=3
додека ако влезот е
 123456789
Доделувањето на вредностите би изгледало вака:
 a=123 b=456 c=789

**Пример 11:**

```
#include <stdio.h>
int main()
{
     int i;
     float f;
     char c;
     ...
     scanf("%3d %5f%c",&i,&f,&c);
     ...
}
```

Ако влезот е:
 10256.875 T
Доделувањето ќе се изврши на следниот начин:
 i=102 f=56.87 c=5

**Треба да се напомене дека функцијата scanf го враќа бројот на прочитани параметри**

**Пример 12:**

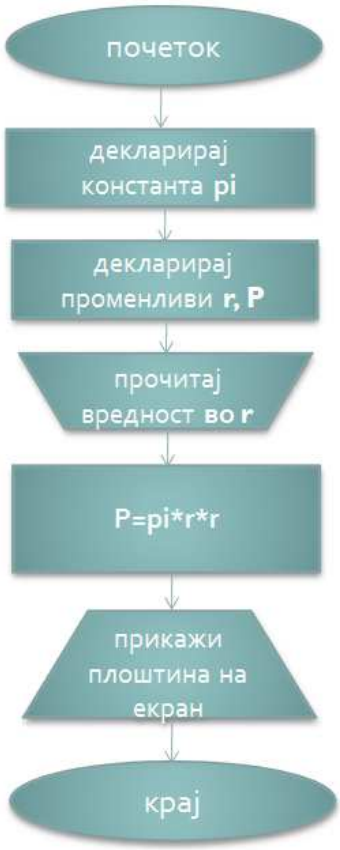
```
#include <stdio.h>
int main()
{
      int i;
      float f = 3.0;
      char c = 'a';
      i = scanf("%f %c",&f,&c);
      printf("%d %f %c",i, f, c);
      return 0;
}
```

За влез: a c
Излезот ќе биде: 0 3.0 a
Додека за влез: 5.0 c
Излезот ќе биде: 2 5.0 c

### *Задачи (втор дел)*

1. Да се напише програма за пресметување и печатење на плоштината на круг која ќе чита реален број кој го претставува радиусот на кругот.

```
#include <stdio.h>

int main()
{
      long double radius = 0.0L;
      long double plostina = 0.0L;
      const long double pi = 3.1415926353890L;

      printf("vnesi radius na krugot ");
      fflush(stdout);
      scanf("%Lf", &radius);

      plostina = pi * radius * radius;

      printf("Plostinata na krugot so radius ");
      printf("%.3Lf e %.12Lf\n", radius, plostina);

      return 0;
}
```




2. Да се напише програма која чита големи букви од тастатура и ги печати истите како мали букви.

```
#include <stdio.h>

int main()
{
      char ch;

      printf("Vnesi golema bukva ");
      fflush(stdout);
      scanf("%c", &ch);
      printf("Mala ekvivalent bukva na ");
      printf("'%c' e '%c'\n", ch, ch-'A'+'a');

      return 0;
}
```

3. Кој ќе биде излезот на следната програма?

```
#include <stdio.h>

int main(void)
{
	int i=0, j, k=7, m=5;
	j = m += 2;
	printf("j = %d\n", j); // j = 7

	j = k++ > 7;
	printf("j = %d\n", j); // j = 0

	j = i == 0 || --k;
	printf("j = %d\tk = %d\n", j, k); // j = 1 k = 8

	return 0;
}
```

4. Да се напише програма која ќе прочита два цели броја и ќе ја испечати нивната сума, разлика, производ и остатокот при делењето. Програмата исто така ќе прочита и со која прецизност треба да ги испечати двата броја.(дијаграмот го прикажува решението без прецизност)

```
#include <stdio.h>
int main()
{
  int prv, vtor, vkupno, dec;

  printf("vnesi dva broja ");
  fflush(stdout);
  scanf("%i %i", &prv, &vtor);
  printf("vnesi preciznost vkupno mesta decimalni mesta ");
  fflush(stdout);
  scanf("%i %i", &vkupno, &dec);

  printf("%i+%i=%0*i\n", prv, vtor, vkupno, prv+vtor);
  printf("%i-%i=%0*i\n", prv, vtor, vkupno,prv-vtor);
  printf("%i*%i=%0*i\n", prv, vtor, vkupno,prv*vtor);
  printf("%i/%i=%0*.*lf\n",prv,vtor,vkupno,dec,prv / vtor);
  printf("%i %% %i=%0*i\n",prv,vtor,vkupno,prv % vtor);

  return 0;
}
```

Излез:
vnesi dva broja 12345 56789
vnesi preciznost vkupno mesta decimalni mesta 9 3
12345 + 56789 = 000069134
12345 - 56789 = -00044444
12345 * 56789 = 701060205
12345 / 56789 = 00000.000
12345 % 56789 = 000012345

5. Кој е излезот од следнава програма

```
#include <stdio.h>
int main()
{
     double f=3.1416, g=1.2e-5, h=5001234567.0;
     printf("f=%f\tg=%f\th=%f\n",f, g, h);
     printf("f=%e\tg=%e\th=%e\n",f, g, h);
     printf("f=%g\tg=%g\th=%g\n",f, g, h);
     printf("f=%lf\tg=%lf\th=%lf\n",f, g, h);
     printf("f=%le\tg=%le\th=%le\n",f, g, h);
     printf("f=%lg\tg=%lg\th=%lg\n",f, g, h);
     printf("f=%7.2lf\tg=%.2le\th=%.4lg\n",f, g, h);

     return 0;
}
```

Излезот од програмата е:

f=3.141600 g=0.000012 h=5001234567.000000

f=3.141600e+000 g=1.200000e-005 h=5.001235e+009

f=3.1416 g=1.2e-005 h=5.00123e+009

f= 3.141600 g=0.000012 h=5001234567.000000

f=3.141600e+000 g=1.200000e-005 h=5.001235e+009

f=3.1416 g=1.2e-005 h=5.00123e+009

f=3.14 g=1.20e-005 h=5.001e-009