2 Константи, оператори, внес/приказ на податоци

1. 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:

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

```
#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
}
```

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.

2. Оператори

Постојат повеќе групи на оператори и тоа:

- Аритметички оператори (+, -, *, /, % (не работи со реални броеви))
- Релациони оператори (>, <, >=, <=, ==, !=)
- Логички оператори (&&, ||)
- Оператори за инкрементирање и декрементирање (++, --)

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.

Реални броеви наспроти 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 деленикот 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 rezultati
    h = f / g;   // realno delenje rezultat 1.25
    h = i / j;   // celobrojno delenje iako h e realen, rezultatot e 1.000

    return 0;
}
```

C 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 компајлерот 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.

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.

1. Ако операторот е пред операндот (++i) тогаш операндот ќе ја промени својата вредност пред да се искористи во понатамошниот тек на наредбата.
2. Ако меѓутоа, операторот е по операндот (i++) тогаш вредноста на операндот се менува откако истата е искористена.

Пример:

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

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 НЕ (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).

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

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

Пример 2:

```
# 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.

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

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:

```
    +=   -=   *=   /=   %=
```

```
израз1 оператор= израз2 ⇔(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).

```
#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
```

3. 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 I/O библиотека (stdio.h). 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: програма која три прочитани знаци ќе ги прикаже во обратен редослед

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

Внесување на податоци со scanf() функцијата

Влезните податоци може да се внесат во компјутерот од стандардниот влез со користење на библиотечната функција scanf(). Обликот на оваа функција е следниот:

```
scanf(Контролна низа од знаци, 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	Значење
c	Податочниот елемент е еден знак
d	Податочниот елемент е децимален цел број
f	Податочниот елемент е реална вредност
h	Податочниот елемент е краток цел број
I	Податочниот елемент е децимален, хексадецимален или октален цел број
o	Податочниот елемент е октален цел број

s	Податочниот елемент е низа од знаци по која следи едно празно место, нов ред или табулатор
u	Податочниот елемент е децимален цел број без знак
x	Податочниот елемент е хексадецимален цел број

Пример 1:

```
#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:

Pc 12345 570.34 или
```
Pc
12345
570.34
```

Пример 2:

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

Влез	**Излез**
1 2 3	a=1, b=2, c=3
123456789	a=123 b=456 c=789

Пример 3:

```
#include <stdio.h>
main()
{
   int i;
```

```
   float f;
   char c;
   ...
   scanf("%3d %5f%c",&i,&f,&c);
   ...
}
```

Влез	**Излез**
10256.875 T	i=102 f=56.87 c=5

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

Пример:

```
#include <stdio.h>
main()
{
   int i;
   float f = 3.0;
   char c = 'a';

   i = scanf("%f %c",&f,&c);
   printf("%d %f %c",i, f, c);
}
```

Влез	**Излез**
a c 5.0 c	0 3.0 a 2 5.0 c

Печатење на податоци со printf() функцијата

Излезните податоци може да се запишат од компјутерската меморија на стандардниот излез со користење на библиотечната функција printf. Општата форма на printf функцијата е

```
printf(Kontrolna niza od znaci, arg1, arg2, …, argn)
```

Каде контролната низа се однесува на низа од знаци која содржи информации за форматирањето. arg1, arg2, …, argn се аргументите кои ги претставуваат индивидуалните излезни податоци.

Пример 1:

```
# 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));
}
```

```
2.000000  3.000000  5.000000  2.236068
```

Пример 2:

```
#include <stdio.h>
int main(void)
{
   double f=3.1416, g=1.2e-5, h=5001234567.0;
   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;
}
```

```
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
```

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:

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

Пример 2:

```
#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:

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

```
#include <stdio.h>

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

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

     plostina = pi * radius * radius;

     printf("P na krug so r %.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(void)
{
     char ch;

     printf("Vnesi golema bukva ");
     scanf("%c", &ch);
     printf("Mala ekvivalent bukva na '%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.

```
#include <stdio.h>

int main(void)
{
    int prv, vtor, vkupno, dec;

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