Тема 4
Типове данни в език C/C++. Константи и променливи. Аритметични оператори.

1. Типове данни - класификация

Данните в програмите се разделят на константи и променливи. ***Променливите*** са данни, чиито стойности се изменят по време на изпълнение на програмата, докато ***константите*** са данни, които не се променят по време на изпълнение на програмата. Основна характеристика на всеки език за програмиране са типовете данни, които поддържа. Типът на дадена данна определя нейното съдържание т.е. от какъв вид е информацията, записана в данната: числов, символен, цели или реални (дробно-десетични) числа и т.н. Типовете на данните определят също и какви са операциите, който могат да бъдат изпълнени с тях.

Размерът на паметта, която се заделя за дадена данна също се определя от нейния тип т.е. за всеки тип данна се отделят точно определен брой байтове. Това се нарича още *размер на тип*. От своя страна, размерът на дадения тип определя обхвата на стойностите, които данната може да заема.

2. Аритметични типове данни

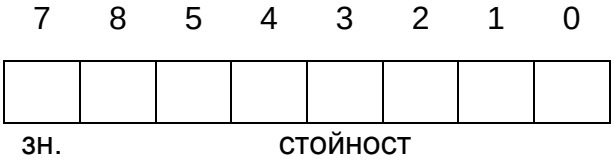
Основни типове данни в език C/C++ са *аритметичните типове*: ***int***, ***char***, ***float***, ***double***. Те могат да бъдат използвани съвместно със следните модификатори:

- ***signed*** - за знакови числа;
- ***unsigned*** - за беззнакови числа;
- ***short*** - за числа с единична точност;
- ***long*** - за числа с двойна точност.

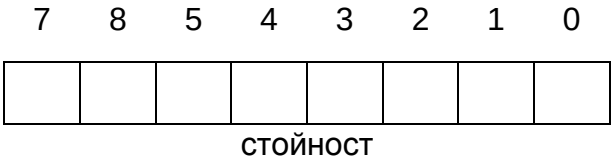
Модификаторите променят (модифицират) типа на данните.

2.1. Аритметичен тип *char*

Тип ***char*** е предназначен за представяне на символни данни (константи и променливи). Размерът на типа е 1 байт (8 бита) т.е. за дадена променлива от този тип се заделя 1 байт. При представяне на знаково число с 8 бита, най-старшият бит се използва като бит за знак: 0 за знак + и 1 за знак -, а останалите 7 бита са за стойността на числото (фиг. 4.1а). При беззнаковите числа за стойността на числото се използват всичките 8 бита (фиг. 4.1б). Като се има предвид, че ***char*** е знаков тип, стойността на дадена променлива или константа от тип е в границите на [-128, +127].



Фиг. 4.1а. Представяне на данна от тип char



Фиг. 4.1б. Представяне на данна от тип unsigned char

Въпреки, че тип ***char*** се използва за описание на символни данни, в програмни езици C и C++ той спада към аритметичните типове. Числената стойност на променлива от този тип е ASCII кода на символа.

Тип ***char*** може да се използва с модификатори ***signed*** и ***unsigned*** като:

- *unsigned char* представя беззнакови числа, чиито стойности са в обхват [0, 255];
- *signed char* е равносилно на ***char***.

2.2. Аритметичен тип *int*

Тип ***int*** е предназначен за представяне на цели числа със знак. За данна от този тип се заделят 2 или 4 байта в зависимост от настройките на компилатора. Обхвата на типа за знакови цели числа при 16 бита е в интервала [-2^{15}, $+2^{15}-1$], докато при 32 бита е в интервала [-2^{31},$+2^{31}$]. За цели беззнакови числа, обхвата при представяне с 16 бита е [0, $2^{16}-1$], а при 32 бита е [0, $2^{32}-1$].

Тип ***int*** може да се използва с всеки един от четирите модификатора. Съответно, вариантите на целочислените типовете са следните:

- *int* – цяло число със знак, представено в 16 или в 32 бита;
- *signed int* - цяло число със знак, представено в 16 или в 32 бита;
- *unsigned int* – цяло число без знак, представено в 16 или в 32 бита;
- *short int* - цяло число със знак, представено в 16 бита;
- *long int* - цяло число със знак, представено в 32 бита;

- *signed short int* – цяло число със знак, представено в 16 бита;
- *unsigned short int* - цяло число без знак, представено в 16 бита;
- *signed long int* - цяло число със знак, представено в 32 бита;
- *unsigned long int*- цяло число без знак, представено в 32 бита.

В езици C и C++ тип ***int*** е по подразбиране. Съответно, ако модификатор се използва без да е посочен тип, то по подразбиране това е тип ***int***.

2.3. Аритметични типове *float* и *double*

Тип ***float*** служи за представяне на реални (дробно-десетични) числа. За данна от този тип се заделят 4 байта. Обхвата на променлива от тип float е [±10E-37, ± 10E+37]. Аритметичен тип ***float*** може да се използва с модификатор ***long***. В този случай, ***long float*** е равносилно на тип ***double***.

Чрез тип ***double*** се представят реални (дробно-десетични) числа с двойна точност. При представяне с 64 бита (8 байта), обхвата на променлива от този тип е [±10E-307, ± 10E+307]. Тип ***double*** може да се използва с модификатор ***long***, като ***long double*** се заделят 10 байта (80 бита).

3. Променливи и константи

3.1. Променливи

Както беше споменато, променливите са данни, чиито стойности се изменят в процеса на изпълнение на програмата. Всяка променлива има символно име (идентификатор), равнозначно на адреса й в паметта, където се съхранява съдържанието (стойността) на променливата. В този аспект, всяка променлива имат три характеристики: **име**, **тип**, **стойност**.

Име на променливата е идентификатор, който служи като символно име за нейното означаване. Името се дефинира по правилата за съставяне на идентификатори – допустими символи са латинските букви (главни и малки), цифрите (0-9) и символ за подчертаване (_). Идентификатор не може да започва с цифра и ключовите думи не могат да са идентификатори.

Типът определя вида на стойността на променливата (цяло число, дробно число, символ), както и необходимия обем памет, който трябва да се задели за променливата.

Стойността на променливата е нейното съдържание, което се записва в клетката памет и може да бъде променено в процеса на изпълнение на програмата.

Използваните променливи задължително трябва да бъдат дефинирани. По този начин компилаторът заделя необходимото количество памет

за дадената променлива, съобразно типа ѝ. Дефиниция на променлива е чрез нейните име и тип:

тип име [=стойност];

Задължителни елементи са името и типа на променливата. Едновременно дефинирането на променлива може да бъде съпроводено и с инициализация т.е. със задаване на начална стойност на променливата. Задаването на начална стойност не е задължително и поради тази причина стойността е поставена в скоби ([]) като незадължителен елемент.

На един ред могат да бъдат дефинирани повече от една променливи, които са от един и същи тип. Общият вид на дефиницията на променливи е следния:

тип име [=стойност][, име =[стойност],…];

Примери за дефиниции на променливи са:

```
int x,y;     // дефиниране на променливи x,y от целочислен тип (int)

float length; // дефиниране на променлива lenght от реален тип (float)

double alpha; /* дефиниране на променлива alpha от реален тип с двойна точност (double) */

unsigned j;  /* дефиниране на променлива j като целочислена без знак (тип unsigned int) */

char ch='Y'; /* дефиниране на променлива ch от символен тип (char) и инициализиране с начална стойност 'Y' */

float pi=3.14;     /* дефиниране на променлива pi от реален тип (float) с начална стойност 3.14 */

unsigned char h;   /* дефиниране на променлива h от тип unsigned char (символен без знак) */

int z=2,k;              /* дефиниране на променливи z с начална
                    стойност 2 и k от тип int */
```

3.2. Константи

Константите са ***данни, чиито стойности не се променят по време на изпълнение на програмата***. Подобно на променливите, константите имат ***тип***, обозначават се чрез ***символни имена (идентификатори)***, но стойностите им са постоянни.

При дефиниране на константа, при всяко нейно срещане в програмата, компилаторът заменя символното име със стойността ѝ. Предимствата на използване на константи са следните: ако в процеса на разработка на програмата се наложи промяна на параметър, определен чрез константа, единствената корекция в тази програма е замяна стойността на една константа. В случай че не е използвана константа, а само стойност, в тогава ще се наложат корекции навсякъде в програмата, където се среща параметърът. В този смисъл, използването на константи е едно улеснение за програмиста.

В език C/C++ константите се разделят на числени (цели или дробни), символни и низови. Обикновено, типа на константата се задава неявно, чрез стойността ѝ. Например, константа със стойност **3.14** е от тип ***float***; **65, -1** - от тип ***int***; **'S'** - от тип ***char***.

Дробните константи по подразбиране са от тип ***float***. Ако реалното число е прекалено малко или прекалено голямо и е извън обхвата на тип ***float***, константата е от тип ***double***.

В език C++ съществува начин за дефиниране на константи чрез използване на ключова дума ***const***. Общият вид на дефиницията е следния:

const [тип] име=стойност;

В случая типът на константата е незадължителен атрибут и се определя от стойността ѝ[1].

Примери за дефиниране на константи са:

```
const int N=100;
const float PI=3.14;
```

Дефинирани са константи: N със стойност 100 (тип **int**) и PI със стойност 3.14 (тип **float**).

Ако типът на константата не бъде явно посочен[2], използва се стойността ѝ за да се зададе тип по подразбиране:

```
const Y=3;   // типът на константата е int
const Z=0.8; // типът на константата е float
```

В език C не съществува ключова дума **const**. За дефиниране на константи в C се използва директива на предпроцесора **#define**.

Примери:

```
#define      N     100
#define      PI    3.14
#define      Y     3
#define      Z     0.8
```

Тези дефиниции са идентични с горепосочените, при които се използва ключова дума **const**.

4. Пресмятане размера на тип

За пресмятането размера на тип в програмен език C/C++ е предвидена операция (оператор) ***sizeof***. Операцията съществува в две форми:

sizeof(тип)

---

[1] В последните версии на Visual Studio е задължително типът на константата да бъде посочен.
[2] Виж 1.

sizeof(унаренИзраз)

В резултат на изпълнението си, оператор ***sizeof*** връща цяло число от тип ***int***, равносилно на количеството памет в байтове, необходими за типа на операнда, посочен в скобите. Операндът или е тип, или е унарен израз. Във втория случай, ***sizeof*** връща броя байтове, необходими за типа на резултата.

Примери:

```
sizeof(short int)  // резултатът от израза е 2
sizeof(long int)   // резултатът от израза е 4
sizeof(char)       // резултатът от израза е 1
sizeof(float)            // резултатът от израза е 4
sizeof(double)     // резултатът от израза е 8
```

Оператор ***sizeof*** намира приложение, когато е необходимо да бъде уточнен размера на обекта. Например, когато е необходимо да бъде заделен блок памет за масив от данни, необходимо е да се знае размера на всеки елемент в брой байтове. В този случай е удобно да се използва оператор ***sizeof***.

5. Аритметични оператори

В програмните езици като елементи от програмите са предвидени оператори, чрез които се извършват различни математически, логически или други действия (операции) върху данните. Език C/C++ поддържа аритметични и логически операции, сравнения (отношения) и побитови операции. Данните, необходими за извършване на дадена операция (оператор) се наричат операнди. В зависимост от броя на операндите операторите се разделят на ***унарни*** и ***бинарни***. Бинарните оператори използват два операнда, докато в унарните участва само един операнд.

Бинарни аритметични оператори в език C/C++ са:

- събиране (+);
- изваждане (-);
- умножение (*);
- деление (/);
- деление по модул (%).

Като оператори ***събиране***, ***изваждане*** и ***умножение*** са напълно идентични със съответните математическите операции. Особеност на оператор ***деление*** е, че когато операндите са реални числа, резултатът също е реално число, докато при деление на целочислени числа, резултатът е цяло число т.е. делението е целочислено. Пример:

```
int x=2, y=9, z;
float p=2, q=9, r;
```

```
z=y/x;        // резултат z=4
r=q/p;        // резултат r=4.5
```

Резултатът от оператор ***% (деление по модул)*** е остатък от целочислено деление. Пример:

```
int x=2, y=6, z;
int p=3, q=5, r;
z=y%x;        // резултат z=0
r=q%p;        // резултат r=2
```

Оператор ***деление по модул*** се използва само за целочислени стойности и е неприложим за числа от тип ***float*** и ***double***.

Унарни аритметични оператори в C/C++ са:

- унарен + (запазване на знака на числото);
- унарен - (промяна на знака на числото);
- инкрементиране ++ (увеличаване на стойността с 1);
- декрементиране -- (намаляване на стойността с 1).

Съществуват две форми на операторите инкрементиране и декрементиране:

- префиксна ++i, --i;
- суфиксна i++, i--.

Разликата между двете форми е, че при префиксната форма първо се инкрементира (декрементира) стойността, след което се изпълнява другият оператор. При суфиксната форма е обратно – инкрементирането (декрементирането) се изпълнява след другият оператор. В следващият пример оператор инкрементиране (++) се използва съвместно с оператор присвояване (=).

```
int x=15;
int y;
y=++x;
// резултантните стойности са: y=16; x=16.

int x=15;
int y;
y=x++;
// резултантните стойности са: y=15; x=16.
```

В резултат, при първият случай, когато се използва префиксната форма на инкрементирането, стойностите на променливите x и y са равни, тъй като стойността на първо се увеличава с единица и след това се

присвоява на променливата y. Във втория, стойността на променливата x първо се присвоята на y и едва след това се инкрементира.

Относно въпроси по темата на адрес:

ln_zh_st@yahoo.com