Тема 3
Основи на програмиране на програмен език C/C++. Азбука, основни елементи. Структура на програма

1. Синтаксис на език C/C++. Основни елементи

1.1. Основни елементи на формалните езици

Според теорията на формалните езици, синтаксисът на всеки език за програмиране се състои от следните основни елементи: ***азбука***, ***лексика***, ***граматика***.

Азбука на език за програмиране се определя като ***множеството допустими символи в езика***. Това множество е крайно и предварително фиксирано.

Азбуката на език C/C++ включва следните символи:

* латински букви: ***A - Z a - z***

* цифри: ***0 – 9***

* специални символи: ***~ ! @ # $ % ^ & * ( ) + - _ { } [ ] : ; ‘ “ < > . , ? / =***

Лексика на език за програмиране определя правилата за образуване на думите, наречени още ***лексеми***. ***Думи*** (лексеми) в даден програмен език са: ***ключови думи; числени и символни константи; идентификатори; символни низове; операции***.

Граматика на език за програмиране включва правилата за образуване на изречения (***statement***). Като изречения в C/C++ са оформени ***декларациите***, ***дефинициите*** и ***операторите***. Като ***оператор*** в език C/C++, за разлика от други езици, се приема изпълним оператор, който указва действие, което ще бъде изпълнено. В C/C++ изреченията завършват със символ **;** (точка и запетая).

1.2. Лексеми в C/C++

Основни елементи в C/C++ са следните лексеми:

- ***Ключови думи***

Ключови думи са резервираните в език за програмиране думи. Чрез тях се означават: оператори, модификации, типове данни, декларации и др.

Например, ключови думи за език C са:

auto	***break***	***case***	***char***	***continue***	***default***
do	***double***	***else***	***enum***	***extern***	***float***
for	***goto***	***if***	***int***	***long***	***while***

В език C++ този набор е допълнен. Например, добавени са нови ключови думи, като:

const class private public protected virtual

Ключовите думи за език C, както и за C++ са дефинирани от стандарт ANSI, но конкретните компилатори могат да поддържат и ключови думи, които са извън този обхват. Например MS Visual C++ 5.00 и следващи версии поддържат ключова дума bool за логически (булев) тип данни. Ключовите думи се записват с малки букви. Списъкът на ключовите думи в C++ е даден в Приложение 1.

- ***Идентификатори***

Идентификаторите са имена, чрез които се означават константи, променливи, типове, функции и други компоненти на програмите. Често терминът ***идентификатор*** се заменя с термина ***име***.

Имената в език C/C++ се образуват от поредица латински букви (главни и малки), цифри и символ за подчертаване (_), като условието е името да не започва с цифра. Ключовите думи са резервирани и не могат да служат като идентификатори.

Примери за валидни идентификатори са:

Alpha beta i i1 j _Load massiv pointer

SaveFile numberOfStudents

Примери за грешно дефинирани имена са:

5i – започва с цифра;

char – ключова дума;

mass#1 – не е позволено използването на символ #.

При дефинирането на имена, в C/C++ се прави разлика между големи и малки букви. Например, идентификатор **ALFA** е различен от **alfa**.

- ***Числени и символни константи***

Числените константи са цели и дробни числа. Дробните числа могат да бъдат представени във формат *фиксирана запетая* (**F** формат) или в *експоненциален* формат (*плаваща десетична запетая* – **E** формат).

Примери за дробни константи в F формат са: **0.15; 12.1; 34.794; 89.5**.

Примери за дробни константи в E формат са: **6.7e-2; 3.8e+5; 4.5E-7**.

Обърнете внимание, че в програмен език C/C++, за десетична запетая се използва символ **.** (десетична точка), както е прието в изчислителната техника. Символи E, e представят експонента. Така записите **6.7e-2** и **6.7E-2** съответстват на математически запис **6,7x10^{-2}**.

Целите константи могат да бъдат задавани в десетична, шестнадесетична или осмична бройна система. Особеност на C/C++ е използването на префикси **0x** (нула x), **0X** за шестнадесетични числа и префикс **0** (нула)

– за осмични числа. Примери за десетични константи са: **254**; **196**; **0**; **-45**; за шестнадесетични константи: **0x23**; **0x1a**; **0X7F**; **0xff**; за осмични константи: **031**; **071**; **023**. Използването на главна или малка буква X(x), определя дали с главна или малка буква ще бъдат записани шестнадесетичните стойности A, B, C, D, E, F.

Докато десетичните константи могат да бъдат положителни и отрицателни числа, осмичните и шестнадесетичните са беззнакови т.е. само положителни числа и нула.

Символните константи са разновидност на числовите. Означават се като символ, заграден в апостроф. Примери за символни константи са: **'a'**, **'H'**, **'2'**. Стойността на една символна константа е нейният ASCII код. Например стойността на константа 'A' е 65 (0x41).

В C/C++ има специални символни константи, съответстващи на управляващи символи. Те се обозначават с **** (наклонена черта). Такива константи са:

'\0' – управляващ символ за край на низ;

'\n' - управляващ символ за нов ред;

'\b' - управляващ символ backspace;

'\t' - управляващ символ за хоризонтална табулация;

'\v' - управляващ символ за вертикална табулация;

'\r' - управляващ символ за начало на ред;

\' – символ апостроф;

\" – символ кавички;

**** - символ обратна наклонена черта.

- ***Символни низове***

Символните низове (***strings***) се състоят от множество символи, заградени в кавички. Например:

"This is a string"

"\nТова също е низ!"

Елементите на символните низове се подреждат последователно в паметта, като всеки елемент (символ) заема 1 байт. Интервал ( ) също е символ и съответно заема един байт. Последният байт на всеки низ е резервиран за край на низа. Това е символа с ASCII код 0 или символ **NULL**, съответстващ на константа **'\0'**. Следователно, ако даден символен низ съдържа **n** символа, в паметта на компютъра, този низ заема **n+1** байта. Последния байт е със стойност '\0'.

- ***Операции (operators)***

Операциите (или оператори) в език C/C++ се разделят на аритметични, сравнения, логически и побитови. Обозначават се чрез мнемонични означения. Например:

+ - * / % != < > <= >= && || ! & | ~ ^

- ***Разделители***

Разделителите в програмен език са символи или групи символи, чрез които се разделят думите една от друга. В език C/C++ като разделители от компилатора се приемат следните символи: интервал; вертикална и хоризонтална табулация; символ за нов ред.

1.3. Коментари

Коментарът е текст, който не се обработва от компилатора и служи за пояснение на програмния код. Добавянето на коментар в програмата подобрява нейната четливост и служи за улеснение, както на програмиста, който създава програмата, така и на други програмисти, които ще доразвиват програмната система. Всички езици за програмиране притежават езикови средства за задаване на коментари.

В език C, коментар се задава като текст, заключен между символи ***/* ... */*** По този начин могат да се задават коментари на един или на повече от един ред. Например:

```
/* Това е коментар! */


/*
Това също е коментар!
*/
```

Освен посочения, в C++ е предвиден друг начин за задаване на коментар, който се разполага само на един ред. В началото на реда се задават символи ***//*** и компилаторът третира като коментар текста до края на реда. Пример:

```
// коментар на един ред
```

2. Структура на програма

Общата структура на програма, написана на език C може да се представи чрез следните компоненти:

```
// включване на заглавни файлове
#include <math.h>
#include <stdio.h>


// дефиниции на външни променливи
int number;
float b;
```

```
// главна функция
 main()
{
// тяло на главната функция
...
}

// дефиниця на функция 1
int function1(int a, int b)
{
……
}

// дефиниця на функция 2
float  function2(int a, int b)
{
……
}
```

Всяка програма, написана на език С се състои от отделни програмни единици, наречени функции. Броят на функциите в програмата е неограничен, но винаги трябва да има една функция, дефинирана като главна – това е функция **main()**.

Функция main() осъществява връзката с операционната система. При стартиране на изпълнимия модул (.COM или .EXE), операционната система предава управлението на главната функция main(). Съответно main функцията може да извиква други функции т.е. да им предава управлението. След завършване, функция main() връща управлението на операционната система.

Функция в език С е набор декларации и последователно написани оператори на езика, осъществяващи действия, които представляват логическа цялост и реализират определен алгоритъм.

Общият вид на една функция е:

```
тип ИмеНаФункция (списък параметри)
{
 // тяло на функцията, състоящо се от:
  // дефиниции на променливи;
  /* оператори и конструкции, които реализират действието на функцията;
```

```
*/
}
```

Общият вид на програма на C++ не се различава съществено от програма на C. Трябва да се има предвид, че в общия случай в нея са включени декларации и дефиниции на класове и обекти. Освен това, в голяма част от случаите модулите са записани в отделни файлове и се прилага т.нар. ***разделно компилиране***. Това означава, че файловете, съдържащи първичен код се компилират поотделно и в последствие се свързват в един общ изпълни модул.

Тема 4
Типове данни в език С. Константи и променливи

1. Общи сведения

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 С са *аритметичните типове*: ***int***, ***char***, ***float***, ***double***. Те могат да бъдат използвани съвместно със следните модификатори:

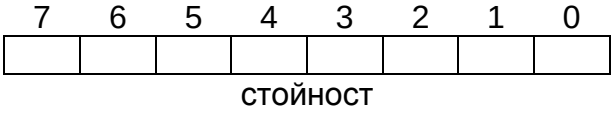
- ***signed*** - за знакови числа;
- ***unsigned*** - за беззнакови числа;
- ***short*** - за числа с единична точност;
- ***long*** - за числа с двойна точност.

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*** както:

- *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 $[-2^{31},+2^{31}-1]$. 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 й. 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 ини-

циализация 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 С 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***.

При целите числови константи се допуска използване на суфикси **U (u)** и **L (l),** при което константата се определя като тип ***unsigned*** и ***long***. Например:

23U, 5u - константите са от тип ***unsigned int***;

19L, 200l - константите са от тип ***long int***;

100UL, 10Ul, 500uL, 1000ul - константите са от тип ***unsigned long int***.

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.

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, 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 предишните два примера, явното посочване типа на константите е безсмислено. Посочените типове са по подразбиране. Явно задаване тип на константа има смисъл само ако този тип не се подразбира от стойността, например, ако желаем константата PI да е от тип **double**, а не от тип **float**:

```
const double PI=3.14;
```

Или, ако желаем да дефинираме N като беззнаково число:

```
const unsigned int N=100;
```

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 е предвидена операция (оператор) ***sizeof***. Операцията съществува в две форми:

sizeof(тип)

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***.