ХИМИКОТЕХНОЛОГИЧЕН И МЕТАЛУРГИЧЕН УНИВЕРСИТЕТ - СОФИЯ

ИНФОРМАТИКА
част първа

лектор: доц. д-р Атанас Атанасов
Катедра “Програмиране и използване на компютърни системи”

Лекция 8

ЕЗИК
ЗА
ПРОГРАМИРАНЕ
C++

продължение

ЕЗИК ЗА ПРОГРАМИРАНЕ C++

- **Структура на програмата на C++**
  - изходен файл
  - заглавен блок с коментари
  - заглавни файлове
  - константи
  - класове
  - глобални променливи
  - функции
- **Скаларни типове данни в C++**
  - Булев
  - Цял
  - Реален
  - Символен

Структура на програмата на C++

Когато една програма написана на езика C++ е малка по обем се записва в един изходен файл.

В случаите, когато програмата е голяма или се разработва от колектив от програмисти е удобно тя да се раздели на части, които се записват в отделни изходни файлове.

Предимствата на този подход са следните:

- Сложният проблем, който ще се решава се разделя на отделни задачи, които могат да се програмират от различни програмисти и се оформят като отделни изходни файлове;

- Оделните файлове могат да се използват многократно и ако няма промени в тях, не се налага да се компилират всеки път.

Когато една програма включва няколко изходни файла, трябва да се укаже на компилатора как да се компилира и изгради цялата програма.

Структура на програмата на C++

Какво представлява една програма, написана на C++?

Тук е даден пример с програма за намиране обема на правоъгълна призма със страни на основата a=6.4, b=5.6 и височина h=12.5.

Програмата е записана като изходен файл с име pr1.cpp. За пресмятане обема на призмата се използват следните зависимости:

V=**S** x **h** , където

S= **a** x **b**

```
// Program Pr1.cpp
# include <iostream.h>
int main()
{
    double a=6.4;
    double b=5.6;
    double h=12.5;
    double s,v;
 /* Namirane liceto na osnovata */
    s=a*b;
 /* Namirane obema na prizmata */
    v=s*h;
 /* Izvejdane liceto na osnovata */
    cout <<"s=" << s <<"\n";
 /* Izvejdane obema na prizmata */
   cout <<"v=" << v <<"\n";
 return 0;
}
```

Структура на програмата на C++

- На първия ред е записан коментар, който може да бъде на кирилица или латиница. Предназначен е за програмиста и показва, че следващия фрагмент е програма написана на **C++** с име **pr1.cpp**.

- На втория ред е записана директива към компилатора на C++, която указва, че към файла **pr1.cpp** трябва да се включи и файл с име **iostream.h**, който съдържа стандартните функции за поточен вход и изход (**cin** и **cout**).

- Текстът от третия до последния ред дефинира една функция, наречена главна (**main**). Всяка програма на C++ задължително има една главна функция, но може да съдържа и други функции.

Описанието на фунцията **main** започва със служебната дума **int,** която означава, че след изпълнението връща като резултат цяло число. Тялото на функцията е заградено с големи скоби {.....} и включва дефиниции и инструкции, реализиращи алгоритъма за решаване на задачата.

Структура на програмата на C++

Инструкциите в тялото се отделят една от друга с “ **;** ” и се изпълняват последователно.

Последната инструкция е **return 0** и означава край на функцията, а стойността 0 показва, че функцията е изпълнена правилно.

Инструкциите, записани на редове от 5 до 8 дефинират, че променливите **a,b,h,s,v** са от реалния тип **double**, като **a,b,h** са инициализирани (зададени са им конкретни стойности), а **s,v** ще получат стойности по време на изпълнение на програмата.

Редовете от програмата /*.....*/ са също коментари. Коментарите в една програма се игнорират от компилатора. В разгледания пример те са използвани за подсещане на програмиста за следващите действия в програмата.

Структура на програмата на C++

Редовете в които е записано: **s=a*b** и **v=s*h** са инструкции за присвояване. Използват се за пресмятане на изрази.

Редовете, които започват със служебната дума **cout** са инструкции за поточен изход на информация върху екрана на дисплея.

Чрез тези инструкции на екрана се извеждат стойностите на променливите **s** и **v.**

Символният низ “**\n**” се използва за преминаване на курсора на нов ред.

Със символите **<<** се означава операцията извеждане на информация на екрана или на прозорец от екрана.

Структура на програмата на C++

След като разгледахме един конкретен пример, нека да обобщим, каква е структурата на изходния файл (модул) в общия случай:

```
<изходен файл> ::= <заглавен блок с коментари>
                   <заглавни файлове>
                   <константи>
                   <класове>
                   <глобални променливи>
                   <функции>
```

Заглавен блок. Всеки модул започва със заглавен блок с коментари. Коментарите могат да съдържат информация за предназначението на модула, за компилатора и операционната система, за програмиста, за датата на създаване, пояснения за данните и др.

Структура на програмата на C++

- Заглавни файлове.

В тази част се изброяват всички необходими файлове, които ще осигурят нормалното изпълнение на модула. За разделител на заглавните файлове се използва знака за нов ред (натискане на клавиша **Enter**).
В примера по-горе заглавният файл е:

 # include <iostream.h>

- Константи.

Описват се всички константи, които са необходими за изпълнението на модула. Описанието на константите включва и дефиниция за тип на данните, които им се присвояват:

const <име на тип> <име на константа> = <израз>

Структура на програмата на C++

Често имената на константите се записват с главни латински букви, за да се различават от имената на променливите.

Например:

```
const  double  PI=3.142;
const  int  MAXINT= 32767;
```

- Класове.

На това място се записват дефинициите на класовете, които ще се използват в модула. В езика C++ се използват стандартни класове/типове от данни (int, double, float, char,..). Този стандартен набор от класове може да се разшири с дефиниране на нови потребителски класове данни (масиви, записи, файлове данни, и др.)

Структура на програмата на C++

-Глобални променливи.

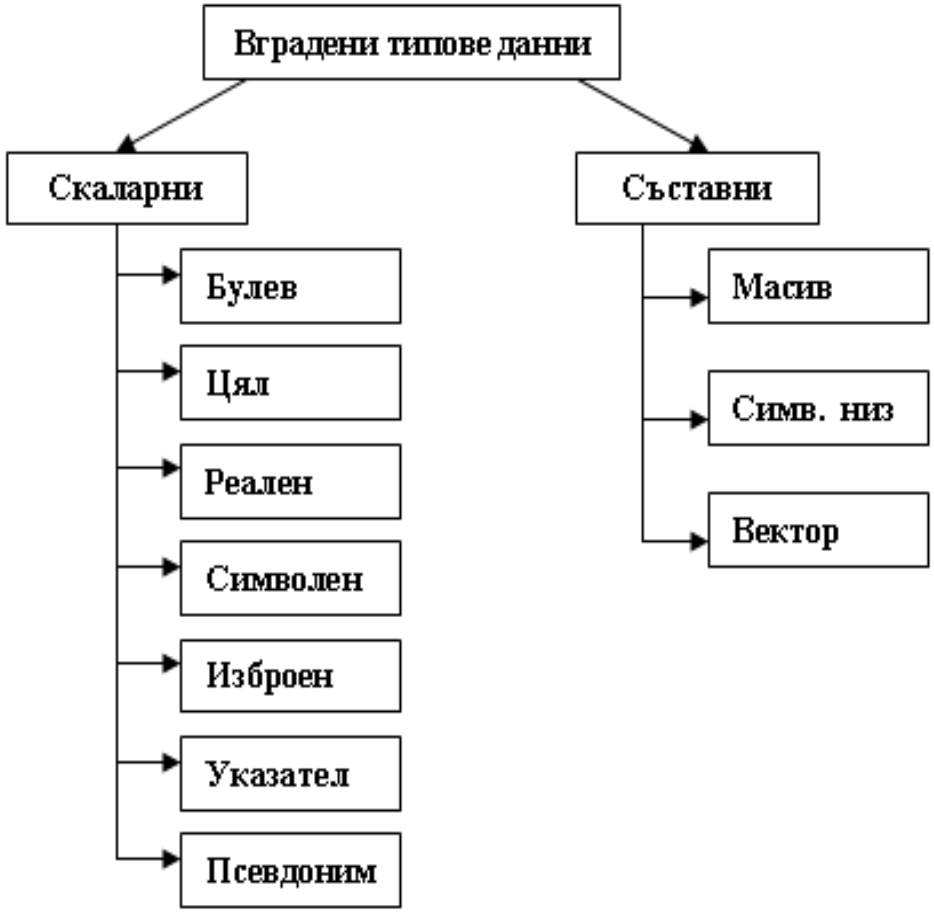
Това са променливи, които се дефинират извън функциите и са валидни за всички функции, дефинирани след тях в модула. Дефинират се по същия начин, както се дефинират и локалните променливи (тези които са валидни само в обхвата на функцията, където са описани).

- Функции.

Всеки модул задължително съдържа функцията **main**, но може да съдържа и други функции и те се изброяват в тази част на модула. Ако при описанието отделните функции са подредени така, че всяка една е описана преди да бъде извикана, то функцията **main** трябва да бъде описана последна.

Скаларни типове данни в C++

Езикът C++ е много мощен по отношение на типовете данни, които се използват в програмите. Най-общо типовете данни, с които разполага езика могат да се разделят на две групи: вградени и абстрактни. Вградените типове данни са предварително дефинирани и се подържат от ядрото на езика C++, а абстрактните типове се дефинират от програмистите. Една класификация на вградените типове данни е дадена вляво.



Скаларни типове данни в C++

Логически тип данни.

Това е стандартен тип данни, който се означава със служебната дума **bool** от boolean (на Джон Бул).

Данните от този тип приемат две възможни стойности: **true**-истина и **false**- лъжа, които се наричат булеви константи. Променливите, които приемат една от тези две стойности се наричат логически (булеви) променливи и за всяка от тях се отделя по 1 байт памет. Пример за дефиниране на булеви променливи:

bool b1,b2; -неинициализирани променливи;

bool b3=true; - инициализирана променrueлива.

При вътрешното представяне на булевите константи **true** се представя с 1-ца или число различно от 0, а **false** с 0.

Скаларни типове данни в C++

Логически операции над данни от тип bool.

-Конюнкция (логическо умножение) - това е бинарна операция, за означаването на която се използват знаците && или and. Нарича се бинарна, защото свързва два операнда. Например:

- **A and B или A && B - Този логически израз приема стойност true само тогава, когато и A и B са true, а във всички останали случаи приема стойност false.**
- **-Дезюнкция (логическо сумиране) – Също е бинарна операция и се означава със знаците or или || . Например:**
- **A or B или A || B – Този логически израз приема стойност true , тогава, когато поне едната от двете логически променливи е true.**

Скаларни типове данни в C++

-Отрицание. Това е унарна операция (участва само един операнд) и се означава със знака not или !. Например:

not A **(!A)** – Този логически израз приема стойност **true**, когато **A** е **false** и обратно.

-Отношение. Два аритметични израза, свързани по между си със знак за съотношение образуват логическото отношение. В C++ се използват следните знаци за съотношение:

- **= =** **знак за равенство;** **!=** **знак за различно;**
- **>** **знак за по-голямо;** **>=** **знак за по-голямо или равно;**
- **<** **знак за по-малко;** **<=** **знак за по-малко или равно.**

Скаларни типове данни в C++

Въвеждането на стойности за булеви променливи е недопустимо с инструкцията за въвеждане. Например:

```
bool  b1;
cin >>b1;
```

е недопустимо, но се допуска

```
bool  b1=true.
```

Извеждането на стойности за булеви променливи може да се извърши с инструкцията за извеждане:

```
cout << b1;
```

Скаларни типове данни в C++

Числови типове данни.

Числовите данни се разделят на две основни групи: целочислени и реални. Диапазонът на изменение на тези данни зависи от хардуерната реализация на езика.

Целочислен тип

Това е стандартен тип от данни за езика C++ и се определя с описателя за тип **int** от integer. Общият вид на записа на едно цяло число в C++ е:

<цяло число> ::= [+|-]<цяло без знак>

Под цяло без знак ще се разбира цифра или последователност от цифри. Както в математиката знакът + може да се пише пред числото, но може и да се пропуска. Езикът C++ допуска целите числа да се записват в десетична, осмична и шестнадесетична бройни системи.

Скаларни типове данни в C++

Осмичните числа започват с 0... , а шестнадесетичните с 0x... За всяка променлива от типа **int** се отделят в паметта по 4 байта, независимо от това дали е инициализирана или не. С данни от тип int са допустими следните операции:

-унарни : +j, -i, -365

-бинарни: +, -, *, /, % .

Например: **13/5** дава стойност **2**, това е целочислено деление, а **13%5** дава стойност **3** и това е остатъка от целочисленото деление.

В C++ се допуска и последователното използване на два знака за аритметични операции един до друг, без това да води до грешки:

5 - +4 дава резултат **1**, а така също и **5 + - 4**.

Изразът **5 * - 4** дава резултат -20.

С данни от тип **int** могат да се извършват и логически операции и да се пресмятат отношения., Всяко число различно от 0 се приема за true, а нулата за false.

Скаларни типове данни в C++

Осмичните числа започват с 0... , а шестнадесетичните с 0x... За всяка променлива от типа **int** се отделят в паметта по 4 байта, независимо от това дали е инициализирана или не. С данни от тип int са допустими следните операции:

-унарни : +j, -i, -365

-бинарни: +, -, *, /, % .

Например: **13/5** дава стойност **2**, това е целочислено деление, а **13%5** дава стойност **3** и това е остатъка от целочисленото деление.

В C++ се допуска и последователното използване на два знака за аритметични операции един до друг, без това да води до грешки:

5 - +4 дава резултат **1**, а така също и **5 + - 4**.

Изразът **5 * - 4** дава резултат -20.

С данни от тип **int** могат да се извършват и логически операции и да се пресмятат отношения., Всяко число различно от 0 се приема за true, а нулата за false.

Скаларни типове данни в C++

С помощта на специални модификатори в езика C++ се използват и други целочислени типове данни:

№	Тип	Диапазон	Необходима памет
1	**short int**	-32768 до +32767	2 байта
2	**unsigned short int**	0 до 65535	2 байта
3	**long int**	-2147483648 до 2147483647	4 байта
4	**unsigned long int**	0 до4294967295	4 байта
5	**unsigned int**	0 до4294967295	4 байта

При дефинирането на тези типове служебната дума **int** може да се пропуска, така че **short int** става **short,** а **long int** става **long.**

Скаларни типове данни в C++

Реален тип

Това са данни, които в процеса на обработка могат да приемат приблизителни стойности. В записа на реалните числа задължително се съдържа десетична точка. В C++ се използват два типа реални данни: **float** и **double.** За всяка данна от тип **float** в паметта се отделят по 4 байта, а за тези от тип **double** по 8 байта. И двата типа са стандартни за езика C++. Реалните данни от тип **double** се изменят в диапазона от -1.74x10308 до +1.7x10308 .

Реалните числа могат да се записват във **F**-форма с фиксирана запетая - използва се за не много големи и не много малки числа и **E**-форма с плаваща запетая (експоненциална форма) - за много големи и много малки числа. Общият вид на едно реално число в **F**-форма е:

<цяло число> . <цяло число без знак>,

Например: 456.123 или -79.7788899

Скаларни типове данни в C++

Общият вид на едно реално число в **E**-форма е:

<цяло число> . <цяло число без знак> E <порядък>

или **<цяло число > E <порядък>**

Например: **123 e-7** **3456 E4**

И в двата случая порядъкът се записва като цяло число. При записа на числата в експоненциален формат може да се използва и малката буква “**e**” и голямата “**E**”

Над реалните числа могат да се прилагат основните аритметични операции **(+, - , * , /).** Ако в един израз участва поне една данна от реален тип, то стойността на целия израз е реална.

Над реалните числа могат да се прилагат основните логически операции (конюнкция, дезюнкция и отрицание) и операциите за сравнение(= =, !=, >, >=, <, <=). Правилото е същото. Реални числа различни от 0 се приемат за **true**, а 0-**false**.

Скаларни типове данни в C++

Над реалните данни могат да се прилагат вградени (стандартни) функции, които при аргумент от цял или реален тип връщат резултат от тип **double**. Някои от по-често използваните функции са дадени в таблицата:

№	функция	пресмята
1	pow(x,n)	x^n , (x,n са реални от тип **double)**
2	sqrt(x)	Корен квадратен от x, $x \geq 0$
3	fabs(x)	Абсолютната стойност на x, $\|x\|$
4	exp(x)	Експонента, e^x
5	log(x)	Натурален логаритъм ln x, $x>0$
6	log10(x)	Десетичен логаритъм lg10 x, $x>0$
7	ceil(x)	Най-малкото цяло $\geq$ x (x е в тип **double**)
8	floor(x)	Най-голямото цяло $\leq$ x (x е в тип **double**)
9	sin(x)	Синус, sin x, **x** е в радиани
10	cos(x)	Косинус, cos x, **x** е в радиани
11	tan(x)	Тангес, tg x, **x** е в радиани
12	asin(x)	Аркуссинус, arcsin x, $\mathbf{x} \in (-\pi/2, \pi/2)$
13	acos(x)	Аркускосинус, arccos x, $\mathbf{x} \in (0 , \pi)$
14	atan(x)	Аркустангес, arctg x , $\mathbf{x} \in (-\pi/2, \pi/2)$
15	sinh(x)	Синус хиперболичен, sh x

Скаларни типове данни в C++

Стандартните математически функции се намират в библиотеката **math.h** и, за да се използват в една програма е необходимо в частта на заглавните файлове да се включи компилаторната директива:

 # include <math.h>

Примери за използване на някои стандартни функции:

$(x-1)^2$ = pow((x-1),2);

|2.45 x Z - 12.54| = fabs(2.45*z-12.54);

ceil(21.328)=22.0 ; ceil(36)=36.0 ;

ceil(-21.328)= -21.0; floor(21.328)=21.0;

floor(36)=36.0; floor(-21.328)=-22.0;

Скаларни типове данни в C++

При пресмятане на аритметични изрази, записани по правилата в C++. се спазва следният приоритет на действия:

- вградените функции (с най-висок приоритет);
- действията в скобите (започва се от най-вътрешните);
- унарните операции – “+” и “-“;
- бинарните операции – “ * ”, “ / ”, “ % ”;
- бинарните операции –“+” и “-“;
- въвеждане и извеждане – “>>” , “<<”(с най-нисък).

Един аритметичен израз може да съдържа операнди от различен тип. При пресмятането на стойността на такъв израз се извършва автоматично преобразуване на типовете на отделните операнди. Без загуба на точност се преобразуват: **bool,** в който и да е числов тип, **short** в **int** и **float** в **double.**

Скаларни типове данни в C++

Пресмятането на булевите изрази се извършва в ред:

- вградените функции (с най-висок приоритет);
- действията в скобите
- унарните операции – “+” и “-“ ,и !(not);
- бинарните операции – “ * ”, “ / ”, “ % ”;
- бинарните операции –“+” и “-“;
- въвеждане и извеждане – “>>” , “<<”;
- сравнение - (= =, !=, >, >=, <, <=);
- конюнкция и – **and** (&&);
- дезюнкция - **or** (||) с най-нисък приоритет.

Пример: Изпълнението на:

```
bool a,b,c;
cout << a && b || c <<”\n”;
```

съобщава за грешка (некоректен аргумент за операцията <<). В този случай коректния запис е:

```
cout << (a && b || c) << ”\n”;
```

Скаларни типове данни в C++

Символен тип данни.

Това е стандартен тип от данни, който се означава със служебната дума **char** и включва наредено множество от символи. Стандартизирани са само част от символите (128 на брой). Това са малките и големите латински букви, цифрите, знаците за пунктуация и управляващи символи. Реализацията на C++ включва два вида символи-графични и управляващи. Графични са тези символи, които имат видимо представяне и се записват оградени с апострофи.

Примери: **‘A’, ‘a’, ‘5’, ‘g’, ‘!’, ‘+’, ‘ ’.**

Последният символ означава интервал. Управляващите символи нямат видимо представяне и се означават като **‘\символ’**.

Скаларни типове данни в C++

Някои от по - важните управляващи символи са дадени в таблицата:

№	символ	предназначение
1	\a	Издава звуков сигнал.
2	\b	Връща курсора 1 символ назад.
3	\n	Курсора преминава на нов ред.
4	\r	Връща курсора в началото на реда.
5	\t	Извършва хоризонтална табулация.
6	\v	Извършва вертикална табулация.
7	\0	Нулев символ-задава край на символен низ.

Скаларни типове данни в C++

Възприето е символите да се наричат символни константи (литерали), а променливите, които приемат такива стойности да се наричат символни променливи. Пример за описание на инициализирани и неинициализирани символни променливи:

```
char a1, a2, a3;
char  b1= '$';
```

За всяка една от тези 4 символни променливи се отделя по 1 байт от оперативната памет. Елементите на множеството стойности от тип **char** са наредени и на всяка една от тях съответства точно определено число (ASCII кода на всеки символ). Числата от 0 до 32 са кодове на управляващи те символи, а от 48 до 57 на арабските цифри. Главните латински букви са с кодове от 65 до 90 (кода на 'A' е 65, а на 'Z' е 90). Кодовете на малките латински букви са от 97 до 122 ('a'-97, 'z'-122). :

Скаларни типове данни в C++

Над символните данни могат да се изпълняват следните операции:

Определяне кода на даден символ:

(int) c1 – където **c1** е символна променлива или символна константа.

Пример:

cout << (int) ‘A’;

Тази инструкция ще изведе 65 (кода на A).

Определяне на символ по зададен код:

(char) p – където **p** е целочислен израз.

Пример:

cout <<(char) 65<<’\t’<<(char)97;

Тази инструкция ще изведе на екрана символите с кодове 65 и 97, разделени с табулация (A....a).

Скаларни типове данни в C++

-Аритметични операции със символни данни.

Всички операции над целочислени данни са допустими и за типа **char,** тъй като се работи с кода на символите.

Присвояване на стойност.

char x,y; x='*'; y=x; x=x + y – 15;

Пресмята израза и го превръща в симовол, който се присвоява на символната променлива x.

Определяне на символ стоящ пред или след зададен символ:

(char) (c-1) и (char) (c+1)- c е константа или променлива от символен тип.

Логически операции и операции за сравнение.

Допустими са и над данни от символен тип, тъй като се работи с техните кодове, които са цели числа:

'a' < 'z' е true, 'a' < '0' е false и т.н.

Скаларни типове данни в C++

-Въвеждане на стойност с операцията <<.

Например:

```
char c1,c2;
cin >>c1 >>c2;
```

очаква въвеждането на два символа различни от интервал, табулация и преминаване на нов ред.

-Извеждане на символи с операцията <<.

Например:

```
char c2, c1='z';
cout <<c1;
```

Извежда символа z, който е присвоен присвоен на **c1**.

```
cout <<c1+c2;
```

Ще изведе цяло число, което е сумата от кодовете на c1 и c2, но **cout <<(char) (c1+c2);** Ще изведе символ, чийто код е равен на сумата от кодовете на c1 и c2.