

Увод в програмирането

Основи на C++

ФМИ, специалност „Софтуерно инженерство“

Съдържание

- Елементи на езика
- Основни типове данни. Представяне в паметта.
- Основни операции
- Вход и изход.

Първа програма на C++

```
#include <iostream>
#include "stdafx.h"

using namespace std;

int main()
{
    cout << "Hello world!";

    return 0;
}
```

- Директиви на препроцесора
- Функция main()
- Инструкции
- Оператор return

Структура на програмата на C++

<изходен_файл> ::=

<заглавен_блок_с_коментари>

<заглавни_файлове>

<константи>

<глобални_променливи>

<класове>

<функции>

Глобални променливи

- Езикът поддържа глобални променливи. Те са променливи, които се дефинират извън функциите и които са “видими” за всички функции, дефинирани след тях. Дефинират се както се дефинират другите (локалните) променливи.
- Съвременните добри практики за стил на програмиране препоръчват минимизиране на използването на глобални променливи и дори напълно ги отхвърлят
- Глобалните променливи задължително трябва да имат коментари

Класове

- В езика C++ има стандартен набор от типове данни като `int`, `double`, `float`, `char` и др. Този набор може да бъде разширен чрез дефинирането на класове.
- Дефинирането на клас въвежда нов тип, който може да бъде интегриран в езика. Класовете са в основата на обектно-ориетираното програмиране, за което е предназначен езикът C++.

Функции

- Всяка програма задължително съдържа функция `main`.
- Възможно е да съдържа и други функции. Тогава те се изброяват в тази част на модула. Ако функциите са подредени така, че всяка от тях е дефинирана преди да бъде извикана, тогава `main` трябва да бъде последна. В противен случай, в началото на тази част на модула, трябва да се декларират всички функции.
- Разлика между декларация и дефиниция на функция

Коментари

- Коментарът е произволен текст, ограден със знаците `/*` и `*/` или от `//` и EOL (знак за край на реда). Игнорира се напълно от компилатора.

`<коментар> ::= /* <редица_от_знаци> */ |`
`// <редица_от_знаци> Край на реда`

- Пояснява програмен фрагмент. Предназначен е за програмиста. Игнорира се от компилатора на езика.
- Хубаво е всеки файл да започва със заглавен блок с коментари, даващи информация за целта му, за използваните компилатор и операционна система, за името на програмиста и датата на създаването му.
- **Писането на коментари значително повишава качеството на кода!!!**

Заглавни (header) файлове

- Съдържа декларации, които се използват в повече от една част на програмата.
- Действа като интерфейс между различните компилируеми части на програмата.
 - Библиотеки
 - Други програми

```
#include <iostream>  
#include "stdafx.h"
```

```
using namespace std;
```

```
int main()  
{  
    cout << "Hello world!";  
    return 0;  
}
```

Азбука на езика C++

- Главните и малки букви на латинската азбука;
- Цифрите
- Специалните символи

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

Идентификатор

- Наименование на елемент в програмата
- Поредица от символи
 - Букви, цифри и знака за долна черта
 - Може да започва само с буква или долна черта
- Валидни идентификатори:
 - abc, ABC, minValue1, _first_setting
- Невалидни идентификатори:
 - 1ba, ab+1, a(1)

Ключови думи

- Вградени в езика идентификатори, които се използват в програмите по стандартен, предварително определен начин
- Ключови думи на C++
 - `int`, `namespace`, `return` и т.н.
- Ключови думи на препроцесора
 - `include`, `undef`, `ifdef` и т.н.
- Ключови думи на компилатора
 - Зависят от конкретния компилатор
 - Например за VS2015: `__finally`, `__event` и т.н.

Оператори

- В C++ има три групи оператори:
 - аритметично-логически оператори
 - управляващи оператори
 - операторите за управление на динамичната памет

Оператор за присвояване

- `<променлива> = <израз>;`

като <променлива> и <израз> са от един и същ тип.

- Пресмята стойността на <израз> и я записва в паметта, именувана с променливата от лявата страна на знака за присвояване =.

Променливи

- Част от паметта, за съхранение на данни което може да съдържа различни стойности по време на изпълнение на програмата
- Имат три характеристики
 - Тип
 - Име
 - Стойност
- Името на променливата трябва да е валиден идентификатор
- Всяка променлива трябва да има тип, който се указва при дефинирането ѝ

Дефиниране на променливи

<име_на_тип> <променлива> [= <израз>]_{опц}
{, <променлива> [= <израз>]_{опц}}_{опц};

<име_на_тип> е дума, означаваща име на типа на променливата

<израз> е правило за получаване на стойност – цяла, реална, знакова и друг тип, съвместимо с името на типа на променливата

```
int i = -123, j = 33;
```

```
char a = -58;
```

```
char b = 58;
```

```
double df;
```

```
string add;
```


Втора програма на C++

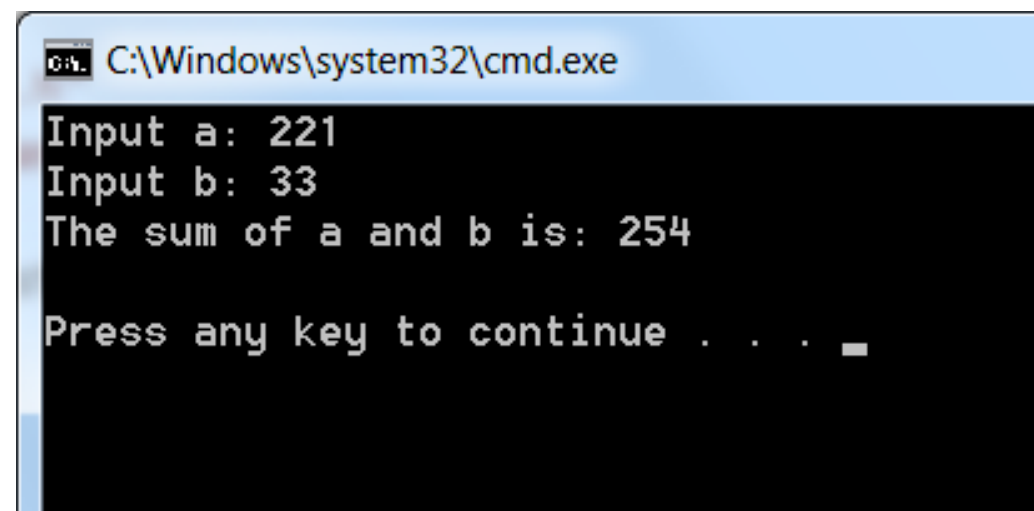
```
#include "stdafx.h"
#include <iostream>

using namespace std;

int main()
{
    int a = 0, b = 0;

    cout << "Input a: "; cin >> a;
    cout << "Input b: "; cin >> b;
    cout << "The sum of a and b is: " << a + b
        << endl << "\n";

    return 0;
}
```



```
C:\Windows\system32\cmd.exe
Input a: 221
Input b: 33
The sum of a and b is: 254
Press any key to continue . . .
```

Константи

- Константи
 - `const` <име_на_тип> <име_на_константа> = <израз>;
 - Добра практика е често повтарящите се стойности да се дефинират като константи, които след това да се използват в програмата
- Конвенция за именуване на константите е те за се записват само с главни букви

Константи

- Примери

```
const int MAXINT = 32767;  
const double RI = 2.5 * MAXINT;  
const double PI = 3.14159265;
```

- Предимства

- Програмите стават по-ясни и четливи.
- Лесно (само на едно място) се променят стойностите им (ако се налага).
- Възможността за грешки, възможни при многократното изписване на стойността на константата, намалява.

Литерал

- Константна стойност на променлива, зададена в кода
 - `int i=2;`
 - `char c = 'a';`
 - `bool isReal = true;`
- `int i = 023` //число в осмична бройна система
- `int i = 0x1F` //число в шеснадесетична бройна система

Използване на константи

```
int main()
{
    float a,b,c;
    const float MULTIPLIER = 4.55;

    cout << "Input a "; cin >> a;
    cout << "Input b "; cin >> b;
    cout << "Input c "; cin >> c;

    cout << "First value is " << a*MULTIPLIER << endl;
    cout << "Second value is " << b*MULTIPLIER << endl;
    cout << "Third value is " << c*MULTIPLIER << endl;
}
```

Основни типове данни (1)

- Скаларни
 - Булев (bool)
 - Символен (char)
 - Целочислен (int)
 - За числа с плаваща запетая (float, double)
 - Изброен (enum)
 - Указател (T*)
 - Псевдоним (T&)

Основни типове данни (2)

- Съставни типове
 - Масив (`T[]`)
 - Символен низ (`char []`)
 - Структура (struct)
 - Клас (class)
 - Обединение (union)

Булев (логически) тип

- **Множество от стойности**

Състои се от два елемента – стойностите true (1) и false (0)

- `<булева_константа> ::= true | false`

- Основни логически операции

- `&&` - AND
- `||` - OR
- `!` - NOT

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

OR

A	not A
0	1
1	0

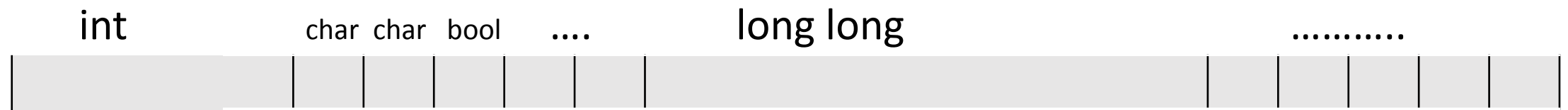
NOT

Символен тип

- Променливите от тип `char` са с размер 1 байт
- Множество от стойности: `[-128; 127]`
 - `unsigned char`: `[0, 255]`
- Литерали
 - `'<символ>'`,
 - `'\<контролен_символ>'`,
 - `'\n'` – нов ред
 - `'\t'` – табулация
 -

Целочислен тип (int)

- Променливите от тип int са с размер от 4 байта
- Множество от стойности: $[-2^{31}; 2^{31}-1]$
- Други целочислени типове
 - short: $[-2^{15}; 2^{15}-1]$ – 2 байта
 - long: $[-2^{31}; 2^{31}-1]$ – най-малко 4 байта
 - long long: $[-2^{63}; 2^{63}-1]$ – най-малко 8 байта
 - unsigned: $[0; 2^n-1]$, $n=16,32,64$



Операции с целочислени типове

- Аритметични операции
 - едноместни операции за знак (+, -)
 - двуместни аритметични операции
 - $a + b$ (събиране)
 - $a - b$ (изваждане)
 - $a * b$ (умножение)
 - a / b (целочислено деление - частно)
 - $a \% b$ (деление по модул - остатък)
- операции за сравнение (предикати)
 - $a == b$ (равно)
 - $a != b$ (различно)
 - $a < b$ (по-малко)
 - $a > b$ (по-голямо)
 - $a <= b$ (по-малко или равно)
 - $a >= b$ (по-голямо или равно)

Представяне на целочислените типове в паметта

- Най-старшият бит на числата със знак определя знака (1 – отрицателно число, 0 – положително число)
- Нека разгледаме тип `short int` (16 бита)
 - Най голямото положително число е 32767

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- Нека добавим към това число 1

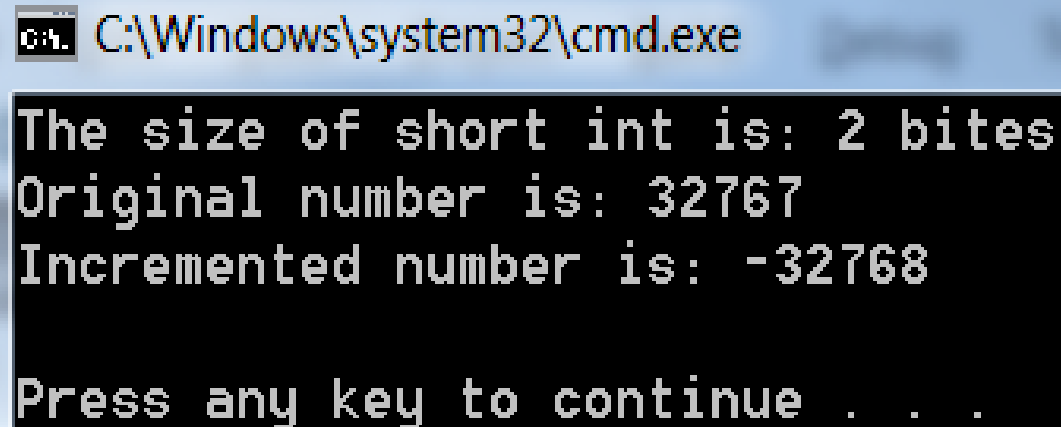
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- Практически, според компютъра се получава -32768

Ако не вярвате 😊

```
int main()
{
    short int a = 32767;

    cout << "The size of short int is: " << sizeof(a)
          << " bites" << endl;
    cout << "Original number is: " << a << endl;
    cout << "Incremented number is: " << ++a << endl << endl;
}
```



C:\Windows\system32\cmd.exe

The size of short int is: 2 bites
Original number is: 32767
Incremented number is: -32768

Press any key to continue . . .

Едноместни операции с целочислени типове

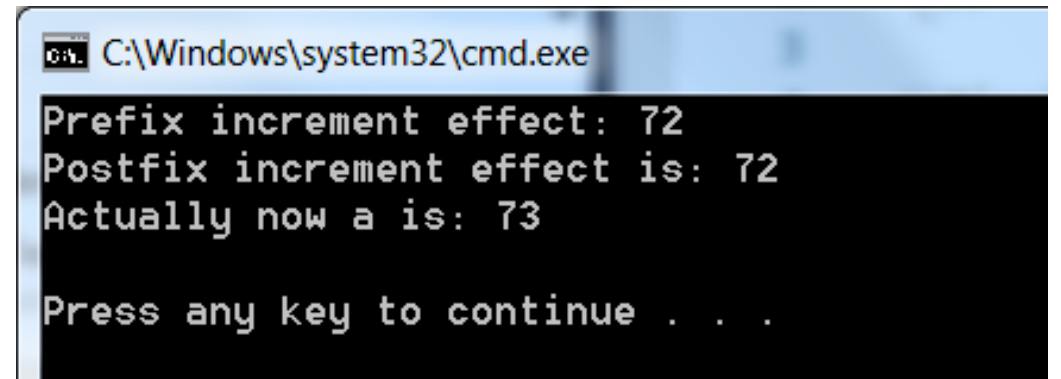
- ++
 - събиране с 1
- --
 - изваждане с 1
- Префиксни операции
--a; ++a
- Постфиксни операции
a--; a++
- А каква е разликата между двете?

Префиксни и постфиксни операции

```
int main()
{
    short int a = 71;

    cout << "Prefix increment effect: " << ++a << endl;

    cout << "Postfix increment effect is: " << a++ << endl;
    cout << "Actually now a is: " << a << endl << endl;
}
```



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Windows\system32\cmd.exe". The window has a black background with white text. The output of the program is displayed as follows:

```
Prefix increment effect: 72
Postfix increment effect is: 72
Actually now a is: 73

Press any key to continue . . .
```

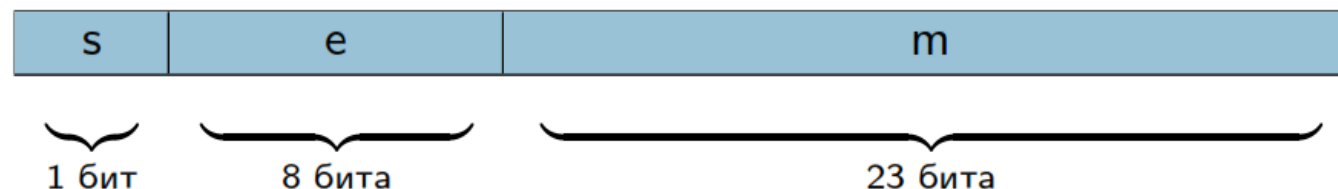
- Прекаленото заиграване с префиксните и постфиксните операции често води до трудно четим код!!

Представяне на реалните числа

- Какво означава реално число?
 - Колко реални числа има между 0 и 1?
 - А между 0.001 и 0.002?
- В компютъра няма безкрайно количество памет с която да се представят всички реални числа
- Числа с плаваща запетая
 - Тип float
 - Тип double

Представяне на реалните числа

- Тип float $f = (-1)^s \cdot m \cdot 2^e$



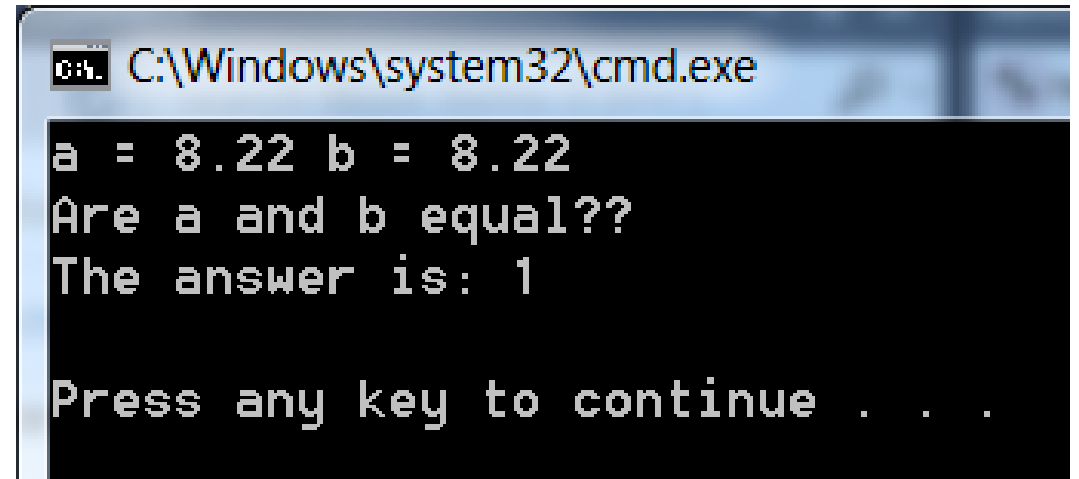
- $s: \{0,1\}$ – знак
 - $m: [0; 2^{23}-1]$ - мантиса
 - $e: [-128; 127]$ – експонента
 - Машинна нула – $[-2^{-23}; 2^{-23}]$
- Тип double – 64 бита

Представяне на реалните числа

```
int main()
{
    double a = 8.22;
    double b = 8.220000000000000004;

    cout << "a= " << a << " b= " << b << endl;
    cout << "Are a and b equal?" << endl
         << "The answer is: " << (a == b) << endl << endl;

    return 0;
}
```



C:\Windows\system32\cmd.exe

```
a = 8.22 b = 8.22
Are a and b equal??
The answer is: 1

Press any key to continue . . .
```

Операции с числа с плаваща запетая

- Валидни са всички операции за целочислените типове, без делението по модул
- Резултатът от операцията „/“ е дробно деление, а не частно
- Трябва да се има предвид точността на представянето на реалните числа при сравняването им

Преобразуване на типовете

- Явно преобразуване

`<преобразуване> ::= (<тип>)<израз>`

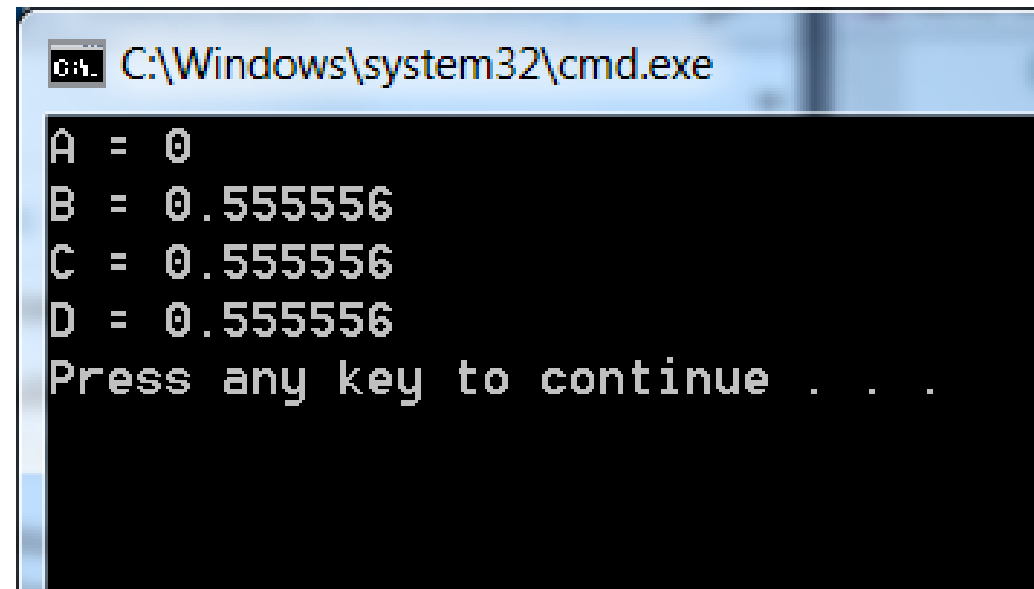
- За да няма загуба на информация се препоръчва преобразуване на типове като се спазва следната посока
 - `bool → char → short → int → long → float → double`
 - `unsigned char → unsigned short → unsigned int → unsigned long`

Ефекти от преобразуване на типовете

```
int main()
{
    float A = 5 / 9;
    float B = 5.0 / 9.0;
    float C = 5.0 / 9;
    float D = 5 / 9.0;

    cout << "A = " << A << endl
         << "B = " << B << endl
         << "C = " << C << endl
         << "D = " << D << endl;

    return 0;
}
```



```
C:\Windows\system32\cmd.exe
A = 0
B = 0.555556
C = 0.555556
D = 0.555556
Press any key to continue . . .
```

Тип изброен

- Дефинира се от програмиста като се изброяват литералите му

дефиниция_на_тип_изброен > ::=

```
enum [<име_на_тип>]_опц {<идентификатор1> [=
<константен_израз1>]_опц,
<идентификатор2> [= <константен_израз2>]_опц,    ...
<идентификаторn> [= <константен_изразn>]_опц};
```

Дефиниране на тип изброен

- `enum Weekday{MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY};`
- `enum Name{IVAN=5, PETER=3, MERY=8, SONIA=6, VERA=10};`
- `enum Id{A1, A2, A3, A4=8, A5, A6=10, A7, A8};`
- Ако не са указани стойности, по подразбиране първият идентификатор получава стойност 0, а всеки следващ – стойност с единица по-голяма от стойността на предходния

typedef

typedef може да се използва за създаване на алтернативно име (alias) на някой вече съществуващ тип.

```
typedef unsigned char BYTE;  
  
int main()  
{  
    BYTE x = 3;  
    return 0;  
}
```

Оператор за вход: „>>“

```
cin >> <променлива> {>> <променлива>;
```

където

- cin е обект (променлива) от клас (тип) istream, свързан с клавиатурата,
- <променлива> е идентификатор, дефиниран, като променлива от “допустим тип”, преди оператора за въвеждане. (Типовете int, long, double са допустими).

Извлича (въвежда) от cin (клавиатурата) поредната дума и я прехвърля в аргумента-приемник <променлива>

Оператор за вход: „>>“

Изразът

```
cin >> <променлива1> >> <променлива2> >> ... >> <променливаn>;
```

е еквивалентен на

```
cin >> <променлива1>;
```

```
cin >> <променлива2>;
```

...

```
cin >> <променливаn>;
```

Оператор за изход: „ <<“

```
cout << <израз> {<< <израз>;
```

където

- cout е обект (променлива) от клас (тип) ostream, свързан с екрана на компютъра,
- <израз> е израз от допустим тип. Представата за израз продължава да бъде тази от математиката. Допустими типове са bool, int, short, long, double, float и др.

Операторът << изпраща (извежда) към (върху) cout (екрана на компютъра) стойността на <израз>.

Оператор за изход: „ <<“

Изразът

`cout << <израз1> << <израз2> << ... << <изразn>;`

е еквивалентен на

`cout << <израз1>;`

`cout << <израз2>;`

...

`cout << <изразn>;`

- За подготовката на тази презентация са използвани слайдове на:
 - Доц. Александър Григоров
 - Доц. Атанас Семерджиев
 - Доц. Трифон Трифонов