ЕЛЕМЕНТИ НА ЕЗИКА. ТИПОВЕ ДАННИ

СЪДЪРЖАНИЕ

1. Допустими символи, запазени думи и структура на програмата

Допустими в програмите са следните символи:

малки и главни латински букви
цифрите от 0 до 9
специални символи:
+ - * / = () [] {} | & ! < > # $ % ^ ~ _ . , : ; ` "
разделители – интервал, табулация и нов ред

asterisk * **ampersand (&)** **quotes " "** **parentheses ( )** */парентезис/*
octothorpe */октоторп/* **#**

less than < > **comma ,** **apostrophe (single quote) '** **semicolon ;** **colon :**
backslash ** **forword slash / **curly braces { } [ ]** **- braces,**

pipe | ***/the backslash key on QWERTY/***

underscore _ **exclamation mark - !** **acute** */екют-ударение/* **`** **caret ^**
period, dot . **dash -**

Езика С е чувствителен към малки и големи букви.

Запазени думи

Стандартът ANSI определя 32 ключови думи, които не могат да се използват в имената на функции или променливи. Много компилатори на C/C++ добавят други ключови думи. В езика С /C++ главните и малките букви се различават (**int**, **Int** и **INT** са различни).

auto, _bool, break, case, char, _complex, const, continue, default, double, else, enum, extern, float, for, goto, if, _imaginary, inline, int, long, register, restrict, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while

Структура на програмата

```
#include <библиотеки>
using namespace std;
int main ()
{
команди;
```

```
return 0;
}
```

Пример: програма за извеждане на „Hello World”

```
#include <iostream> //вкл. библиотека
using namespace std;  //използва std именовано пространство
int main()  //гл. функция
{  // начало на тялото на функцията
   cout<<"Hello World!";     //оператор за изход
   return 0 ;   // край на програмата
}  //край на тялото на функцията
```

#include <*библиотеки*> глобални декларации на константи, променливи и типове данни, декларации на други функции

/ octothorpe/ **-** директиви на предпроцесора /указание към предпроцесора/. Предпроцесора извършва подготовка на кода за компилация.
Той маха празни пространства, коментари и др. и оставя съществената част от написаната програма.

Първият ред представлява **директива #include към предпроцесора** за включване на **външен файл** <**iostream**> който съдържа описанията на някои стандартни функции за вход и изход на данни. В случая е нужен за да може компилаторът да намери описанието на функцията за **форматиран изход cout** Триъгълните скоби указват да се търси в каталога със стандартни заглавни файлове.

http://en.wikipedia.org/wiki/C++_standard_library

using namespace std; - използване на именовано пространство std /стандартната библиотека на C/C++)

Пример: Използване на собствено namespace / mystuff/

```
#include <iostream>
using namespace std;
namespace mystuff{
int value = 5;
}
using namespace mystuff;
```

```
int main()
{
    cout << value; //outputs 5
    return 0;}
int main();
```

Структурата на тази главна функция се задава по един от следните начини:

```
int main()
{
тяло на функцията
return 0;
}
```

или:

```
main()
{
тяло на функцията
return 0
}
```

или:

```
void main()
{
тяло на функцията
}
```

Следващият ред представляват дефиниция на функция с име **main** като **int** пред името на функцията указва че тя връща стойност int (целочислен тип). Всяка програма трябва да има функция **main** и това **е** първата функция която се изпълнява при стартиране на приложението. В кръглите скоби се описват параметрите които функцията приема, но в случая тя не изисква никакви параметри и това се заявява със **()** / braces/

- Във фигурните скоби **{ }** /curly braces/ е поместено тялото на функцията което ще се изпълни.

- Започва с извикване на функция от стандартната библиотека на *C++* — cout, която в случая приема единствен параметър — указател към низа от символи „ Hello World!”. Двойните кавички **“ “**създават

анонимен низ (низова константа) от символите заградени в тях в паметта и връща указател към него. Нарича се анонимен защото не е присвоен на променлива и съответно по-късно няма да можем да се обърнем към него чрез такава променлива.

- Символът **;** **/ semicolon/** поставя края на израза който е основния градивен блок на програмите.

- Изразът **return 0;** прекратява изпълнението на функцията и връща целочислената стойност нула. Често функциите, освен ако нулата не е валидна резултантна стойност на задачата изпълнявана от функцията, връщат нула за да сигнализират че не е възникнал проблем при изпълнението им. Но това е различно за всяка функция и за смисъла на връщаните стойности програмистът трябва да се допита до документацията.

Константи – числа или последователност от символи с постоянна стойност.Константите се използват често в началото на програмата за инициализиране на променливи.

Видове константи

• ***Числови:***

Цели 12; -12

Дробни десетични -12.55; -321.0; 0.123

• ***Символни***: един символ, затворен в ' '; всеки символ в ASCII таблицата има числен код – 'a'; '34'; '#'

http://www.asciitable.com/ - ASCII таблицата

• ***Низови***: низ, затворен в двойни кавички; дължината е неограничена

„Hello World!”

„12345”

“ 23+44”

Има разлика между символна и низова константа т.е .*”a”* и *'a'*

'a' е символ *a*, чийто код от ASCII таблицата е 97

”a” едномерен масив (**виж масиви**)

Деклариране на константа

const тип име на константа = израз;

Пример:

```
const char s=”Hello world”;
const char p=’#’;
const int p=5;
const double pi=3.14;
const float v=50.23;
```

Ако числото което искаме да зададем е дробно, но дробната част е 0, трябва да го зададем като примерно 1000.0..по този начин казваме на компилатора, че числото 1000 е число с плаваща запетая.

Пример: Изчисление обиколката и лицето на кръг по зададен радиус

```
#include <iostream>
using namespace std;

int main()
{
   int r,P,S;
   const double p=3.14;
   cout<<"Napishi radius r=  ";
   cin>>r;
   P=2*p*r;
   S=p*r*r;
   cout<<S<<"  "<<P;
   return 0;
}
```

Променливи - величини от определен тип, на които е присвоено име и чиято стойност се променя при изпълнение на програмата.

• Под понятието ***променлива*** **се крие клетка от паметта**, в която се съхранява текущата стойност на променливата.

• Всяка клетка има ***адрес***, използването на името е обръщение към адреса.

• **Всяка област от паметта, която се използва за запис на стойности се нарича *обект*. Променливата е най-простия обект**.

• **Всички *променливи* предварително трябва да се декларират и с това се определя размерът или броят байтове памет, които трябва да се резервират за нея. !!!**

• Имената на променливите позволяват да се отличават една от друга. Те се задават с *идентификатори*. Идентификатор е комбинация от букви (латински) и цифри, като винаги започват с буква. (suma; br; k1). Идентификаторът трябва да е различен от ключовите думи. C/C++ прави разлика между големи и малки букви и затова идентификатор с име *Suma* е различен от *suma*.

Деклариране на променлива

тип *на променлива* **име** *на променлива;*

Пример:
```
int a,b;
float c;
```

char –байт, съдържащ 1 знак пр. A /8 bits/
int- цяло число – Пр. 100 /16 bits, 32 bits, 64 bits/
float – число с плаваща запетая с точност до 6 десетици Пр. 0.123456
double - число с плаваща запетая с точност до 10 десетици Пр. 0.0123456789
bool - булева стойност (истина-неистина, true,false), всяка нулева стойност представлява истина, т- е true

Инициализация на променливи

тип *на променлива* **име** *на променлива* **= израз**;

Променливата се инициализира, като след името й се поставя символ за присвояване (=) и желаната начална стойност.

Пример:
```
int a=5;
float c=3.13;
char g='@';
```

Локални и глобални променливи

- ***Глобалните*** променливи се описват в началото на програмата преди всички функции. Имената им трябва да бъдат уникални. Тяхната област на действие е определена за цялата програма.
- ***Локалните*** променливи се дефинират в тялото на функцията и тяхното действие е за самата функция.

Коментари

Коментарът е бележка за вас или други програмисти, която поставяте във вашия сорс код.

Всички коментари се игнорират от програмата.

Коментара започва с **/*** и завършва с ***/**

Коментарите могат да се разпростират на няколко реда.

Коментара може да се постави навякъде, освен по средата на някоя ключова дума, име на функция или променлива.

Може да използвате коментар, за да премахнете някой ред от кода

Въпреки, че още не е дефинирано в стандарта ANSI, , можете да използвате още един стил на коментиране, който се използва в C++ . Това е символа **// - за коментар на един ред.** **В езика C** този вид коментар е технически невалиден , но повечето компилатори го приемат

Коментарите не могат да се влагат!/коментар в коментар/

2. Основни типове данни

Езика за програмиране C/C++ притежава ***вградени*** / предварително дефинирани и се поддържат от неговото ядро/ и ***абстрактни*** / дефинират от програмиста/

Вградени типове данни:

скаларни типове: **съставни типове:**

булев; цял; реален; символен; изборен; указател масив; символен; вектор

Всеки тип се определя с *множество от допустими стойности, оператори и вградени функции,* които могат да се прилагат над елементите му

С притежава 5 основни типове данни: **void, char, int, float , double.** Тези основни типове данни с изключение на void, могат да се модифицират чрез използването на модификаторите на типове в С.

Модификаторът на тип предшества името на типа.
Пример: тази конструкция декларира long цяло число:
long int i;

1. Името на променливата трябва да се състои само от знаци, цифри и долна черта.
2. Името на променлива не трябва да започва с число.
3. Името на променливата не трябва да се състои от интервал.
4. Името на променливата не трябва да се състои от ключова дума.

3. Скаларен тип данни

3.1 Целочислен

Предназначен за обработка на цели числа.
Запазената дума за деклариране на променливи от този тип е int.
Количеството памет за една променлива от тип int е 2B (байта).

Тип	Диапазон	Памет
char, signed char	-128 до 128	1B
unsigned char	0 до 255	1B
short, signed short	-32 768 до 32 767	2B
unsigned short	0 до 65 535	2B
int, signed short	-32 768 до 32 767	2B
unsigned, unsigned int	0 до 65 535	2B
long, signed long	-2 147 483 648 до 2 147 483 648	4B
unsigned long	0 до 4 294 967 295	4B

Множеството от допустими стойности зависи от компилатора.
В таблицата по-горе е даден диапазонът на целочислен тип в средата за програмиране Borland C/C++.
В MS Visual C++ диапазонът е от -2 147 483 648 до 2 147 483 648.

Аритметични операции над целочислен тип
Операциите над даден тип се извършват посредством оператори, които се изпълняват над даден операнд или операнди.

Унарни операции – извършват се над един операнд с операторите + и -, записани преди операнда. Те имат същото действие, както и математиката – да потвърдят или да променят знака на дадено число.

Примери:

```
int a=21,b=-12, c=10;
a=+в;      a=-b;    //унарни операции a=-(-12)=12
```

Бинарни операции – извършват се точно над два операнда с операторите +, - , *, /, % , записани между операндите.

Примери:

```
c=25+4; c=25-4;
c=a+b; c=a-b;
a=c*2; b=c/a;
```

Два последователни аритметични оператора може да се допуснат само ако единият от тях е унарен.

Оператор за:	Означение	Примери	
		Операция	Резултат
Събиране	+	24+7 -32+86	31 54
Изваждане	-	8-4 322-421	4 -99
Умножение	*	5*4 5*-3	20 -15
Частно при целочислено деление	/	10/3 22/5	3 4
Остатък при целочислено деление	%	10%3 22%5	1 2

Приоритет на операторите в низходящ ред:

Най-висок	**()**	**изразите в скоби**
	+, -	**унарни**
	***, /, %**	
Най-нисък	**+, -**	**бинарни**

За смяна на приоритета се използва операторът „златни скоби“ ().

Първо се изчислява изразът, записан в скобите.
Ако в даден израз са включени няколко израза в скоби, те се изчисляват в реда на задаването им отляво надясно.
Броят на отварящите и затварящите скоби трябва да е равен.

Примери:
Изразът
(5+6)*3=33 е различен от 5+6*3=23
(3+6)/(8-5)=3 е различен от 3+6/8-5=3+0-5=-2 //6/8 =0.7 рез ще е 0
Могат да се използват вложени изрази със скоби, като при тях изчислението ще започне от най-вътрешните скоби и се спазва приоритетът на операциите.

Пример:
2*(2+(3*(5-2)))/5 = 4

Оператори за сравнение

Като резултат от срабнението се връща логическа стойност 1 или 0 (true или false)

Оператор	Описание	Примери	
<	по-малко	cout<<5>4	Рез. 1
<=	по-малко или равно	cout<<(6>=6)	Рез. 1
>	по-голямо	cout<<6>6	Рез. 0
>=	по-голямо или равно	cout<<(8>=9)	Рез. 0
==	равно	cout<<(6==6)	Рез. 1
!=	различно(неравно)	cout<<(4!=16)	Рез. 1

Примери:

```
#include <iostream>
using namespace std;

int main() {
   cout<<(5>4);
   return 0;
  }
```

```
#include <iostream>
using namespace std;

int main() {
   cout<<(6>6);
   return 0;

}
```

```
#include <iostream>
using namespace std;

int main() {
   cout<<(4!=16);
   return 0;
  }
```

Вградени функции при реален тип

За целочислен тип данни е вградена фунцията abs(x), която връща абсолютната стойност на аргумента x. За да се използва тази функция, е необходимо в началото на програмата да се включи библиотеката math.h

Пример:

```
cout<<abs(-12);     //извежда 12

int a=abs(12);
cout<<a;            //извежда 12
```

3.2 Реален (float и double)

В математиката реалните числа могат интуитивно да бъдат дефинирани като елементи на множество, съответстващи на всички точки на една права. Множеството на всички реални числа обикновено се отбелязва със символа R. Множеството на реалните числа обединява множеството на рационалните R и множеството на ирационалните числа I. Формално то трябва да удоволетворява следните аксиоми:

R е поле, тоест дефинирани са операциите събиране и умножение със стандартните им свойства.
В R е въведена релация на пълна наредба "≤" за която е изпълнено:
ако $x \leq y$, то $x+z \leq y+z$
ако $0 \leq x$ и $0 \leq y$, то $0 \leq xy$
Реалните числа могат да бъдат представени като десетични дроби. При това, ако едно число е рационално, представянето му винаги е като крайна или безкрайна периодична десетична дроб, докато ирационалните числа се представят като безкрайни непериодични десетични дроби. Това на практика означава, че при конкретни пресмятания се използват приближения на реалните числа. Така например дробите 1; 1,4; 1,41; 1,414 са приближения на ирационалното число √2

Име	Диапазон	Памет
float	**$-3.4*10^{38}$ до $3.4*10^{38}$**	**4B**

double	**$-1.74*10^{308}$ до $1.7*10^{308}$**	**8B**

Точността на тип float е до 7 значещи цифри след десетичната точка.
float a =12.1234567;

Точността на тип double е до 14 значещи цифри след десетичната точка.
double b=32.12345678999999;

При извеждане на реални числа компилаторът игнорира последните нули след десетичната точка (ако не е указано допълнително изискване от програмиста).

За записване на реални числа се използват два начина:
1. в дробнодесетичен формат
2. в експоненциална форма

Пример:
3.54 -4.0 65. 0.55 .25
3.15e+2 -7.8E+5 12.8e-7

Пример:
```
float a;  // деклариране на променлива от тип float - едноредов коментар
double b=31.1;  /*деклариране на променлива от тип double и инициализация - многоредов коментар*/
```

Аритметични операции при реален тип данни: унарни +, -
бинарни - *, /, +, -
Операциите се извършват както при целочислен тип с едно изключение – делението не е целочислено и на реални числа не може да се намира остатък от деление.
Операции за сравнение – както при целочислен тип

Вградени математически функции

Функции	Описание
ceil(x)	Намира най-малкото цяло число >=x . То е от тип double
floor(x)	Намира най-голямото цяло число <=x. То е от тип double
sqrt(x)	Намира корен квадратен на числото x

pow(x)	**Степенува x^n**
fabs(x)	**Намира абсолютната стойност на числото x. То е от тип double**
sin(x)	**Намира sin x. Числото x се задава в радиани**
cos(x)	**Намира cos x. Числото x се задава в радиани**
tan(x)	**Намира tg x . Числото x се задава в радиани**

Примери:

ceil(3.4) **Рез. 4**
sqrt(9) **Рез. 3**
pow(2,3) **Рез. 8**
fabs(-20.6) **Рез. 20.6**

3.3 Логически тип

Логическият тип се декларира със запазената дума bool
Името му идва от името на английския математик Джордж Бул.
За променливите и константите /по-долу са обяснени/, се заделя памет от 1B. Допустими стойности са двете константи true и false(истина и лъжа), като true се представя с 1, а false с 0.

Пример:

```
#include <iostream>
#include<math.h>
using namespace std;
int main(){
bool a1,a2=true;
a1=0;
cout<<a1<<endl;
cout<<a2<<endl;
  return 0;
}
```

Резултат:
0
1

Логически оператори

Оператор за логическо отрицание (!) – използва се за промяна на логическата стойност на даден операнд. Действието на оператора (!) върху операнда A се представя както е описано по-долу

!<операнд>; //променя ст-та на <операнд> от true на false и обратно

Пример:
bool a=true,b=false;
a=!a; //ст-та на променливата a става false
b=!!b; //ст-та на променливата b остава false

Оператор за логическо „И” (&&) – логическо умножение

A	**B**	**A&&B**
True (1)	True (1)	True 1*1 ==1
true	false	false
false	true	false
false	false	false
A	B	A&&B
1	1	1
1	0	0
0	1	0
0	0	0

Оператор за логическо „ИЛИ” (||) – логическо събиране

A	**B**	**A||B**
true 1	True 1	True 1+1==1
true	false	true
false	true	true
false	false	false

Приоритет на логическите оператори

! най-висок

&&
|| **най-нисък**

Пример:

```
#include <iostream>
using namespace std;

int main() {
    bool a=true;
    bool b=true;
    cout<<(a&&b);
    b=false;
    cout<<(a&&b);
   a=false;
   b=true;
   cout<<(a&&b);
   b=false;
   cout<<(a&&b);
    return 0;
   }
```

Резултат:
1000

Пример:

```
#include <iostream>
using namespace std;

int main() {
    bool a=true;
    bool b=true;
    cout<<(a||b);
    b=false;
    cout<<(a||b);
   a=false;
   b=true;
   cout<<(a||b);
```

```
  b=false;
  cout<<(a||b);
   return 0;
  }
```

Резултат:
1110

Примери за логически изрази

1||1&&0	true
!1||0&&1	false
false&&true||false	false
!(1||0&&1)	false
false&&(true||false)	false
(!1||0&&1)||(false&&true||false)	false
!(!1||0&&1)||(false&&true||false)	true

3.4 Символен тип

Декларира се със запазената дума char. Необходимата памет е 1B.
Пример:

```
char a,b;
char A,B;
char symbol;
char Symbol;
```

В C++ символният тип използва символите от така нарчената ASCII кодова таблица, вградена в компютъра. Броят на тези символи е 256, като на всеки съответства код от 0 do 255.
Първите 128 символа са стандартни и за всички компютри са едни и същи, докато при останалите съществуват различия.
На екрана се изобразява символ, а в паметта се разполага число, което е кодът на този символ.
Символите са графични (букви, цифри и други знаци) и управляващи (Enter-нов ред, Backspace- връщане на курсур и др.)

Инициализация на символен тип –присвоява се символ заграден в апострофи

```
char c= `A`;
```

За задаване на управляващи символи се използват следните специални означения:

Озн.	Описание на символа
\n	нов ред
\t	хориз. табулация
\v	верт.табулация
\b	връщане на курсора с 1 символ назад
\r	връщане на курсора в началото на реда
\a	издава звуков сигнал
\0	нулев символ, знак за край на низ

По зададения символ може да се стигне до неговият ASCII код чрез израза (int) ch, където ch е символна константа или променлива.
Пример:

```
cout<<(int)`a`; //извежда на екрана 97
```

Извеждане на символ по зададен негов ASCII код

Пример:

```
cout<<(char)97;  //извежда символа a
cout<<(char)(20+45); //извежда на екрана A
```

Над символни данни могат да се прилагат всички аритметични операции, допустими за целочислен тип данни. Тези операции се извършват над ASCII кодовете на съответните символи.

Пример:

```
#include <iostream>
using namespace std;

int main() {
```

```
   int b='3';
cout<<'a'+4<<endl; //извежда 101
cout<<'b'+2;   //извежда 53

   return 0;
  }
```

СПЕЦИФИЧНИ операции за числени величини

Оператори за инкрементиране и декрементиране

Означават се с ++ и --
Прилагат се над предварително декларирана променлива.
++променя текущата стойност на променливата, като добавя 1,
--променя текущата стойност, като изважда 1
Употребяват се в два варианта – преди и след променливата.
Когато използваме a++(суфикс) , към a се добавя 1 след като израза се изчисли със старата (неувеличена) стойност на a.
Когато използваме израза ++a (префикс), отново a се увеличава с 1, но изразът вече се изчислява с новополучената (увеличена) стойност.

Пример:

```
1. a = a+1;      // и трите увеличават стойността на a с 1
2. a += 1;
3. a++;
```

Пример

```
1. a = 12;
2. b = a++;
```

Резултат: a=13,b=12

Пример:

```
1. a = 12;
```

2. **b = ++a;**

   Резултат: a=13, b=13

Оттук можем да направим извода: Когато операторът е пред името на променливата, стойността й първо се увеличава с 1 и тогава се присвоява. Когато операторът е след името на променливата, стойността й първо се присвоява, а след това се увеличава.

Пример

```
#include <iostream>
using namespace std;

int main() {
   int a=20, b=15;
cout<<"a="<<++a<<endl;   //извежда a=21
cout<<"a="<<a<<endl;     //извежда a=21

cout<<"b="<<b++<<endl;   //извежда b=15
cout<<"b="<<b<<endl;     //извежда b=16
cout<<"a="<<--a<<endl;   //извежда a=20
cout<<"a="<<a<<endl;    //извежда a=20
cout<<"b="<<b--<<endl;   //извежда b=16
cout<<"b="<<b<<endl;     //извежда b=15

   return 0;
  }
```

КОМБИНИРАНИ оператори за присвояване

Операция	Пълна форма	Кратка форма
+-	a=a+20	a+=20
-=	b=b-15	b-=15
*=	c=c*a	c*=a
/=	d=d/b	d/=b
%=	e=e%2	e%=2

Пример

```
#include <iostream>
using namespace std;

int main() {
    int k=20, p=18;
    double s=8.0, r=3.0;
    cout<<(k+=p)<<endl;
   cout<< (k/=p)<<endl;
    cout<<(k%=p-3)<<endl; //?
    cout<<(s*=r);

    return 0;
   }
```

Резултат:

```
38
2
2
24
```

4.Връзка между типовете в C++

При работа със скаларните типове много често се налага те да се преобразуват до точно определен тип. За тази цел в езика C++ са реализирани два начина:

Неявно преобразуване на типовете – извършва се от компилатора при определени условия, без да се оказва от програмиста.

Правила за неявно преобразуване:

а) Винаги когато се използват логически оператори се извършва неявно преобразуване на вградените типове в C++ до логически тип.
При числовите типове числото 0 се преобразува до false, а всички останали числа до true.

```
cout<<(24&&0);  //извежда 0
```

Пример

```
#include <iostream>
using namespace std;

int main() {
  cout<<(24&&0);

   return 0;
  }
```

б) При символен тип само символът `\0`се преобразува във false, а всички останали символи до true.

```
cout<<('\0'||2.5);    //извежда true
```

Пример

```
#include <iostream>
using namespace std;

int main() {
   cout<<('\0'||2.5);

   return 0;
  }
```

в) Прилагането на аритметични оператори над операнд от символен тип води до преобразуване на символа в цяло число, равно на неговия код

```
cout<<('5'*2);
```

Пример

```
#include <iostream>
using namespace std;

int main() {
  cout<<('5'*2);  //53*2  ASCII

   return 0;
  }
```

Резултат: 106

г) При пресмятане на аритметичен израз, в който участват операнди, заемащи различна по големина памет, всеки операнд от тип с по-малък брой байтове се преобразува до тип, равен на операнда с най-голям брой байтове.

Пример

```
#include <iostream>
using namespace std;

int main() {
  cout<<(2-'A')*2.1;  /*най-голям брой байтове ползва числото 2.1 което е от реален тип */

   return 0;
  }
```

Резултат: -132.3 //затова резултатът е от реален тип

Тип	Преобразува се до
bool	Всички числови типове
short	int
unsigned short	unsigned int

float	double

Пример без загуба на точност
double a=5; // стойността на a се преобразува до 5.0

Пример със загуба на точност
int b=3.52; //стойността на b се преобразува до 3

Явно преобразуване на типовете – става като се използва една от следните две конструкции
(„име на тип“) <израз>
Преобразува стойността на <израз> до указания в скобите тип

Пример

```
#include <iostream>
using namespace std;

int main() {
  cout<<(int)3.8<<endl;
  cout<<(char)42<<endl;
  cout<<(double)10/4<<endl; //10 се преобр. в реално преди делението, рез. също е реален
  cout<<(double)(10/4);  /* по-висок приоритет имат скобите и рез. се преобразува в реално число*/

   return 0;
  }
```

Резултат:
3
*****
2.5
2

5.Манипулатори за форматиране на изхода

За да се укаже как точно трябва да се изведат резултатите, се използват манипулаторите за форматиране на изхода. Те са стандартно дефинирани във файла iomanip.h.
Този файл трябва да бъде включен в началото на програмата.

Манипулатор setw(<цял израз>) – задава брой позиции на следващото извеждане, подравнено отдясно.

Пример

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   cout<<215;
   cout<<setw(5)<<215;
   cout<<setw(6)<<430;

   return 0;
}
```

Резултат:

```
215  215   430
```

Манипулатор setprecision(<цял израз>) – задава броя на цифрите след десетичната точка, с които ще бъде изведено следващото реално число, като прави закръгление.

Пример

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
   cout<<setprecision(3)<<3.3123<<endl;
   cout<<setprecision(1)<<562.01<<endl;
```

```
    cout<<32.47<<endl;
    cout<<8;

    return 0;
}
```

Резултат:

3.31
6e+02
3e+01

Задачи:

Задача 1. Да се напише програма за събиране на две числа, като потребителят пише числата от клавиатурата.

```
#include<iostream>
using namespace std;
int main()
{
  int a, b, c;
  cout<<"Enter two numbers to add";
  cin>>a>>b;
  c = a + b;
  cout<<"Sum of entered numbers = "<<endl;
  cout<<c;
  return 0;
```

Резултат:

Enter two numbers to add

3 6

Sum of entered numbers = 9

Задача 2. Демонстрация на аритметичните операции

```
#include<iostream>
using namespace std;
int main() {

  int a = 21;
  int b = 10;
  int c ;
  c = a + b;
  cout<<"Line 1 - Value of c is %d\n"<<c ;
  c = a - b;
  cout<<"Line 2 - Value of c is %d\n"<<endl;
  c = a * b;
  cout<<"Line 3 - Value of c is "<< c <<endl;
```

```
   c = a / b;
   cout<<"Line 4 - Value of c is " <<c <<endl;
   c = a % b;
   cout<<"Line 5 - Value of c is "<<c <<endl;
   c = a++;
   cout<<"Line 6 - Value of c is "<<c <<endl;
   c = a--;
   cout<<"Line 7 - Value of c is "<<c <<endl;

return ();
}
```

Резултат:

```
Line 1 - Value of c is 31
Line 2 - Value of c is 11
Line 3 - Value of c is 210
Line 4 - Value of c is 2
Line 5 - Value of c is 1
Line 6 - Value of c is 21
Line 7 - Value of c is 22
```

Задача 3. Да се състави програма, която въвежда от клавиатурата цяло положително трицифрено число и извежда на екран числото написано в обратен ред

Решение:

```
#include <iostream>
using namespace std;

int main ()
{
   int a,c1,c2,c3;
   cin>>a; //въвежда от клавиатурата трицифрено число a
   c1=a/100; /*цифрата на стотиците получаваме
            като делим трицифреното число на 100, примерно 345/100=3
            цяла част 3 */
c2=a/10%10; /*цифрата на десетиците получаваме, като делим
последователно 2 пъти на 10: примерно 345 , първи път 345/10=34 и втори
път  34%10=3 с остатък 4 - пишем остатъка 4*/
```

```
   c3=a%10; /*c3 -цифра на единиците: 345/10=34 с остатък 5 което е
числото, което търсим*/
   cout<<c3<<c2<<c1<<endl; //числото се изписва в обратен ред

    return 0;
}
```

Същата задача може да се реши и по втори начин: цифрите на числото последователно се отделят отдясно наляво и се извеждат на екран:

```
#include <iostream>
using namespace std;

int main ()
{
   int a;
   cin>>a;
   cout<<(a%10); //245/10 остатък 5
   cout<<(a/10%10); //245/10=24, 24%10 е 20 с остатък 4
   cout<<a/100; /*245/100 е цяло число 2 /понеже числата са int
              програмата ще отреже дробната част 45*/
    return 0;
}
```

Задача 4. Да се състави програма, която въвежда от клавиатурата цяло положително четирицифрено число и извежда на екран числото написано в обратен ред

Решение:

```
#include <iostream>
using namespace std;

int main ()
{
   int a;
   cin>>a;
   cout<<(a%10); //2435/10 остатък 5
   cout<<(a/10%10);  //2435/10=243, 243%10 е 24 с остатък 3
   cout<<a/100%10; /*2435/100 е цяло число 20 с остатък 4
```

```
   cout<<a/1000; /*2435/1000 е 2,435 но понеже числото е int, програмата ще отреже дробната част 435 и ще изведе само 2*/
    return 0;
}
```

Задача 5. Да се състави програма, която въвежда от клавиатурата цяло положително четирицифрено число. Програмата извежда сумата и произведението на цифрите на числото

Решение:

```
#include <iostream>
using namespace std;
int main ()
{
   unsigned int a,c1,c2,c3,c4;
   cin>>a;
   c1=a/1000;
   c2=a/100%10;
   c3=a/10%10;
   c4=a%10;
   cout<<c1+c2+c3+c4<<"   ";
   cout<<c1*c2*c3*c4;
    return 0;
}
```

Задача 6. Да се състави програма, която въвежда от клавиатурата цяло положително петцифрено число.

Програмата да извежда:

а/ броя на нечетните цифри на числото

б/ сумата на четните цифри на числото

Решение а:

```
#include <iostream>
using namespace std;

int main ()
{
   unsigned long a; /*числото е long защото едно петцифрено число може да надвиши стойността на int , което е 2 байта, тоест числото което можем да въведем може да бъде примерно над 65535 */
   int c1,c2,c3,c4,c5;
```

```
   cin>>a;
   c1=a/10000;  //отделяме всички цифри на числото
   c2=a/1000%10;
   c3=a/100%10;
   c4=a/10%10;
   c5=a%10;
   cout<<c1%2+c2%2+c3%2+c4%2+c5%2<<endl; /*при деление с модул 2 всяко от горните числа  ще даде остатък 1 ако е нечетно а четните ще са с остатък 0 , следователно събираме единиците   и намираме колко са те, тоест тяхната сума, което търсим; ако търсехме нечетнните за cout щяхме да напишем: cout<<5-(c1%2+c2%2+c3%2+c4%2+c5%2); тоест от общия брой на цифрите които са 5 в случая изваждаме нечетните цифри*/
   return 0;
}
```

Решение б:

```
#include <iostream>
using namespace std;

int main ()
{
   unsigned long a;
   int c1,c2,c3,c4,c5,sum;
   cin>>a;
   c1=a/10000;  //отделяме всички цифри на числото
   c2=a/1000%10;
   c3=a/100%10;
   c4=a/10%10;
   c5=a%10;
   sum=c1+c2+c3+c4+c5;
   cout<<sum-(c1%2*c1+c2%2*c2+c3%2*c3+c4%2*c4+c5%2*c5)<<endl;
/*ако имаме числото 12345 sum =15, след което от нея изваждаме сумата на нечетните числа, която се намира като се фиксира чрез деление модул 2 при остатък  съответното нечетно число*/
   return 0;
}
```

Задача 7. Създайте програма, която конвертира земните дни в юпитерски години

1 година на Юпитер /времето за което Юпитер обикаля около Слънцето/ е

равна на 12 земни години, като използвате коментари, за да поясните кода на програмата ви.

```
#include<iostream>
using namespace std;
int main() {
   float Earth_days; /*earth days*/
   float Jupiter_years; /*Jupiter years*/

   cout<<"Enter Number of Earth days:";
   cin>>Earth_days; /*enter the number of earth days*/
   Jupiter_years = Earth_days/(365*12.0);  /*make calculation*/
   cout<<"Equivalent Jupiter years:"<<Jupiter_years ;  /*rezultat*/
   return 0;
}
```

Резултат:
Enter Number of Earth days:45
Equivalent Jupiter years: 0.010274

Задача 8. Демонстриране на квадратен корен

```
#include <stdio.h>
#include<iostream>
using namespace std;
int main(){
   double answer;  /*връщаната ст-т е от тип double*/
  answer = sqrt(10.0);   /*присвояване*/
  cout<<answer;
  return 0;
}
```

Задача 9. Намиране корени на квадратно уравнение

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   double a,b,c,d,x1,x2,x;
```

```
   cout<<"Namira korenite na uravneniq ot tipa: ax^2 +- bx +- c=0\n";
   cout<<"a=";
   cin>>a;
   cout<<"b=";
   cin>>b;
   cout<<"c=";
   cin>>c;
   d=b*b-4*a*c;
   cout<<"D="<<d<<"\n";
   if (d>0)
{
  x1 = (-b+sqrt(d))/(2*a);
  x2 = (-b-sqrt(d))/(2*a);
 cout << "x1=" << x1 <<"\n";
 cout << "x2=" << x2 <<"\n";
}
  if (d==0)
{
 x=-b/2*a;
 cout <<"X="<< x <<"\n";
}
   if (d<0)
{
 cout << "Uravnenieto nqma reshenie"<<"\n";
}
system( "pause" );
return 0;
}
```

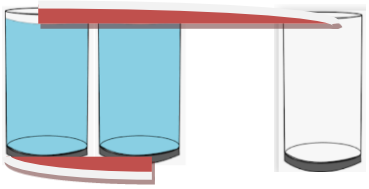
Задача 10. Напишете програма, която да изчислява броя на секундите в една година

```
#include <iostream>
using namespace std;
int main() {
   cout<<"Number of seconds in a year is:";
   cout<<60*60*24*365;
   return 0;
}
```

Резултат:
Number of seconds in a year is:31536000

Задача 11. Да се декларират две променливи a и b от тип int и да им се въведат стойности от клавиатурата. Да се разменят стойностите им и да се изведат на екрана

```
#include<iostream>
using namespace std;
  int main() {
    int a ,b,r; //въвеждаме помощна променлива r
    cout<<"a=";
    cin>>a;
    cout<<"b=";
    cin>>b;
    r=a;
    a=b;
    b=r;
    cout<<"a="<<a<<endl;
    cout<<"b="<<b<<endl;
   return 0;
}
```



1. Течността от чаша a се прехвърля в допълнителната чаша r
2. Течността от чаша b се прехвърля в чаша a
3. Течността от чаша r се прехвърля в чаша b

Резултат:
a=5
b=7
a=7
b=5

Задача 12. Да се състави програма, която въвежда от клавиатурата цяло число *к* и пресмята и извежда стойността на булевия израз:
!((к+325)%2==1

```
#include<iostream>
using namespace std;
int main()
{
  int k;
  cin>>k;
```

```
  cout<<(!((k+325)%2==1))<<endl;
  return 0;
}
```

Резултат:
35
1

Задача 13. Да се състави програма, която въвежда от клавиатурата цяло положително осемцифрено число. Програмата да пресмята и извежда средноаритмитичната стойност на цифрите на въведеното число.

```
#include<iostream>
using namespace std;
int main()
{
  unsigned long int a;
  double sum;
  cout<<"Въведете цяло осемцифрено число";
  cin>>a;
  sum=a/10000000;
  sum+=a/1000000%10; //sum=sum+a/10000000
  sum+=a/100000%10;
  sum+=a/10000%10;
  sum+=a/1000%10;
  sum+=a/100%10;
  sum+=a/10%10;
  sum+=a%10;
  cout<<sum/8<<endl;
  return 0;
}
```

Резултат:
Въведете цяло осемцифрено число88888888

8

ТЕСТ

1. **Кои са допустимите символи в езика C/C++ ?**
2. **Кои са петте основни типове данни в C/C++?**
3. **Какъв е размерът на int в битове?**
4. **Какъв е размерът на char в байтове?**
5. **Как ще напишете b^x използвайки матем. функция pow ?**
6. **С кои цифри се представят true и false ?**
7. **С коя дума се декларира логическият тип?**
8. **Какъв е отговорът на логическото умножение false&&true?**
9. **Какъв е отговорът на логическото събиране true||false?**
10. **Какъв отговор ще изкара на изхода израза cout<<(char)98; ?**
11. **За какво се използват двойните кавички?**
12. **Как ще инициализирате дробната променлива b със стойност 13.23?**

ТЕСТ -отговори

1. **Кои са допустимите символи в езика C/C++ ?**

малки и главни латински букви
цифрите от 0 до 9
специални символи:
+ - * / = () [] {} | & ! < > # $ ^ % ^ ~ _ . , : ; ‘ "
разделители – интервал, табулация и нов ред

2. **Кои са петте основни типове данни в C/C++?**

void, char, int, float , double

3. **Какъв е размера на int в битове?**

16 или 32

4. **Какъв е размера на char в байтове?**

1

5. **Какво прави floor() ?**

- намира най-голямото цяло число

6. **Как ще напишете b^x използвайки матем. функция pow ?**

pow(b,x)

7. **С кои цифри се представят true и false ?**

1 и 0

8. С коя дума се декларира логическият тип?

Bool

9. Какъв е отговорът на логическото умножение false&&true?

false

10.Какъв е отговорът на логическото събиране true||false?

true

11. Какъв отговор ще изкара на изхода израза cout<<(char)98; ?

ще изведе символа b

12.За какво се използват двойните кавички?

за извеждане на символен низ

13.Как ще инициализирате дробната променлива b със стойност 13.23?

float b=13.23;

РАЗЛИКИ МЕЖДУ „C” И „C++”

- В **„C”** ако дадена функция не приема параметри, то нейният прототип съдържа думата **void** в списъка с параметри
  char f1(void)
  В **C++** употребата на **void** не е задължителна . Поради тази причина прототипът обикновено се изписва така:
  char f1();
- В **C++** всички функции трябва да имат прототип, за разлика от **C,** където проттотипите не са задължителни
- В **C++** ако една функция е декларирана като връщащ резултат, то тя трябва да върне резултат. Тоест, ако типът на връщащият резултат на дадена функция е различен от **void**, всяка конструкция return в тялото на функцията трябва да връща стойност. Докато в **C**, една не-**void** функция не е задължена да връща резултат – ако тя не върне реазултат, се връща неопределен резултат.
- В **C** локалните променливи могат да се декларират само в началото на блок, преди някаква конструкция за действие. В **C++** локалните променливи могат да бъдат декларирани навсякъде и предимството в случая е, че те могат да бъдат декларирани непосредствено на мястото където се използват за първи път.
- **C** дефинира тип данни **bool**, който се използва за съхранение на булеви стойности (истина/лъжа/ и ключовите думи **true** и **false**, които са единствените стойности, които може да приема една променлива от типа **bool**. В **C++** резултата от релационните и логически оператори стойност тип **bool**, а всички условни конструкции трябва да връщат резултат от тип **bool**.