

# Технология программирования на C++

Начальный курс

Лекция 1

Pascal ABC

Файл Правка Вид Программа Сервис Помощь

\*квадратное уравнение2.pas

```
program kvur;
uses crt;
var a,b,c,d,x1,x2:real;
begin

write('введите коэффициенты=');
readln(a,b,c);
d:=b*b-4*a*c;

if d=0 then
begin
x1:=(-b+sqrt(d))/(2*a);
writeln('x1=x2=', x1);
end;

if d>0 then
begin
x1:=(-b+sqrt(d))/(2*a);
x2:=(-b-sqrt(d))/(2*a);
writeln('x1=', x1);
writeln('x2=', x2);
end;

if d<0 then
writeln('корней нет');

end.
```

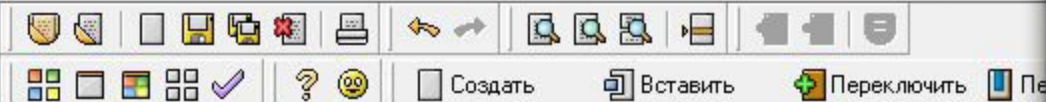
Строка: 28 Столбец: 11

```
#include <conio.h>
#include <stdio.h>
#include <math.h>
main()
{
    int a,b,c,d;
    float x1,x2;
printf("Введите коэффициенты \n");
scanf("%d%d%d", &a, &b, &c);
d=(b*b)-(4*a*c);

if (d==0)
{
x1=(-b+sqrt(d))/(2*a);
printf("x1=x2= %f \n",x1);
}

if (d>0){
x1=(-b+sqrt(d))/(2*a);
x2=(-b-sqrt(d))/(2*a);
printf("x1= %f \n",x1);
printf("x2= %f \n",x2);
}

if (d<0)
    printf("корней нет \n");
getch();
return 0;
}
```



```
#include <conio.h>
#include <stdio.h>
#include <math.h>
main()
{
    int a,b,c,d;
    float x1,x2;
    printf("Введите коэффициенты \n");
    scanf("%d%d%d", &a, &b, &c);
    d=(b*b)-(4*a*c);

    if (d==0)
    {
        x1=(-b+sqrt(d))/(2*a);
        printf("x1=x2= %f \n",x1);
    }

    if (d>0){
        x1=(-b+sqrt(d))/(2*a);
        x2=(-b-sqrt(d))/(2*a);
        printf("x1= %f \n",x1);
        printf("x2= %f \n",x2);
    }

    if (d<0)
        printf("корней нет \n");
    getch();
    return 0;
}
```



```
program kvur;
uses crt;
var a,b,c,d,x1,x2:real;
begin

write('введите коэффициенты=');
readln(a,b,c);
d:=b*b-4*a*c;

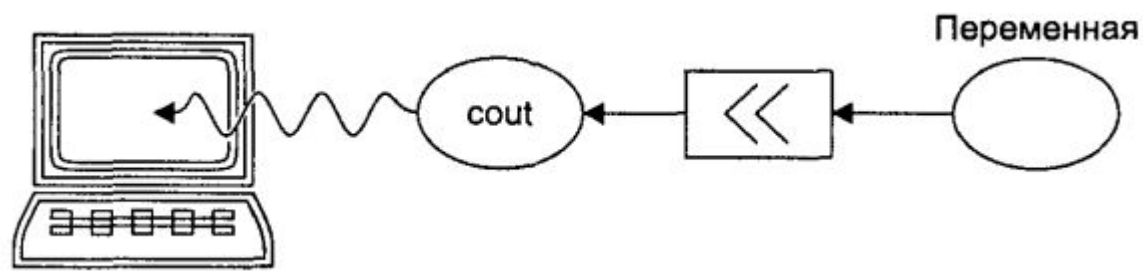
if d=0 then
begin
x1:=(-b+sqrt(d))/(2*a);
writeln('x1=x2=', x1);
end;

if d>0 then
begin
x1:=(-b+sqrt(d))/(2*a);
x2:=(-b-sqrt(d))/(2*a);
writeln('x1=', x1);
writeln('x2=', x2);
end;

if d<0 then
writeln('корней нет');

end.
```

```
#include <iostream>
int main()
{
    using namespace std;
    int carrots;
    cout << "How many carrots do you have?" << endl;
    cin >> carrots; // ввод C++
    cout << "Here are two more. ";
    carrots = carrots + 2;
    // следующая строка выполняет конкатенацию вывода
    cout << "Now you have" << carrots << " carrots." << endl;
    return 0;
}
```



# Структура С-программы

```
#Директивы препроцессора // включение библиотеки  
using namespace std; // стандартная область имен  
Объявление функций (их прототипов)  
// описание глобальных переменных и констант  
int main(параметры)  
{  
    // описание локальных переменных и констант  
    // операторы  
return 0;  
}  
Описание функций
```



Операторы — это выражения C++,  
завершаемые точкой с запятой

# Препроцессор

Выполняет набор инструкций перед началом компиляции программы. Все директивы препроцессора начинаются с #.

## **#include**

Директива включения файла. Используется для вызова файлов включений, содержащих описания библиотечных функций.

```
#include <stdlib.h>
```

```
#include "menu.h"
```

Если имя файла заключено в угловые скобки (<>), считается, что нужен некий стандартный заголовочный файл, и компилятор ищет этот файл в predetermined местах. (Способ определения этих мест сильно различается для разных платформ и реализаций.) Двойные кавычки означают, что заголовочный файл – пользовательский, и его поиск начинается с того каталога, где находится исходный текст программы.



# Препроцессор

**#define**

Определение символического обозначения.

```
#define PI 3.14159
```

# Описание переменных

Переменные, используемые программой, должны быть предварительно описаны.

Начинается имя с буквы или символа подчеркивания "\_", может содержать буквы латинского алфавита, цифры и знак подчеркивания.

*int i,j,k;*

Возможна инициализация переменных при описании: *int i = 0;*

# Описание переменных

Переменные целого типа: bool, char, short, int, long

Переменные с плавающей точкой: float, double, long double

# Символьные константы

Состоят из символа, заключенного в одинарные кавычки: 'a', 'b', '0'. Константа является целым числом равным коду символа.

Для ввода специальных символов зарезервированы управляющие символьные константы.

# Символьные константы

- `\a` звуковой сигнал
- `\b` возврат курсора на одну позицию
- `\f` перевод страницы
- `\n` перевод строки
- `\r` возврат каретки
- `\t` табуляция
- `\v` вертикальная табуляция
- `\\` обратный слеш
- `\'` одинарная кавычка
- `\"` двойная кавычка
- `\?` знак вопроса
- `\bbb` любой символ, где `bbb` - восьмеричное число
- `\hhh` любой символ, где `hhh` - шестнадцатеричное число

# Строковые константы

Последовательность символов, заключенная в двойные кавычки ("текст"), рассматривается как текстовая константа. В C++ принято считать признаком конца текста символ '\0', который добавляется автоматически. Таким образом, любая текстовая константа занимает в памяти область на 1 байт больше количества символов в последовательности.

# Переменные-константы

*const double PI = 3.14;*

*const int NULL = 0;*

# Операторы и операции

операция присваивания:

операнд = выражение;

$i = j + k;$

многократное присваивание

$i = j = k = 0;$

$i = 2 + (k = 3);$



# Арифметические операции

Операция	Пояснение
*	умножение
/	деление
%	остаток от деления ( <b>для целых</b> )
+	сложение
-	вычитание
функция <b>pow(x, n)</b>	операция возведения в степень

# Приведение типов

```
double avg, sum;
```

```
int n;
```

```
avg = sum/n;
```

```
double num = n;
```

```
int a = 3, b = 2;
```

```
double r = a/b;
```

1. `(double)a` – базовый оператор классического C.
2. `double(a)` – расширение языка, преобразование типа как функция.
3. `static_cast<double>(a)` – современный стиль.

```
double r = double(a)/b; //получим: r = 1.5;
```

# Операции ++ и --

- Увеличение ++ (increment)
- Уменьшение -- (decrement) значения переменной на 1.
- префиксная операция ++a; --a;
- постфиксная операция a++; a--;

*i = 0;*

*j = ++i; // j = 1, i = 1*

*k = i--; // k = 1, i = 0*

# Комбинированные операции

*i += j;      //i = i + j;*

*i -= j;      //i = i - j;*

*i \*= j;      //i = i \* j;*

*i /= j;      //i = i / j;*

*i %= j;      //i = i % j;*

*i <<= j;     //i = i << j;*

*i >>= j;     //i = i >> j;*

*i &= j;      //i = i & j;*

*i |= j;      //i = i | j;*

*i ^= j;      //i = i ^ j;*

# Операции отношения

- > больше* ( $a > b$ )
- >= больше или равно* ( $a \geq b$ )
- < меньше* ( $i < 0$ )
- <= меньше или равно* ( $i \leq j$ )
- == равно*** ( $i == k$ )
- != не равно* ( $ch \neq 'y'$ )

# Логические операторы

<code>&amp;&amp;</code>	И	<code>( i &gt; j ) &amp;&amp; ( k != 1 )</code>
<code>  </code>	ИЛИ	<code>(ch == 'y')    (ch == 'Y')</code>
<code>!</code>	НЕ	<code>!( i &gt; 1)</code>

# Операторы языка C++

Простой – заканчивается символом «;»

Составной – последовательность в фигурных скобках { }

# Условный оператор `if . . . else`

Сокращенная форма:

*if (логическое выражение) оператор;*

Полная форма:

*if (логическое выражение) оператор 1;*

*else оператор 2;*

*if ( i & 1 ) a = 0; else a = 1;*



# Условный арифметический оператор

*(условие)? выражение\_1 : выражение\_2;*

*a = (i & 1)? 0 : 1;*

# Операторы цикла

## 1. Цикл с предусловием `while`

*while* (логическое выражение) оператор;

Оператор выполняется до тех пор, пока логическое выражение истинно. Если условие сразу не выполняется, оператор ни разу не выполнится. Например:

```
a = i = 0;
```

```
while(i < n) a += ++i;
```

# Операторы цикла

## 2. Цикл с постусловием `do - while`

*do*

*оператор;*

*while (логическое выражение);*

Оператор выполняется пока логическое выражение истинно. Оператор будет выполнен хотя бы один раз.

*s = i = 0;*

*do*

*s += ++i;*

*while(i < n);*

# Операторы цикла

## 3. Цикл for

*for(Нач\_установки; Лог\_выражение;  
Приращение\_переменных) оператор;*

*for (i = 0, s = 0.0; i < n; i++) s += x[i];*

В операторе цикла могут отсутствовать все три выражения *for(;;) {...}*

# Вспомогательные операторы: **break, continue**

Используются в операторах цикла, а оператор `break` и в переключателе `switch`.

```
char ch;  
cout << " Enter number from 1 up to 9, 0 - the end\n";  
for (;;)   
{  
cin >> ch;  
if (ch < '0' || ch > '9') continue;  
if (ch == '0') break;  
...  
}
```

# Переключатель `switch`

```
switch (арифметическое выражение)
{
    case константа_1 : операторы_1;
    ...
    case константа_n : операторы_n;
    default : операторы;
}

switch (ch)
{
    case '+': c = a + b; break;
    case '-': c = a - b; break;
    default : c = 0;
}
```