

# Разработка приложений средствами С++ на основе технологии структурного программирования

Лекции 1,2.

8-917-225-21-45

[Nk\\_petrova@mail.ru](mailto:Nk_petrova@mail.ru)

Наталья Константиновна  
Петрова

## Задача №1

1. Построить таблицу функции

$$F(x, N) = \sum_{i=1}^N (-1)^i \frac{(2x)^{2i-1}}{(3i)!}$$

для  $x \in [-3 \cdot 10^1; +3 \cdot 10^1]$  с шагом 5.  $N$  – любое конечное целое число.

Использовать рекуррентные соотношения и не использовать функцию pow для получения степени (-1).

2. Результаты вычислений занести в массив и найти в нём элемент, которого ближе всего по значению к некоторому наперед заданному числу  $F$ .

### Алгоритм решения задачи состоит из следующих шагов:

1. Разработка функции  $F(x, N)$ , обеспечивающей вычисление суммы знакпеременного ряда.
2. Построение таблицы заданной функции на интервале от  $x \in [x_n, x_k]$  с шагом  $dx$ .
3. Формирование вещественного массива с вычисляемым количеством членов.
4. Поиск индекса необходимого элемента, наиболее близкого к  $F$ .

# Какие средства языка потребуются для решения задачи?

1. Описание и инициализация переменных
2. Способы консольного ввода/вывода данных
3. Арифметические операции и функции
4. Операции логического сравнения и логические функции
5. Логические операторы и условная операция
6. Циклические операторы
7. Описание функций и формальных параметров
8. Понятия «указателя» и «ссылки»
9. Описание динамических массивов
0. Способы передачи данных в функцию/из функции
1. Сборка проекта

# 1. Базовые типы данных языка C++

тип	байт	Диапазон принимаемых значений
целочисленный (логический) тип данных		
<b>bool</b>	1	true и false   0 до 255 (от 0 до $2^8$ )
целочисленный (символьный) тип данных		
<b>*signed char</b>	1	-128 до 127 (от $-2^7$ до $+2^7-1$ )
<b>char</b>	1	0 до 255 (от 0 до $2^8-1$ )
целочисленные типы данных		
<b>*signed short int</b>	2	-32 768 до 32 767 (от $-2^{15}$ до $+2^{15}-1$ )
<b>unsigned short int</b>	2	0 до 65 535 (от 0 до $+2^{16}-1$ )
<b>int</b>	4	<b>-2 147 483 648 до 2 147 483 647 (от <math>-2^{31}</math> до <math>+2^{31}-1</math>)</b>
<b>unsigned int</b>	4	0 до 4 294 967 295 (от 0 до $+2^{32}-1$ )
<b>long long</b>	8	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 ( $-2^{63}$ до $+2^{63}-1$ )
<b>unsigned long long</b>		от 0 до $2^{64}-1$ (до 18 446 744 073 709 551 615)
типы данных с плавающей точкой		
<b>float</b>	4	от 3.4E-38 до 3.4E+38 (точность до 7 цифр)
<b>double</b>	8	от 1.7E-308 до 1.7E+308 (точность до 15 цифр)
<b>long double</b>	10	от 3.4E-4932 до 3.4E+4932 (точность до 15 цифр)

**int** a, c=1, d=-1000, g; **float** a, b=1e2,h, c=1.2e-2;

## 2.a Консольный ввод

Для использования этого способа необходимо включить заголовочный файл `<iostream>`.

При запуске консольного приложения открываются потоки:

**`cin` (console input)** — для ввода с клавиатуры, и

**`cout` (console output)** — для вывода на монитор. Эти потоки определены посредством `<iostream>`.

```
cin >> a >> b >> c;
```

**`>>`** – оператор извлечения или направление потока.

оператор позволит ввести с консоли три значения, разделенных или пробелом, или клавишей Enter, в переменные `a`, `b`, `c`, соответственно.

## 2.b Консольный вывод

При использовании *cout* для вывода на экран необходимо так же указать одну или несколько переменных, значения которых будут помещены в выходной поток на экран.

Оператор *cout* помещает данные в поток с помощью оператора вставки (<<).

```
cout <<" a = " << a << " b = " << b;
```

позволит вывести на экран значения переменных *a*, *b* с соответствующими подписями.

### Управляющие символы C++

<code>\n</code>	новая строка	= манипулятору <i>endl</i> (endline)
<code>\t</code>	горизонтальная табуляция	

```
cout <<" \n a = " << a << "\t" << " b = " << b << endl;
```

## 3.а Унарные арифметические операции по убыванию приоритетов

Знак операции	Название	Краткое описание
<b>++</b> <переменная>	Инкремент (префикс)	Увеличение на 1
<b>--</b> <переменная>	Декремент (префикс)	Уменьшение на 1
<b>+</b> <переменная>	Унарный плюс	+x-2
<b>-</b> <переменная>	Унарный минус	Изменение знака на противоположный -x+3
<b>!</b> <переменная>	Логическое отрицание	Not (инверсия)
<переменная> <b>++</b>	Инкремент (постфикс)	Увеличение на 1 после выполнения операции
<переменная> <b>--</b>	Декремент (постфикс)	Уменьшение на 1 после выполнения операции
<b>( &lt;тип&gt; )</b> переменная	Преобразование типа (дескриптор)	(int) x

Пусть  $x = 0$

$10 + ++x$  равно 11,  $x$  равен 1

a)  $x=0+1=1$ ; b)  $10+1=11$

Пусть  $x = 0$

$10 + x++$  равно 10,  $x$  равен 1

a)  $10+0=10$ : b)  $x=0+1=1$

## 2.b Арифметические операции

Знак операции	Название	Примечание
<b>+</b>	Сложение	
<b>-</b>	Вычитание	
<b>*</b>	Умножение	
<b>/</b>	Деление	Если оба операнда целочисленные, то работает как целочисленное деление, в противном случае тип результата определяется правилами преобразования
<b>%</b>	Остаток от деления	Применяется только к целочисленным операндам

**10 / 3** при целочисленном делении равно 3

**10./3** равно 3.333...(в числителе вещественная константа)

**10 % 3** равно 1 (это остаток от деления)

**int a = 17, b = 5; float c = a/b; c = 3.**

**float c = (float) a/b = a/ (float) b = (float)a / (float) b;**



## 2.c Операции присваивания

<i>Знак операции</i>	<i>Название</i>
<b>=</b>	Присваивание
<b>+=</b>	Сложение с присваиванием
<b>-=</b>	Вычитание с присваиванием
<b>*=</b>	Умножение с присваиванием
<b>/=</b>	Деление с присваиванием
<b>%=</b>	Остаток от деления с присваиванием

Операторы составных (кратких) присваиваний упрощают написание конструкций присваивания и обеспечивают более эффективный программный код.

Вместо  **$x = x + 10;$**  можно записать  **$x += 10;$**

Операторная пара **+=** указывает компилятору, что переменной **x** следует присвоить значение **x** плюс **10**.

### 3.с Математические функции из библиотеки `<cmath>`

Матем. функция	Имя функции	Название	Пример
$ x $	<code>abs(x)</code>	абсолютное значение X	<code>abs(-3.0)= 3.0</code> <code>abs(5.0)= 5.0</code>
	<code>sqrt(x)</code>	квадратный корень X	<code>sqrt(9.0)=3.0</code>
$\ln x$	<code>log(x)</code>	натуральный логарифм X	<code>log(1.0)=0.0</code>
$\lg x$	<code>log10(x)</code>	десятичный логарифм X	<code>log10(100)=2</code>
$e^x$	<code>exp(x)</code>	e в степени X	<code>exp(0)=1</code>
$a^x$	<code>pow(a,x)</code>	a в степени X	<code>pow(2,3)=8</code>
$\sin x$	<code>sin(x)</code>	синус x (x задаётся в радианах)	
$\cos x$	<code>cos(x)</code>	косинус x (x задаётся в радианах)	
$\operatorname{tg} x$	<code>tan(x)</code>	тангенс x (x задаётся в радианах)	
$\arcsin x$	<code>asin(x)</code>	Возвращает угол, синус которого равен x	
$\arccos X$	<code>acos(x)</code>	Возвращает угол, косинус которого равен x	
$\operatorname{arctg} X$	<code>atan(x)</code>	Возвращает угол, тангенс которого равен x	

<code>floor(x)</code>	Округляет x до целого в меньшую сторону	<code>floor(12.4)=12</code> <code>floor(-2.9)=-3</code>
<code>ceil(x)</code>	Округляет x до целого в большую сторону	<code>ceil(2.3)=3.0</code> <code>ceil(-2.3)=-2.0</code>
<code>fmod(x, y)</code>	Возвращает остаток от деления вещественных x / y	<code>fmod(4.4, 7.5) = 4.4 10</code> <code>fmod( 7.5, 4.4) = 3.1</code>

# 4. Операции логического сравнения и логические функции

## с. Операторы сравнения

Операция (выражение)	Оператор	Синтаксис выражения	Краткое описание/примечание
Равенство	<code>==</code>	<code>a == b</code>	$a = 1; b = 2; c = 1;$ $a == b ? false; a == c ? true$
Неравенство	<code>!=</code>	<code>a != b</code>	$a = 1; b = 2; c = 1;$ $a != b ? true; a != c ? false$
Больше	<code>&gt;</code>	<code>a &gt; b</code>	
Меньше	<code>&lt;</code>	<code>a &lt; b</code>	
Больше или равно	<code>&gt;=</code>	<code>a &gt;= b</code>	
Меньше или равно	<code>&lt;=</code>	<code>a &lt;= b</code>	

## д. Логические функции

Операция (выражение)	Оператор	Синтаксис выражения	Краткое описание/примечание
Инверсия, НЕ	<code>!</code>	<code>!a</code>	Не a
Конъюнкция, И	<code>&amp;&amp;</code>	<code>a &amp;&amp; b</code>	a И b
Дизъюнкция, ИЛИ	<code>  </code>	<code>a    b</code>	a ИЛИ b

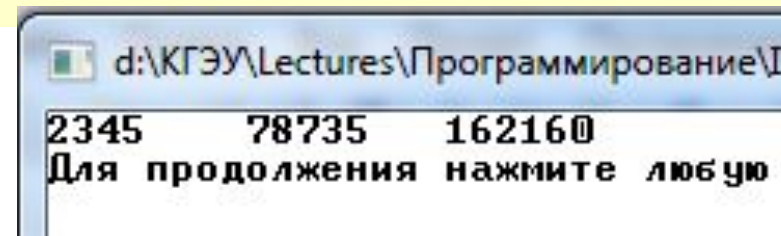
## 5.a Логический оператор *if*

```
if (/*проверяемое условие*/)
{
    /*тело оператора выбора 1*/;
} else
    {
        2,345· 103,
        /*тело оператора выбора 2*/;
    }
```

Формат числа с плавающей точкой

Пример 1: Если в ячейке А денег меньше, чем в ячейке В, то в ячейку С положить удвоенное значение суммы ячеек А и В, иначе в ячейку С положить разность денег между А и В.

```
double A=2.345e3, B=7.8735e4, C;
if (A<B)
    C=2*(A+B);
else C=B-A;
cout << A <<'\t' <<B <<'\t'<<C<<endl;
```



```
d:\КГЭУ\Lectures\Программирование\
2345 78735 162160
Для продолжения нажмите любую
```

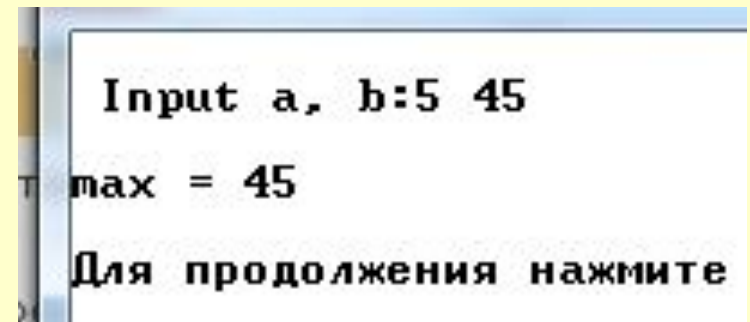
## 5.b Условная (тернарная) операция

A=условие ? операторы 1 : операторы 2;

```
double A=2.345e3, B=7.8735e4, C;  
if (A<B)  
C=2*(A+B);      C=A<B?2*(A+B):B-A;  
else C=B-A;  
cout << A << '\t' << B << '\t' << C << endl;
```

Пример 2: напечатать наибольшее из двух чисел.

```
float a,b,c;  
cout<<"\n Input a, b:"; cin>>a>>b;  
c=a>b?a:b;  
cout <<"\nmax = "<<c<<endl<<endl;
```



```
Input a, b:5 45  
max = 45  
Для продолжения нажмите
```

# 6.1 Счетный оператор цикла **for**

Параметр  
цикла

Начальное  
значение

Изменение  
параметра цикла

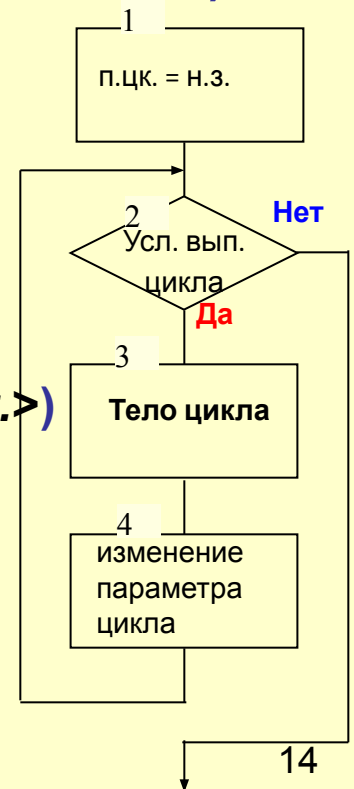
```
for(<п.цк.> = <н.з.>; <условие выполнения цикла>; <изм. п.цк.>)  
    <оператор>;
```

← **тело**  
цикла

Любой оператор

```
for(<п.цк.> = <н.з.>; <условие выполнения цикла>; <изм. п.цк.>)  
{<оператор1>; <оператор2>; ... <операторn>; }
```

**тело цикла**



## 7.а Общая форма определения функции

```
Тип_функции Имя_функции ( [Список_параметров ] )  
{  
Операторы; // Тело_функции  
    return [значение];  
}
```

Здесь **Тип\_функции** определяет тип величины, возвращаемого функцией. Функция может возвращать любой тип за исключением массива. Если функция ничего не возвращает, то тип возврата должен быть **void – пустой**.

**Имя\_функции** – любой допустимый идентификатор.

**Список параметров** представляет собой **последовательность пар типов и идентификаторов**, разделяемых запятыми.

**Параметры** – это переменные, которые получают значения аргументов, передаваемых функции при ее вызове. Если функция не требует параметров, то список параметров будет пуст.

Фигурные скобки окружают **тело функции**. Тело функции состоит из операторов, определяющих, что именно эта функция делает.

## 7.b Передача значений в функцию

В функцию можно передать одно или несколько значений.

Значение, передаваемое в вызываемую функцию называется аргументом или фактическим параметром и указывается в *обращении* к функции.

Соответствующие параметры в функции называются формальными параметрами.

Формальные параметры объявляются в *определении* (*прототипе*) функции.



## Разработка функции $F(x,N)$ , обеспечивающей вычисление суммы знакопеременного ряда.

$$F(x,N) = \sum_{i=1}^N (-1)^i \frac{(2x)^{2i-1}}{(3i)!}$$

Функция зависит от двух величин – вещественного аргумента  $x$  и целого положительного числа  $N$ . Она должна вернуть ОДНО значение вещественного типа. Тогда её прототип можно описать так:

**float F(float x, int N);**

Прототип показывает, КАК надо обратиться к функции и какого типа значения в неё можно передать.

$$-\frac{(2x)^1}{1 \cdot 2 \cdot 3} + \frac{(2x)^3}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6} \dots (-1)^N \frac{(2x)^{2N-1}}{1 \cdot 2 \cdot 3 \dots (3N-2) \cdot (3N-1) \cdot 3N}$$

Анализ формулы общего члена позволяет сделать вывод, что, **начиная с  $i = 2$** , для числителя, выполняется условие:

$$ch_i = ch_{i-1} \cdot (2x)^2$$

А для знаменателя - соотношение:

$$zn_i = zn_{i-1} \cdot (3 \cdot i - 2) \cdot (3 \cdot i - 1) \cdot (3 \cdot i)$$

Таким образом, нами получены **рекуррентные соотношения**, позволяющие, каждый последующий член вычислять через предыдущий используя простые выражения. Эти операции могут помещены в цикл.

Знак (+/-) отношения меняется при каждой итерации.

# Код функции F

```
float F(float x, int N)
{float tx=2*x,ch=tx, zn=1*2*3,U=ch/zn, s=U;
int i=1;
for(i=1;i<=N;i++)
{
    ch*=-tx*tx;
    zn*=(3*i-2)*(3*i-1)*3*i;
    U=ch/zn;
    s+=U;
}
return s;
}
```

## Построение таблицы заданной функции на интервале от $x \in [x_n, x_k]$ с шагом $dx$ .

Функция принимает 4 параметра – вещественные значения для границ интервала и шага табулирования, и целое значение  $N$ , определяющее количество членов в ряду функции  $F$ . Функция печатает результаты не возвращает никакого значения, поэтому она имеет тип `void`

```
void table(float xn, float xk, float dx, int N)
{
    int i=0;
    for(float x=xn; x<xk+dx/2;x+=dx,i++)
    {
        float y=F(x,N);

        cout<<"\t"<<x<<"\t"<<y<<endl;
    }
    cout<<endl<<endl;
    return;
}
```

# Главная программа

```
#include <iostream>
using namespace std;
float F(float x, int N); // вычисление суммы ряда функции F(x,N)
void table(float xn, float xk, float dx, int N); // Построение таблицы
функции
int main()
{
    float xn, xk, dx;
    int N;
    cout<<"Input xn xk dx:"; cin>>xn>>xk>>dx;
    cout<<"\n Input number of terms in a series:";cin>>N;
    table (xn, xk, dx, N);
    system("pause");
    return 0;
}
```

# Результат работы программы

```
d:\КГЭУ\Lectures\Программирование\II семестр\Лаб_раб\2018_
Input xn xk dx:-3.e1 +3e1 5
Input number of terms in a series:7
-30      363014
-25      14470.6
-20      1595.79
-15      205.583
-10      -88.034
-5       7.22039
0        0
5        -7.22039
10       88.034
15       -205.583
20       -1595.79
25       -14470.6
30       -363014
Для продолжения нажмите любую клавишу . . .
```

# Порядок ввода программы

## 1. Загружаем среду разработки

Microsoft Visual C++ 2008 Express Edition.Ink  
или Visual Studio2017

## 2. Меню Файл – Создать – Проект

## 3. В диалоговом окне:

- *Общие – Пустой проект*
- Вводим *Имя*
- Задаем – Расположение
- Кнопка ОК

# Порядок ввода программы (продолжение)

**4. Меню Проект – *Добавить новый элемент***

**5. В диалоговом окне**

Visual C++

ФайлC++(.cpp)

Набираем Имя

Кнопка ДОБАВИТЬ

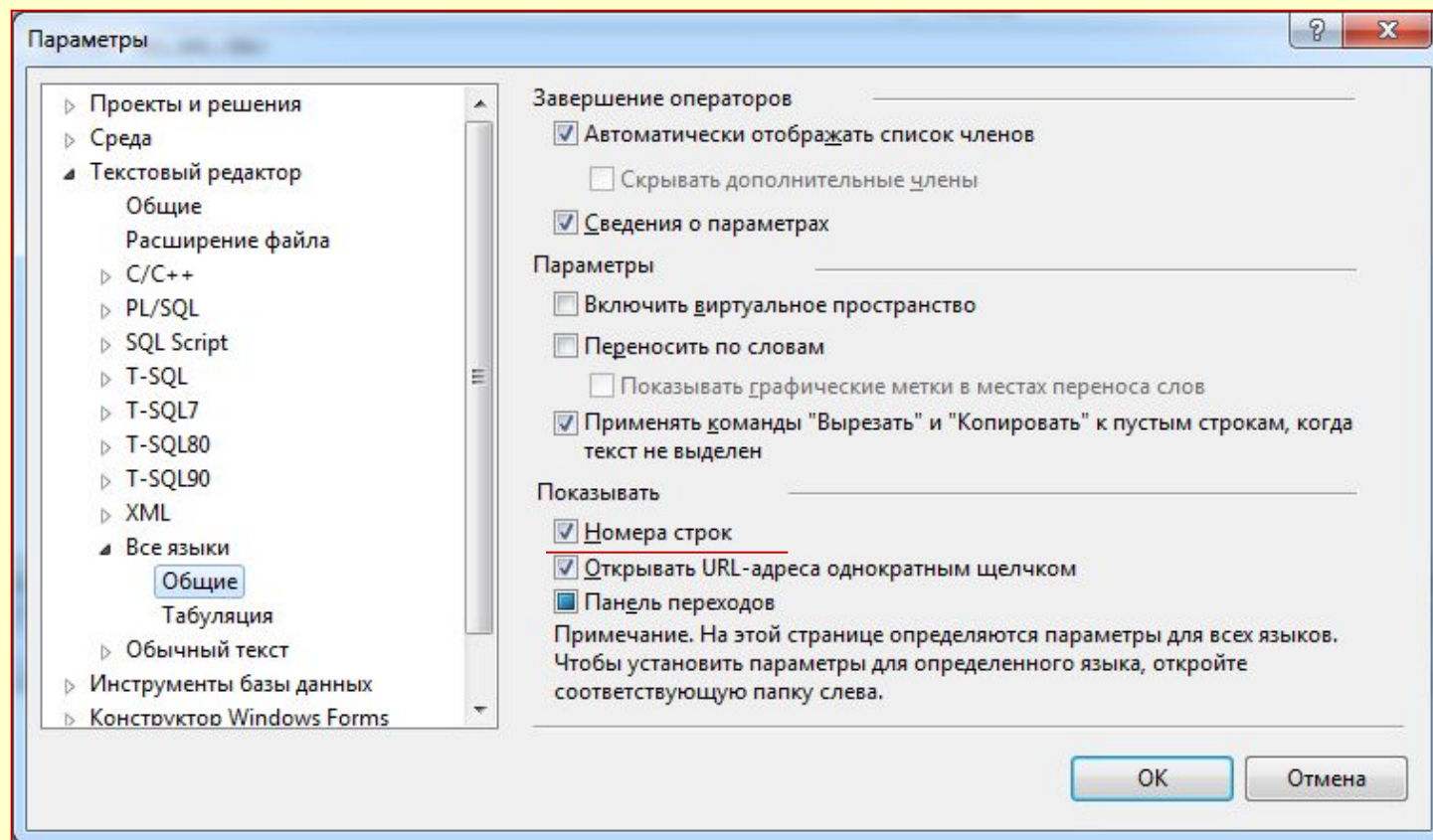
**6. В открывшемся окне набираем текст программы**

**7. Запускаем программу клавишей F5 или значком**



# Вставка нумерации строк в текст программы

Сервис/Параметры/Текстовый редактор/Все языки/Общие





# Проект задачи будет иметь вид

```
(Глобальная область)
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4 float f(float x, int N); // вычисление суммы ряда функции F(x,N)
5 void table(float xn, float xk, float dx, int N); // Построение таблицы функции
6 int main()
7 {
8     float xn, xk, dx;
9     int N;
10    cout<<"Input xn xk dx:"; cin>>xn>>xk>>dx;
11    cout<<"\n Input number of terms in a series:";cin>>N;
12    table (xn, xk, dx, N);
13    system("pause");
14    return 0;
15 }
16 float f(float x, int N)
```

Вывод

Показать выходные данные от: Отладка

```
"KP_ЗПИт.exe": Загружено: "D:\КГЗУ\Lectures\Программирование\II семестр\Лаб_раб\2018_осень\KP_ЗПИт\Debug\KP_ЗПИт.exe", Символы загружены.
"KP_ЗПИт.exe": Загружено "C:\Windows\SysWOW64\ntdll.dll"
"KP_ЗПИт.exe": Загружено "C:\Windows\SysWOW64\kernel32.dll"
"KP_ЗПИт.exe": Загружено "C:\Windows\SysWOW64\KernelBase.dll"
"KP_ЗПИт.exe": Загружено "C:\Windows\winsxs\x86_microsoft.vc90.debugcrt_1fc8b3b9a1e18e3b_9.0.30729.1_none_bb1f6aa1308c35eb\msvcp90d.dll"
"KP_ЗПИт.exe": Загружено "C:\Windows\winsxs\x86_microsoft.vc90.debugcrt_1fc8b3b9a1e18e3b_9.0.30729.1_none_bb1f6aa1308c35eb\msvcr90d.dll"
"KP_ЗПИт.exe": Загружено "C:\Windows\SysWOW64\apphelp.dll"
"KP_ЗПИт.exe": Загружено: "ImageAtBase0x4ac90000", Символы не загружены.
"KP_ЗПИт.exe": Выгружено: "ImageAtBase0x4ac90000"
Программа "[2636] KP_ЗПИт.exe: Машинный код" завершилась с кодом 0 (0x0).
```

Готово

Строка 15    Столбец 2    Знак 2    ВСТ

ОБЛАСТЬ СООБЩЕНИЙ  
ОБ ОШИБКАХ

# Домашнее задание

1. Составить программу по вычислению полной поверхности и объема конуса по радиусу его основания и высоте.
2. Вводится номер семестра  $N \leq 10$ . Вывести фразу "Я проучился  $N$  семестров", согласовав слово "семестр" с числом  $N$ . Определить номер курса.

# Материалы в Moodle

## Информатика и программирование

[В начало](#) ▶ [Разработка](#) ▶ [ИнфПр](#)


### НАВИГАЦИЯ

[В начало](#)

- ▶ [Личный кабинет](#)
- ▶ [Страницы сайта](#)
- ▼ Текущий курс
  - ▼ **ИнфПр**
    - ▶ [Участники](#)
    - ▶ [Значки](#)
    - ▶ [Общее](#)
    - ▶ [Тема 1: Линейные алгоритмы.](#)
    - ▶ [Тема 2: Алгоритмы ветвления](#)
    - ▶ [Тема 3: Циклические операторы на VBA](#)
    - ▶ [Тема 4: Работа с массивами](#)
    - ▶ [Тема 5: Работа с символьными данными](#)
    - ▶ [Материалы для подготовки к зачёту](#)
    - ▶ [Тестирование по VBA](#)
    - ▶ [Материалы для ЗПИ и ЗПИ I курс](#)
    - ▶ [Материалы для ЗПИ II курс](#)
- ▶ [Мои курсы](#)

[+](#)  [Новости](#)

## + [Материалы для ЗПИ II курс](#)

- [+](#)  [Лекция 1 2 с++](#)
- [+](#)  [Лекция 3 do function](#)

## Программирование на C++

[В начало](#) ▶ [Разработка](#) ▶ [C++](#)

### НАВИГАЦИЯ





[В начало](#)

- ▶ [Личный кабинет](#)
- ▶ [Страницы сайта](#)
- ▼ Текущий курс
  - ▼ **C++**
    - ▶ [Участники](#)
    - ▶ [Значки](#)
    - ▶ [Общее](#)
    - ▶ [Алгоритмы и способы их описания](#)
    - ▶ [Базовые алгоритмы в реализации C++](#)
    - ▶ [Циклические алгоритмы](#)
    - ▶ [Структурное программирование](#)
    - ▶ [Динамические массивы на C++](#)
    - ▶ [Работа с указателями на функции](#)
    - ▶ [Пользовательские типы данных](#)
    - ▶ [Материалы к зачётам и экзаменам](#)
    - ▶ [Тема 9](#)
    - ▶ [Тема 10](#)
- ▶ [Мои курсы](#)

 [Новостной форум](#)

 [Порядок запуска консольных приложений](#)

## Алгоритмы и способы их описания

-  [Лекция № 1-2 Введение в программирование](#)
-  [Лекция 3 Типы данных на C](#)
-  [Лекция 4 Математические операции и функции](#)
-  [Лабораторная работа №1\\_линейные алгоритмы](#)

## Базовые алгоритмы в реализации C++