|  |
| --- |
| Насрутдинов М.Ф. |
| **Материалы к лекциям по C++ (1 семестр)** |
| Marat.Nasrutdinv@kpfu.ru |

|  |
| --- |
| Marat Nasrutdinov  18.09.2022 |

Оглавление

[Лекция 1 3](#_Toc130570848)

[Минимальный набор для компиляции программ C++ 3](#_Toc130570849)

[Установка MinGW 3](#_Toc130570850)

[Первая программа 4](#_Toc130570851)

[Типы данных. Простейшие операторы. 5](#_Toc130570852)

[Простейшие операторы 6](#_Toc130570853)

[Лекция 2. Управляющие конструкции. 7](#_Toc130570854)

[Условный оператор if 7](#_Toc130570855)

[Оператор выбора switch 8](#_Toc130570856)

[Условный тернарный оператор 9](#_Toc130570857)

[Циклы. 9](#_Toc130570858)

[Массивы. 11](#_Toc130570859)

[Лекция 3 12](#_Toc130570860)

[Работа с файлами 12](#_Toc130570861)

[Функции 13](#_Toc130570862)

[Лекция 4 16](#_Toc130570863)

[Указатели 16](#_Toc130570864)

[Выделение и освобождение памяти 17](#_Toc130570865)

[Создание динамического массива 17](#_Toc130570866)

[Лекция 5. Структуры 19](#_Toc130570867)

[Указатели 19](#_Toc130570868)

[Инициализация полей структуры 19](#_Toc130570869)

[Массивы структур 20](#_Toc130570870)

[Указатели на структуры и динамические списки 21](#_Toc130570871)

[Ссылки и литература 23](#_Toc130570872)

# Лекция 1

Языки программирования. Язык программирования С++. Как установить компилятор C++. Какие IDE можно использовать. Первая программа на C++

## Минимальный набор для компиляции программ C++

Компилятор вместе с IDE

* <https://www.embarcadero.com/ru/free-tools/dev-cpp>
* <https://visualstudio.microsoft.com/ru/vs/community/>
* <http://www.codeblocks.org/>

Только компилятор

* <https://sourceforge.net/projects/mingw/>

IDE без компилятора

* <https://www.jetbrains.com/clion/> (необходимо сначала установить компилятор)

## Установка MinGW

Для компиляции программ на C++ необходим компилятор. Под Windows можно поставить компилятор C/C++ MinGW.

MinGW (англ. Minimalist GNU for Windows), ранее mingw32 — набор инструментов разработки программного обеспечения для создания приложений под Windows. Включает в себя компилятор, родной программный порт GNU Compiler Collection (GCC) под Windows вместе с набором свободно распространяемых библиотек импорта и заголовочных файлов для Windows API.

Ссылки на инструкции

* <http://ci-plus-plus-snachala.ru/?p=3056>
* <http://microsin.net/programming/pc/mingw-w64-install.html>
* <https://o7planning.org/ru/10467/installing-c-cpp-compiler-mingw>
* <http://www.cyberforum.ru/cpp-cross-platform/thread83661.html>

1. Загружаем установщик mingw-get-setup.exe (или mingw-w64-install.exe для 64-битных систем)

2. Запустить установщик и пройти согласно подсказкам до конца установки (при выборе пакетов выбрать все пакеты в  **"Basic Setup"**).

3. Вместо папки по умолчанию (в C:\Program Files\ ) лучше выбрать папку на диске.

4. Прописываем путь к компилятору в переменной окружения.

После установки компилятора можно компилировать файлы. Сами файлы можно создавать в любом текстовом редакторе.

Например, если у нас создан файл test.cpp, который находится в папке C:\Папка1\...\Папка\_с\_программой, то **в командной строке** переходим в папку с файлом программы (cd C:\Папка1\...\Папка\_с\_программой). И отправляем файл на компиляцию (файл с программой, хотим получить файл mytest.exe)

Для компилятора g++

|  |
| --- |
| g++ -o mytest test.cpp |

Для компилятора VS

|  |
| --- |
| cl -o mytest test.cpp |

## Первая программа

Версия 1 - Hello world!

|  |
| --- |
| #include <iostream>  int main()  {  std::cout << "Hello World! \n";  return 0;  } |

Версия 2 - ввод и вывод чисел

|  |
| --- |
| #include <iostream>  //пространство имен std, можно не писать std::  using namespace std;  int main()  {  //русские буквы в консоли  setlocale(LC\_ALL, "Russian");  cout << "введите a: \n";  int a;  //читаем переменную с клавиатуры (потока ввода)  cin >> a;  //выводим результат  cout << "p= "<<4\*a<<"\n";  return 0;  } |

## Типы данных. Простейшие операторы.

Перед началом работы с переменной она должна быть описана. То есть надо указать компилятору какого типа должна быть переменная.

|  |  |
| --- | --- |
| Название типов данных |  |
| int; long; short; unsigned; | целые |
| char; | символьные |
| float; double | вещественные |

|  |
| --- |
| //Примеры описания  int a;  float c,b=4.9;  char ch='a'; |

Пример программы - 3

|  |
| --- |
| // В программе вводятся целые числа a и b, находится их сумма  #include "iostream"  int main()  {  //описание переменных  int a,b;  std::cout<<"enter a and b: \n";  std::cin>>a>>b;  std::cout<<"a + b= " << a+b <<”\n”;  return 0;  } |

## Простейшие операторы

**Арифметические операции:** +; -; \*; /; %; ++; --

a%b - остаток от деления a на b (a и b - целые)

++ - увеличение на 1 целого числа

-- - уменьшение на 1 целого числа

Для целых чисел операция / – это целочисленное деление. Например, 1/7 равно 0.

**Операции присваивания:**  =, +=, -=, \*=, /=, %=

**Логические операции:** &&, ||, !

**Операции отношения:**  < <= > >= == !=

**Упражнение:** найти сумму цифр четырехзначного числа.

# Лекция 2. Управляющие конструкции.

## Условный оператор if

Оператор **if** служит для того, чтобы выполнить какую-либо операцию в том случае, когда условие является верным.

Синтаксис записи оператора

|  |
| --- |
| if (/\*проверяемое условие\*/)  {     /\*тело оператора выбора 1\*/;  }  else  {     /\*тело оператора выбора 2\*/;  } |

Часть else может отсутствовать.

Пример.

|  |
| --- |
| // IfEx.cpp : Вводятся два числа. Выводится 1, если (a-b)>0,  //-1 если (a-b)<0 и 0, если (a-b)=0  #include <iostream>  using namespace std;  int main()  {  double a, b;  int sign = 0;  cout << "input a,b: \n";  cin >> a >> b;    if (a > b)  {  sign = 1;  }  else  {  if (a < b) {sign = -1;}  }  cout << "compare(a,b)= " << sign << "\n";  return 0;  } |

## Оператор выбора switch

Оператор выбора switch является удобной заменой множественного использования операторов if. Оператор switch сравнивает значение одной переменной с несколькими константами.

|  |
| --- |
| switch ( /\*variable\*/ ) {  case const1:    /\*Тут находится код, который необходимо выполнить, если переменная  variable будет равна const1\*/    break;  case const2:    /\*этот код выполнится, если variable будет равна const2\*/    break;  /\*...\*/  default:    /\*Код, который выполнится, если ниодно из константых значению не соответствует значение в переменной variable\*/    break;  } |

Пример.

|  |
| --- |
| …  int str, first, second;  cout << "Введите первое число: "; cin >> first;  cout << "Введите второе число: "; cin >> second;  cout << "Укажите какую операцию использовать: "cin >> str;  switch (str) {  case 5:  cout << "Сумма: " << first + second;  break;  case 8:  cout << "Вычитание: " << first - second;  break;  case 9:  cout << "Умножение: " << first \* second;  break;  case 7:  cout << "Деление: " << first / second;  break;  default:  cout << "Что-то пошло не так. Повторите попытку";  } |

## Условный тернарный оператор

|  |
| --- |
| (выражение)? Выражение\_1: Выражение\_2 |

|  |
| --- |
| int a,b;  int max = (a < b) ? b : a; |

## Циклы.

**Цикл** заставляет программу многократно выполнять определенное количество операций до тех пор, пока заданное условие не станет ложным.

**В языке C++ есть 4 типа циклов**:

* цикл while;
* цикл do while;
* цикл for;
* цикл foreach (добавили в C++11).

Цикл **while**

|  |
| --- |
| while (условие)  {     тело цикла;  } |

Пример. Вывод чисел от 0 до 10

|  |
| --- |
| #include <iostream>  int main()  {  int count = 0;  while (count < 10)  {  std::cout << count << " ";  ++count;  }  std::cout << "done!";  return 0;  } |

Цикл for

|  |
| --- |
| for (инициализация переменной; условие; изменение значения переменной)  {    // тело цикла (тут находится код который будет повторяться)  } |

Пример 1.

|  |
| --- |
| // Вывод чисел от 0 до 10  #include<iostream>  using namespace std;  int main()  {  for (int i = 1; i < 10; i++)  {  cout << "i=" << i << "\n";  }    return 0;  } |

Пример 2.

|  |
| --- |
| // вводим x и выводим сумму x/1+x2/2!+ … +x10/10!  #include<iostream>  #include <locale>  using namespace std;  int main()  {  double s = 0, p = 1, fact = 1, x = 0;  //для отображения русских букв  setlocale(LC\_ALL, "Russian");  cout << "введите x \n";  cin >> x;  for (int i = 1; i <= 10; i++)  {  fact \*= i;  p \*= x;  s = s + p / fact;  }  cout << "сумма x+x^2+...+x^10=" << s << "\n";  return 0;  } |

Цикл do **while**

|  |
| --- |
| do  {  операторы  }  while(условие) |

При необходимости можно прервать выполнение цикла с помощью операторов перехода

break - прервать цикл или оператор switch

continue - продолжить; начать новую итерацию цикла, пропустив остаток тела цикла

## Массивы.

Описание массивов

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  //описание целочисленного массива из 10 элементов  int x[10];  //ввод элементов с клавиатуры  //индексация массива всегда с 0  for (int i = 0; i < 10; i++)  {  cout << "input x[" << i << "]=";  cin >> x[i];  }  //вывод элементов массива  for (int i = 0; i < 10; i++)  {  cout << "x[" << i << "]=" << x[i] << "\n";  }  //задание массива поэлементно  int y[5] = { 1,2,3,4,5 };  //вывод элементов массива y  for (int i = 0; i < 5; i++)  {  cout << "y[" << i << "]=" << y[i] << "\n";  }  return 0;  } |

# Лекция 3

## Работа с файлами

<http://cppstudio.com/post/446/>

Для работы с файлами необходимо подключить заголовочный файл <fstream>**.** Файловый ввод/вывод аналогичен стандартному вводу/выводу, единственное отличие – это то, что ввод/вывод выполнятся не на экран, а в файл. Если ввод/вывод на стандартные устройства выполняется с помощью объектов cin и cout, то для организации файлового ввода/вывода достаточно создать собственные объекты, которые можно использовать аналогично операторам cin иcout.

|  |
| --- |
| // file\_ex.cpp : Пример работы с файлами.  //  #include <iostream>  #include <fstream>  using namespace std;  int main()  {  /\* связываем объект с файлом, по умолчанию файл ищется  в папке программы, лучше прописать полный путь \*/  ifstream input\_file("input.txt");  int N, M;  //проверяем открылся ли файл  if (!input\_file.is\_open()) // если файл не открыт  cout << "Файл не может быть открыт!\n";  else  {  input\_file >> N >> M;  cout << "sum =" << N + M << endl;  }  // связываем объект с файлом, при этом файл открываем в режиме  // записи, предварительно удаляя все данные из него  ofstream file\_output("data.txt");  //ofstream file\_output("data.txt", ios\_base::out);  // проверка отрытия файла  if (!file\_output.is\_open())  {  cout << "Файл не может быть открыт или создан \n";  // выполнить выход из программы  return 1;  }  file\_output << "sum="<<N+M ;  // программа больше не использует файл,  //поэтому его нужно закрыть  file\_output.close();  return 0;  } |

Файл для записи и чтения может быть открыт в разных режимах

Режимы открытия файлов

| Константа | Описание |
| --- | --- |
| ios\_base::in | открыть файл для чтения |
| ios\_base::out | открыть файл для записи |
| ios\_base::ate | при открытии переместить указатель в конец файла |
| ios\_base::app | открыть файл для записи в конец файла |
| ios\_base::trunc | удалить содержимое файла, если он существует |
| ios\_base::binary | открытие файла в двоичном режиме |

## Функции

Для написания больших программ лучше выделять блоки кода программы в функции. Это позволяет логически структурировать программу и повторно использовать код. В С++ предусмотрено объявление своих функций. Помимо своих функций программист может воспользоваться функциями определёнными в стандартных заголовочных файлах и сторонних библиотек.

Объявление функции

|  |
| --- |
| // структура объявления функций  Тип\_возвр\_значения имя\_функции (/\*параметры функции\*/)  {  // тело функции  } |

Пример

|  |
| --- |
| // func\_ex.cpp : Пример описания функции  //  #include <iostream>  using namespace std;  //функция вычисления факториала n!  float fact(int n)// заголовок функции  {  float result = 1;  for (int i = 2; i <= n; i++) // цикл вычисления значения n!  result \*= i; // накапливаем произведение в переменной result    return result;  }  int main(int argc, char\* argv[])  {  int n;  cout << "Enter number: \n";  cin >> n;  cout<<"n!=" << fact(n)<<endl;// запуск функции нахождения факториала    return 0;  } |

Параметры функции передаются по значению, то есть при входе в функцию создаются локальные копии переменных, изменения которых не отражаются на исходные данные (если мы не используем ссылки или указатели).

Для изменения значений переменной внутри функции используют механизм ссылок или указатели. В примере используются ссылки.

|  |
| --- |
| #include <iostream>  using namespace std;  //прототипы функций  void swap(int x, int y);  void swap2(int &x, int&y);  int main(int argc, char\* argv[])  {  int x = 2, y = 5;  cout << "before func: \n";  cout << "x=" << x << endl;  cout << "y=" << y << endl;  //пробуем поменять местами x и y  swap(x, y);  cout << "after func: \n";  cout << "x=" << x << endl;  cout << "y=" << y << endl;  cout << "before func2: \n";  cout << "x=" << x << endl;  cout << "y=" << y << endl;  swap2(x, y);  cout << "after func2: \n";  cout << "x=" << x << endl;  cout << "y=" << y << endl;    return 0;  }  //описание функции (эта функция не будет менять переменные местами)  void swap(int x, int y) {  int z = x;  x = y;  y = z;  cout << "inside func: \n";  cout << "x=" << x<<endl;  cout << "y=" << y << endl;  }  //описание функции (а эта функция меняет переменные местами)  void swap2(int &x, int&y) {  int z = x;  x = y;  y = z;  cout << "inside func2: \n";  cout << "x=" << x << endl;  cout << "y=" << y << endl;  } |

Разбор примеров

1. Задана функция f(x). Найти ее корни на отрезке [a,b] методом деления пополам

2. Задана функция f(x). Найти ее корни на отрезке [a,b] методом касательных (Ньютона).

3. Задана функция f(x). Найти определенный интеграл на отрезке [a,b].

# Лекция 4

## Указатели

<http://cppstudio.com/post/423/>

Указатель – переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных  из области памяти, причём на самое его начало.

Указатель может ссылаться на переменную или функцию. Для этого нужно знать адрес переменной или функции.

Чтобы узнать адрес конкретной переменной в С++ существует унарная операция взятия адреса &.

Указатели могут использоваться для передачи по ссылке данных, что намного ускоряет процесс обработки этих данных (в том случае, если объём данных большой), так как их не надо копировать, как при передаче по значению, то есть, используя имя переменной.

В основном указатели используются для организации динамического распределения памяти, например при объявлении массива.

Любой указатель необходимо объявить перед использованием, как и любую переменную.

|  |
| --- |
| #include <iostream>  using namespace std;  int main(int argc, char\* argv[])  {  // инициализация переменной var числом 123  int var = 123;  // указатель на var (присвоили адрес переменной указателю)  int\* ptrvar = &var;  // адрес переменной var содержащийся в памяти,  // извлечённый операцией взятия адреса  cout << "&var = " << &var << endl;  // адрес переменной var, является значением указателя ptrvar  cout << "ptrvar = " << ptrvar << endl;  cout << "var = " << var << endl; // значение в переменной var  // вывод значения содержащегося в переменной var  // через указатель, операцией разыменования указателя  cout << "\*ptrvar = " << \*ptrvar << endl;  return 0;  } |

## Выделение и освобождение памяти

<https://code-live.ru/post/cpp-pointers/>

**Выделение памяти** осуществляется с помощью оператора **new.** После удачного выполнения такой операции, в оперативной памяти компьютера происходит выделение диапазона ячеек, необходимого для хранения переменной типа **int**.

Для того, чтобы освободить память, выделенную оператором **new**, используется оператор delete.

|  |
| --- |
| #include <iostream>  int main()  {  int\* a = new int; // Объявление указателя для переменной типа int  int\* b = new int(5); // Инициализация указателя  \*a = 10;  \*b = \*a + \*b;  std::cout << "b is " << \*b << endl;  delete b;  delete a;  return 0;  } |

## Создание динамического массива

<https://code-live.ru/post/cpp-dynamic-arrays/>

При объявлении статического массива, его размером должна являться числовая константа, а не переменная. В большинстве случаев, целесообразно выделять определенное количество памяти для массива, значение которого изначально неизвестно.

Например, необходимо создать динамический массив из **n** элементов, где значение **n** задается пользователем.

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  int num; // размер массива  cout << "Enter integer value: ";  cin >> num; // получение от пользователя размера массива  int\* p\_darr = new int[num]; // Выделение памяти для массива  for (int i = 0; i < num; i++) {  // Заполнение массива и вывод значений его элементов  p\_darr[i] = i;  cout << "Value of " << i << " element is " << p\_darr[i] << endl;  }  delete[] p\_darr; // очистка памяти  return 0;  } |

Разбор примеров

1. Задана функция f(x). Найти ее корни на отрезке [a,b] методом деления пополам

2. Задана функция f(x). Найти ее корни на отрезке [a,b] методом касательных (Ньютона).

3. Задана функция f(x). Найти определенный интеграл на отрезке [a,b].

# Лекция 5. Структуры

## Указатели

Структуры. Создание списка. Примеры.

<https://prog-cpp.ru/c-struct/>

Структура -- это объединение нескольких объектов, возможно, различного типа под одним именем, которое является типом структуры. В качестве объектов могут выступать переменные, массивы, указатели и другие структуры.

Структуры позволяют трактовать группу связанных между собой объектов не как множество отдельных элементов, а как единое целое. Структура представляет собой сложный тип данных, составленный из простых типов.

Общая форма объявления структуры:

|  |
| --- |
| struct StructName {  type\_1 elem\_name\_1;  type\_2 elem\_name\_2;  …  int elem\_name\_n;  }; |

После закрывающей фигурной скобки } в объявлении структуры обязательно ставится точка с запятой. Пример объявления структуры

|  |
| --- |
| #include <string>  struct date{  int day;  std::string month;  int year;  }; |

## Инициализация полей структуры

Инициализация полей структуры может осуществляться двумя способами:

* присвоение значений элементам структуры в процессе объявления переменной, относящейся к типу структуры;
* присвоение начальных значений элементам структуры с использованием функций ввода-вывода

В первом способе инициализация осуществляется по следующей форме (пример):

|  |
| --- |
| date bd = {11,"nov", 2020}; |

Второй способ инициализации объектов языка Си с использованием функций ввода-вывода.

|  |
| --- |
| date bd;  std::cin >> bd.day >> bd.month >> bd.year; |

## Массивы структур

Работа с массивами структур аналогична работе со статическими массивами других типов данных.

|  |
| --- |
| #include <iostream>  using namespace std;  struct book{  char title[15];  char author[15];  int price;  };  int main()  {  book library[3];    for (int i = 0; i < 3; i++)  {  cout<<"Введите название книги с номером " << (i + 1)<<endl;  cin >> library[i].title;  cout << "Введите автора книги : "<<endl;  cin >> library[i].author;  cout << "Введите цену книги : " << endl;  cin>> library[i].price;  }  } |

## Указатели на структуры и динамические списки

Доступ к элементам структуры можно осуществить с помощью указателей. Для этого необходимо инициализировать указатель адресом структуры или объединения.

Для организации работы с массивом можно использовать указатель. При этом обращение к полям структуры через указатель будет выглядеть как:

|  |
| --- |
| Указатель -> поле |

или

|  |
| --- |
| (\*указатель).поле |

Динамический список представляет из себя некоторое количество компонентов (узлов), содержащих данные, а также ссылку на следующий компонент (возможно, что узел содержит 2 ссылки: на следующий и на предыдущий, в таком случае, список называется двусвязным). Таким образом, получаем следующую структуру, описывающую компонент списка для нашей задачи:

|  |
| --- |
| struct comp {  char name[20]; // Имя переменной  char value[10]; // Значение переменной  comp\* next; // Ссылка на следущий элемент списка  }; |

Сам же список это совокупность его узлов. В этом семестре мы просто будем хранить ссылку на начало и конец списка. Обычно список оформляется в виде класса (структуры).

|  |
| --- |
| comp\* head; // Первый элемент (голова) списка  comp\* tail; // Последний элемент (хвост) списка |

Создание списка (наивная версия)

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  struct comp {  //string name; // Имя переменной  int value; // Значение переменной  comp\* next; // Ссылка на следущий элемент списка  };  int main(int argc, char\* argv[])  {  comp\* head;  comp \*tail, \*curr;  head = new comp;  head->value = -1;  head->next = NULL;  tail = head;  for (int k = 0; k < 5; k++) {  curr = new comp;  curr->value = k;  curr->next = NULL;  tail->next = curr;  tail = curr;  }  curr = head;  while (curr) {  cout << curr->value << "\n";  curr = curr->next;  }    return 0;  } |

# Ссылки и литература

<http://cppstudio.com/cat/274>/ - краткое описание конструкций языка