# Программирование

Две концепции с которыми работает компьютер и к которой привязаны все языки программирования

* Алгоритм
* Данные

# Процедурное программирование

**В данной концепции программирования делается акцент на алгоритм.**

# СП

**В данной концепции программирования делается акцент на алгоритм.** Тот же что и в процедурном только более упорядоченном и дисциплинированном.

# ООП

**В данной концепции программирования делается акцент на данные.**

В языке программирования C++ существуют понятия класса, который представляет

собой спецификацию, описывающую такую новую форму данных, и объекта, который

представляет собой индивидуальную структуру данных, созданную в соответствии с

этой спецификацией.

# Обобщенное программирование

**В обобщенном программировании акцент делается — на независимость от конкретного типа данных.**

# Парадигмыпрограммирования

1. Процедурное программирование
2. Структурное программирование
3. Объектно-ориентированное программирование
4. Обобщенное программирование

# С++



1. С помощью текстового редактора напишите свою программу и охраните ее в файле. Это будет **исходный код** вашей программы.

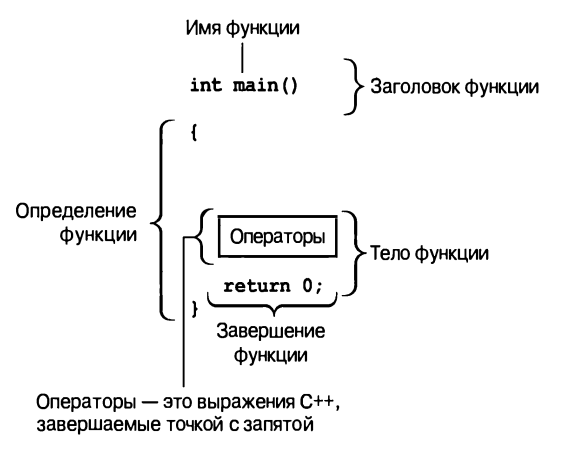
2. Скомпилируйте исходный код. Для этого необходимо запустить программу, которая транслирует исходный код во внутренний язык, называемый машинным языком рабочего компьютера. Файл, содержащий

транслированную программу — **это объектный код** вашей программы.

3. Свяжите объектный код с дополнительным кодом. Например, программы на C++ обычно используют библиотеки. Библиотека C++ содержит объектный код для набора компьютерных подпрограмм, называемых функциями, которые решают такие задачи, как отображение информации на экране монитора, нахождение квадратного корня числа и т.д. В процессе связывания ваш объектный код комбинируется с объектным кодом для используемых функций и с некоторым стандартным кодом запуска для формирования версии времени выполнения вашей программы. Файл, содержащий этот финальный продукт, называется **исполняемым кодом.**

Int main() заголовок функции

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Int main ( ) заголовок функции

Int возвращаемый результат

Main имя функции

( ) параметры или аргумент который сообщается вызываемой функции

Return 0; - может и не писаться но только для функции main( )

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Препроцессор — это программа, которая выполняет обработку файла исходного кода перед началом собственно компиляции.

### Директивы

#**include** –включение

**Using**- использовать

#include <**cmath**> или **math.h**

Пространство имен используется для стандартизации …и объединения старой и новой версии. То есть до введения пространства имен и после пространства имен

**Using namespace std;**  – говорится о том что в файле iostream находится набор стандартных имен для выведения на экран или другие действия и

Iostream – файл содержащий в себе набор действий такие как вывод ввод .

1. i-input

2. o- output

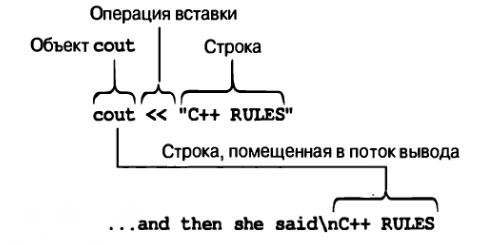
### Объект cout

**cout** –выводит информационный поток на экран “«” указывает направления следования информации

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

cout « "Come up and C++ me some time.";

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

В C++ любая последовательность символов, заключенных в двойные

кавычки, называется символьной строкой,

### Манипулятор endl

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

cout << endl;

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Здесь **endl** — это специальное обозначение в C++, которое представляет важное

понятие начала новой строки. Вставка endl в поток вывода означает, что курсор на

экране будет перемещен на начало следующей строки.

при выводе строки **cout** не переходит автоматически на следующую строку,

### Символ новой строки

Обозначить новую строку в выводе в C++ можно и старым способом —

посредством обозначения “**\n”** , принятого в языке С:

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

cout << "What's next?\n"; // \п обозначает начало новой строки

cout « "\n"; // начинает новую строку

cout « endl; // начинает новую строку

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

При вводе кода необходимо соблюдать некоторые правила. В частности, в С и C++ нельзя размещать пробел, табуляцию или возврат каретки в середине элемента, например в имени, и не допускается помещать возврат каретки в середину строки.

### Стиль написания исходного кода C++

* Один оператор в одной строке.
* Открывающая и закрывающая фигурные скобки для функции, каждая из которых находится в своей строке.
* Операторы в функции записаны с отступом от фигурных скобок.
* Вокруг круглых скобок, связанных с именем функции, пробельные символы отсутствуют.

### Операторы в С++

#### Оператор присваивани

**Оператор присваивания** –присваивает некоторое значение переменной (**«cat»** переменная **«=»** знак присваивания )

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Cat=25;

carrots = carrots - 1; // изменяет значение переменной

===========================================

// пример применения оператор дважды

int steinway;

int baldwin;

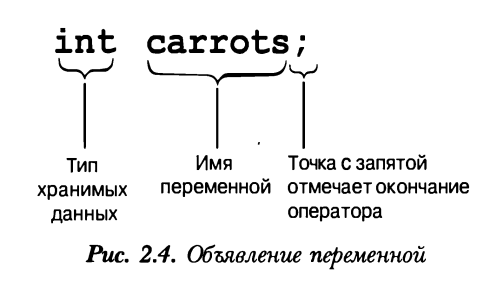
int yamaha;

yamaha = baldwin = steinway =88;

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

#### Оператор объявления переменной

**Оператор объявления переменной** –объявляет переменную и тип переменной (**«int»** –тип переменной **«cat»**–переменнная )

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Int cat;

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Совет

В языке C++ принято объявлять переменную как можно ближе к той строке, в которой она впервые используется.

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

// carrots.срр -- программа по технологии производства пищевых продуктов

// использует и отображает переменную

#include <iostream>

int main ()

{

using namespace std;

int carrots; // объявление переменной целочисленного типа

carrots = 25; // присваивание значения переменной

cout << "I have ";

cout << carrots; // отображение значения переменной

cout « " carrots.";

cout « endl;

carrots = carrots - 1; // изменение переменной

cout << "Crunch, crunch. Now I have " << carrots << " carrots." << endl;

return 0;

}

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

#### Оператор ввода данных

**Оператор ввода данных** **cin>>** используетсядляввода данных, различает между символьными значениями и цифрами

При помощи определения переменной

cin » carrots;

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

// getinf о. срр — ввод и вывод

#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

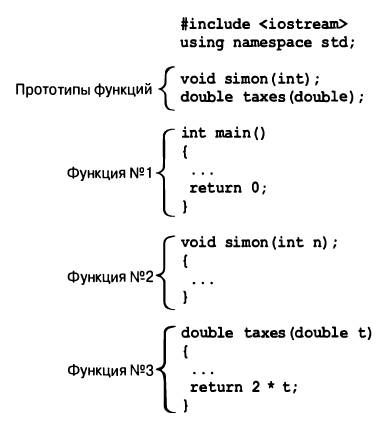
cout << "Now you have" << carrots « " carrots." << endl;

return 0;

}

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

### Функции

Функция main () возвращает при помощи оператора return значение 0 в ОС

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

squeeze = main(); // такого нет в наших программах

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Функция cout.put ()в программе отображается значение переменной ch и символьная константа.

#### Прототип не принимающих аргументов

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

int rand(void); // прототип функции, не принимающей аргументов генерирует случайное число

myGuess = rand(); // вызов функции без аргументов

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

#### прототип не возвращающий значения

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

void bucks(double); // прототип для функции, не имеющей возвращаемого значения

bucks(1234.56); // вызов функции; возвращаемого значения нет

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

#### Создание новой функции

Для случаев когда требуется ввести свою функцию ее нужно описать создать прототип…описать возвращаемые значения

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

тип имя\_функции(список\_аргументов)

{

операторы

}

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Return можно не употреблять если функция не возвращает никакого значения

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

// ourfunc.cpp — определение собственной функции

#include <iostream>

void simon(int); // прототип функции simon()

int main ()

{

using namespace std;

simon(3); // вызов функции simon()

cout « "Pick an integer: ";

int count;

cin » count;

simon(count); // еще один вызов simon()

cout « "Done!" << endl;

return 0;

}

//описание новой функции

void simon(int n) // определение функции simon()

{

using namespace std;

cout << "Simon says touch your toes " << n « " times." << endl;

} // функции void не требуют операторов return

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

#### Математические функции

Для начало что бы компилятор понял с какой функцией и с каким прототипом работать его нужно объявить в директиве

#include <cmath > или math.h

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя функции | Выполняемые действия | Прототип |
|  | Sqrt ( ); | Корень квадратный | Double sqrt (значение double) |
|  | Pow( ); | Возведение в степень  Указывается два значения | Pow(double, double);  X=pow(5.0, 8.0); x= 58 ; |
|  |  |  |  |
|  |  |  |  |

### Типы данных

### Фундаментальные типы данных

#### Целочисленные типы данных

Инициализация

Int cars =12;

Int cars={24};

Int cars {24};

Int cars (24);

Int cars={}; устанавливает значение 0

Int cars{} устанавливает значение 0

|  |  |  |  |
| --- | --- | --- | --- |
| № |  | Кол-во байтов |  |
|  | Char | 8 бит |  |
|  | Short | 16 бит |  |
|  | Integer (int) | 16-64 бит 4 байт в основном |  |
|  | Long | 32 бит |  |
|  | Long long | 64 бит |  |

|  |  |  |  |
| --- | --- | --- | --- |
| № |  |  |  |
|  | CHAR\_BIT | Количество битов в char |  |
|  | CHAR\_MAX | Максимальное значение char |  |
|  | CHAR\_MIN | Минимальное значение char |  |
|  | SCHAR\_MAX | Максимальное значение unsigned char |  |
|  | SCHAR\_MIN | Максимальное значение short |  |
|  | UCHAR\_MAX | Минимальное значение short |  |
|  | SHRT\_MAX | Максимальное значение unsigned short |  |
|  | SHRT\_MIN | Максимальное значение int |  |
|  | USHRT\_MAX | Минимальное значение int |  |
|  | INT\_MAX | Максимальное значение unsigned int |  |
|  | INT\_MIN | Максимальное значение long |  |
|  | UINT\_MAX | Минимальное значение long |  |
|  | LONG\_MAX | Максимальное значение unsigned long |  |
|  | LONG\_MIN | Максимальное значение long long |  |
|  | ULONG\_MAX | Минимальное значение long long |  |
|  | LLONG\_MAX | Максимальное значение unsigned long long |  |
|  | LLONG\_MIN | Максимальное значение unsigned char |  |
|  | ULLONG MAX | Максимальное значение short |  |

Без знаковые целочисленные переменные

|  |  |  |  |
| --- | --- | --- | --- |
| № |  |  |  |
|  | unsigned short change ; | // тип short без знака |  |
|  | unsigned int rovert; | // тип int без знака |  |
|  | unsigned quarterback; | // тоже тип int без знака |  |
|  | unsigned long gone; | // тип long без знака |  |
|  | unsigned long long lang\_lang; | // тип long long без знака |  |

**Десятичное представление числа**

12

**Восьмеричное представление числа**

012

cout « oct; // манипулятор для изменения основания системы счисления

std: :oct

**Шестнадцатеричное представление числа**

0xFC1

cout << hex; // манипулятор для изменения основания системы счисления

std: :hex

**char-** целочисленные тип данных который хранит принимает символьную информацию

содержит 8 бит для машин в которых байт =8 бит

**wchar\_t -** для более больших размеров где один символ превышает размер 8 битов

**wcout -** используют для вывода на поток

**wcin** - используют для ввода в поток

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Название символа | Название символа | Название символа | Название символа | Название символа |
|  | Новая строка | NL (LF) | \n | 10 | 0xA |
|  | Горизонтальная табуляция | НТ | \t | 9 | 0x9 |
|  | Вертикальная табуляция | ѴТ | \v | 11 | 0xB |
|  | Забой | BS | \b | 8 | 0x8 |
|  | Возврат каретки | CR | \r | 13 | 0xD |
|  | Предупреждение | BEL | \a | 7 | 0x7 |
|  | Обратная косая черта | \ | \\ | 92 | 0x5C |
|  | Знак вопроса | ? | \? | 63 | 0x3F |
|  | Одинарная кавычка | ‘ | \’ | 39 | 0x27 |
|  | Двойная кавычка | “ | \" | 34 | 0x22 |

char - // может быть со знаком, а может быть и без знака

signed char - //явное указание типа со знаком -128..127

unsigned char - //явное указание беззнакового типа 0..255

charl6\_t chl = u'q'; // базовый символ в 16-битной форме

char32\_t ch2 = U'/U0000222B'; // универсальное имя символа в 32-битной форме

#### Логические типы данных

**Bool**

**bool** start = -100; // start присваивается true

**bool** stop =0; // stop присваивается false

#### Квалификатор константа

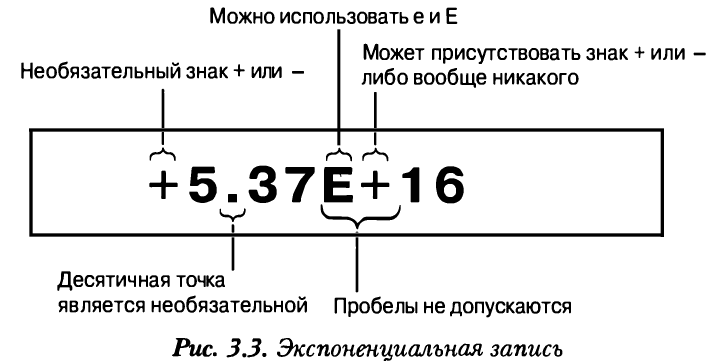
const int Months = 12; // Months — это символическая константа для 12

Ключевое слово const называется квалификатором, т.к. оно уточняет, что означает объявление.

const тип имя = значение;

#### Вещественные типы данных

Вещественные числа или числа с плавающей точкой-числа вида 2.35Е6

Цифра после символа Е это порядок, а все что левее то это мантисса

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Имя | Размер занимаем. | Минимальное количество значащих чисел #define | Кол-во битов, исп-мых для пред-ния мантиссы #define | Макс. значения порядка #define | Мин. значения порядка #define |
|  | Float | 32 бит | DBL\_DIG 15 | DBL\_MANT\_DIG 53 | DBL\_MAX\_10\_EXP +308 | DBL\_MIN\_10\_EXP -307 |
|  | Double | 64 бит | FLT\_DIG 6 | FLT\_MANT\_DIG 24 | FLT\_MAX\_10\_EXP +38 | FLT\_MIN\_10\_EXP -37 |
|  | Long double | 80-128 бит | LDBL\_DIG 18 | LDBL\_MANT\_DIG 64 | LDBL\_MAX\_10\_EXP +4932 | LDBL\_MIN\_10\_EXP -4931 |
|  |  |  |  |  |  |  |

Константы с плавающей точкой по умолчанию будет Double если только не указать другой тип

Пример

1.234f // константа float

2.45E20F // константа float

2.345324Е28 // константа double

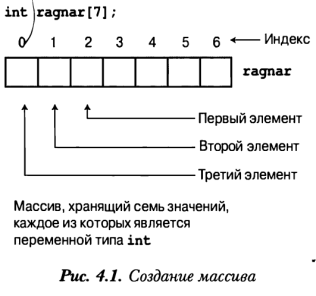
2.2L // константа long double

2.2l // константа long double

### структурные типы данных

К структурированным относится много типов

#### Массив

Массив это структура данных, которая представляет набор элементов значения которых состоят из одного и того же типа.

1. **Обьявление массива**

Тип значения

Имя массива

Количество элементов

Пример

**Int array[12];** Массив с именем **array**  и состоящий из **12** элементов

1. **Инициализация**

**Int array[3]={3,4,7};**

1. **Свойства**

**Массив имеет индексы которые начинаются с нуля**

**Для того что бы узнать размер массива**

**Sizeof array**

**Для того что бы узнать размер одного элементам**

**Sizeof array[1]**

1. **Также квадратные скобки могут быть пусты, если они внутри функции или при инициализации**
2. **При установки фигурных скобок пустыми, им автоматически присваивается нулевое значение**

#### Строки

Символьный массив

1. **Обьявление**

Char cat [8];

Тип

Имя

Количество элементов(символов)

1. **Инициализация**

**Двойные ковычки –строковая константа или строковой литерал**

Char day[10]={‘A’,’b’,’c’,d \0};

Char bird[12]=”Mr.Burbes”;

Char car[]=”Hello world”

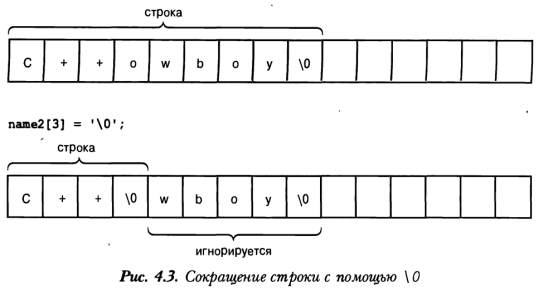
1. Получение размера и длины строки

Sizeof car

Strlen( )- только для начала нужно включить строковую директиву cstring

1. Установка нулевого символа после которого массив прекратит работу.

name2[3] = '\0'; // установка нулевого символа



### Арифметические операции

целочисленные арифметические операции

сумма +

разность –

произведение \*

частное /- получаемые результаты будет целое число бес остатка

остаток % получает остаток от деления двух целых чисел(только для целочисленных типов данных)

преобразования типов при инициализации и присвоение или арифметических операций

присваивается к значению меньшего типа, типа большего значения.

(long) thorn // возвращает результат преобразования thorn в тип long

long (thorn) // возвращает результат преобразования thorn в тип long

cout « int('Q'); // отображает целочисленный код для ' Q'

преобразование

В C++11 этот список претерпел некоторые изменения и представлен ниже.

1. Если один из операндов имеет тип long double, то другой операнд

преобразуется в long double.

2. Иначе, если один из операндов имеет тип double, то другой операнд

преобразуется в double.

3. Иначе, если один из операндов имеет тип float, то другой операнд

преобразуется в float.

4. Иначе, операнды имеют целочисленный тип, поэтому выполняется

целочисленное расширение.

5. В этом случае, если оба операнда имеют знак или оба операнда беззнаковые,

и один из них имеет меньший ранг, чем другой, он преобразуется в больший

ранг.

6. Иначе, один операнд имеет знак, а другой беззнаковый. Если беззнаковый

операнд имеет больший ранг, чем операнд со знаком, последний преобразует в тип

беззнакового операнда.

7. Иначе, если тип со знаком может представить все значения беззнакового типа,

беззнаковый операнд преобразуется к типу операнда со знаком.

8. Иначе, оба операнда преобразуются в беззнаковую версию типа со знаком.

# IDE

«Integrated Development Environment»

«Интегрированная среда разработки исходного кода»

# Языки программирования

«С»СИ создатель Деннис Ритчи

«С++» создатель Бьярне Страуструп,

«Fortran» процедурное программирование

«Basic» процедурное программирование

# Терминология

* **Нисходящее проектирование** - идея в том, чтобы разделить большую программу на небольшие, поддающиеся управлению задачи. Если после разбиения одна из задач все равно остается крупной, этот процесс продолжается до тех пор, пока программа не будет разделена на небольшие модули, с которыми будет просто работать.
* **Переносимой программой называется** – программа которую можно перекомпилировать, ничего в ней не меняя, и без помех запустит на других ОС.
* **Исходный код**- код который набранный в редакторе
* **Объекный** **код** – код который транслируется на машинный язык
* **Исполняемый** **код**-финальный код который связывает объектный код функций, дополнительный код
* **Оператор**- это законченная инструкция
* **Оператор** **возврата** – это финальная часть функции
* **Лексемами** называются неделимые элементы в строке кода
* **Двойные ковычки-**строковой литерал или строковая константа
* **Операнд-** значения переменных или констант над которыми производят арифметические действия.
* **Выражения-**это инструкция
* **Конкатенация –**комбинирование строковых литералов в один строковой литерал

# Стандарты

## «ANSI» -

American National Standards Institute.

## «ISO»

International Organization for Standardization

1. ISO 10646-является международным стандартом, разработка которого пока не завершена; он предлагает числовые коды для широкого диапазона символов.
2. ASCII
3. Unicode

## «IEC»

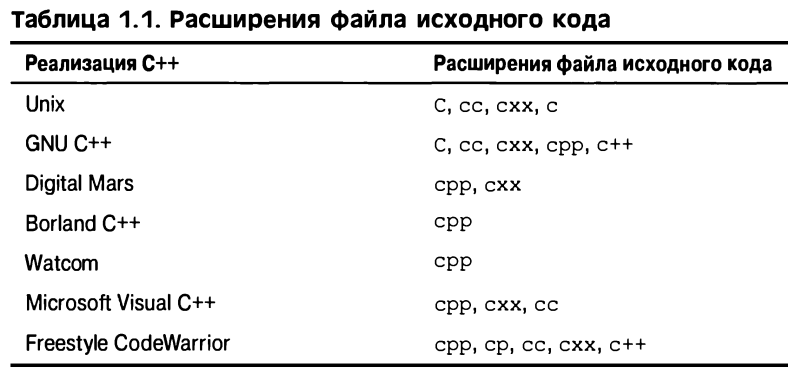
International Electrotechnical Commission

1. C++
2. C++98/ 800 страниц
3. C++03/
4. C++11/1350 страниц
5. C
6. K&R
7. C99

## DLL

Dynamic Link Library - Эта библиотека содержит код, который могут использовать другие Windows-программы.

# Расширение



# Компиляция

После того как проект настроен, нужно скомпилировать и скомпоновать свою программу. В IDE-среде обычно предлагается несколько вариантов, такие как **Compile (Компилировать)**, **Build (Построить), Make (Построить), Build All (Построить все), Link (Скомпоновать), Execute (Выполнить), Run (Выполнить) и Debug (Отладить)** но не обязательно все варианты сразу.

* **Compile** обычно означает компиляцию кода в открытом в настоящий момент файле.
* **Build** или **Make** обычно означает компиляцию кода для всех файлов исходного кода, входящих в состав данного проекта. Часто этот процесс является инкрементным. То есть, если в проекте было три файла, и вы изменили только один из них, то повторно скомпилирован будет только этот файл.
* **Build All** обычно означает компиляцию всех файлов исходного кода с самого начала.
* **Link** означает объединение скомпилированного исходного кода с необходимым библиотечным кодом.
* **Run** или **Execute** означает запуск программы. Обычно если вы еще не завершили выполнение предыдущих этапов, команда **Run** выполнит их перед запуском программы. Начало работы с C++ 47
* **Debug** означает запуск программы с возможностью ее пошагового выполнения.

Компилятор может поддерживать создание версий Debug (Отладочная) и Release (Выпуск). Версия Release содержит дополнительный код, который увеличивает размер программы и замедляет ее выполнение, но зато делает доступными средства отладки.

# Директивы

#include –включение

#define – Эта директива сообщает препроцессору следующее: найти в программе экземпляры символической

константы

Using- использовать

# Синтаксис **helper**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| A |  |  |  |
|  |  |  |  |
| B |  |  |  |
|  |  |  |  |
| C | Cin.get () | Остановка до нажатия клавиши |  |
|  | Cout<< | Вывод на поток |  |
|  | Cin >> | Ввод в поток |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| D |  |  |  |
|  |  |  |  |
|  |  |  |  |
| E |  |  |  |
|  |  |  |  |
|  |  |  |  |
| F |  |  |  |
|  |  |  |  |
| H |  |  |  |
|  |  |  |  |
| G |  |  |  |
|  |  |  |  |
|  |  |  |  |
| I |  |  |  |
|  |  |  |  |
| J |  |  |  |
|  |  |  |  |
| K |  |  |  |
|  |  |  |  |
|  |  |  |  |
| L |  |  |  |
|  |  |  |  |
|  |  |  |  |
| M |  |  |  |
|  |  |  |  |
| N |  |  |  |
|  |  |  |  |
|  |  |  |  |
| O |  |  |  |
|  |  |  |  |
| P |  |  |  |
|  |  |  |  |
| R |  |  |  |
|  |  |  |  |
| S | Sizeof | Оператор для вывода размера типа данных |  |
|  | System(“cls”) | Очистка консоли |  |
|  |  |  |  |
| T |  |  |  |
|  | // | Коментарии |  |
|  | {} | Тело программы (скобки операндов) |  |
|  |  |  |  |

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Программа будет ждать нажатия Enter после продолжит выполнения алгоритма

cin.get( ); // добавьте этот оператор

cin.get( ); //и, возможно, этот тоже

return 0;

}

= = = = = = = = = = = = = = = = =

// обозначение коментариев

= = = = = = = = = = = = = = = = =

#include директива препроцессора

= = = = = = = = = = = = = = = = = = =

Int main() заголовок функции

= = = = = = = = = = = = = = = = = = = =

Using namespase; директива

= = = = = = = = = = = = = = = = = = = = =

{} тело функции

= = = = = = = = = = = = = = = = = = =

Cout оператор для отображения выведения

= = = = = = = = = = = = = = = = = = = = = =

Printf() вывод (операторы СИ)

Scan () (операторы СИ)

= = = = = = = = = = = = = = = = = = = = = = =

Return возвращает в main () значение

= = = = = = = = = = = = = = = = = = = = = = = =

Stdio.h включение Си файла

Setlocale(LC\_CTYPE, “rus”); //CP866 кодировка

Getch; //остановка консоли до нажатия любой клавиши

# Бибдиотека С++

<iostream> ввод вывод

<cmath> математические функции

<conio.h> остановка консоли

<clocale> правка кодировки

<climits> установленное компилятором ограничение

<cfloat > значение принимающие вещественные типы или float.h

<cstring> для того что бы открыть строковую директиву или string.h