

МАКСИМ ДИНМАН

C++

ОСВОЙ НА ПРИМЕРАХ

ОСНОВЫ ПРОГРАММИРОВАНИЯ

ФУНКЦИИ, МАССИВЫ, СТРУКТУРЫ
ДАНЫХ И УКАЗАТЕЛИ

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ
ПРОГРАММИРОВАНИЕ

ПРИМЕРЫ АЛГОРИТМОВ ТЕОРИИ
ГРАФОВ И ШИФРОВАНИЯ



Максим Динман



ОСВОЙ НА ПРИМЕРАХ

Санкт-Петербург

«БХВ-Петербург»

2006

УДК 681.3.068+800.92C++
ББК 32.973.26-018.1
Д44

Динман М. И.

Д44 С++. Освой на примерах. — СПб.: БХВ-Петербург, 2006. — 384 с.: ил.

ISBN 5-94157-917-9

Подробно и доступно на занимательных примерах рассмотрены синтаксис, семантика и техника программирования на языке С++. Описаны все этапы проектирования программ, приведены подробные комментарии программного кода, проанализированы результаты вычислений, показаны типичные проблемы и пути их решения. Большое внимание уделяется алгоритмам и примерам решения задач при помощи графов, а также алгоритмам шифрования. Каждая глава содержит упражнения для самостоятельной работы.

Для учащихся и начинающих программистов

УДК 681.3.068+800.92C++
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Натали Каравасовой</i>
Корректор	<i>Наталия Першакова</i>
Дизайн серии	<i>Игоря Цырульников</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 27.06.06.

Формат 60×90^{1/16}. Печать офсетная. Усл. печ. л. 24.

Тираж 3000 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-917-9

© Динман М. И., 2006
© Оформление, издательство "БХВ-Петербург", 2006

Оглавление

Введение.....	1
B1. О названии	2
B2. О разработке	3
B3. В общих чертах о C++.....	3
Глава 1. Знакомство с приложением.....	5
1.1. Установка C++.....	5
1.2. Рабочее окно программы.....	7
1.3. Создание и сохранение документа	8
1.4. Компиляция программы	9
Глава 2. Основы программирования.....	13
2.1. Этапы разработки программы	13
Спецификация	14
Разработка алгоритма	14
Кодирование	14
Отладка	14
Тестирование	15
2.2. Элементы языка C++	15
Алфавит.....	15
Комментарии	17
Переменные	17
Типы данных	18
Преобразование типов	21
Константы	22
Арифметические и логические операции.....	23
2.3. Директивы препроцессора C++	25
Директива <i>#include</i>	25
Директива <i>#define</i>	26
Директива <i>#undef</i>	26
Директива <i>#error</i>	27
Директива <i>#typedef</i>	27
2.4. Ввод/вывод	28

2.5. Управляющие структуры.....	29
Структуры выбора.....	29
Структура выбора <i>if/else</i>	30
Структура выбора <i>switch</i>	32
Примеры структур выбора	33
Пример 1	33
Пример 2	34
Пример 3	35
Пример 4	36
Пример 5	38
2.6. Структуры повторения	38
Структура повторения <i>for</i>	38
Структура повторения <i>while</i>	39
Оператор цикла <i>do...while</i> с условием	39
Операторы <i>continue</i> и <i>break</i>	40
Оператор <i>goto</i>	42
Примеры структур повторения	43
Пример 1	43
Пример 2	44
Пример 3	45
Пример 4	47
Пример 5	49
Пример 6	51
2.7. Упражнения	52
Глава 3. Функции.....	55
3.1. Понятие функции	55
Свойства функции	56
Встроенные функции	56
Перегрузка функций	57
Арифметические и алгебраические функции	58
Тригонометрические функции	59
3.2. Функция-подпрограмма.....	59
Передача параметров функции <i>main()</i>	62
3.3. Ссылочный тип.....	62
Возврат значений из функции с помощью переменных ссылочного типа.....	63
3.4. Рекурсия	63
Шаблоны функций	65
3.5. Примеры программ с использованием функций	65
Пример 1	65
Пример 2	68
Пример 3	69

Пример 4	71
Пример 5	73
3.6. Упражнения	77
Глава 4. Массивы.....	79
4.1. Понятие массива	79
Ввод элементов массива	81
Вывод элементов массива	81
Операция <i>sizeof</i>	84
Передача одномерных массивов в функцию	84
Максимальный объем памяти, занимаемой массивом.....	85
4.2. Многомерные массивы	86
Обработка матриц	86
Обработка матрицы по столбцам и строкам	86
Обработка всей матрицы	87
Передача двумерных массивов в функцию.....	88
4.3. Символьные массивы	89
4.4. Сортировка	89
Пузырьковая сортировка	90
Выборочная сортировка	92
Быстрая сортировка	95
Сортировка методом Шелла	101
Сортировка массива при известном интервале значений элементов.....	104
4.5. Поиск заданного элемента в массиве	106
Последовательный поиск элемента	106
Двоичный поиск	107
4.6. Арифметика больших чисел.....	109
Вычитание чисел.....	109
Сложение чисел.....	117
Умножение чисел.....	121
4.7. Примеры использования массивов.....	124
Пример 1	124
Пример 2	127
Пример 3	129
Пример 4	131
Пример 5	134
4.8. Упражнения	136
Глава 5. Структуры данных	139
5.1. Очередь	139
Операции над очередями.....	141
Добавление элемента в очередь.....	141
Проверка очереди на наличие элементов.....	142

Удаление элемента из очереди.....	142
5.2. Стеки	142
Операции над стеками	143
Добавление элемента в стек	144
Проверка стека на наличие элементов	144
Удаление элемента из стека	144
5.3. Списки.....	145
Реализация операций над списками	146
Создание пустого списка	146
Добавление в список нового элемента	147
Поиск элемента в списке	149
Удаление элемента из списка	149
Добавление элемента в список после заданного элемента	150
Сортировка списков	151
Слияние списков	153
5.4. Примеры использования структур данных	154
Пример 1	154
Пример 2	159
Пример 3	163
Пример 4	165
Пример 5	167
5.5. Упражнения	173
Глава 6. Файлы	177
6.1. Обращение к файлам	178
Последовательная запись в файл	179
Последовательное чтение из файла	180
Произвольная запись в файл	181
Произвольное чтение из файла	181
6.2. Поиск в файле.....	182
Поиск и замена в файле	186
6.3. Примеры программ обработки файлов	189
Пример 1	189
Пример 2	190
Пример 3	192
Пример 4	194
Пример 5	198
6.4. Упражнения	200
Глава 7. Техника указателей	203
7.1. Понятие указателя.....	203
7.2. Указатели на массивы.....	204
Операции над указателями.....	206

Доступ к значениям, адресуемым указателями	207
7.3. Указатели на строку	208
7.4. Указатели на функцию	208
Функции, возвращающие указатель	209
7.5. Динамическое распределение памяти	210
Динамическое выделение памяти	210
Освобождение динамически выделенной памяти	212
7.6. Примеры использования указателей	213
Пример 1	213
Пример 2	215
Пример 3	219
Пример 4	221
Пример 5	223
Пример 6	225
7.7. Упражнения	227
Глава 8. Объектно-ориентированное программирование	231
8.1. Структуры	232
8.2. Классы	236
Вложенные классы	242
Дружественные функции	242
Производные классы	245
Построение производного класса	246
Инкапсуляция	248
Статические члены класса	249
Класс <i>ios</i>	251
Указатель <i>this</i>	251
8.3. Конструктор и деструктор	252
8.4. Наследование	256
8.5. Упражнения	257
Глава 9. Основы теории графов	261
9.1. Понятие графа	261
Представление неориентированного графа	262
Представление неориентированного графа в памяти компьютера ...	263
Представление ориентированного графа	266
Представление ориентированного графа в памяти компьютера	267
Смежность и инцидентность	270
Цепи и циклы	272
9.2. Обходы графов	273
Обход в ширину	273
Реализация поиска в ширину	275

Обход в глубину	282
Реализация поиска в глубину	288
9.3. Примеры графов.....	299
Связные и несвязные графы.....	300
Регулярные графы.....	303
Двудольные графы	305
Пустые и полные графы	310
9.4. Эйлеровы графы.....	312
Алгоритм и реализация построения эйлера цикла	314
9.5. Построение кратчайших путей	316
Алгоритм Дейкстры	318
Вывод кратчайшего пути.....	328
9.6. Раскраска вершин графа.....	332
9.7. Примеры решения задач при помощи графов	336
Пример 1	336
Пример 2	342
9.8. Упражнения	345
Глава 10. Основы шифрования.....	349
10.1. Классы алгоритмов шифрования	349
10.2. Требования к криптографическим системам.....	350
Криптоанализ.....	352
Безопасность алгоритмов	354
10.3. Перестановочные алгоритмы.....	356
Простой столбцевой перестановочный шифр	356
Перестановочный шифр с ключевым словом.....	359
10.4. Подстановочные шифры	362
Шифр Полибия.....	362
10.5. Шифрование сдвигом ASCII-значений	366
10.6. Криптосистема RSA.....	368
Математические идеи алгоритма.....	369
Алгоритм шифрования	370
Алгоритм дешифрования.....	370
Пример использования алгоритма RSA	371
10.7. Цифровые подписи	372
Предметный указатель	374

Глава 1



Знакомство с приложением

1.1. Установка C++

Для установки среды разработки C++ запустите файл Setup.exe, который находится в папке инсталляции программы. На экране появится диалоговое окно установки (рис. 1.1). Для продолжения нажмите кнопку **Next** (Далее), оставив все значения без изменения.

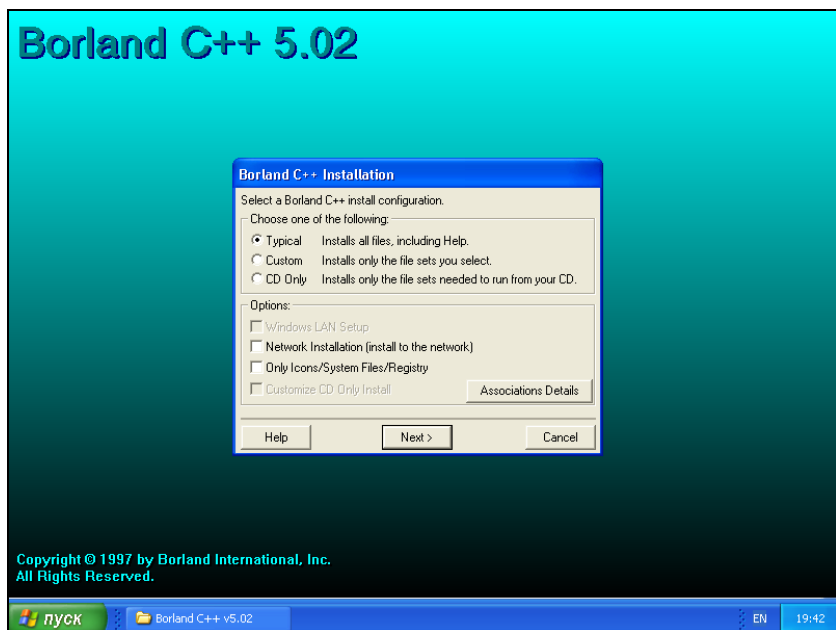


Рис. 1.1. Установка среды разработки C++

Затем укажите папку, в которой вы желаете хранить программу, например, C:\BC5, и нажмите кнопку **Next** (Далее). Значения параметров в следующих двух диалоговых окнах также оставьте без изменений.

На этом этап установки приложения завершен. Открыть среду разработки C++ для работы можно с помощью команды меню **Пуск | Все программы | Borland C++ 5.02 | Borland C++** (рис. 1.2).



Рис. 1.2. Открытие приложения Borland C++ 5.02

1.2. Рабочее окно программы

После открытия на экране вы увидите рабочее окно программы (рис. 1.3), которое состоит из следующих частей:

- ☐ меню;
- ☐ панель инструментов;
- ☐ рабочая область;
- ☐ строка состояния.

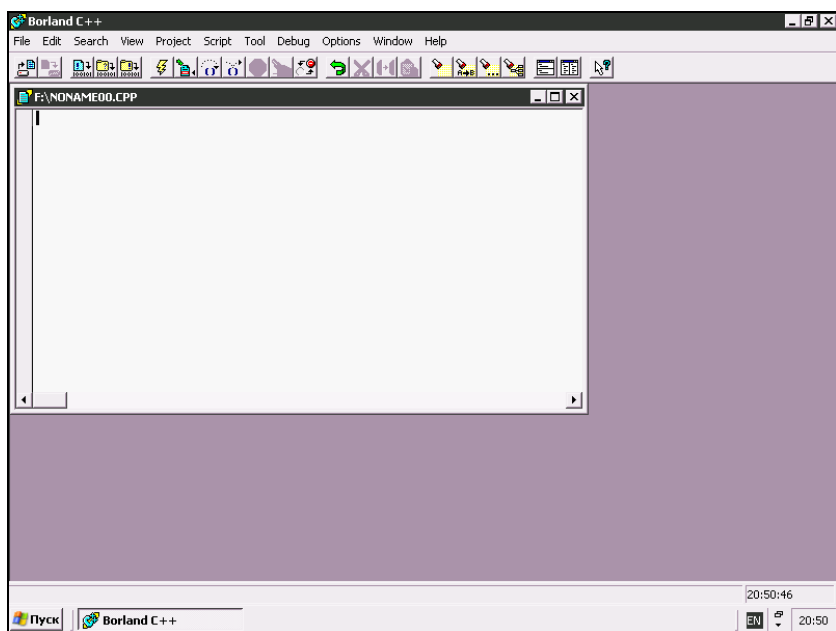


Рис. 1.3. Окно C++

В *строке состояния* программа отображает краткие подсказки, связанные с тем или иным объектом.

Приведем назначение некоторых пунктов меню:

- ☐ **File** (Файл) — это один из самых главных пунктов меню; он отвечает за открытие и сохранение проекта;

- ❑ **Edit** (Правка) — используется для копирования, вставки, вырезания информации, а также для отмены последних операций над текстом;
- ❑ **Search** (Поиск) — отвечает за поиск информации в тексте;
- ❑ **View** (Вид) — открывает окна, которые помогают при написании программы; подробно об этом пункте меню будет рассказано позже;
- ❑ **Project** (Проект) — предназначен для управления созданным проектом;
- ❑ **Options** (Опции) — этот пункт меню служит для настройки параметров программы и проекта;
- ❑ **Window** (Окно) — помогает расположить окна каскадом, а также переключаться между ними.

1.3. Создание и сохранение документа

Чтобы создать новый документ, выберите пункт меню **File | New | Text edit** (Файл | Создать | Текстовая область). В рабочей области появится окно — *редактор кода*, в котором и будет содержаться основной код программы (см. рис. 1.3).

Чтобы сохранить документ, выберите пункт меню **File | Save as...** (Файл | Сохранить как...). В результате откроется окно сохранения документа (рис. 1.4).

В этом окне укажите папку, в которой будет размещаться документ, и имя, под которым он будет храниться. Для подтверждения сохранения нажмите кнопку **Открыть**. Данная команда дублируется нажатием комбинации клавиш <Ctrl>+<K>+<S>.

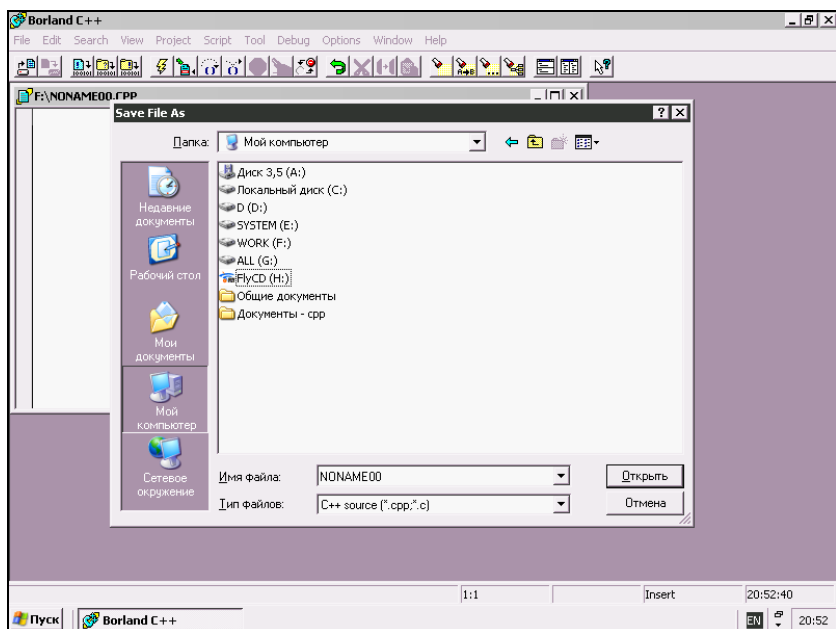


Рис. 1.4. Сохранение документа

1.4. Компиляция программы

После того как исходный текст набран (это можно сделать в любом текстовом редакторе, хотя существуют и специальные приложения), его необходимо преобразовать в программу, которая и будет выполняться на компьютере. Важно понять, что сам текст только формально описывает алгоритм вычислений — он не является программой.

Дело в том, что процессор понимает только двоичный код, представляющий собой очень простые машинные команды. В машинном коде написать более-менее сложную программу практически невозможно. В обычном языке программирования для операции над парой переменных достаточно одной команды, а процессору на это могут потребоваться десятки машинных команд.

В том, что машинные команды очень просты, а написать приличную программу с их помощью невероятно трудно, нет противоречия. Все детали механической части современного автомобиля совершают очень простые вращательные или поступательные движения, но невозможно создать автомобиль из куска металла с помощью молотка и напильника, нужны специальные инструментальные средства.

Точно так же дело обстоит и в вычислительной технике. Здесь тоже нужны специальные инструментальные средства. Перевод из текста в двоичный код осуществляется специальными программами, которые называются *трансляторами*. Они транслируют, т. е. переводят тексты, написанные на языке программирования, в машинный код.

В вычислительной технике существуют два класса трансляторов: интерпретаторы и компиляторы. *Интерпретатор* работает как синхронный переводчик — просматривает исходный текст строку за строкой, переводит каждую строку в промежуточный или в машинный код и передает его на исполнение. Если ошибок нет, интерпретатор приступает к следующей строке. *Компилятор* (а именно к этому классу относится рассматриваемая нами система программирования Borland C++) работает как литературный переводчик. Сначала он просматривает весь текст, иногда и не один раз, находит общие повторяющиеся места, тщательно готовит стратегию перевода, подбирает самые эффективные аналоги и только после этого переводит весь исходный текст целиком и полностью, создав при этом новый документ, который называется *объектным кодом*. Объектный код можно считать законченной программой, хотя и не вполне.

Итак, *компиляция* — это процесс преобразования исходной программы в исполняемую. Процесс компиляции состоит из двух этапов. На первом этапе выполняется проверка текста программы на отсутствие ошибок, на втором — генерируется исполняемая программа (exe-файл).

После ввода текста функции обработки события и сохранения проекта можно из меню **Project** (Проект) выбрать команду **Compile**

(Компиляция) или выполнить компиляцию, нажав клавишу <F9>. Диалоговое окно компилятора представлено на рис. 1.5.

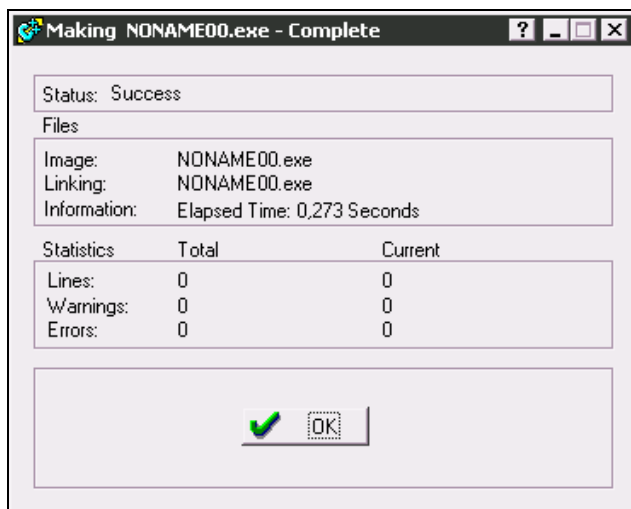


Рис. 1.5. Результаты компиляции в диалоговом окне

Если ошибок не найдено (**Errors** = 0), то можно запустить приложение, нажав комбинацию клавиш <Ctrl>+<F9>.

Компиляцию можно осуществлять и в пошаговом режиме с помощью клавиши <F7>. После ее нажатия строка, с которой начинается программа, будет выделена синим цветом. При повторном нажатии клавиши операции, находящиеся в этой строке, будут выполнены, и синим цветом подсветится следующая строка.

Запуск программы можно начать с любой желаемой строки, установив в ней курсор и нажав клавишу <F4>.

Глава 2



Основы программирования

В данной главе вы познакомитесь с основами программирования на C++, получите представление о программах и разработке алгоритмов для них.

2.1. Этапы разработки программы

Программа, которую выполняет компьютер, нередко отождествляется с самим компьютером, т. к. человек, использующий программу, вводит в компьютер исходные данные (например, при помощи клавиатуры), а компьютер выводит результат на экран, на принтер или в файл. Процессор преобразует исходные данные в результат по определенному алгоритму, который, будучи записан на специальном языке, называется *программой*. Таким образом, чтобы компьютер решил некоторую задачу, необходимо разработать последовательность команд, обеспечивающую решение данной задачи, или, как говорят, написать программу.

Выражение "написать программу" означает только один из этапов создания компьютерной программы, когда разработчик (программист) действительно пишет команды (инструкции) на бумаге или при помощи текстового редактора.

Программирование — это процесс создания (разработки) программы, который может быть представлен с помощью последовательности шагов:

1. Спецификация (определение, формулирование требований к программе).

2. Разработка алгоритма.
3. Кодирование (запись алгоритма на языке программирования).
4. Отладка.
5. Тестирование.

Спецификация

Спецификация — определение требований к программе — один из важнейших этапов, на котором подробно описывается исходная информация, формулируются требования к результату, поведение программы в особых случаях (например, при вводе неверных данных).

Разработка алгоритма

На этапе *разработки алгоритма* необходимо определить последовательность действий, которые надо выполнить для получения результата. Если задача может быть решена несколькими способами, то программист, используя некоторый критерий (например, скорость работы), выбирает наиболее подходящий алгоритм. Результатом этапа разработки алгоритма является его подробное словесное описание или блок-схема.

Кодирование

После того как определены требования к программе и составлен алгоритм решения, он записывается на выбранном языке программирования. В результате получается исходная программа.

Отладка

Отладка — это процесс поиска и устранения ошибок. Ошибки в программе разделяют на две группы: синтаксические (ошибки в тексте) и алгоритмические. Синтаксические ошибки устранить довольно легко, а алгоритмические ошибки обнаружить труднее. Этап отладки можно считать законченным, если программа правильно работает при любом правильном наборе входных данных.

Тестирование

Тестирование особенно важно, если вы предполагаете, что вашей программой будут пользоваться другие. На этом этапе следует проверить, как ведет себя программа при как можно большем количестве входных наборов данных, в том числе и заведомо неверных.

2.2. Элементы языка C++

Алфавит

Множество символов языка C++ включает прописные и строчные буквы латинского алфавита и цифры.

Внимание

C++ чувствителен к регистру. Это означает, что "a" и "A" — совсем разные буквы.

Язык программирования, как и любой другой язык, имеет свой словарь. *Словарь языка программирования* представляет собой совокупность зарезервированных или так называемых "ключевых" слов. Все зарезервированные слова содержат только строчные буквы (символы нижнего регистра) и написаны на английском языке.

Внимание

Пробел используется не только в качестве разделительного знака, но и как символьный знак.

Приведем список ключевых слов языка C++:

asm	char	delete	extern
auto	class	do	float
break	const	double	for
case	continue	else	friend
catch	default	enum	goto

if	protected	static	typedef
inline	public	struct	union
int	register	switch	unsigned
long	return	template	virtual
new	short	this	void
operator	signed	throw	volatile
private	sizeof	try	while

Примечание

Использование части ключевых слов будет объяснено в процессе изложения материала, остальные войдут в словарь программиста по мере изучения C++.

Кроме того, в C++ используются следующие символы:

- ❑ знаки препинания: . , : ;;
- ❑ скобки: () [] { };
- ❑ знаки: | ^ ? _;
- ❑ двойные и одинарные кавычки: " и '.

Алгоритм заключается в фигурные скобки {} после выражения `main()`, в котором круглые скобки показывают, что это программный блок, называемый *функцией*. Обычно программа состоит из нескольких функций, но все же `main()` присутствует всегда, с нее и начинается выполнение программы. Данная функция всегда оканчивается оператором `return 0;`, показывающим, что программа завершена. Вот пример простой программы:

```
main()
{
    return 0;
}
```

Внимание

Зарезервированные (ключевые) слова запрещается использовать в качестве имени переменных.