

А. П. Побегайло

С/С ++
ДЛЯ СТУДЕНТА

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.068+800.92С/С++

ББК 32.973.26-018.1

П41

Побегайло А. П.

П41 С/С++ для студента. — СПб.: БХВ-Петербург, 2006. — 528 с.: ил.

ISBN 5-94157-647-1

Подробно рассматриваются языки программирования С и С++. Описаны типы данных, функции, классы, шаблоны, а также библиотеки стандартных функций. Язык программирования С++ рассматривается как объектно-ориентированное расширение языка С, что позволяет последовательно изучить процедурное программирование, объектно-ориентированное программирование и обобщенное программирование. Изложение материала отличается краткостью и снабжено большим количеством простых примеров и листингов, которые поясняют технику программирования на языках С и С++.

Для студентов и программистов

УДК 681.3.068+800.92С/С++

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>Андрей Смышляев</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Игоря Цырульникова</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.05.06.

Формат 60×90^{1/16}. Печать офсетная. Усл. печ. л. 33.

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 5-94157-647-1

© Побегайло А. П., 2006

© Оформление, издательство "БХВ-Петербург", 2006

Оглавление

Введение	17
ЧАСТЬ I. ЯЗЫК ПРОГРАММИРОВАНИЯ C	21
Глава 1. Структура языка C.....	23
1.1. Элементы языка C.....	23
1.2. Символы.....	25
1.3. Ключевые слова	26
1.4. Идентификаторы.....	27
1.5. Константы.....	27
1.6. Инструкции.....	29
1.7. Комментарии	30
Глава 2. Встроенные типы данных и переменные	31
2.1. Базовые типы данных	31
2.2. Модификаторы типов	32
2.3. Спецификаторы типов	33
2.4. Переменные	35
2.5. Фундаментальные типы данных	36
2.6. Квалификаторы типов.....	37
Глава 3. Операторы и выражения	39
3.1. L-value и R-value.....	39
3.2. Арифметические операторы	40
3.3. Побитовые операторы	41
3.4. Операторы сравнения	43
3.5. Логические операторы.....	44

3.6. Выражения.....	45
3.7. Приведение типов в выражениях.....	47
3.8. Оператор преобразования типов.....	48
3.9. Оператор присваивания.....	49
3.10. Составные операторы присваивания.....	51
3.11. Операторы инкремента и декремента.....	52
3.12. Условный оператор.....	53
3.13. Оператор "запятая".....	55
3.14. Побочные эффекты.....	56
3.15. Точки последовательности.....	57

Глава 4. Управляющие инструкции..... 59

4.1. Инструкции выбора <i>if</i> и <i>if...else</i>	59
4.2. Инструкция выбора <i>switch</i>	60
4.3. Инструкции цикла <i>while</i> и <i>do...while</i>	62
4.4. Инструкция цикла <i>for</i>	63
4.5. Инструкция перехода <i>break</i>	64
4.6. Инструкция перехода <i>continue</i>	64
4.7. Метки инструкций и инструкция перехода <i>goto</i>	65

Глава 5. Указатели и массивы..... 67

5.1. Указатели.....	67
5.2. Преобразование типов указателей.....	68
5.3. Операторы определения адреса и обращения по адресу.....	69
5.4. Указатели на константы и константные указатели.....	70
5.5. Одномерные массивы.....	71
5.6. Строки.....	74
5.7. Арифметические действия с указателями.....	74
5.8. Многомерные массивы.....	75

Глава 6. Функции..... 78

6.1. Объявление функции.....	78
6.2. Определение функции.....	80
6.3. Стандартные функции.....	81
6.4. Вызов функции.....	82
6.5. Передача массивов в функции.....	84
6.6. Указатели на функции (функторы).....	86

Глава 7. Структура программы на языке C..... 89

7.1. Область видимости идентификатора.....	89
7.2. Время существования переменных и функций.....	90

7.3. Связывание идентификаторов.....	91
7.4. Спецификаторы классов памяти.....	92
7.5. Структура программы.....	96
7.6. Функция <i>main</i>	97

Глава 8. Типы данных, определяемые программистом 100

8.1. Введение	100
8.2. Перечисления	101
8.3. Структуры.....	103
8.4. Объединения.....	107
8.5. Битовые поля	110
8.6. Передача структур в функции.....	111
8.7. Объявление <i>typedef</i>	114
8.8. Оператор <i>sizeof</i>	116
8.9. Сравнимые типы	119

Глава 9. Директивы препроцессора..... 122

9.1. Препроцессор	122
9.2. Директивы <i>#define</i> и <i>#undef</i>	123
9.3. Директивы <i>#if</i> , <i>#elsif</i> , <i>#else</i> и <i>#endif</i>	124
9.4. Директивы <i>#ifdef</i> и <i>#ifndef</i>	126
9.5. Директива <i>#include</i>	127
9.6. Директивы <i>#error</i>	129
9.7. Директива <i>#line</i>	129
9.8. Директива <i>#pragma</i>	130
9.9. Пустая директива	130
9.10. Оператор <i>#</i>	130
9.11. Оператор <i>##</i>	131
9.12. Предопределенные макрокоманды.....	132

ЧАСТЬ II. ЯЗЫК ПРОГРАММИРОВАНИЯ C++ 133

Глава 10. Дополнения к типам данных языка C..... 135

10.1. Тип данных <i>bool</i>	135
10.2. Тип данных <i>wchar_t</i>	135
10.3. Анонимные объединения	137
10.4. Ссылки	138
10.5. Новые операторы явного преобразования типов	140
10.6. Типизированные операторы распределения памяти.....	143

Глава 11. Дополнение к функциям языка C	146
11.1. Передача аргументов по умолчанию.....	146
11.2. Встроенные функции	147
11.3. Передача аргументов и возврат значения через ссылки.....	148
11.4. Перегрузка функций	149
11.5. Искажение имен функций	153
Глава 12. Пространства имен.....	155
12.1. Определение пространства имен	155
12.2. Анонимные пространства имен	156
12.3. Стандартное пространство имен.....	157
12.4. Оператор разрешения области видимости.....	157
12.5. Объявление <i>using</i>	160
12.6. Директива <i>using</i>	161
12.7. Псевдонимы.....	162
12.8. Искажение имен переменных	163
Глава 13. Обработка исключений.....	165
13.1. Механизм обработки исключений.....	165
13.2. Обработка нескольких исключений	169
13.3. Перехват всех исключений.....	170
13.4. Вложенные исключения	171
13.5. Реализация исключений	174
13.6. Спецификация исключений.....	175
Глава 14. Классы	177
14.1. Введение	177
14.2. Определение класса	178
14.3. Объекты, доступ к членам класса	180
14.4. Указатель <i>this</i>	181
14.5. Спецификаторы доступа к членам класса.....	181
14.6. Друзья класса.....	183
14.7. Встроенные функции-члены класса	187
14.8. Функции-члены класса с квалификаторами <i>const</i> и <i>volatile</i>	188
14.9. Данные-члены класса с квалификатором <i>mutable</i>	190
14.10. Статические члены класса.....	190
14.11. Указатели на нестатические члены класса.....	192
14.12. Вложенные классы.....	194
14.13. Локальные классы.....	196

Глава 15. Конструкторы и деструкторы.....	197
15.1. Конструктор класса.....	197
15.2. Список инициализации.....	198
15.3. Конструктор по умолчанию.....	199
15.4. Конструктор копирования.....	200
15.5. Явный вызов конструкторов.....	202
15.6. Деструкторы.....	203
15.7. Исключения в деструкторах.....	205
Глава 16. Перегрузка операторов.....	206
16.1. Общие правила.....	206
16.2. Унарные операторы.....	209
16.3. Оператор присваивания.....	210
16.4. Составные операторы присваивания.....	212
16.5. Оператор индексирования [].....	213
16.6. Оператор вызова функции.....	214
16.7. Оператор доступа к членам класса.....	215
16.8. Операторы инкремента и декремента.....	216
16.9. Бинарные операторы.....	217
16.10. Перегрузка операторов >> и << для ввода и вывода.....	218
16.11. Операторы преобразования типов (конвертеры).....	219
16.12. Операторы <i>new</i> и <i>delete</i>	219
Глава 17. Наследование классов.....	222
17.1. Определение наследования.....	222
17.2. Доступ к членам, наследуемым от базового класса.....	225
17.3. Конструкторы, деструкторы и наследование.....	226
17.4. Наследование и присваивание.....	229
17.5. Виртуальные функции.....	230
17.6. Полиморфизм и позднее связывание.....	233
17.7. Передача аргументов по умолчанию в виртуальные функции.....	234
17.8. Виртуальные деструкторы.....	235
17.9. Абстрактные классы.....	236
17.10. Множественное наследование.....	238
17.11. Виртуальное наследование.....	239
Глава 18. Шаблоны функций.....	241
18.1. Определение шаблона функции.....	241
18.2. Конкретизация шаблона функции.....	243
18.3. Вывод аргументов шаблона функции.....	246

18.4. Явная специализация шаблона функции.....	249
18.5. Модели компиляции шаблонов.....	250
18.6. Перегрузка шаблонов функций.....	252
18.7. Шаблоны функций как члены класса.....	255

Глава 19. Шаблоны классов.....257

19.1. Определение шаблона класса.....	257
19.2. Конкретизация шаблона класса.....	259
19.3. Использование ключевого слова <i>typename</i>	262
19.4. Явная специализация шаблона класса.....	264
19.5. Частичная специализация шаблона класса.....	265
19.6. Статические члены шаблона класса.....	267
19.7. Шаблоны как члены шаблона класса.....	268
19.8. Объявление друзей в шаблоне класса.....	269
19.9. Шаблоны класса и наследование.....	273
19.10. Шаблоны класса как параметры шаблонов функций.....	274
19.11. Шаблоны класса как параметры шаблона класса.....	275

ЧАСТЬ III. СТАНДАРТНАЯ БИБЛИОТЕКА ЯЗЫКА ПРОГРАММИРОВАНИЯ C.....277

Глава 20. Стандартные определения <stddef.h>.....279

20.1. Типы данных.....	279
20.2. Макрос <i>NULL</i>	280
20.3. Макрос <i>offsetof</i>	280

Глава 21. Стандартные функции <stdlib.h>.....282

21.1. Динамическое распределение памяти.....	282
21.2. Динамические массивы.....	284
21.3. Стандартные функции сортировки и поиска.....	287
21.4. Взаимодействие с системой.....	290
21.5. Преобразования строк в числа.....	294
21.6. Арифметические функции.....	298
21.7. Генерация случайных чисел.....	299
21.8. Обработка длинных символов.....	301

Глава 22. Диагностика ошибок <assert.h>.....305

Глава 23. Функции с переменным количеством параметров <stdarg.h>.....307

Глава 24. Диапазоны целочисленных данных <limits.h>	311
Глава 25. Диапазоны чисел с плавающей точкой <float.h>.....	313
Глава 26. Обработка ошибок <errno.h>	316
Глава 27. Математические функции <math.h>.....	318
27.1. Обработка ошибок	318
27.2. Тригонометрические функции	319
27.3. Экспоненциальная и логарифмическая функции	320
27.4. Гиперболические функции	321
27.5. Степень и квадратный корень	321
27.6. Экспонента и мантисса	322
27.7. Целая и дробная часть числа с плавающей точкой	323
27.8. Целые границы числа с плавающей точкой.....	323
27.9. Остаток от деления	324
27.10. Абсолютное значение числа.....	324
Глава 28. Функции классификации символов <ctype.h>	325
28.1. Определение типа символа.....	325
28.2. Отображения прописных и строчных символов	326
Глава 29. Функции для работы со строками <string.h>.....	328
29.1. Сравнение строк	328
29.2. Копирование строк.....	329
29.3. Соединение строк.....	330
29.4. Поиск символа в строке.....	331
29.5. Поиск подстроки в строке	333
29.6. Разбор строки на лексемы	334
29.7. Определение длины строки	335
29.8. Функции для работы с памятью.....	336
29.9. Сравнение строк, содержащих локальные символы	337
29.10. Получение строки, описывающей код ошибки	339
Глава 30. Функции для работы с файлами <stdio.h>	341
30.1. Файлы и потоки	341
30.2. Открытие файла	343
30.3. Перенаправление потока	344
30.4. Закрытие файла	344

30.5. Работа с индикатором ошибки.....	344
30.6. Работа с индикатором конца файла.....	345
30.7. Работа с индикатором позиции файла.....	345
30.8. Блочный ввод/вывод.....	347
30.9. Ввод/вывод символов.....	350
30.10. Запись/чтение строк из файла.....	352
30.11. Форматированный вывод.....	354
30.12. Форматированный ввод.....	360
30.13. Форматирование и сканирование строк.....	364
30.14. Работа с буферами.....	366
30.15. Стандартные потоки.....	368
30.16. Удаление файла.....	369
30.17. Переименование файла.....	370
30.18. Работа с временными файлами.....	371

Глава 31. Организация нелокальных переходов

<setjmp.h>.....	373
31.1. Сохранение информации о состоянии стека.....	373
31.2. Нелокальный переход.....	374

Глава 32. Обработка исключительных ситуаций

<signal.h>.....	377
32.1. Сигналы.....	377
32.2. Установка функции обработки сигнала.....	378
32.3. Возбуждение сигнала.....	379

Глава 33. Поддержка локализации <locale.h>.....

33.1. Локальность.....	382
33.2. Локальные категории.....	383
33.3. Установка локальности.....	383
33.4. Форматы числовых и денежных данных.....	384
33.5. Определение текущих форматов данных.....	387

Глава 34. Работа с датами и временем <time.h>.....

34.1. Арифметические типы для представления времени.....	389
34.2. Структура для представления календарного времени.....	389
34.3. Определение времени работы программы.....	390
34.4. Определение текущего времени.....	391
34.5. Нахождение разности времен.....	392
34.6. Преобразование представления времени из числа в структуру.....	393

34.7. Преобразование представления времени из структуры в число ...	395
34.8. Преобразование представления времени из числа в строку.....	395
34.9. Преобразование представления времени из структуры в строку..	396
34.10. Форматирование представления времени в строке.....	397

ЧАСТЬ IV. СТАНДАРТНАЯ БИБЛИОТЕКА ЯЗЫКА ПРОГРАММИРОВАНИЯ C++401

Глава 35. Структура стандартной библиотеки C++403

Глава 36. Обработка исключений <exception>407

36.1. Классы и функции	407
36.2. Класс <i>exception</i>	408
36.3. Класс <i>bad_exception</i>	408
36.4. Аварийное завершение программы	409
36.5. Обработка не специфицированного исключения	410
36.6. Определение наличия выброшенного исключения.....	412

Глава 37. Классы стандартных исключений <stdexcept>414

Глава 38. Динамическая идентификация типов <typeinfo>.....418

38.1. Динамическая идентификация типов	418
38.2. Класс <i>type_info</i>	419
38.3. Оператор <i>typeid</i>	420
38.4. Обработка исключения <i>bad_typeid</i>	421
38.5. Обработка исключения <i>bad_cast</i>	423

Глава 39. Работа со строками в C++ <string>.....425

39.1. Характеристики символов.....	426
39.2. Строки.....	430
39.2.1. Ассоциированные типы	431
39.2.2. Специальное значение символьного типа.....	432
39.2.3. Конструкторы.....	432
39.2.4. Шаблоны конструкторов	434
39.2.5. Операторы присваивания.....	435
39.2.6. Функции, возвращающие итераторы.....	435
39.2.7. Получение ссылки на элемент строки.....	436
39.2.8. Оператор индексирования	437

39.2.9. Добавление символа в конец строки	437
39.2.10. Преобразование в строку языка C	437
39.2.11. Получение адреса первого элемента строки.....	438
39.2.12. Функции, работающие с длиной строки	438
39.2.13. Соединение строк	439
39.2.14. Присваивание строк.....	441
39.2.15. Вставка строк и символов в строку	442
39.2.16. Удаление последовательности символов из строки	443
39.2.17. Очистка строки.....	444
39.2.18. Замена символов и подстрок в строке.....	444
39.2.19. Копирование подстроки в массив символов	446
39.2.20. Обмен строк.....	447
39.2.21. Поиск символов и подстрок в строке	447
39.2.22. Выделение подстроки.....	449
39.2.23. Сравнение строк.....	450
39.2.24. Получение распределителя памяти для символа	451
39.3. Шаблоны операторов.....	451
39.3.1. Соединение строк	451
39.3.2. Сравнение строк.....	452
39.3.3. Ввод/вывод строк.....	453
39.4. Шаблоны функций	454
39.4.1. Ввод строк	454
39.4.2. Обмен строк.....	455
Глава 40. Поток ввода/вывода в C++	456
40.1. Структура стандартной библиотеки ввода/вывода	456
40.2. Базовые классы потоков <ios>	458
40.2.1. Класс <i>ios_base</i>	459
40.2.1.1. Флаги	460
40.2.1.2. Состояния	462
40.2.1.3. Режимы открытия потока.....	463
40.2.1.4. Направления поиска	463
40.2.1.5. События	464
40.2.1.6. Конструктор	466
40.2.1.7. Оператор присваивания.....	466
40.2.1.8. Класс <i>failer</i>	466
40.2.1.9. Класс <i>Init</i>	466
40.2.1.10. Управление разрешением дисплея	467
40.2.1.11. Управление шириной поля ввода/вывода	467
40.2.1.12. Управление памятью	467

40.2.1.13. Управление локальностью	468
40.2.1.14. Синхронизация с потоками языка программирования C	468
40.2.2. Шаблон класса <i>basic_ios</i>	469
40.2.2.1. Ассоциированные типы	469
40.2.2.2. Конструкторы	470
40.2.2.3. Деструктор	470
40.2.2.4. Инициализация объектов	471
40.2.2.5. Управление состоянием потока	472
40.2.2.6. Управление исключениями	472
40.2.2.7. Копирование состояния потока	473
40.2.2.8. Перегруженные операторы	473
40.2.2.9. Установка символа заполнителя	474
40.2.2.10. Управление буфером	474
40.2.2.11. Управление потоком вывода	474
40.2.2.12. Управление локальностью	475
40.2.3. Шаблон класса <i>fops</i>	475
40.2.4. Манипуляторы	477
40.3. Базовые потоки вывода <code><ostream></code>	479
40.3.1. Шаблон класса <i>basic_ostream</i>	479
40.3.1.1. Ассоциированные типы	480
40.3.1.2. Конструктор	480
40.3.1.3. Деструктор	480
40.3.1.4. Перегруженный оператор вывода <code><<</code>	480
40.3.1.5. Вывод символа	482
40.3.1.6. Вывод блока памяти	482
40.3.1.7. Освобождение буфера потока	482
40.3.1.8. Управление индикатором позиции	483
40.3.1.9. Класс <i>sentry</i>	484
40.3.2. Манипуляторы	485
40.3.3. Перегруженный шаблон оператора вывода <code><<</code>	485
40.4. Базовые потоки ввода и ввода/вывода <code><istream></code>	486
40.4.1. Шаблон класса <i>basic_istream</i>	487
40.4.1.1. Ассоциированные типы	488
40.4.1.2. Конструктор	488
40.4.1.3. Деструктор	488
40.4.1.4. Перегруженный оператор ввода <code>>></code>	488
40.4.1.5. Ввод символов	490
40.4.1.6. Ввод строки	492
40.4.1.7. Просмотр символа в потоке	493

40.4.1.8. Возврат символа в буфер потока ввода	494
40.4.1.9. Ввод блока памяти	496
40.4.1.10. Игнорирование символов	496
40.4.1.11. Определение количества прочитанных символов	497
40.4.1.12. Управление индикатором позиции	497
40.4.1.13. Синхронизация потока с файлом	499
40.4.1.14. Класс <i>sentry</i>	499
40.4.2. Манипулятор	499
40.4.3. Перегруженный шаблон оператора ввода >>	500
40.4.4. Шаблон класса <i>basic_iostream</i>	501
40.5. Стандартные потоки ввода/вывода <iostream>	502
40.6. Базовые потоки ввода/вывода в файлы <fstream>	503
40.6.1. Шаблон класса <i>basic_ofstream</i>	504
40.6.2. Шаблон класса <i>basic_ifstream</i>	505
40.6.3. Шаблон класса <i>basic_fstream</i>	506
40.6.4. Работа с текстовыми файлами	507
40.6.4.1. Создание текстового файла	508
40.6.4.2. Чтение текстового файла	508
40.6.4.3. Модификация текстового файла	509
40.6.5. Работа с бинарными файлами	509
40.6.5.1. Создание бинарного файла	510
40.6.5.2. Чтение бинарного файла	511
40.6.5.3. Модификация бинарного файла	512
40.6.6. Копирование файла	513
40.7. Манипуляторы <iomanip>	514
40.7.1. Простые манипуляторы	514
40.7.2. Параметризованные манипуляторы	515
Список литературы	517
Предметный указатель	518

Рецензия на книгу
"Языки программирования С и С++"
доцента кафедры технологии программирования,
факультета прикладной математики и информатики
Белорусского государственного университета,
к. т. н. Побегайло Александра Павловича

В рецензируемой книге дано довольно полное изложение языков программирования С и С++. Книга представляет собой учебное пособие по программированию на языках С и С++ для студентов высших учебных заведений, специальностей "Информатика" и "Прикладная математика". В книге полностью представлены конструкции языков программирования С и С++: типы данных, операторы и выражения, функции, классы, шаблоны. Поэтому можно считать, что рецензируемая книга представляет собой полное и замкнутое пособие по языкам программирования С и С++.

Книга "Языки программирования С и С++" может использоваться как начинающими, так и подготовленными программистами. А также может быть полезной профессиональным программистам.

Книга имеет несколько особенностей. Во-первых, изложение отличается краткостью и простотой. Во-вторых, все вопросы рассматриваются подробно и полностью. Каждое положение снабжено простым примером, что позволяет студентам быстро перейти к практическому программированию. В-третьих, процедурно-ориентированная часть языка программирования С++ излагается на базе и в сравнении с языком программирования С. Это позволяет построить курс программирования следующим образом: процедурное программирование, объектно-ориентированное программирование, обобщенное программирование.

Считаю, что книга "Языки программирования С и С++" может быть рекомендована к изданию и использоваться в учебном процессе.

Зам. генерального директора
Объединенного института проблем информатики НАН
Республики Беларусь
доктор физико-математических наук, профессор

Тузиков А. В.

Рецензия на книгу
"Языки программирования С и С++"
доцента кафедры технологии программирования,
факультета прикладной математики и информатики
Белорусского государственного университета,
к. т. н. Побегайло Александра Павловича

В рецензируемой книге рассмотрены актуальные в настоящее время языки программирования С и С++. Изложение материала отличается простотой и логической стройностью, что позволяет изучать процедурное и объектно-ориентированное программирование последовательно, обращая внимание на расширение языка программирования С конструкциями для объектно-ориентированного программирования.

Книга "Языки программирования С и С++" состоит из четырех частей:

Часть I. Язык программирования С.

Часть II. Язык программирования С++.

Часть III. Стандартная библиотека языка программирования С.

Часть IV. Стандартная библиотека языка программирования С++.

Материалы, представленные в частях I и II книги, соответствуют программе курса "Программирование" для студентов специальностей "Информатика" и "Прикладная математика" университетов и поэтому могут использоваться в качестве основы курса лекций по данному предмету. Части III и IV позволяют использовать книгу в качестве справочного пособия по программированию, так как содержат большое количество справочного материала по стандартным библиотекам языков программирования С и С++.

Рекомендую книгу "Языки программирования С и С++" к печати и использованию в учебном процессе.

Зав. кафедрой технологии программирования
факультета прикладной математики и информатики
Белорусского государственного университета
доктор технических наук, профессор

Курбацкий А. Н.

Введение

Язык программирования С был разработан в период с 1969 по 1973 годы. Прежде чем переходить к изучению языка программирования С, скажем немного о целях, которые преследовали создатели этого языка, американские программисты Кен Томпсон и Деннис Ритчи. Язык С разрабатывался как язык системного программирования и предназначался для кодирования операционной системы Unix, которая была бы переносима на различные аппаратные платформы. Поэтому в языке программирования С так много возможностей для программирования на низком уровне. Вследствие этого язык программирования С часто также рассматривают как язык ассемблера высокого уровня. Следует отметить, что язык С получился очень мощным и компактным. С годами стало понятно, что разработка этого языка явилась выдающимся достижением в области информационных технологий.

В 1999 году разработчики этого языка были удостоены Национальной медали США за достижения в области технологии.

Язык программирования С++ был разработан с 1980 по 1983 годы Бьерном Страуструпом, сотрудником AT&T Bell Laboratories. Цель разработки этого языка программирования состояла в добавлении в язык программирования С конструкций для объектно-ориентированного программирования, а именно — классов. В 1986 году Бьерн Страуструп опубликовал книгу "Язык программирования С++", которая стала классическим руководством по программированию на языке С++ и переиздавалась не-

сколько раз. Первая спецификация языка программирования C++ была опубликована Бьерном Страуструпом и Маргарет Еллис в 1990 году и называлась "Annotated C++ Reference Manual" или сокращенно ARM. В дальнейшем эта спецификация стала основой для стандартизации языка программирования C++. Международный стандарт языка C++ был утвержден в 1998 году. По этому стандарту язык программирования C++ базируется на стандарте языка программирования C, утвержденном в 1990 году. Эти версии языков программирования C и C++ и рассматриваются в данном пособии.

Настоящая книга представляет собой курс по изучению языков программирования C и C++ для студентов университетов. Содержание и форма изложения курса обусловлены временем, отводимым программами университетов на изучение данного курса.

Сначала кратко коснемся содержания курса. Первые две части книги содержат описание языков программирования C и C++. Предполагается, что вопросы, изложенные в этих частях, рассматриваются и обсуждаются на лекциях. Третья и четвертая части разбираются студентами самостоятельно при выполнении лабораторных работ, т. к. на изучение стандартных библиотек языков программирования C и C++ времени в программе не хватает.

Теперь перейдем к форме изложения курса. Я считаю, что на лекциях нужно излагать концепции и технику программирования, а не решать алгоритмические задачи. Для этого предназначены другие курсы. Поэтому форма изложения книги базировалась на стандартах языков программирования C и C++, при этом каждое положение сопровождается максимально простым и понятным работающим примером. Вопросы, касающиеся практического программирования, как, например, работа со структурами, обработка строк, организация файлов, рассматриваются на семинарах и при выполнении лабораторных работ.

Особенно отмечу, что курс разбивается на две части: процедурное программирование и объектно-ориентированное программирование, которые условно совпадают с первыми двумя частями книги. В последнее время можно часто слышать, что изучать про-

граммирование надо сразу с объектно-ориентированных концепций. Мой опыт показывает, что это примерно то же самое, что начинать изучение математики с теории множеств и математической логики. Пока студент четко не усвоил технику и принципы процедурного программирования, он не сможет даже понять, почему класс имеет такой интерфейс, а не другой, не говоря уже о грамотной реализации этого интерфейса. Поэтому сначала нужно обратить внимание на основы и технику программирования, особенно при работе с такими языками программирования, как С и С++, и только затем переходить к объектно-ориентированному программированию.

Представленный материал предназначен для первого ознакомления с предметом, после усвоения изложенного материала можно переходить к более продвинутым пособиям, которые приведены в списке литературы. Изложение материала краткое, но довольно полное. Все приведенные в книге программы были проверены на работоспособность компилятором Microsoft Visual C++ 7.0 (.NET).



ЧАСТЬ I

Язык программирования C

- Глава 1. Структура языка C
- Глава 2. Встроенные типы данных и переменные
- Глава 3. Операторы и выражения
- Глава 4. Управляющие инструкции
- Глава 5. Указатели и массивы
- Глава 6. Функции
- Глава 7. Структура программы на языке C
- Глава 8. Типы данных, определяемые программистом
- Глава 9. Директивы препроцессора

ГЛАВА 1



Структура языка C

1.1. Элементы языка C

Как и любой другой, язык программирования C включает следующие элементы: символы, слова и предложения. *Символы* — это базовые графические элементы языка, из которых строятся слова. *Слово* — это последовательность символов, которая имеет смысл в данном языке. Слова, используемые в языке программирования C, разбиваются на три группы: ключевые слова, идентификаторы и константы. Подробнее об этих группах слов будет рассказано в следующих разделах этой главы. Из слов и символов формируются предложения, которые в языке программирования C называются инструкциями. То есть, *инструкция* — это последовательность слов и символов языка программирования C, которая имеет смысл в этом языке. Последовательность инструкций языка C является *программой* на языке C. Программа на языке C преобразуется компилятором в последовательность машинных команд, которые исполняются микропроцессором. Схематически алгоритм работы компилятора может быть представлен блок-схемой, которая показана на рис. 1.1.

Вообще, *компилятор* — это такая программа, входными данными для которой является программа, написанная на определенном языке программирования, а выходными данными — программа в кодах команд микропроцессора, которая функционально соответствует входной программе. Другими словами можно сказать, что

компилятор переводит программу с некоторого языка программирования высокого уровня на машинный код. В соответствии с приведенной схемой процесс компиляции исходной программы можно разбить на следующие этапы:

- лексический анализ;
- синтаксический анализ;
- генерацию кода.

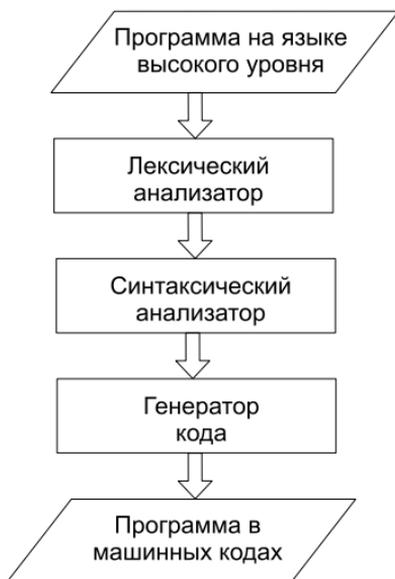


Рис. 1.1. Структурная схема работы компилятора

На этапе лексического анализа компилятор выделяет те элементы языка, из которых строятся предложения (к таким элементам относятся слова и символы). При этом символы должны иметь самостоятельное назначение и использоваться для специальных целей. К таким символам, например, относятся символы для обозначения знаков препинания и операторов. Элементы языка C, которые распознаются на этапе лексического анализа, называются *лексемами* (или *токенами*) (lexical elements или tokens). При этом компилятор проверяет правильность написания слов языка C. То есть определяется, состоят ли слова из допустимых символов языка и правильно ли эти символы используются.

На этапе синтаксического анализа компилятор проверяет, правильно ли составлены инструкции из лексем. А на этапе генерации кода в соответствие каждой инструкции ставится последовательность кодов команд микропроцессора. Если компилятор может выполнять оптимизацию кода, то генератор кода имеет более сложную структуру. В этом случае отображение инструкций языка программирования в коды команд микропроцессора не такое прямолинейное.

1.2. Символы

Каждый язык допускает только определенный набор символов. Не является исключением и язык программирования C, в котором могут использоваться только следующие символы:

- строчные латинские буквы: a ... z;
- прописные латинские буквы: A ... Z;
- цифры: 0 1 2 3 4 5 6 7 8 9;
- специальные символы: . , : ; ' " # { } [] () < > & | ^ !
_ + - * / \ % = ? ~ ;
- символ "пробел";
- нулевой символ или "пусто" (NULL).

Некоторые из символов могут обозначаться специальным образом:

- \? — знак вопроса;
- \' — апостроф;
- \" — кавычки;
- \\ — обратная косая черта;
- \0 — пусто.

Кроме того, каждый символ может быть обозначен своим кодом в восьмеричной или шестнадцатеричной системе счисления:

- \ddd — где буквы ddd обозначают код символа в восьмеричной системе счисления;

- `\xdd` — где буквы `dd` обозначают код символа в шестнадцатеричной системе счисления.

Каждый из символов языка C имеет свое назначение. Буквы и цифры используются главным образом для написания идентификаторов и литералов. Как следует из самого названия специальных символов, они имеют специальное назначение и используются главным образом для обозначения операторов.

Дополнительно в языке программирования C используются *управляющие символы*. Это такие символы, при вставке которых в текст происходит некоторое действие. К ним относятся следующие символы, которые обозначаются специальным образом:

- `\a` — сигнал тревоги;
- `\b` — возврат на шаг;
- `\f` — переход на следующую страницу;
- `\n` — переход на следующую строку;
- `\r` — переход на первую позицию текущей строки;
- `\t` — горизонтальная табуляция;
- `\v` — вертикальная табуляция.

В заключение этого раздела отметим символы, которые используются для разделения слов языка C, такие символы называются *символами разделителями* или *пробельными символами*. К этим символам относятся: пробел, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`.

1.3. Ключевые слова

Ключевые слова языка программирования C это такие слова, которые имеют predetermined назначение в этом языке и не могут использоваться для других целей. Ниже перечислены все ключевые слова языка C:

<code>auto</code>	<code>double</code>	<code>int</code>	<code>struct</code>
<code>break</code>	<code>else</code>	<code>long</code>	<code>switch</code>
<code>case</code>	<code>enum</code>	<code>register</code>	<code>typedef</code>
<code>char</code>	<code>extern</code>	<code>return</code>	<code>union</code>

const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Иногда ключевые слова также называют *зарезервированными словами* языка программирования.

1.4. Идентификаторы

Идентификатор — это такое слово языка C, которое может использоваться для обозначения имени переменной, имени функции, имени типа или метки инструкции. Идентификаторы могут включать только алфавитно-цифровые символы языка программирования C, а также символ подчеркивания "_". Кроме того, при написании идентификаторов должны учитываться следующие правила:

- идентификатор должен отличаться от ключевых слов языка C;
- идентификатор не должен начинаться с цифры;
- допускается любая длина идентификатора, но компилятор различает только первые 31 символ;
- в идентификаторе прописные и строчные буквы считаются различными.

Кроме того, не рекомендуется использовать символ "_" в качестве первого символа идентификатора, т. к. этот символ часто используется для именования системных переменных и функций.

В заключение этого раздела скажем, что стиль программирования языка C предлагает для имен переменных, функций и типов использовать строчные буквы, а для имен макросов — прописные буквы.

1.5. Константы

Константами или *литералами* называются некоторые фиксированные значения данных, т. е. такие значения, которые не могут изменяться.

В языке программирования C различаются четыре типа констант:

- целые константы;
- плавающие константы;
- символьные константы;
- строковые константы.

Целая константа может быть записана в десятичной, восьмеричной или шестнадцатеричной системе счисления. В десятичной системе целая константа записывается как обычное десятичное число, при условии, что первая цифра не является нулем. Например, следующие числа являются целыми десятичными целыми константами:

12, 234, 1009

В восьмеричной системе счисления целая константа записывается восьмеричными цифрами и должна начинаться с нуля. Примерами восьмеричных констант являются следующие числа:

012, 0234, 01007

В шестнадцатеричной системе счисления целая константа записывается шестнадцатеричными цифрами и должна начинаться с символов 0x или 0X. При этом для обозначения шестнадцатеричных цифр от 10 до 15 могут использоваться как строчные буквы a, b, c, d, e, так и прописные буквы A, B, C, D, E. Например, следующие целые числа являются целыми шестнадцатеричными константами:

0x12, 0X120xABC, 0Xавс

Кроме того, в языке программирования C разрешается объявление длинных целых констант, под которые компилятор отводит в два раза больше памяти, чем под целые константы. Для этой цели в конце целой константы ставится буква l или L. При этом заметим, что если заданное значение целой константы превышает диапазон целого типа данных, то она автоматически представляется длинной целой константой.

Константа с плавающей точкой представляет некоторое действительное число и имеет следующий вид:

[целая часть] . [дробная часть] [E|e [+|-] экспонента]

где целая часть, дробная часть и экспонента записываются при помощи десятичных цифр. В определении константы с плавающей точкой должна присутствовать, по крайней мере, одна из частей, заключенных во внешние квадратные скобки. Чтобы получить действительное число, которое представляется константой с плавающей точкой, необходимо целую и дробную часть этой константы умножить на десять в степени, которая задается экспонентой этой константы. Ниже приведены примеры констант с плавающей точкой:

```
3., .14      3.14, 0.314e1, 314e-2.
```

Символьная константа состоит из одного символа, который заключается в апострофы. Ниже приведены примеры символьных констант:

```
'c', 'y', '5', '\101'
```

Сам символ апостроф, используемый в качестве символьной константы, нужно обозначать как `\'`. Отметим, что символьные константы могут содержать символы, не входящие в язык программирования C, например, русские буквы.

Строковая константа представляет собой последовательность символов, заключенную в кавычки. По стандарту длина строковой константы не может превышать 509 символов. Ниже приведены примеры строковых констант.

```
"This is a string.", "Это строка.", "a", "1"
```

Сам символ кавычки, используемый в строковой константе, нужно обозначать как `\"`. Отметим, что строковые константы также могут включать символы, не принадлежащие языку программирования C. Кроме того, в конце каждой строковой константы компилятор помещает нулевой символ `\0`, который отмечает конец строки. В языке программирования C строковые константы обычно называются *строковыми литералами*.

1.6. Инструкции

Инструкцией называется любое синтаксически правильно составленное предложение языка программирования C. Инструкция должна заканчиваться символом `;`. Инструкции описывают неко-

торые действия, которые должна выполнять программа. В языке программирования C допускается пустая инструкция, которая состоит только из символа ; и не выполняет никаких действий.

Любое количество инструкций, заключенное в фигурные скобки { и }, называется *составной инструкцией* или *блоком*. Особенно отметим, что после блока точку с запятой ставить не нужно.

1.7. Комментарии

Комментарий — это предложение на естественном языке, которое поясняет ход выполнения программы. Компилятор игнорирует комментарии. В языке программирования C комментарии начинаются парой символов /* и заканчиваются парой символов */. Комментарии могут содержать символы, не принадлежащие языку программирования C. Комментарии разрешается применять везде, где используются пробелы. Например, комментарий может выглядеть следующим образом:

```
/* подсчет количества вариантов */
```

Замечание

В языке программирования C++ комментарий может начинаться символами //. В этом случае весь текст до конца строки считается комментарием.

Например, в языке программирования C++ этот же комментарий можно написать следующим образом:

```
// подсчет количества вариантов
```

В заключение этого раздела сделаем следующее замечание. Так как программы могут использоваться десятилетиями, то поддержка и модификация программного обеспечения может обходиться значительно дороже, чем его разработка. Поэтому считается, что программа, не содержащая комментариев, ничего не стоит и не может быть принята в эксплуатацию.

ГЛАВА 2



Встроенные типы данных и переменные

2.1. Базовые типы данных

Тип данных определяется как множество значений и множество операций, допустимых над этими значениями. В табл. 2.1 приведены *базовые типы данных*, предопределенные в языке программирования C.

Таблица 2.1. Базовые типы данных

Тип данных	Длина в байтах	Диапазон значений
char	1	0...255 или -128 ... 127
int	Не менее 2	Зависит от длины
float	4	-3.4E-38 ... 3.4E+38
double	8	-1.8E-308 ... 1.8E+308

В стандарте языка C длина в байтах и, соответственно, диапазон типа данных `int` не определены точно и зависят от системы, на которой реализован компилятор. Под системой понимается микропроцессор и операционная система, на которой функционирует компилятор. Однако предполагается, что этот тип данных, по крайней мере, содержит диапазон целых чисел от -32768 до 32767 .

Диапазоны типов данных `float` и `double` приведены с округлением до первого знака после точки. Точные границы этих диапазонов указаны в заголовочном файле `float.h`.