

Содержание

Предисловие	15
Часть I. Императивное программирование	17
Глава 1. Элементы программ	19
1.1. Алфавит языка	19
1.2. Ключевые слова	21
1.3. Идентификаторы	22
1.4. Литералы	24
1.5. Операторы и знаки пунктуации	26
Глава 2. Структура программы	27
2.1. Главный модуль	27
2.2. Объявление и определение	28
2.3. Функции	31
2.4. Заголовочные файлы	35
2.5. Директивы препроцессора	36
2.6. Комментарии	38
2.7. Фазы компиляции	39
Глава 3. Переменные	41
3.1. Атрибуты переменных	41
3.2. Основные типы данных	45
3.2.1. Символы	45
3.2.2. Логические переменные	46
3.2.3. Целые числа	46
3.2.4. Числа с плавающей точкой	48
3.3. Спецификаторы хранения	49
3.3.1. Статические переменные	49
3.3.2. Автоматические переменные	51
3.3.3. Динамические переменные	52
3.3.4. Внешние переменные	53
3.3.5. Регистровые переменные	54
3.3.6. Пространства имен	55
3.4. Квалификаторы доступа	60
3.4.1. Константы	60
3.4.2. Изменчивые сущности	61
3.5. Приведение типов	62
3.5.1. Приведение типа в стиле языка C	62
3.5.2. Оператор <code>static_cast</code>	63
3.5.3. Оператор <code>const_cast</code>	64
3.5.4. Оператор <code>dynamic_cast</code>	65
3.5.5. Оператор <code>reinterpret_cast</code>	65
3.6. Указатели и ссылки	66
3.6.1. Указатели	66
3.6.2. Ссылки	73
Глава 4. Сложные типы	75
4.1. Операторы <code>typedef</code> и <code>sizeof</code>	75
4.1.1. Оператор <code>typedef</code>	75

4.1.2. Оператор sizeof	77
4.2. Перечисления	78
4.3. Массивы	80
4.3.1. Одномерные массивы	81
4.3.2. Строки	82
4.3.3. Двухмерные массивы	84
4.3.4. Свободные массивы	85
4.4. Структуры POD	87
4.4.1. Операции над структурами	88
4.4.2. Массивы и структуры	90
4.4.3. Вложенные структуры	90
4.4.4. Указатели на структуры	91
4.5. Битовые поля	93
4.6. Объединения	94
Глава 5. Операторы	97
5.1. Арифметические операторы	97
5.1.1. Приоритеты арифметических операторов	104
5.1.2. Правила ассоциативности	104
5.1.3. Скобки	106
5.1.4. Преобразование типов	106
5.2. Операторы сравнения	111
5.3. Логические операторы	112
5.4. Побитовые операторы	114
5.5. Операторы присваивания	118
Глава 6. Управляющие конструкции	121
6.1. Блок	121
6.2. Оператор последовательного вычисления	122
6.3. Оператор if	123
6.4. Оператор else	125
6.5. Тернарная альтернатива	125
6.6. Оператор switch	126
6.7. Операторы цикла	129
6.7.1. Цикл со счетчиком for	129
6.7.2. Цикл while	132
6.7.3. Цикл do-while	133
6.8. Операторы перехода	134
6.8.1. Оператор return	134
6.8.2. Оператор break	134
6.8.3. Оператор continue	135
6.8.4. Оператор goto	135
6.8.5. Функция exit()	138
6.8.6. Функция abort()	139
6.8.7. Функция atexit()	139
Глава 7. Функции	141
7.1. Прототипы и определения	141
7.1.1. Основные понятия	141
7.1.2. Функция main()	143
7.2. Передача аргументов по значению	144
7.3. Передача аргументов по ссылке	145
7.3.1. Указатель как параметр функции	145
7.3.2. Ссылка как параметр функции	146
7.3.3. Одномерный массив как параметр функции	146
7.3.4. Многомерный массив как параметр функции	147

7.4. Возвращаемое значение	148
7.4.1. Возврат переменных простых типов	148
7.4.2. Возврат переменных сложных типов	149
7.5. Вызов функции	150
7.5.1. Механизм активации	151
7.5.2. Вызов функции с помощью указателей	152
7.6. Многоточие	154
7.7. Параметры, заданные по умолчанию	156
7.8. Перегрузка функций	156
7.9. Подставляемые функции	159
7.9.1. Макросы	159
7.9.2. Подставляемые функции	160
Глава 8. Средства ввода-вывода	161
8.1. Стандартные потоки	161
8.2. Ввод-вывод символов	162
8.3. Ввод-вывод строк	163
8.4. Форматный вывод	165
8.5. Форматный ввод	172
8.6. Чтение и запись текстовых файлов	177
8.7. Чтение и запись бинарных файлов	184
8.7.1. Последовательный доступ	184
8.7.2. Произвольный доступ	185
Глава 9. Ассемблер	187
9.1. Анатомия языковых конструкций	187
9.1.1. Простейшая программа	188
9.1.2. Функция без параметров и возвращаемых значений	190
9.1.3. Полноценная функция	191
9.1.4. Арифметические операции	194
9.1.5. Операции сравнения	196
9.1.6. Логические операции	197
9.1.7. Управляющие конструкции	199
9.2. Оператор <code>asm</code>	202
9.3. Программные прерывания	205
Глава 10. Оптимизация	207
10.1. Оптимизация по скорости	207
10.1.1. Выбор алгоритма	208
10.1.2. Ясность и простота кода	208
10.1.3. Скорость выполнения операций	209
10.1.4. Упрощение выражений	209
10.1.5. Макросы	210
10.1.6. Развертывание циклов	211
10.1.7. Объединение циклов	213
10.1.8. Инверсия цикла	214
10.1.9. Замена выражений	214
10.1.10. Инварианты циклов	215
10.1.11. Исключение общих подвыражений	215
10.1.12. Префиксная и постфиксная форма инкрементации	217
10.2. Оптимизация работы с памятью	217
10.2.1. Доступ к элементам многомерных массивов	217
10.2.2. Копирование больших массивов	218
10.2.3. Использование объединений	219
10.2.4. Уплотнение структур	220
10.2.5. Алгоритмические трюки	222

Часть II. Объектно-ориентированное программирование	225
Глава 11. Классы	227
11.1. Члены класса и управление доступом	229
11.1.1. Структуры и объединения как разновидности классов	231
11.1.2. Подставляемые функции-члены класса	232
11.1.3. Определение функций-членов вне класса	233
11.1.4. Указатель this	233
11.1.5. Статические переменные-члены класса	235
11.1.6. Статические функции-члены класса	236
11.1.7. Константные функции-члены	237
11.1.8. Вложенные классы	238
11.1.9. Указатели на члены класса	239
11.2. Конструкторы и деструктор	242
11.2.1. Конструкторы с одним параметром	243
11.2.2. Список инициализации	243
11.2.3. Конструктор по умолчанию	245
11.2.4. Конструктор копирования	245
11.2.5. “Именованные конструкторы”	247
11.2.6. Деструктор	248
11.2.7. Передача аргументов и возврат значений	250
11.3. Дружественные функции и классы	252
Глава 12. Перегрузка операторов	257
12.1. Операторные функции	257
12.2. Перегрузка унарных операторов с помощью функций-членов	258
12.2.1. Унарные операторы “плюс” и “минус”	258
12.2.2. Операторы инкремента и декремента	259
12.2.3. Унарные операторы !, & и ~	261
12.2.4. Перегрузка оператора ->	262
12.3. Перегрузка бинарных операторов с помощью функций-членов	263
12.3.1. Перегрузка арифметических операторов	265
12.3.2. Перегрузка оператора присваивания	266
12.3.3. Перегрузка сокращенных операторов присваивания	268
12.3.4. Перегрузка оператора последовательного вычисления	269
12.3.5. Перегрузка операторов new и delete	270
12.3.6. Перегрузка операторов new[] и delete[]	272
12.3.7. Перегрузка оператора []	272
12.3.8. Перегрузка оператора ()	274
12.4. Перегрузка унарных операторов с помощью дружественных функций	275
12.4.1. Перегрузка унарного минуса	275
12.4.2. Перегрузка операторов инкремента и декремента	276
12.5. Перегрузка бинарных операторов с помощью дружественных функций	278
12.6. Преобразование типов	279
Глава 13. Наследование	281
13.1. Производные классы	282
13.1.1. Открытое наследование	282
13.1.2. Закрытое наследование	285
13.1.3. Защищенное наследование	285
13.1.4. Множественное наследование	286
13.1.5. Конструкторы и деструкторы	288
13.1.6. Виртуальные базовые классы	290
13.2. Динамический полиморфизм	293

13.2.1. Виртуальные функции-члены	294
13.2.2. Чисто виртуальные функции и абстрактные классы	297
13.2.3. Механизм виртуальных функций	299
13.2.4. Виртуальный деструктор	300
13.2.5. Приемы программирования	302
13.3. Информация о типе на этапе выполнения	304
13.3.1. Оператор <code>dynamic_cast</code>	304
13.3.2. Оператор <code>typeid</code>	307
Часть III. Обобщенное программирование	309
Глава 14. Исключительные ситуации	311
14.1. Механизм обработки исключительных ситуаций	311
14.1.1. Обработка исключительных ситуаций	312
14.1.2. Перехват классов исключительных ситуаций	318
14.1.3. Иерархия исключительных ситуаций	319
14.2. Тонкости обработки исключительных ситуаций	323
14.2.1. Тотальный перехват исключительных ситуаций	323
14.2.2. Объявление исключительных ситуаций	326
14.2.3. Повторные исключительные ситуации	330
14.2.4. Непредвиденные исключительные ситуации	332
14.3. Стандартные исключительные ситуации	335
14.3.1. Класс <code>bad_alloc</code>	336
14.3.2. Класс <code>bad_cast</code>	337
14.3.3. Класс <code>bad_typeid</code>	338
14.3.4. Класс <code>bad_exception</code>	338
Глава 15. Шаблоны	341
15.1. Шаблонные функции	341
15.1.1. Явная специализация шаблонной функции	345
15.1.2. Выведение шаблонных аргументов	347
15.1.3. Перегрузка шаблонной функции	348
15.1.4. Шаблонная функция-член класса	349
15.2. Шаблонные классы	350
15.2.1. Объявление объекта шаблонного класса	350
15.2.2. Шаблонные функции-члены шаблонного класса	353
15.2.3. Шаблонные аргументы, заданные по умолчанию	355
15.2.4. Статические члены классов	356
15.2.5. Явная специализация класса	357
15.2.6. Шаблоны в качестве шаблонных параметров	358
15.2.7. Иерархия шаблонов	359
15.2.8. Распознавание типов	361
15.2.9. Виртуальные функции-члены шаблонных классов	361
15.2.10. Частичная специализация	362
Глава 16. Шаблонные идиомы	365
16.1. Интеллектуальные указатели	365
16.1.1. Тип членов класса	366
16.1.2. Применение интеллектуальных указателей	367
16.1.3. Подсчет ссылок	371
16.2. Характеристики и стратегии	372
16.2.1. Настройка интеллектуальных указателей	372
16.2.2. Настройка алгоритмов с помощью характеристик	374
16.2.3. Настройка алгоритмов с помощью стратегий	377
16.3. Шаблоны проектирования	378
16.3.1. Идиома Singleton	379

16.3.2. Идиома Adapter	380
16.3.3. Идиома Template Method	380
Глава 17. Шаблонное метапрограммирование	383
17.1. Статические метапрограммы	383
17.1.1. Перечислимый тип	386
17.1.2. Рекурсия	386
17.2. Основные управляющие конструкции	387
17.2.1. Оператор is-else	387
17.2.2. Оператор switch	387
17.2.3. Цикл do	388
17.2.4. Цикл while	389
17.2.5. Цикл for	389
17.3. Статические метафункции	390
17.3.1. Элементарные метафункции	390
17.3.2. Вычисление нормы вектора	392
17.3.3. Сравнение типов	393
17.3.4. Статическая диспетчеризация	395
17.3.5. Метафункции высшего порядка	395
17.3.6. Дискриминация типов	397
Часть IV. Стандартная библиотека шаблонов	403
Глава 18. Итераторы	405
18.1. Введение	405
18.1.1. Операторы	406
18.1.2. Пара	406
18.2. Концепция итераторов	408
18.3. Итераторы ввода	410
18.4. Итераторы вывода	412
18.5. Однонаправленные итераторы	413
18.6. Двухнаправленные итераторы	414
18.7. Итераторы произвольного доступа	416
18.8. Свойства итераторов	417
18.9. Дескрипторы итераторов	418
18.10. Операции с итераторами	420
Глава 19. Функторы	423
19.1. Базовые классы	423
19.2. Арифметические операции	425
19.3. Предикаты	427
19.4. Логические операции	428
19.5. Адаптеры	430
19.5.1. Связыватели	430
19.5.2. Инверторы	431
19.5.3. Адаптеры указателей на функцию	432
19.5.4. Адаптеры указателей на функции-члены класса	433
19.5.5. Адаптеры ссылок на функции-члены класса	434
19.6. Распределители	435
Глава 20. Контейнеры	437
20.1. Общая структура контейнеров	437
20.2. Последовательности	439
20.2.1. Вектор	440
20.2.2. Список	446
20.2.3. Двусторонняя очередь	453

20.3. Адаптеры последовательностей	459
20.3.1. Стек	459
20.3.2. Очередь	461
20.3.3. Очередь с приоритетом	462
20.4. Ассоциативные контейнеры	465
20.4.1. Множество	466
20.4.2. Мультимножество	472
20.4.3. Ассоциативный массив	478
20.4.4. Ассоциативный мультимассив	484
20.5. Битовое множество	490
20.5.1. Конструкторы и деструктор	490
20.5.2. Логические операции	491
20.5.3. Операции преобразования	492
20.5.4. Операции поиска и определения размера	492
20.5.5. Операции сравнения	492
20.5.6. Операции вставки	493
Глава 21. Алгоритмы	497
21.1. Алгоритмы, не изменяющие содержимое контейнеров	497
21.1.1. Алгоритм <code>for_each</code>	497
21.1.2. Алгоритм <code>find</code>	498
21.1.3. Алгоритм <code>find_if</code>	498
21.1.4. Алгоритм <code>find_end</code>	499
21.1.5. Алгоритм <code>find_first_of</code>	500
21.1.6. Алгоритм <code>adjacent_find</code>	501
21.1.7. Алгоритм <code>count</code>	502
21.1.8. Алгоритм <code>count_if</code>	502
21.1.9. Алгоритм <code>mismatch</code>	503
21.1.10. Алгоритм <code>equal</code>	504
21.1.11. Алгоритм <code>search</code>	506
21.1.12. Алгоритм <code>search_n</code>	507
21.2. Модифицирующие алгоритмы	509
21.2.1. Алгоритм <code>copy</code>	509
21.2.2. Алгоритм <code>copy_backward</code>	509
21.2.3. Алгоритм <code>swap</code>	510
21.2.4. Алгоритм <code>swap_ranges</code>	511
21.2.5. Алгоритм <code>iter_swap</code>	511
21.2.6. Алгоритм <code>transform</code>	512
21.2.7. Алгоритм <code>replace</code>	513
21.2.8. Алгоритм <code>replace_if</code>	514
21.2.9. Алгоритм <code>replace_copy</code>	515
21.2.10. Алгоритм <code>replace_copy_if</code>	515
21.2.11. Алгоритм <code>fill</code>	516
21.2.12. Алгоритм <code>fill_n</code>	517
21.2.13. Алгоритм <code>generate</code>	517
21.2.14. Алгоритм <code>generate_n</code>	518
21.2.15. Алгоритм <code>remove</code>	519
21.2.16. Алгоритм <code>remove_if</code>	520
21.2.17. Алгоритм <code>remove_copy</code>	520
21.2.18. Алгоритм <code>remove_copy_if</code>	521
21.2.19. Алгоритм <code>unique</code>	522
21.2.20. Алгоритм <code>unique_copy</code>	523
21.2.21. Алгоритм <code>reverse</code>	524
21.2.22. Алгоритм <code>reverse_copy</code>	525
21.2.23. Алгоритм <code>rotate</code>	526
21.2.24. Алгоритм <code>rotate_copy</code>	527

21.2.25. Алгоритм random_shuffle	527
21.2.26. Алгоритм partition	529
21.3. Алгоритмы сортировки и поиска	531
21.3.1. Алгоритм sort	531
21.3.2. Алгоритм stable_sort	532
21.3.3. Алгоритм partial_sort	534
21.3.4. Алгоритм partial_sort_copy	535
21.3.5. Алгоритм nth_element	537
21.3.6. Алгоритм lower_bound	538
21.3.7. Алгоритм upper_bound	540
21.3.8. Алгоритм equal_range	541
21.3.9. Алгоритм binary_search	543
21.3.10. Алгоритм merge	544
21.3.11. Алгоритм inplace_merge	546
21.3.12. Алгоритм includes	548
21.3.13. Алгоритм set_union	551
21.3.14. Алгоритм set_intersection	553
21.3.15. Алгоритм set_difference	555
21.3.16. Алгоритм set_symmetric_difference	557
21.3.17. Алгоритм make_heap	559
21.3.18. Алгоритм push_heap	560
21.3.19. Алгоритм pop_heap	562
21.3.20. Алгоритм sort_heap	564
21.3.21. Алгоритм min	566
21.3.22. Алгоритм max	567
21.3.23. Алгоритм min_element	567
21.3.24. Алгоритм max_element	569
21.3.25. Алгоритм lexicographical_compare	570
21.3.26. Алгоритм next_permutation	572
21.3.27. Алгоритм prev_permutation	574
21.4. Численные алгоритмы	575
21.4.1. Алгоритм accumulate	575
21.4.2. Алгоритм inner_product	576
21.4.3. Алгоритм partial_sum	578
21.4.4. Алгоритм adjacent_difference	579
Часть V. Потоки и буфера	581
Глава 22. Потоки	583
22.1. Классы потоков	583
22.2. Флаги форматирования	585
22.2.1. Класс ios_base	585
22.2.2. Класс basic_ios	591
22.2.3. Класс basic_istream	593
22.2.4. Класс basic_ostream	598
22.2.5. Класс basic_iostream	601
22.2.6. Применение флагов форматирования	601
22.3. Манипуляторы	604
22.3.1. Стандартные манипуляторы	604
22.3.2. Программирование манипуляторов без параметров	611
22.4. Функции извлечения и вставки	612
22.4.1. Программирование манипуляторов, имеющих параметры	615
Глава 23. Буфера потоков	617
23.1. Класс basic_streambuf	618
23.1.1. Типы членов класса	618

23.1.2. Открытый интерфейс	619
23.1.3. Защищенный интерфейс	621
23.2. Класс <code>basic_stringbuf</code>	624
23.2.1. Типы членов класса	624
23.2.2. Открытый интерфейс	624
23.2.3. Защищенный интерфейс	625
23.3. Класс <code>strstringbuf</code>	627
23.3.1. Типы членов класса	628
23.3.2. Открытый интерфейс	628
23.3.3. Защищенный интерфейс	629
23.4. Класс <code>istream</code>	631
23.5. Класс <code>ostream</code>	631
23.6. Класс <code>stringstream</code>	632
23.7. Класс <code>basic_filebuf</code>	633
23.7.1. Типы членов класса	633
23.7.2. Открытый интерфейс	633
23.7.3. Защищенный интерфейс	635
Глава 24. Файлы	637
24.1. Класс <code>basic_ifstream</code>	637
24.1.1. Типы членов класса	638
24.1.2. Открытый интерфейс	638
24.2. Класс <code>basic_ofstream</code>	639
24.2.1. Типы членов класса	639
24.2.2. Открытый интерфейс	640
24.3. Класс <code>basic_fstream</code>	641
24.3.1. Типы членов класса	641
24.3.2. Открытый интерфейс	642
24.4. Бинарные файлы	644
24.5. Произвольный доступ	647
Приложение. Директивы препроцессора	649
Макроподстановка	649
Директива <code>#define</code>	649
Директива <code>#undef</code>	651
Встроенные макросы	652
Операторы препроцессора <code>#</code> и <code>##</code>	652
Директива <code>#line</code>	653
Условная компиляция	654
Директивы условной компиляции	654
Директивы <code>#ifdef</code> и <code>#ifndef</code>	655
Директива <code>#pragma</code>	656
Оператор <code>defined</code>	657
Директива <code>#error</code>	657
Директива <code>#include</code>	658
Библиография	659
Предметный указатель	660

