Содержание

Предисловие ... 13
Благодарности ... 15
Обращение к читателю ... 16

Часть I ... 29

Глава 1. Предыстория C++ ... 30
1.1. Язык Simula и распределенные системы ... 30
1.2. Язык C и системное программирование ... 33
1.3. Немного об авторе книги ... 33

Глава 2. Язык C with Classes ... 36
2.1. Рождение C with Classes ... 36
2.2. Обзор языковых возможностей ... 38
2.3. Классы ... 39
2.4. Эффективность исполнения ... 41
2.4.1. Встраивание ... 42
2.5. Модель компоновки ... 43
2.5.1. Простые реализации ... 46
2.5.2. Модель размещения объекта в памяти ... 48
2.6. Статический контроль типов ... 49
2.6.1. Сужающие преобразования ... 50
2.6.2. О пользе предупреждений ... 51
2.7. Почему C? ... 52
2.8 Проблемы синтаксиса ... 54
2.8.1. Синтаксис объявлений в языке C ... 54
2.8.2. Тэги структур и имена типов ... 56
2.8.3. Важность синтаксиса ... 58
2.9. Производные классы ... 58
2.9.1. Полиморфизм без виртуальных функций ... 59
2.9.2. Контейнерные классы без шаблонов ... 60
2.9.3. Модель размещения объекта в памяти ... 61
2.9.4. Ретроспектива ... 62
2.10. Модель защиты ... 62

2.11. Гарантии времени исполнения ... 65
2.11.1. Конструкторы и деструкторы ... 65
2.11.2. Распределение памяти и конструкторы ... 66
2.11.3. Функции call и return ... 67
2.12. Менее существенные средства ... 67
2.12.1. Перегрузка оператора присваивания ... 67
2.12.2. Аргументы по умолчанию ... 68
2.13. Что не реализовано в C with Classes ... 69
2.14. Рабочая обстановка ... 70

Глава 3. Рождение C++ ... 73
3.1. От C with Classes к C++ ... 73
3.2. Цели C++ ... 74
3.3. Компилятор Cfront ... 76
3.3.1. Генерирование C-кода ... 77
3.3.2. Синтаксический анализ C++ ... 79
3.3.3. Проблемы компоновки ... 80
3.3.4. Версии Cfront ... 80
3.4. Возможности языка ... 82
3.5. Виртуальные функции ... 82
3.5.1. Модель размещения объекта в памяти ... 85
3.5.2. Замещение и поиск подходящей виртуальной функции ... 87
3.5.3. Сокрытие членов базового класса ... 87
3.6. Перегрузка ... 88
3.6.1. Основы перегрузки ... 89
3.6.2. Функции-члены и дружественные функции ... 91
3.6.3. Операторные функции ... 93
3.6.4. Перегрузка и эффективность ... 94
3.6.5. Изменение языка и новые операторы ... 96
3.7. Ссылки ... 96
3.7.1. Lvalue и Rvalue ... 98
3.8. Константы ... 99
3.9. Управление памятью ... 101
3.10. Контроль типов ... 103
3.11. Второстепенные возможности ... 104
3.11.1. Комментарии ... 104
3.11.2. Нотация для конструкторов ... 104
3.11.3. Квалификация ... 105
3.11.4. Инициализация глобальных объектов ... 106
3.11.5. Предложения объявления ... 109
3.12. Языки C и C++ ... 111
3.13. Инструменты для проектирования языка ... 114

3.14. Книга «Язык программирования C++» ... 116
3.15. Статья «Whatis?» ... 117

Глава 4. Правила проектирования языка C++ ... 120
4.1. Правила и принципы ... 120
4.2. Общие правила ... 121
4.3. Правила поддержки проектирования ... 125
4.4. Технические правила ... 128
4.5. Правила поддержки низкоуровневого программирования ... 132
4.6. Заключительное слово ... 134

Глава 5. Хронология 1985–1993 гг. ... 135
5.1. Введение ... 135
5.2. Версия 2.0 ... 136
5.2.1. Обзор возможностей ... 137
5.3. Аннотированное справочное руководство ... 138
5.3.1. Обзор ARM ... 139
5.4. Стандартизация ANSI и ISO ... 140
5.4.1. Обзор возможностей ... 143

Глава 6. Стандартизация ... 144
6.1. Что такое стандарт? ... 144
6.1.1. Детали реализации ... 145
6.1.2. Тест на реалистичность ... 146
6.2. Работа комитета ... 146
6.2.1. Кто работает в комитете ... 148
6.3. Как велась работа ... 148
6.3.1. Разрешение имен ... 149
6.3.2. Время жизни объектов ... 153
6.4. Расширения ... 157
6.4.1. Критерии рассмотрения предложений ... 159
6.4.2. Текущее состояние дел ... 161
6.4.3. Проблемы, связанные с полезными расширениями ... 162
6.4.4. Логическая непротиворечивость ... 163
6.5. Примеры предлагавшихся расширений ... 164
6.5.1. Именованные аргументы ... 164
6.5.2. Ограниченные указатели ... 168
6.5.3. Наборы символов ... 169

Глава 7. Заинтересованность и использование ... 174
7.1. Рост интереса к C++ ... 174
7.1.1. Отсутствие маркетинга C++ ... 175
7.1.2. Конференции ... 175

7.1.3. Журналы и книги ... 176
7.1.4. Компиляторы ... 177
7.1.5. Инструментальные средства и среды программирования ... 177
7.2. Преподавание и изучение C++ ... 178
7.3. Пользователи и приложения ... 183
7.3.1. Первые пользователи ... 183
7.3.2. Сферы применения C++ ... 184
7.4. Коммерческая конкуренция ... 184
7.4.1. Традиционные языки ... 185
7.4.2. Современные языки ... 186
7.4.3. Как выдержать конкуренцию ... 187

Глава 8. Библиотеки ... 189
8.1. Введение ... 189
8.2. Проектирование библиотеки C++ ... 189
8.2.1. Альтернативы при проектировании библиотеки ... 190
8.2.2. Языковые средства и построение библиотеки ... 190
8.2.3. Как работать с разнообразными библиотеками ... 191
8.3. Ранние библиотеки ... 192
8.3.1. Библиотека потокового ввода/вывода ... 193
8.3.2. Поддержка параллельности ... 196
8.4. Другие библиотеки ... 198
8.4.1. Базовые библиотеки ... 199
8.4.2. Устойчивость и базы данных ... 200
8.4.3. Библиотеки для численных расчетов ... 200
8.4.4. Специализированные библиотеки ... 201
8.5. Стандартная библиотека ... 201

Глава 9. Перспективы развития языка C++ ... 203
9.1. Введение ... 203
9.2. Оценка пройденного пути ... 203
9.2.1. Достигнуты ли основные цели C++? ... 204
9.2.2. Является ли C++ логически последовательным языком? ... 204
9.2.3. Основная недоработка языка ... 207
9.3. Всего лишь мост? ... 208
9.3.1. Мост нужен надолго ... 208
9.3.2. Если C++ – это ответ, то на какой вопрос? ... 209
9.4. Что может сделать C++ более эффективным ... 213
9.4.1. Стабильность и стандарты ... 213
9.4.2. Обучение и приемы ... 213
9.4.3. Системные вопросы ... 213
9.4.4. За пределами файлов и синтаксиса ... 214
9.4.5. Подведение итогов и перспективы ... 215

Часть II 217

Глава 10. Управление памятью 218
10.1. Введение 218
10.2. Отделение распределения памяти и инициализации 219
10.3. Выделение памяти для массива 220
10.4. Размещение объекта в памяти 221
10.5. Проблемы освобождения памяти 222
10.5.1. Освобождение памяти для массивов 224
10.6. Нехватка памяти 225
10.7. Автоматическая сборка мусора 226
10.7.1. Необязательный сборщик мусора 226
10.7.2. Как должен выглядеть необязательный сборщик мусора? 228

Глава 11. Перегрузка 230
11.1. Введение 230
11.2. Разрешение перегрузки 230
11.2.1. Детальное разрешение 231
11.2.2. Управление неоднозначностью 233
11.2.3. Нулевой указатель 236
11.2.4. Ключевое слово overload 238
11.3. Типобезопасная компоновка 239
11.3.1. Перегрузка и компоновка 239
11.3.2. Реализация компоновки в C++ 240
11.3.3. Анализ пройденного пути 241
11.4. Создание и копирование объектов 244
11.4.1. Контроль допустимости копирования 244
11.4.2. Управление распределением памяти 244
11.4.3. Управление наследованием 245
11.4.4. Почленное копирование 246
11.5. Удобство нотации 248
11.5.1. «Умные» указатели 248
11.5.2. «Умные» ссылки 249
11.5.3. Перегрузка операторов инкремента и декремента 252
11.5.4. Перегрузка ->* 254
11.5.5. Перегрузка оператора «запятая» 254
11.6. Добавление в C++ операторов 254
11.6.1. Оператор возведения в степень 254
11.6.2. Операторы, определяемые пользователем 257
11.6.3. Составные операторы 258
11.7. Перечисления 259
11.7.1 Перегрузка на базе перечислений 261
11.7.2. Тип Boolean 261

Глава 12. Множественное наследование ... 263
12.1. Введение ... 263
12.2. Базовые классы ... 264
12.3. Виртуальные базовые классы ... 265
12.3.1. Виртуальные базовые классы и виртуальные функции ... 267
12.4. Модель размещения объекта в памяти ... 270
12.4.1. Размещение в памяти объекта виртуального базового класса ... 272
12.4.2. Виртуальные базовые классы и приведение типов ... 273
12.5. Комбинирование методов ... 274
12.6. Полемика о множественном наследовании ... 276
12.7. Делегирование ... 279
12.8. Переименование ... 280
12.9. Инициализаторы членов и базовых классов ... 282

Глава 13. Уточнения понятия класса ... 284
13.1 Введение ... 284
13.2. Абстрактные классы ... 284
13.2.1. Абстрактные классы и обработка ошибок ... 284
13.2.2. Абстрактные типы ... 286
13.2.3. Синтаксис ... 288
13.2.4. Виртуальные функции и конструкторы ... 288
13.3. Константные функции-члены ... 291
13.3.1. Игнорирование const при приведении типов ... 291
13.3.2. Уточнение определения const ... 292
13.3.3. Ключевое слово mutable и приведение типов ... 293
13.4. Статические функции-члены ... 294
13.5. Вложенные классы ... 295
13.6. Ключевое слово inherited ... 297
13.7. Ослабление правил замещения ... 299
13.7.1. Ослабление правил аргументов ... 301
13.8. Мультиметоды ... 303
13.8.1. Когда нет мультиметодов ... 305
13.9. Защищенные члены ... 307
13.10. Улучшенная генерация кода ... 308
13.11. Указатели на функции-члены ... 309

Глава 14. Приведение типов ... 311
14.1. Крупные расширения ... 311
14.2. Идентификация типа во время исполнения ... 312
14.2.1. Зачем нужен механизм RTTI ... 313
14.2.2. Оператор dynamic_cast ... 313
14.2.3. Правильное и неправильное использование RTTI ... 319

14.2.4. Зачем давать «опасные средства» ... 321
14.2.5. Оператор typeid() ... 322
14.2.6. Модель размещения объекта в памяти ... 326
14.2.7. Простой ввод/вывод объектов ... 327
14.2.8. Другие варианты ... 329
14.3. Новая нотация для приведения типов ... 333
14.3.1. Недостатки старых приведений типов ... 334
14.3.2. Оператор static_cast ... 335
14.3.3. Оператор reinterpret_cast ... 337
14.3.4. Оператор const_cast ... 339
14.3.5. Преимущества новых приведений типов ... 340

Глава 15. Шаблоны ... 343
15.1. Введение ... 343
15.2. Зачем нужны шаблоны ... 344
15.3. Шаблоны классов ... 346
15.3.1. Аргументы шаблонов, не являющиеся типами ... 347
15.4. Ограничения на аргументы шаблонов ... 348
15.4.1. Ограничения за счет наследования ... 349
15.4.2. Ограничения за счет использования ... 350
15.5. Устранение дублирования кода ... 351
15.6. Шаблоны функций ... 353
15.6.1. Выведение аргументов шаблона функции ... 354
15.6.2. Задание аргументов шаблона функции ... 355
15.6.3. Перегрузка шаблона функции ... 357
15.7. Синтаксис ... 360
15.8. Методы композиции ... 361
15.8.1. Представление стратегии реализации ... 362
15.8.2. Представление отношений порядка ... 363
15.9. Соотношения между шаблонами классов ... 365
15.9.1. Отношения наследования ... 365
15.9.2. Преобразования ... 367
15.9.3. Шаблоны-члены ... 368
15.10. Инстанцирование шаблонов ... 369
15.10.1. Явное инстанцирование ... 371
15.10.2. Точка инстанцирования ... 372
15.10.3. Специализация ... 378
15.10.4. Нахождение определений шаблонов ... 381
15.11. Последствия введения шаблонов ... 383
15.11.1. Отделение реализации от интерфейса ... 384
15.11.2. Гибкость и эффективность ... 384
15.11.3. Влияние на другие компоненты C++ ... 385

Глава 16. Обработка исключений ... 387
16.1. Введение ... 387
16.2. Цели и предположения ... 388
16.3. Синтаксис ... 389
16.4. Группировка ... 390
16.5. Управление ресурсами ... 391
16.5.1. Ошибки в конструкторах ... 393
16.6. Возобновление или завершение? ... 394
16.6.1. Обходные пути для реализации возобновления ... 397
16.7. Асинхронные события ... 398
16.8. Распространение на несколько уровней ... 399
16.9. Статическая проверка ... 399
16.9.1. Вопросы реализации ... 401
16.10. Инварианты ... 402

Глава 17. Пространства имен ... 403
17.1. Введение ... 403
17.2. Для чего нужны пространства имен ... 404
17.2.1. Обходные пути ... 404
17.3. Какое решение было бы лучшим? ... 406
17.4. Решение: пространства имен ... 408
17.4.1. Мнения по поводу пространств имен ... 410
17.4.2. Внедрение пространств имен ... 411
17.4.3. Псевдонимы пространства имен ... 412
17.4.4. Использование пространств имен для управления версиями ... 413
17.4.5. Технические детали ... 415
17.5. Классы и пространства имен ... 421
17.5.1. Производные классы ... 421
17.5.2. Использование базовых классов ... 423
17.5.3. Исключение глобальных статических объявлений ... 424
17.6. Совместимость с C ... 425

Глава 18. Препроцессор C ... 427

Алфавитный указатель ... 431

Предисловие

Кто не пашет, должен писать.

Мартин А. Хансен

На второй конференции ACM по истории языков программирования (HOPL-2) меня попросили написать статью по истории C++. Мысль показалась мне разумной, предложение – лестным, поэтому я приступил к работе. Чтобы работа получилась более полной и неискаженной, я обратился за помощью к друзьям. Вслед за этим молва разнесла известие о готовящемся проекте. Не обошлось без преувеличений – и однажды я получил по электронной почте письмо с вопросом, где можно купить мою новую книгу о дизайне C++. Оно и положило начало этой серьезной работе.

Обычно в книгах по программированию и его языкам объясняется, чту представляет собой язык и как им пользоваться. Но многих интересует вопрос, почему язык оказался именно таким и как он создавался. Именно об этом, применительно к C++, и говорится в данной книге. В ней рассказывается, как шла эволюция языка от первоначального проекта до нынешнего состояния, объясняются основные задачи и цели проектирования, заложенные в языке идеи и ограничения, описывается развитие языковых концепций.

Естественно, ни C++, ни представления о его дизайне и применении не изменялись сами собой. В действительности эволюционировали сознание пользователей и их представления о том, какие перед ними стоят цели и какие инструменты необходимы для решения. А значит, в книге говорится о наиболее важных вопросах, которые снимались с помощью C++, а также о людях, решавших эти задачи и оказавших влияние на развитие языка, и их воззрениях.

C++ – все еще сравнительно молодой язык. Некоторые из обсуждаемых в книге проблем пока неизвестны широкой публике. Для полного осознания всех последствий описываемых решений потребуется еще немало лет. В этой книге представлена только моя точка зрения на то, почему появился C++, что он собой представляет и каким ему следовало бы быть. Надеюсь, что данная работа будет способствовать дальнейшей эволюции этого языка и поможет людям понять, как лучше всего использовать его.

Основное внимание в книге уделено общим целям дизайна, практическим ограничениям. Ключевые проектные решения, относящиеся к языковым средствам, излагаются в историческом контексте. Я прослеживаю эволюцию языка от C with Classes (C с классами) к версиям 1.0 и 2.0 и далее к проводимой в настоящее время комитетом ANSI/ISO работе по стандартизации. В книге анализируются и такие аспекты: резкое увеличение количества приложений по C++, возросший

интерес пользователей к языку, усиление коммерческой активности, появление большого количества компиляторов, инструментальных средств, сред программирования и библиотек. Подробно обсуждаются связи между языками C++ и Simula. Взаимодействие C++ с другими языками затрагивается лишь мимоходом. Дизайн основных языковых средств, к которым можно отнести классы, наследование, абстрактные классы, перегрузку, управление памятью, шаблоны, обработку исключений, идентификацию типов во время исполнения и пространства имен, рассматривается довольно подробно.

Основная цель книги – дать программистам на C++ правильное представление о фундаментальных концепциях языка и побудить их к экспериментам с теми его возможностями, о которых они не подозревали. Книга будет полезна как опытным программистам, так и студентам и, возможно, поможет решить, стоит ли тратить время на изучение C++.

Благодарности

Я выражаю глубокую признательность Стиву Клэмиджу (Steve Clamage), Тони Хансену (Tony Hansen), Лорейн Джуль (Lorraine Juhl), Питеру Джулю (Peter Juhl), Брайану Кернигану (Brian Kernighan), Ли Найту (Lee Knight), Дугу Леа (Doug Lea), Дугу Макилрою (Doug MacIlroy), Барбаре Му (Barbara Moo), Йенсу Палсбергу (Jens Palsberg), Стиву Рамсби (Steve Rumsby) и Кристоферу Скелли (Christopher Skelly) за то, что они прочли все черновые варианты этой книги. Их конструктивные замечания привели к кардинальным изменениям в содержании и организации работы. Стив Бурофф (Steve Buroff), Мартин Кэролл (Martin Caroll), Шон Корфилд (Sean Corfield), Том Хагельскяер (Tom Hagelskjaer), Рик Холлинбек (Rick Hollinbeck), Деннис Манкль (Dennis Mancl) и Стэн Липпман (Stan Lippman) высказали замечания по отдельным главам. Отдельное спасибо Арчи Лахнеру (Archie Lachner), поинтересовавшемуся, где достать эту книгу еще до того, как я всерьез задумался о ее написании.

Естественно, я признателен всем тем, кто участвовал в создании языка C++. В некотором смысле моя книга – дань уважения этим людям; имена некоторых из них упомянуты в разных главах и в предметном указателе. Впрочем, если отмечать конкретных людей, то более всех помогали мне и поддерживали меня Брайан Керниган (Brian Kernighan), Эндрю Кениг (Andrew Koenig), Дуг Макилрой (Doug MacIlroy) и Джонатан Шопиро (Jonathan Shopiro). Они также щедро делились своими идеями на протяжении более десяти лет. Спасибо Кристену Найгаарду (Kristen Nygaard) и Деннису Ричи (Dennis Ritchie) – разработчикам соответственно Simula и C, откуда я позаимствовал многие ключевые компоненты. Со временем я оценил Кристена и Денниса не только как блестящих разработчиков языков, но и как очень порядочных и симпатичных людей.

Бьерн Страуструп

Обращение к читателю

**Писательство – это единственное искусство,
которым нужно овладевать посредством писания.**

Анонимный автор

Введение

C++ проектировался с целью обеспечить средства организации программ, присущие языку Simula, а также необходимую для системного программирования эффективность и гибкость, свойственные C. Предполагалось, что от замысла до его первой реализации пройдет примерно полгода. Так оно и вышло.

Тогда – в середине 1979 г. – я еще не осознавал, насколько эта цель была скромной и в то же время абсурдной. Скромной – потому что не предполагалось вводить какие бы то ни было новшества, абсурдной – из-за слишком жестких временных рамок и драконовских требований к эффективности и гибкости языка. Новшества со временем все же появились, но в вопросах эффективности и гибкости ни на какие компромиссы я не пошел. С годами цели C++ уточнялись, видоизменялись и формулировались более четко, но и сегодня язык в точности отражает первоначально поставленные цели.

Назначение книги, которую вы держите в руках, – документировать эти цели, проследить их эволюцию и представить на суд читателей тот C++, который появился в результате. Я старался уделять равное внимание историческим фактам (именам, местам и событиям) и техническим вопросам дизайна, реализации и применения языка. Я стремился фиксировать не каждое событие, а лишь ключевые идеи и направления развития, которые уже повлияли на определение C++ и, возможно, еще скажутся на его дальнейшей эволюции и практике применения.

Если я упоминаю некоторое событие, то стараюсь не выдавать желаемое за действительное. Там, где это уместно, приводятся цитаты из различных статей, иллюстрирующие, как разные цели, принципы или свойства языка представлялись в то время. Я не пытаюсь давать оценку прошлому с позиций сегодняшнего дня. Воспоминания о событиях и замечания по поводу последствий принятых тогда решений отделены от основного текста и помечены. Я вообще питаю отвращение к историческому ревизионизму и стараюсь этого избегать. Приведу в пример свое старое высказывание: «Я пришел к выводу, что система типов в языке Pascal не просто бесполезна– это смирительная рубашка, которая создает проблем больше, чем решает, заставляя меня жертвовать чистотой дизайна ради удовлетворения причуд компилятора». Так я думал в то время, и данное мнение существенно повлияло на эволюцию C++. Было ли это резкое осуждение Pascal справедливым и стал бы я сегодня

(по прошествии более десяти лет) судить точно так же, не имеет значения. Я не могу притвориться, что этого не было (чтобы пощадить чувства приверженцев Pascal или избавить себя от чувства неловкости и лишних споров) или как-то пригладить свое высказывание (приведя более полную и сбалансированную точку зрения), не исказив при этом историю C++.

Когда я говорю о людях, которые внесли вклад в дизайн и эволюцию C++, то по возможности уточняю, в чем именно их заслуга и когда это происходило. Но тут есть опасность. Поскольку память моя несовершенна, я вполне мог что-то забыть. Приношу свои извинения. Я называю имена тех, кто способствовал принятию того или иного решения относительно C++. Увы, не всегда это именно тот человек, который впервые столкнулся с определенной проблемой или задумался о ее решении. Досадно, однако вообще отказаться от упоминания имен было бы еще хуже. Не стесняйтесь сообщать мне обо всем, что могло бы внести ясность в этот вопрос.

Когда описываешь исторические события, всегда задумываешься об объективности. Неизбежную предвзятость я старался компенсировать получением информации о событиях, в которых сам не принимал участия, и беседами с другими участниками событий. Еще я попросил нескольких человек, имеющих отношение к эволюции C++, прочесть эту книгу. Их имена приведены в конце предисловия. Ну и, кроме того, статья, представленная на второй конференции по истории языков программирования [Stroustrup, 1993] и содержащая основные исторические факты, упоминаемые в этой книге, была тщательно переработана и освобождена от излишней предвзятости.

Как читать эту книгу

В части I описывается дизайн, эволюция, практика применения и процесс стандартизации C++ в относительной хронологической последовательности. Такая схема выбрана потому, что решения, принимавшиеся в ранние годы, выстраиваются в четкую, логичную временную цепочку. В главах 1, 2 и 3 описываются истоки C++ и его эволюция от C with Classes к версии 1.0. В главе 4 излагаются правила, по которым C++ развивался в течение этого периода и позже. Главы 5, 6 посвящены соответственно хронологии разработки после выхода версии 1.0 и процессу стандартизации C++ под эгидой ANSI/ISO. О перспективе развития языка говорится в 7 и 8 главах, где анализируются приложения, инструментальные средства и библиотеки. И, наконец, в главе 9 представлены ретроспективный взгляд и некоторые общие мысли о перспективах развития C++.

В части II описывается разработка C++ после выхода версии 1.0. Язык развивался в тех направлениях, которые были определены уже ко времени выпуска версии 1.0: добавление желательных свойств (к примеру, шаблонов и обработки исключений) и принципов их дизайна. После появления версии 1.0 хронологический порядок событий был уже не так важен для разработки C++. Определение языка в основных чертах осталось бы таким же, как сейчас, даже если бы последовательность реализации расширений была бы иной. То, в каком порядке решались задачи и как к языку добавлялись новые свойства, представляет лишь исторический интерес. Строго хронологическое изложение помешало бы логически естественному

представлению идей, поэтому в основу построения части II положено описание важнейших свойств языка. Отдельные главы части II не зависят друг от друга, так что читать их можно в любом порядке. Глава 10 посвящена управлению памятью, 11 – перегрузке, 12 – множественному наследованию, 13 – уточнениям концепции класса, 14 – приведению типов, 15 – шаблонам, 16 – обработке исключений, 17 – пространствам имен, 18 – препроцессору С.

Люди ждут от книги по дизайну и эволюции языка программирования разного. Не найдется двух человек, имеющих одинаковое мнение о степени детализации, необходимой при обсуждении некоторой темы. Так, все рецензии на варианты статьи, представленной на конференции HOPL-2, строились по одинаковому шаблону: «Статья слишком длинная... пожалуйста, уделите больше внимания темам x, y и z». И пока одни рецензенты говорили: «Исключите философскую дребедень, пусть будет побольше технических деталей», другие требовали: «Избавьте от этих утомительных деталей, мне интересна философия проектирования».

Чтобы выйти из этого положения, я написал книгу внутри книги. Если вас не интересуют детали, то для начала пропустите все подразделы, имеющие номера x.y.z, где x – номер главы, а y – номер раздела. А из оставшегося читайте то, что вам интереснее. Впрочем, читать можно и последовательно, от первой до последней страницы. Правда, при этом вы рискуете увязнуть в деталях. Я вовсе не хочу сказать, что они несущественны. Напротив, никакой язык программирования нельзя понять, рассматривая только его общие принципы. Без конкретных примеров никак не обойтись. Но, если изучать только детали и не представлять общей картины, можно окончательно запутаться.

Стремясь облегчить участь читателя, я поместил в часть II обсуждение почти всех новых свойств языка, которые принято считать «продвинутыми». Поэтому о базовых конструкциях говорится в I части. Почти вся информация о нетехнических аспектах эволюции C++ сосредоточена здесь же. Те, кто не любит «философских материй», могут пропустить главы с 4 по 9 и сразу перейти к техническим деталям, описанным в части II.

Я предвижу, что кое-кто захочет использовать эту книгу как справочный материал, а многие прочтут лишь отдельные главы, не обращая внимания на то, что им предшествует. Такой подход тоже возможен, поскольку я постарался сделать главы по возможности замкнутыми и, как опытный программист на C++, не поскупился на перекрестные ссылки и составил подробный предметный указатель.

Пожалуйста, имейте в виду, что в книге я не пытаюсь строго определить возможности языка C++. Здесь излагаются лишь подробности, необходимые для описания того, как появилось на свет то или иное средство. Я не ставлю себе целью научить программированию или проектированию с использованием C++. Тех, кому это интересно, отсылаю к работе [2nd].

Хронология C++

Приведенная ниже хронологическая таблица поможет вам лучше представить то, о чем говорится дальше.

Таблица 1

Год	Месяц	Событие
1979	май	Начало работы на C with Classes
	октябрь	Готова первая реализация C with Classes
1980	апрель	Первый внутренний документ по C with Classes в Bell Labs [Stroustrup, 1980]
1982	январь	Первая опубликованная статья по C with Classes [Stroustrup, 1982]
1983	август	Готова первая реализация C++
	декабрь	C++ получил имя
1984	январь	Вышло первое руководство по C++
1985	февраль	Первая распространяемая версия C++ (Release E)
	октябрь	Cfront Release 1.0 (первая коммерческая версия)
	октябрь	Вышла книга *The C++ Programming Language* [Stroustrup, 1986]
1986	август	Статья «Whatis?» [Stroustrup, 1986b]
	сентябрь	Конференция OOPSLA (положившая начало безудержной рекламе объектно-ориентированного программирования, крутящейся вокруг Smalltalk)
	ноябрь	Первая коммерческая версия Cfront для ПК (Cfront 1.1, Glockenspiel)
1987	февраль	Cfront Release 1.2
	ноябрь	Первая конференция USENIX, посвященная C++ (Санта Фе, штат Нью-Мексико)
	декабрь	Первая версия GNU C++ (1.13)
1988	январь	Первая версия Oregon Software C++
	июнь	Первая версия Zortech C++
	октябрь	Первый симпозиум разработчиков C++ в рамках USENIX (Эстес Парк, штат Колорадо)
1989	июнь	Cfront Release 2.0
	декабрь	Организационная встреча комитета ANSI X3J16 (Вашингтон, округ Колумбия)
1990	март	Первое техническое совещание комитета ANSI X3J16 (Сомерсет, штат Нью-Джерси)
	май	Первая версия Borland C++
	май	Вышла книга *The Annotated C++ Reference Manual* [ARM]
	июль	Одобрены шаблоны (Сиэтл, штат Вашингтон)
	ноябрь	Одобрен механизм обработки исключений (Пало Альто, штат Калифорния)
1991	июнь	Вышло второе издание The C++ Programming Language [2nd]
	июнь	Первая встреча рабочей группы ISO WG21 (Лунд, Швеция)
	октябрь	Cfront Release 3.0 (включающая шаблоны)
1992	февраль	Первая версия DEC C++ (включающая шаблоны и обработку исключений)
	март	Первая версия Microsoft C++
	май	Первая версия IBM C++ (включающая шаблоны и обработку исключений)
1993	март	Одобрена идентификация типов во время исполнения (Портленд, штат Орегон)
	июль	Одобрены пространства имен (Мюнхен, Германия)
1994	август	В комитете ANSI/ISO зарегистрирован предварительный стандарт

Пользователь превыше всего

Эта книга написана для пользователей C++, то есть для программистов и проектировщиков. Я старался избегать запутанных и понятных лишь немногим экспертам предметов, стремясь увидеть C++, его средства и эволюцию глазами пользователя. Узкоспециальные аспекты языка обсуждаются только тогда, когда связаны с вопросами, непосредственно касающимися пользователей. В качестве примеров можно привести рассмотрение правил разрешения имен в шаблонах (см. раздел 15.10) и периода существования временных объектов (см. пункт 6.3.2).

Специалисты по языкам программирования, языковые пуристы и разработчики найдут в этой книге немало поводов для критики, но я стремился дать общую картину, а не точные и исчерпывающие разъяснения по поводу каждой мелочи. Если вам нужны технические детали, обратитесь к книге *The Annotated C++ Reference Manual* [ARM], где дано определение языка, или к книге *Язык программирования C++* (второе издание) [2nd][1], или к рабочим материалам комитета ANSI/ISO по стандартизации.

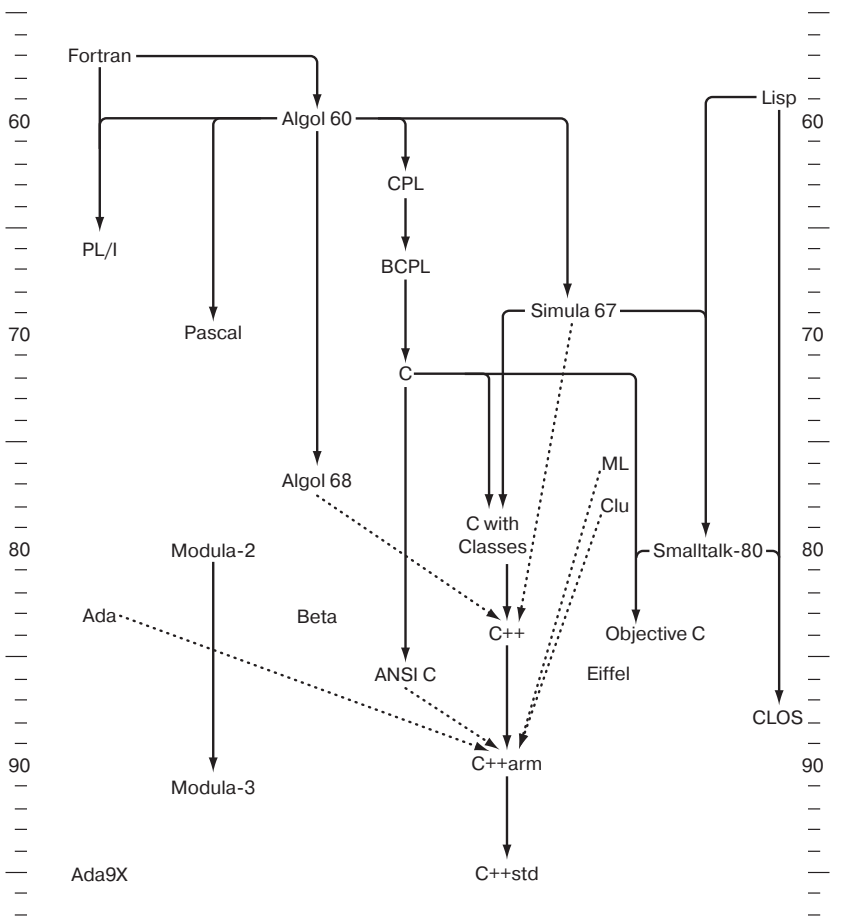
Языки программирования

Несколько рецензентов просили меня сравнить C++ с другими языками. Но я решил этого не делать. Еще раз выскажу свою неизменную точку зрения: сравнение языков редко бывает осмысленным, а еще реже честным. Для хорошего сравнительного анализа основных языков программирования требуется больше времени, чем можно себе позволить, нужно иметь опыт работы во многих прикладных областях, быть беспристрастным и объективным. Времени у меня нет, и вряд ли я, как создатель C++, могу рассчитывать, что многие поверят в мою беспристрастность по этому поводу.

Еще меня беспокоит некий феномен, который я неоднократно наблюдал, когда другие пытались заняться честным сравнением языков. Авторы изо всех сил стараются быть объективными, но обязательно делают упор на каком-то одном виде приложений, одном стиле или определенной культуре программирования. А уж если один язык более известен, чем другой, то в оценке чуть ли изначально возникает некая предубежденность: дефекты хорошо известного языка признаются несущественными, для них предлагаются простые способы обхода, тогда как аналогичные изъяны в других языках объявляются фундаментальными проблемами. Но ведь часто бывает, что в менее популярном языке давно известны методы решения подобных задач, только автор сравнительного обзора о них не знает или считает неудовлетворительными, поскольку в знакомом ему языке они работать не будут.

Кроме того, самая свежая информация о хорошо известном языке доступна. Говоря о языке менее популярном, авторы подчас вынуждены полагаться на устаревшие сведения. Если языки вообще уместно сопоставлять, то сравнивать язык X на основе определения трехлетней давности с языком Y в том виде, в каком он

[1] В настоящее время уже вышел русский перевод третьего издания этой книги (Б. Страуструп «Язык программирования C++», издательство «Бином», 1999). – *Прим. перев.*



существует в данный момент, нечестно и бессмысленно. Поэтому, как уже было сказано выше, в отношении отличных от C++ языков я ограничусь общими высказываниями и небольшими замечаниями на узкие темы.

Чтобы прояснить историческое место C++, на рисунке ниже показано, когда появились на свет другие языки программирования, часто упоминаемые в связи с данным.

Данная диаграмма ни в коей мере не претендует на полноту и отражает лишь важнейшие влияния на C++. В частности, воздействие концепции классов из Simula на диаграмме отражено явно неполно: на языки Ada [Ichbiah, 1979] и Clu [Liskov, 1979] Simula повлияла не очень сильно, а на Ada9X [Taft, 1992], Beta [Madsen, 1993], Eiffel [Meyer, 1988] и Modula-3 [Nelson, 1991] – весьма заметно. Диаграмма не отражает влияние C++ на другие языки. Сплошными линиями обозначено воздействие на структуру языка, пунктирными – на его специфические

средства. Под датой появления языка в большинстве случаев понимается дата первой версии. Например, Algol68 [Woodward, 1974] датирован 1977 г., а не 1968 г.

Анализируя отзывы на статью, представленную на конференции HOPL-2, и многие другие источники, я пришел к выводу: нет согласия по поводу того, что же такое язык программирования и какой главной цели он должен служить. Как его определить? Это инструмент для составления компьютерных заданий? Средство для общения программистов? Способ представить высокоуровневые проекты? Система обозначений для записи алгоритмов? Возможность выразить отношения между концепциями? Средство для проведения экспериментов? Механизм управления компьютеризованными устройствами? По-моему, язык программирования общего назначения должен отвечать всем вышеперечисленным критериям и удовлетворять потребности самых разных групп пользователей. Единственное, чем язык быть не может, – это простым набором «изящных штучек».

Разнобой мнений также отражает существующие точки зрения на то, что такое информатика и как следует проектировать языки. Является ли информатика частью математики или это, скорее, отрасль инженерного проектирования? Или архитектуры? Или искусства? Или биологии? Или социологии? Или философии? А может, информатика заимствует методы и подходы, применяемые во всех этих дисциплинах? Я склоняюсь к последнему предположению.

Отсюда следует, что проектирование языка отличается от более строгих и формальных наук вроде математики и философии. Чтобы стать полезным, универсальный язык программирования должен быть эклектичным и учитывать множество прагматических и социальных факторов. В частности, каждый язык проектируется для решения определенного вида задач в определенное время в соответствии с представлениями определенной группы людей. С появлением новых требований исходный дизайн начинает расширяться, чтобы соответствовать современному пониманию задач. Эта точка зрения прагматична, но не беспринципна. Я твердо убежден, что все успешные языки программирования именно развивались, а не были просто спроектированы на базе первоначально заложенных принципов. Принципы лежат в основе исходного дизайна и определяют последующую эволюцию языка, однако и сами они не остаются застывшими.

Библиография

В этом разделе перечислены работы, на которые даются ссылки во всех главах книги.

Таблица 2

[2nd]	see [Stroustrup,1991].
[Agha,1986]	Gul Agha: *An Overview of Actor languages*. ACM SIGPLAN Notices. October 1986.
[Aho,1986]	Alfred Aho, Ravi Sethi, and Jeffrey D. Ullman: *Compilers: Principles, Techniques, and Tools.* Addison-Wesley, Reading, MA. 1986. ISBN 0-201-10088-6.
[ARM]	see [Ellis.1990].
[Babcisky,1984]	Karel Babcisky: *Simula Performance Assessment.* Proc. IFIP WG2.4 Conference on System Implementation Languages: Experience and Assessment. Canterbury, Kent, UK. September 1984.

Таблица 2 (продолжение)

[Barton, 1994]	John J. Barton and Lee R. Nackman: *Scientific and Engineering C++: An Introduction with Advanced Techniques and Examples.* Addison-Wesley, Reading, MA. 1994. ISBN 0-201-53393-6.
[Birtwistle,1979]	Graham Birtwistle, Ole-Johan Dahl, Bjцrn Myrhaug, and Kristen Nygaard: *SIMULA BEGIN.* Studentlitteratur, Lund, Sweden. 1979. ISBN 91-44-06212-5.
[Boehm,1993]	Hans-J. Boehm: *Space Efficient Conservative Garbage Collection.* Proc. ACM SIGPLAN '93 Conference on Programming Language Design and Implementation. ACM SIGPLAN Notices. June 1993.
[Booch,1990]	Grady Booch and Michael M. Vilot: *The Design of the C++ Booch Components.* Proc. OOPSLA'90. October 1990.
[Booch,1991]	Grady Booch: *Object-Oriented Design.* Benjamin Cummings, Redwood City, CA. 1991. ISBN 0-8053-0091-0.
[Booch,1993]	Grady Booch: *Object-Oriented Analysis and Design with Applications, 2nd edition.* Benjamin Cummings, Redwood City, CA. 1993. ISBN 0-8053-5340-2.
[Booch,1993b]	Grady Booch and Michael M. Vilot: *Simplifying the C++ Booch Components.* The C++Report. June 1993.
[Budge, 1992]	Ken Budge, J.S. Perry, and A.C. Robinson: *High-Performance Scientific Computation using C++.* Proc. USENIX C++ Conference. Portland, OR. August 1992.
[Buhr,1992]	Peter A. Buhr and Glen Ditchfield: *Adding Concurrency to a Programming Language.* Proc. USENIX C++ Conference. Portland, OR. August 1992.
[Call, 1987]	Lisa A. Call, et al.: *CLAM - An Open System for Graphical User Interfaces.* Proc. USENIX C++ Conference. Santa Fe, NM. November 1987.
[Cameron,1992]	Don Cameron, et al.: A *Portable Implementation of C++ Exception Handling.* Proc. USENIX C++ Conference. Portland, OR. August 1992.
[Campbell,1987]	Roy Campbell, et al.: *The Design of a Multiprocessor Operating System.* Proc. USENIX C++ Conference. Santa Fe, NM. November 1987.
[Cattell,1991]	Rich G.G. Cattell: *Object Data Management: Object-Oriented and Extended Relational Database Systems.* Addison-Wesley, Reading, MA. 1991. ISBN 0-201-53092-9.
[Cargill,1991]	Tom A. Cargill: *The Case Against Multiple Inheritance in C++.* USENIX Computer Systems. Vol4, no 1, 1991.
[Carroll,1991]	Martin Carroll: *Using Multiple Inheritance to Implement Abstract Data Types.* The C++Report. April 1991.
[Carroll,1993]	Martin Carroll: *Design of the USL Standard Components.* The C++ Report. June 1993.
[Chandy,1993]	K. Mani Chandy and Carl Kesselman: *Compositional C++: Compositional Parallel Programming.* Proc. Fourth Workshop on Parallel Computing and Compilers. Springer-Verlag. 1993.
[Cristian,1989]	Flaviu Cristian: *Exception Handling.* Dependability of Resilient Computers, T. Andersen, editor. BSP Professional Books, Blackwell Scientific Publications, 1989.
[Cox, 1986]	Brad Cox: *Object-Oriented Programming: An Evolutionary Approach.* Addison-Wesley, Reading, MA. 1986.
[Dahl,1988]	Ole-Johan Dahl: Personal communication.
[Dearle, 1990]	Fergal Dearle: *Designing Portable Applications Frameworks for C++.* Proc. USENIX C++Conference. San Francisco, CA. April 1990.