Летопись языков. Си++

Руслан Богатырев

Пожалуй, не найдется в мире ни одного другого языка, которому было бы посвящено такое несметное количество книг и статей. Мы вроде бы должны знать о нем все или почти все. Но так ли это на самом деле? Почему чем дальше, тем больше он становится уделом узкого круга профессионалов? Что ждет его в будущем?

Что бы нового мы ни совершали,
мы должны дать возможность людям переходить
от старых инструментов и идей к новым.
Бьерн Страуструп

Как и в предыдущих статьях цикла «Летопись языков» [1, 2], мы попробуем здесь посмотреть на этот язык глазами его создателя и вернуться к первоистокам. Быть может, это даст нам новую пищу для размышлений и позволит иначе взглянуть на казалось бы привычные вещи?

Позиции Си++ в современном мире

Подобно Си (Деннис Ритчи), Паскалю (Никлаус Вирт) и Java (Джеймс Гослинг), язык Си++ был задуман одним человеком. Имя его — Бьерн Страуструп (см. врезку). В конце 1980-х годов Си++, стремительно ворвавшийся в элиту языков программирования, очень быстро завоевал миллионы поклонников даже без проведения мощной маркетинговой кампании. Как не без горечи отмечает Страуструп [3], корпорация AT&T в течение первых трех лет после появления языка затратила на рекламу Си++ целых 3 тыс. долларов.

Становление ИТ-индустрии в 1990-х годах пришлось на время его безмятежного царствования. Положив конец эре Си и Паскаля, он стал фундаментом великих программистских строек конца прошлого столетия. В отличие от своих новых конкурентов, он не стал собственностью какой-либо конкретной компании, хотя и Microsoft (Visual Basic), и Borland (Delphi), и Sun (Java) делали и продолжают немало делать для его совершенствования. Правда, сейчас все чаще говорят о закате эры Си++. Отсутствие конкретного хозяина, способствовавшее триумфальному восхождению на трон, похоже, сослужило ему плохую службу. Едва лишь мир поразила Интернет-лихорадка, едва лишь на горизонте забрезжила перспектива больших и быстрых денег, как от Си++ один за другим стали отворачиваться те, кто еще недавно поклонялся своему идолу. Java и сценарные языки серьезно пошатнули его престол и, кажется, вот-вот свергнут ненавистного конкурента. Того хуже — они поставили (по крайней мере, так видится со стороны) под сомнение сам факт дальнейшего существования Си++.

Ситуация сложилась весьма запутанная и напоминает смутное время. Судя по прессе, на Си++ давно уже пора ставить крест. Но при этом как отрицать тот факт, что практически все используемые в мире ключевые средства, в том числе компиляторы, операционные системы, СУБД, системы телекоммуникаций написаны на Си++ и тенденции к изменению этой ситуации пока не наблюдается. Приведу лишь несколько примеров:

- практически все программные продукты Microsoft (Windows XP, Office XP, Internet Explorer, MS Exchange, MS SQL Server, Visual Studio, FrontPage, все игры);
- ведущие продукты Adobe Systems (Photoshop, Illustrator, Acrobat, InDesign);
- базовые компиляторы Sun (кодогенераторы для SPARC, компиляторы переднего плана для Си++ и Fortran 90, Java HotSpot);
- все компиляторы и оптимизаторы кода, созданные Intel;
- компиляторы и редакторы связей (включая платформу HP IA64), созданные Hewlett-Packard;
- операционная система IBM OS/400 для бизнес-компьютеров AS/400;
- графическая оболочка KDE для Linux (Gnome реализована на Си);
- многие компоненты Mac OS X.

В сфере военных систем (по крайней мере, тех, что находятся в ведении Министерства обороны США) наметился постепенный переход от Ады в сторону Си++. Не вызывает сомнений подавляющее превосходство Си++ в области встроенных систем и уж тем более в индустрии компьютерных игр (Doom III,

StarCraft, Diablo I/II, Warcraft III, World of Warcraft, Kings Quest, Civilization, Master of Orion III — перечислять можно очень долго). По данным агентства Venture Development Corporation (http://www.vdc-corp.com), играющего ведущую роль в области анализа рынка передовых технологий, наиболее активно в сфере встроенных систем используется язык Си++ (75,7%); далее идут Ассемблер (50%) и Си (50%), Java (25,7%) и Ада (6,8%), доля же других языков крайне ничтожна (1,4%). Не стоит забывать и о том, что на Си++ реализованы ведущие поисковые Web-системы и крупнейшие Web-порталы: Google, Altavista, Yahoo, Amazon, eBay, а также системы резервирования Amadeus — самого большого в Европе центра данных с онлайновым доступом (200 тыс. терминалов, 5 тыс. транзакций в секунду).

Взглянем на цифры. По данным IDC, к 2001 г. около 3 млн. человек рассматривало Си и Си++ в качестве основного языка программирования. Затем шли Visual Basic — 2,3 млн, 4GL (языки четвертого поколения) — 1,8 млн., Java — 1,2 млн. и Кобол — 1,1 млн. человек По заявлению представителей Microsoft, сделанному на конференции разработчиков Developers 2000 Conference, за два последних года прошедшего столетия в Северной Америке был зафиксирован заметный рост числа программистов, работающих на Си++: с 2,8 млн. до 3,6 млн. (в абсолютном выражении это составляет 800 тыс.). Из них от 70 до 80% используют Microsoft Foundation Classes (MFC), т.е. Си++ фактически стал языком платформы Windows.

В отношении преподавания есть разные примеры: американский MIT (Массачусетский технологический институт) делает ставку на Java. По мнению профессора Даниэля Джексона, Java вскоре вытеснит все языки, в особенности Си++. А швейцарский ETH (альма-матер Паскаля и Оберона), ведущий европейский университетский центр, который всегда достаточно скептически относился к Си++, наоборот, стал уделять преподаванию этого языка более серьезное внимание, и главным образом, таким его особенностям, как шаблонное (template programming) и обобщенное (generic programming) программирование. Для этой цели были приглашены весьма компетентные специалисты, в том числе Евгений Зуев из МГУ, консультант Microsoft Research и ведущий разработчик ISO-эталонного компилятора переднего плана для Си++.

Если говорить об общей ситуации в преподавании компьютерных наук, то, по оценке Gartner Group, в 2000 г. язык Java изучали в 87% университетов (в обязательном порядке — в 56%), доля Си++ была такой же — 87% университетов (однако в обязательных дисциплинах выше — 67%). Но в 2001 г. этот паритет был нарушен в пользу Java: 96% против 79%.

Причины очевидны: преподавать и осваивать Java несколько проще, к тому же востребованность таких знаний на рынке труда все время повышается. По данным английского агентства Bloor Research (2001 г.), анализ 40 тыс. заявок от различных компаний Великобритании на заполнение вакансий позволил выявить, что наибольший интерес проявлялся к владению Java — 37% (годом ранее этот показатель был вдвое меньше), доля идущего на втором месте Си++ составляла 25%. Связано это с изменением роли Java в создании корпоративного ПО, где критичны скорость разработки, простота миграции на разные платформы и стоимость сопровождения заказных систем. Как показывают исследования Cutter Consortium, к началу 2001 г. Java превосходил Си++ в качестве языка разработки систем для электронного бизнеса в отношении 51% к 37%.

В конце 2000 г. Gartner Group опросил 400 респондентов из крупных ИТ-компаний, связанных с разра-

Биография

Бьерн Страуструп (Bjarne Stroustrup) родился 30 декабря 1950 г. в городе Аархус (Дания). Как подчеркивает он сам, его имя и фамилию на многих языках произносят неверно. В действительности по-датски следует говорить Бьярне Строуструп. Однако, как известно, звучание иностранных имен собственных не должно в точности сохраняться в русском языке, так что нет серьезных оснований изменять сложившуюся традицию.

В 1975 г. Бьерн Страуструп закончил университет Аархуса, где получил степень магистра математики и компьютерных наук. Вслед за этим поступил в Вычислительную лабораторию (Computing Laboratory) Кембриджского университета (Великобритания). В 1979 г. там же защитил диссертацию, посвященную распределенным компьютерным системам, и получил степень доктора философии. В том же году вместе с женой Мариан и дочерью Аннемари переехал в шт. Нью-Джерси (США), в исследовательский центр Bell Labs (Computer Science Research Center of Bell Telephone Laboratories). В 1980 г. в ходе экспериментов с собственным диалектом языка Си, использующим заимствованные из языка Симула-67 средства объектно-ориентированного программирования, создал язык C with Classes, который в 1984 г. перерос в язык Си++.

Бьерн Страуструп — автор известных книг «The C++ Programming Language» (1986; перевод на русский язык см. [11]) и «The Design and Evolution of C++» (1994; перевод на русский язык см. [12]). Активный участник процесса стандартизации Си++ в рамках ANSI и ISO. Отмечен рядом международных наград, среди которых ACM Grace Murray Hopper Award (1993), входит в списки «12 лучших молодых ученых Америки» (1990, журнал *Fortune*) и «20 самых выдающихся людей в компьютерной индустрии за последние 20 лет» (1995, журнал *Byte*).

Сейчас доктор Страуструп в AT&T Bell Laboratories возглавляет департамент исследований в области промышленного программирования (Large-scale Programming Research). Семья Бьерна Страуструпа долгое время проживала в городке Мейерсвилль. Там и родился сын Николас. В последние годы семья обосновалась в Вотчунге, в непосредственной близости от Мюррей-Хилл (шт. Нью-Джерси), где расположен исследовательский центр AT&T Labs. Официальная страница Бьерна Страуструпа: http://www.research.att.com/~bs

боткой ПО. Доля Java оказалась выше: 80% против 68% у Си++. Правда, лидером был Visual Basic — 82%. Данные Microsoft за тот же период заметно отличаются (это связано с опросом иных компаний и другой методикой оценки долей): Visual Basic набрал 44%, Си++ получил 11% и Java — 6%.

Генеалогия языков, или Как Бьерн чуть не лишился своей шевелюры

Не секрет, что новые языки создаются, как правило, на основе идей и конструкций, апробированных другими языками. Но язык не коктейль, а автор его не бармен. Бессмысленно пытаться подбирать состав и долю ингредиентов. Работа архитектора языка скорее напоминает работу скульптора, который уверенными движениями отсекает все лишнее, чтобы в конце концов на свет появилось настоящее творение.

В отличие от зарождения Си, Паскаля и Java, история Си++, казалось бы, весьма ординарна и не окутана ореолом таинственности. Хотя на самом деле это далеко не так. Впрочем, судите сами.

Как известно, своим рождением Си++ обязан Си и Симуле-67: «Си++ проектировался с целью объединения средств Симулы по организации программ с эффективностью и гибкостью системного программирования на Си» [3]. От Си была взята форма (синтаксис и базовые конструкции), от Симулы — содержание (классы). Причина выбора Си вроде бы понятна — тогда это был основной язык системного программирования. А вот почему на проектирование Си++ столь большое влияние оказал язык Симула? Дело здесь не только в нескрываемой симпатии к этому языку со стороны Страуструпа и не только в том, что Симула появился в Скандинавии (правда, не в родной для Бьерна Дании, а в соседней Норвегии). Важным катализатором послужили работы (1978—1979, Кембридж) над экспериментальным компьютером Cambridge CAP, где самое непосредственное участие в ходе подготовки своей диссертации принимал Страуструп. Он писал эмулятор распределенной операционной системы для CAP (сама ОС для CAP была реализована на Алголе-68). Работы велись на языке Симула в рамках мэйнфрейма IBM 360/165. Именно в те годы при организации мультизадачности Страуструп по достоинству оценил простой и удобный механизм сопрограмм Симулы. Ему нравилось почти все, кроме одного: система получалась элегантной, но крайне неэффективной.

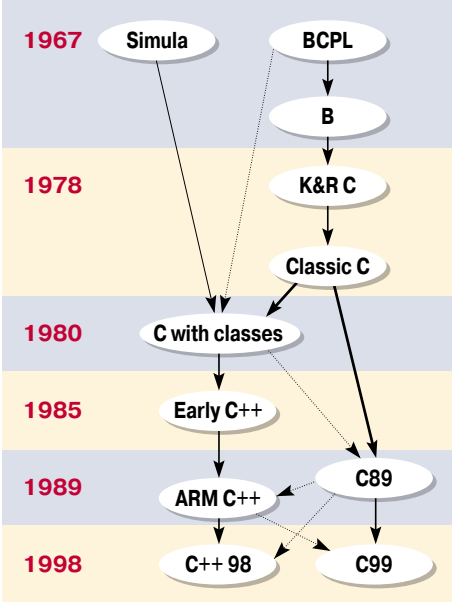

Рис. 1. Генеалогия языков Си-семейства

Simula **(Симула-67) — язык, созданный Кристианом Нигаардом (1967);**
BCPL **— язык, созданный Майком Ричардсом (1966);**
B **— язык, созданный автором UNIX Кеном Томпсоном (1969);**
K&R C **— версия языка Си в соответствии с канонической книгой Кернигана и Ритчи;**
C with Classes **— прототип Си++, играющий такую же роль, как NB для Си и Oak для Java;**
ARM C++ **— версия Си++ в соответствии с канонической книгой The Annotated C++ Reference Manual;** ***C89*** **— ISO-стандарт Си (1989);**
C99 **— ISO-стандарт Си (1999);**
C++98 **— ISO-стандарт Си++ (1998)**

Проверка типов на этапе выполнения, сборка мусора, поддержка параллельных процессов, гарантированная инициализация переменных — все эти особенности Симулы в данном проекте оборачивались с точки зрения эффективности против нее. Машинное время на мэйнфрейме было ограничено. До появления мощных рабочих станций оставалось еще без малого восемь лет. Диссертация находилась на грани срыва. Но с позором покидать престижный Кембридж очень уж не хотелось. Страуструп был вынужден переписать эмулятор на BCPL. Однако процесс кодирования и отладки оставил у него ужасное впечатление. «Это было крайне, крайне неприятно. Я потерял почти половину волос у себя на голове. Моя жена до сих пор пересказывает те ночные кошмары, которые мне чудились при отладке программы» [3]. Для полноты картины стоит добавить и то, что процесс подготовки диссертации совпал с появлением очаровательной малютки Аннемари. Вряд ли стоит говорить, что подобная ситуация наложила огромный отпечаток на всю последующую работу Страуструпа. И эффективность для будущего Си++ была задачей номер один.

Вот и ответ на вопрос о выборе Си. «Причина, почему я выбрал Си, была проста, — вспоминает Страуструп. — Это теперь думают, что связано все с тем, что Си был языком, рожденным в AT&T... Но в то время дело было в другом. Я считал Си языком, в наибольшей степени подходящим для того, что я хотел делать. Мне требовалось прибегать к системному программированию, необходим был доступ к низкоуровневым средствам, нужны были эффективность, гибкость, переносимость» [3].

Ясно, что недостаточно было просто объединить Си и Симулу. Требовались также другие идеи и механизмы. К тому же язык редко удается сразу. Сначала

вырисовываются лишь основные контуры, и только спустя месяцы и годы он начинает обретать законченный облик.

На рис. 1 приводится генеалогическое древо языков Си-семейства, данное самим Страуструпом. Стоит обратить внимание, что здесь остались «вне игры» Java и C#, хотя многие эксперты также относят их к данному семейству.

Связь языков столь замысловата, и авторы их так редко и неохотно признаются в каких-либо заимствованиях, что только спустя годы правда начинает всплывать наружу. В середине 1990-х годов Страуструп писал: «В то время я видел в качестве альтернатив для Си такие языки, как Modula-2, Ада, Smalltalk, Mesa и CLU, которые и рассматривал как источники идей для Си++. Однако в версии 1985 г. свой заметный след оставили Си, Симула, Алгол-68 и в одном случае BCPL. Симула дала классы, Алгол-68 — перегрузку операций, ссылки (references) и способность объявлять переменные в любом месте программного блока, а BCPL дал значок комментариев (//)... Эволюция Си++ после 1985 г. отражает воздействие идей языков Ада, CLU и ML» [3].

Решающим моментом в создании нового творения является личность самого автора. Она формируется в том числе и под влиянием первого изучаемого языка программирования. Для Денниса Ритчи (автора Си) первым языком был Кобол, для Джеймса Гослинга (Java) — Focal-5 (сценарный язык для PDP-8). «Моим первым языком программирования, — вспоминает Страуструп, — был Алгол-60 на компьютере GIER. Это был датский компьютер с 1 Кслов длиной 42 разряда» [4].

От того, что знает и умеет автор языка на момент пришедшего к нему озарения, зависит немало, по крайней мере целостность и выживаемость будущего творения. В программировании озарение — результат разочарований: в случае Си++, как и с Java, это был естественный процесс поиска выхода из критической ситуации. Нелишне вспомнить, что язык Java (Oak) родился

Эволюция языка Си++

Год, месяц	События
1979, октябрь	Препроцессор Cpre, добавлявший Симула-подобные классы в Си (реализован Страуструпом)
1980, апрель	В Bell Labs публикуется технический отчет по языку C with Classes
1982	Первое публичное описание прямого предшественника Си++. В журнале *ACM SIGPLAN Notices* появилась статья Страуструпа «Adding Classes to the C Language: An Exercise in Language Evolution»
1983	Язык C with Classes переименовывается в C84
1983	Реализация Cfront, компилятора переднего плана для C84
1983, август	В AT&T начинает использоваться первая версия языка Си++
1983, декабрь	Рик Маскитти предложил для нового языка название «C++»
1984, январь	Первое официальное описание языка Си++: «The C++ Reference Manual» (AT&T Bell Labs Computer Science Technical Report #108)
1986	Первое издание книги «The C++ Programming Language» (Addison-Wesley). C++ Release 1.0
1986, июнь	C++ Release 1.1
1987, февраль	C++ Release 1.2
1987, ноябрь	Международная конференция USENIX, посвященная Си++ (Санта-Фе, США)
1987, декабрь	Компилятор GNU C++ 1.13 (Майк Тайманн)
1988, январь	Компилятор TauMetric C++ (Oregon Software C++, Майк Болл)
1988, июнь	Первый компилятор для PC, отличный от Cfront (Zortech C++, Вальтер Брайт)
1989, июнь	C++ Release 2.0. Множественное наследование
1989, декабрь	В Вашингтоне прошло организационное собрание нового комитета X3J16 по созданию стандарта ANSI C++. Председателем избран Дмитрий Ленков. Усилия по стандартизации поддержали AT&T, HP, IBM, DEC
1990, май	Компилятор Borland C++
1990	ARM. Публикация технического руководства Эллиса и Страуструпа «The Annotated C++ Reference Manual»
1990, апрель	C++ Release 2.1
1990	В Hewlett-Packard Labs начались работы по расширению Си++ с помощью механизма шаблонов (STL)
1991, июнь	В Лунде (Швеция) прошло организационное собрание рабочей группы WG21 по созданию стандарта ISO C++. Усилия по стандартизации поддержали представители семи стран: Канады, Дании, Франции, Японии, Швеции, Великобритании и США
1991	Второе издание книги «The C++ Programming Language» (Addison-Wesley)
1991, сентябрь	C++ Release 3.0. Шаблоны
1992, февраль	Компилятор DEC C++
1992, март	Компилятор Microsoft C++
1992, май	Компилятор IBM C++
1994	Александр Степанов и Менг Ли в Hewlett-Packard Labs публикуют технический отчет по STL: «The Standard Template Language»
1994	Книга «The Design and Evolution of C++» (Addison-Wesley)
1997	Третье издание книги «The C++ Programming Language» (Addison-Wesley)
1998	Международный стандарт C++98 (ISO/IEC 14882—1998)
2000	Специальное издание книги «The C++ Programming Language» (Addison-Wesley)

в ходе неудачных экспериментов Гослинга по реализации компактного интерпретатора Си++ под давлением жестких сроков, установленных для вполне конкретного проекта разработки миниатюрного устройства.

Как создавался Си++

В апреле 1979 г. Страуструп после того, как успешно защитил диссертацию, вместе со своей семьей покинул Кембридж, чтобы приступить к работе в знаменитом исследовательском центре корпорации AT&T (Computer Science Research Center of Bell Telephone Laboratories) в Мюррей-Хилл (шт. Нью-Джерси, США). Там он занялся изучением ядра UNIX на предмет совершенствования коммуникационных средств и реализации распределенных сред в рамках локальных сетей. Ему потребовался хороший инструмент для анализа сетевого трафика и поиска разумного подхода к модуляризации ядра UNIX. Страуструп впоследствии писал: «Мой опыт в создании операционных систем и мой интерес к модуляризации и обмену данными оказали существенное воздействие на Си++» [3]. Интересное наблюдение: если Ритчи начинал с языка программирования и затем сконцентрировался на операционных системах, то Страуструп пошел по обратному пути.

Уже в октябре 1979 г. (см. таблицу) Бьерн написал препроцессор Cpre, позволяющий использовать Симула-подобные классы в Си. Этот препроцессор оказался весьма полезным инструментом и был задействован в полутора десятках проектов внутри AT&T. Именно с апреля по октябрь 1979 г. и были заложены идеи, воплощенные в том языке, который поддерживал новый препроцессор. Этот язык Страуструп назвал «C with Classes». Первоначально он был реализован на той же самой платформе, что дала жизнь Си и UNIX, — на знаменитом компьютере DEC PDP-11. Вскоре препроцессор был перенесен на VAX и компьютеры на базе процессора Motorola 68 000.

Основу реализации Си++ составлял Cfront — компилятор переднего плана (front-end) для языка C84 (так стал называться C with Classes). Он проектировался и реализовывался Страуструпом с весны 1982 г. до лета 1983 г. на PDP-11. В тот же самый период, когда Бьерн проектировал C84, он реализовывал на C with Classes библиотеку complex (вместе с Леони Роуз) и первый класс string (вместе с Джонатаном Шопиро).

Cfront представлял собой традиционный компилятор переднего плана, производивший полную проверку синтаксиса и семантики языка, построение, анализ и оптимизацию внутреннего представления программы, а также вывод всей необходимой информации для последующей генерации объектного кода. Cfront первоначально был создан на C with Classes и потому очень скоро стал первым компилятором Си++, написанным на самом Си++. Уже в первой версии Cfront активно использовался механизм классов, хотя и без виртуальных функций.

Наиболее необычным тогда было то, что Cfront генерировал код на Си. Страуструп рассматривал Си как переносимый ассемблер. Впоследствии подобный подход завоевал популярность, и для многих других языков (Ада, Smalltalk, CLOS, Modula-3, Eiffel) стали активно создаваться компиляторы переднего плана с генерированием кода на Си.

Технологически цепочка подготовки объектного кода выглядела так: сначала исходный текст обрабатывался Cpp (препроцессором Си), затем полученный результат транслировался компилятором Cfront, после чего в дело вступал компилятор Си.

Но название «C with Classes» оказалось неудачным. «Помимо Симулы-67 моим любимым языком в те годы был Алгол-68, — рассказывает Страуструп. — Я думаю, что название «Algol68 with Classes» точнее бы отражало суть языка, чем «C with Classes». Однако оно было бы мертворожденным» [4]. Ни «C with Classes», ни «C84» не устраивали Страуструпа, поскольку пользователи говорили просто о Си. В конце концов, в декабре 1983 г. Рик Маскитти (AT&T) нашел выход из этого положения — он предложил подчеркнуть преемственность нового языка, использовав конструкцию инкрементирования. Так появилось нынешнее название Си++.

Разумеется, с годами язык не только менял названия. Он совершенствовался. В версию C with Classes образца 1980 г. вошли такие средства, как классы и производные классы, управление доступом к полям и методам классов, механизм конструкторов и деструкторов, проверка типов и преобразование аргументов функций. В 1981 г. к ним добавились аргументы по умолчанию, функции подстановки (inline), перегрузка присваивания. В 1983 г., непосредственно перед переименованием C with Classes в Си++, в язык были включены виртуальные функции, перегрузка имен функций и операций, ссылки (в дополнение к указателям Си), константы (const), управление памятью (new и delete), усовершенствованный механизм проверки типов. При оттачивании идей языка C with Classes большую помощь оказали Деннис Ритчи, Стив Джонсон и Дуг Макилрой, коллеги Страуструпа по AT&T.

Так уж повелось, что версии языка Си++ совпадают с номерами релизов компилятора Cfront. В первом издании книги «The C++ Programming Language» (1986) был зафиксирован C++ Release 1.0 . Затем друг за другом появились Release 1.1 (июнь, 1986) и Release 1.2 (февраль, 1987), где были исправлены основные ошибки Cfront и неточности Си++. Два года спустя (июнь, 1989) вышел Release 2.0, и в него было введе-

но множественное наследование, а в сентябре 1991 г. был выпущен и Release 3.0 вместе с механизмом шаблонов, закрепленным в руководстве ARM (1990). В реализации компилятора приняли участие многие коллеги Страуструпа по AT&T, среди них Стив Дьюхарст, Лаура Ивз, Стэн Липпман, Джордж Логотезис, Джуди Уорд, Нэнси Уилкинсон, Барбара Му, Эндрю Кениг. При создании двух важнейших механизмов — шаблонов (1989) и исключений (1992) — помогали и специалисты других фирм (Object Design и Hewlett-Packard соответственно). Кроме того, ведущие компании (HP, Sun, Apple, ParcPlace, Saber) работали над собственными версиями Cfront.

В конце 1983 г. отделение Bell Labs, занимавшееся разработкой и поддержкой UNIX, выказало интерес к Си++ и выделило ресурсы для создания инструментальных средств для этого языка. Однако этот интерес начал довольно быстро угасать, едва стало ясно, что практически нереально обеспечить 100%-ную совместимость Си и Си++.

Осенью 1983 г. Эл Ахо предложил Страуструпу написать книгу по Си++, взяв за основу структуру справочника Кернигана и Ритчи по языку Си. В 1986 г. благодаря помощи Денниса Ритчи и Брайана Кернигана в издательстве Addison-Wesley была выпущена книга «The C++ Programming Language». За прошедшие годы она пережила четыре издания, вышла на 14 языках, а ее общий тираж превзошел 1 млн. экземпляров.

Ниже приведена динамика распространения Си++ (по данным Б. Страуструпа).

К 1989 г. число пользователей достигло отметки 50 тыс. Язык миновал начальный этап, и созрели все предпосылки для выхода новой версии C++ Release 2.0. В нее были добавлены множественное наследование, редактирование связей с контролем типов, рекурсивное объявление инициализации и присваивания, усовершенствованные средства для управления памятью, абстрактные классы.

Помимо разработки компиляторов для языка было важно создание библиотек. Пионерами здесь были библиотеки NIH (Кейт Горлен, Smalltalk-подобный набор классов), Interviews (Марк Линтон, использование X Window), G++ (GNU C++, Дуг Ли), а также библиотеки компаний Rogue Wave и Dyad (преимущественно для научных расчетов). Фирма Rational начала поставки The Booch Components (адаптация для Си++ Ада-библиотек, написанных Греди Бучем и Майком Вилотом). Но дальше рост интереса замедлился. Не последнюю роль в этом сыграло прохладное отношении к Си++ со стороны корпорации AT&T, из недр которой язык и вышел. Страуструп вспоминает: «Пропаганда Си++ имела очевидный успех там, где это сначала было важнее всего, — языком стали пользоваться миллионы программистов. Поразительно, но факт: подобный результат был достигнут без организации и практически без ресурсов. Возможно, именно поэтому сообщество пользователей Си++ получилось таким самодовольным, явно разобщенным и подверженным враждебной пропаганде» [5].

Помимо сложности освоения языка немалую роль в крушении его позиций вызвали несовместимость с Си (при том, что Си++ позиционировался как органичное расширение Си), отсутствие хорошей стандартной библиотеки (Страуструп считает это самой серьезной ошибкой), использование множественного наследования (Java и C# от этого отказались) и отказ от штатного механизма сборки мусора (этот механизм для Си++ был достаточно эффективно реализован разными компаниями и организациями).

Парадигмы, исключения, шаблоны и обобщенное программирование

Язык Си++, как и большинство других универсальных языков программирования, гибридный. Он поддерживает разные стили и модели программирования, или, как сейчас уже стало модным говорить, разные парадигмы. «Выводы, к которым я пришел (около 1980 г.), сводились к тому, что язык программирования общего назначения должен поддерживать несколько парадигм, — высказывал свою точку зрения Страуструп, — причем одинаково качественно и с близкими к оптимальным показателями эффективности по памяти и по времени выполнения. Тем самым, на мой взгляд, принятие новых идей существенно замедляется из-за консерватизма, питательную почву для которого поставляют мифы о сложности и высоких накладных расходах» [5].

Си++ не только объединил в себе идеи системного, структурного и объектно-ориентированного программирования (ООП). Когда поутихли страсти вокруг ООП и развеялся дым маркетинговых сражений, то стало ясно, что у него есть и другие плюсы, другие парадигмы, о которых нередко забывают в практической деятельности. «Я считаю шаблоны и обобщенное программирование центральными концепциями в современном Си++, — отмечает Страуструп. — Они служат ключом к крайне эффективному и надежному коду (с

Динамика распространения Си++ (по данным Б. Страуструпа)

Год	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991
Число пользователей	1	16	38	85	180	330	500	2000	4000	15 000	50 000	150 000	400 000

точки зрения контроля типов), однако не вписываются в классическую объектно-ориентированную парадигму»[6].

Шаблоны (templates) пришли на смену макросам, весьма активно применяемым в языке Си. Макросы из-за неконтролируемой и лавинообразной подстановки строк крайне опасны и подчас даже вредны при разработке и сопровождении программ. Шаблоны вносят в этот хаос заметный порядок. Одним из признанных достижений в программировании является стандартная библиотека шаблонов Си++ (STL, Standard Template Library), немало взявшая из функционального программирования. В период 1990—1994 гг. в исследовательском центре Hewlett-Packard (HP Labs) ее создавал Алексей Степанов. Страуструп вспоминает: «Мы с Алексеем потратили немало времени на обсуждение языковых средств, прежде чем шаблоны стали частью Си++. На мой взгляд, он находился под сильным влиянием своего опыта работы с языком Ада. В то же время, в отличие от меня, он обладал ценными практическими навыками обобщенного программирования... Нам обоим не нравились макросы, мы были сторонниками строгого контроля типов» [6].

Обобщенное, или родовое, программирование (generic programming), хорошо известное специалистам по таким языкам, как Ада и Modula-3, дает в руки программистам очень важный инструмент параметризации программного текста. Однако время его признания, похоже, еще впереди.

Думается, в разговоре о парадигмах стоит отметить и исключения (exceptions). Нередко их воспринимают просто как средство контроля и реакции на ошибки исполнения программ. И хотя термин exception-oriented programming не встречается в публикациях, исключения — это не просто механизм. Подобно событийно-ориентированному программированию они могут активно использоваться и в «мирных целях» как удобные средства программных прерываний (это особенно актуально при мультипрограммировании).

Механизм обработки исключений для Си++ проектировался самим Страуструпом в 1984—1989 гг. (определенную помощь в этом ему оказал Энди Кениг). При обсуждении особенностей его работы Страуструп провел немало встреч в Apple, DEC, Microsoft, Sun, IBM. «Я находился под воздействием механизма языка CLU, — признается Страуструп, — немного взял из Ады и посмотрел, как это было сделано в Modula-2+, а также в ML для использования типов» [3]. От принятого имени raise ему пришлось отказаться, поскольку такое название носила стандартная функция в библиотеке Си. Страуструп остановил свой выбор на catch и throw, позаимствовав эти имена из Лиспа.

Итак, программирование со множеством парадигм — отличительная черта Си++, заслуживающая поддержки и развития. Что же думает по этому поводу сам автор языка? «Думаю, что «парадигма» — это затасканное слово, — замечает Страуструп. — Я предпочитаю использовать более скромное словосочетание — стиль программирования... Не думаю, что Си++ будет когда-нибудь поддерживать новую парадигму, но, быть может, я чересчур консервативен в том, что называю парадигмой. Надеюсь, что грядущий стандарт Си++ будет поддерживать распределенное программирование, но это должно достигаться в основном с помощью стандартной библиотеки» [7].

Стандартизация языка

Нужно ли добиваться стандартизации языка? Вопрос непростой. С одной стороны, промышленное использование языка на разных платформах невозможно без закрепленных, стабильных спецификаций. В идеале они должны пройти горнило жестких обсуждений в национальных, региональных и международных комитетах по стандартизации (ANSI, ECMA, ISO). Но стандарт (не корпоративный) в значительной степени лишает владельца языка возможности самостоятельно решать, что в нем подлежит изменениям. По понятным причинам Microsoft не горит желанием стандартизировать Visual Basic, а Borland — Delphi. Корпорация Sun Microsystems также не осталась в стороне: она пересмотрела свою исходную позицию в отношении Java и отозвала заявку на стандартизацию. В то же время компромиссы при выработке стандартов могут убить язык, как это произошло, в частности, с Алголом и Modula-2.

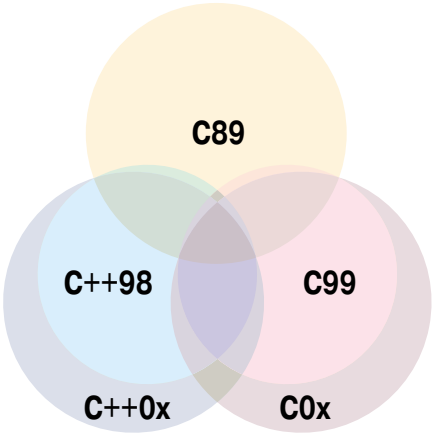

Рис. 2. Взаимосвязь разных версий Си и Си++

В случае Си++ ситуация сложилась едва ли не уникальная. Хотя не стоит забывать о том, что ISO-стандарт имеет также Си и уже вот-вот его должен получить C#. Между Си и Си++ наблюдается явный конфликт интересов: разные группы по стандартизации, разные цели и, как следствие, рассогласование этих языков, некогда бывших едва ли не сиамскими близнецами.

«Моим идеалом по-прежнему остается единый язык, и техническая возможность достаточно логично слить Си++ с C99 пока сохраняется. На мой взгляд, такой язык мог бы соответствовать любым разумным техническим условиям. Однако я не уверен, что это приемлемо политически», — с нескрываемым сожалением признает Страуструп [5].

На рис. 2 [8] показаны области пересечения разных версий Си (C89, C99, C0x) и Си++ (C++98, C++0x). Стандарты C0x и C++0x пока формируются в ISO (выпуск ISO-стандарта Си++ планируется на 2005 г.), поэтому вместо года здесь ставится «x».

Будущее языка Си++

Бьерну Страуструпу грядущее представляется не в таких уж черных тонах: «Я не вижу признаков того, что Си++ начинает устаревать» [9]. И отмечает также, что «в мире Microsoft .NET наблюдается заметный акцент на Си++: он по-прежнему остается здесь ведущим языком» [10]. Страуструп, находясь под впечатлением нового языка C# (Андерс Хейльсберг и др.), играющего роль связующего механизма на теперь уже стандартизированной платформе CLI в рамках Microsoft .NET, делает важный вывод о том, как будет развиваться Си++: «Нужны качественные средства для составления программ из отдельных частей, возможно написанных на разных языках».

Новая идеология распределенных Web-вычислений привела к изменению роли Си++. На передний план стали выдвигаться сценарные языки, Perl, Python, JavaScript. Как же реагирует на это Си++? «Никогда не существовало языка, подходящего для всех случаев жизни, и я сомневаюсь, что он когда-нибудь появится, — считает Страуструп. — Как только универсальный язык вроде Си++ начнет дополняться специализированными языками и инструментами, специализация будет приносить значительный успех... Я не уверен, что большинство программного кода будет Web-центричным... Для Web-среды и сетей вообще нам крайне нужна подлинная модель безопасности» [9].

Что бы нового ни приносил нам день завтрашний, но преемственность идей и инструментов все равно остается незыблемым принципом эволюции. «Я не слишком хорошо умею предсказывать, — признает автор Си++, — поэтому не стану и пытаться. Замечу только, что завтрашний день обычно больше похож на вчерашний, чем нам хотелось бы. Обратите внимание: Кобол, Фортран и Си до сих пор остаются в числе главных языков» [5]. Какие бы новые веяния ни сеяли смуту в душах программистов, потребность в эффективном коде была всегда и остается сейчас. Не случайно до сих пор живы и востребованы ассемблеры для разных процессоров, не случайно Си++, несмотря ни на что, по-прежнему служит фундаментом программной индустрии. Конечно, Страуструп не может не переживать за судьбу своего детища. Но трудно обвинять его в предвзятости приводимых им аргументов: «Си++ является наилучшим языком для многих приложений, где требуется системное программирование, имеются определенные ограничения по ресурсам и выдвигаются серьезные требования к производительности. Одним из примеров служит Google, другим — встроенные системы для миниатюрных устройств» [10].

Если говорить не только о прикладных областях, но и о фундаментальных исследованиях, то и здесь Си++ остается хорошим источником новых идей и технологий. «Я думаю, — говорит Страуструп, — что существует серьезный рост использования Си++ в области встроенных систем... Я знаю, что наметился интерес в области шаблонного метапрограммирования (template metaprogramming), обобщенного программирования (generic programming) и порождающего программирования (generative programming). Я ожидаю, что эти горячие темы будут будоражить первопроходцев и ученых и что некоторые — но не все — новые подходы будут задавать магистральное направление развития на ближайшие несколько лет» [10]. ■

Литература

1. Богатырев Р. Летопись языков. Паскаль // Мир ПК. 2001. № 4.

2. Богатырев Р. Летопись языков. Си // Мир ПК. 2001. № 8.

3. Stroustrup B. A History of C++: 1979-1991 // History of Programming Languages. ACM-Press, 1996.

4. The C Family of Languages: Interview with Dennis Ritchie, Bjarne Stroustrup, and James Gosling // C++ Report, 12(7), July/August, 2000.

5. Калев Д. Будущее по Бьерну Страуструпу // Мир ПК. 2001. № 5.

6. Interview with Bjarne Stroustrup // C++ View. China. August, 2001.

7. Tran P. Interview with Bjarne Stroustrup // Developpeur Reference. France. March, 2002.

8. Stroustrup B. C and C++: Siblings // The C/C++ Users Journal. July, 2002.

9. Nelson E. Interview with Bjarne Stroustrup // Visual C++ Developers Journal. Vol. 3. № 5. June, 2000.

10. A Conversation with Bjarne Stroustrup // Rogue Wave User Conference. July, 2002.

11. Страуструп Б. Язык программирования C++: Спец. изд.: Пер. с англ. М.; СПб.: Бином: Невский диалект, 2001.

12. Страуструп Б. Дизайн и эволюция языка C++. ДМК, 2000.