Краткая история Паскаля

Алгол

Язык Компонентный Паскаль является кульминацией нескольких десятилетий исследовательской работы. Это самый младший член семейства алголоподобных языков. Алгол, определенный в 1960, был первым языком высокого уровня с синтаксисом, который был легко читаем, четко структурирован и описан формальным образом. Несмотря на его успешное использование в качестве нотации для математичесих алгоритмов, в нем недоставало важных типов данный, таких как указатели и литеры.

Паскаль

В конце 60-х гг. было выдвинуто несколько предложений об эволюционном развитии Алгола. Самым успешным оказался Паскаль, определенный в 1970 г. профессором Никлаусом Виртом из ETH, швейцарского Федерального Технологического Института в Цюрихе [Eidgenossische Technische Hochschule]. Наряду с очищением языка от некоторых непрозрачных средств Алгола, в Паскале была добавлена возможность объявления новых структур данных, построенных из уже существующих более простых. Паскаль также поддерживал динамические структуры данных, т.е. такие, которые могут расти или уменьшаться во время выполнения программы. Паскаль получил сильный импульс к распространению, когда в ETH был выпущен компилятор, порождавший простой промежуточный код для виртуальной машины (P-код) вместо кода для конкретного процессора. Это существенно упростило перенос Паскаля на другие процессорные архитектуры, т.к. для этого нужно было только написать новый интерпретатор для P-кода вместо всего нового компилятора. Один из таких проектов был предпринят в Университете Калифорнии в Сан-Диего. Замечательно, что эта реализация (UCSD Pascal) не требовала большого компьютера [mainframe] и могла работать на новых тогда персональных компьютерах Apple II. Это дало распространению Паскаля второй важный импульс. Третьим был выпуск компанией Borland продукта ТурбоПаскаль, содержавшего быстрый и недорогой компилятор вместе с интегрированной средой разработки программ для компьютеров IBM PC. Позднее Борланд возродил свою версию Паскаля, выпустив среду быстрой разработки приложений Дельфи.

Паскаль сильно повлиял на дизайн и эволюцию многих других языков, от Ады до Visual Basic.

Модула-2

В середине 70-х гг., вдохновленный годичным академическим отпуском, проведенным в исследовательском центре PARC компании Xerox в Пало Альто, Вирт начал проект по созданию нового компьютера класса рабочая станция <*проект Lilith; см. Краткая история Модулы и Лилит (на англ.) — прим. перев.*>. Компьютер должен был полностью программироваться на языке высокого уровня, так что язык должен был обеспечить прямой доступ к аппаратному уровню. Далее, он должен был поддерживать коллективное программирование и современные принципы разработки программного обеспечения, такие как абстрактные типы данных. Эти требования были реализованы в языке программирования Модула-2 (1979). Модула-2 сохранила хорошо зарекомендовавшие себя средства Паскаля и добавила систему модулей, а также контролируемые возможности обойти систему типов языка для целей программирования низкого уровня (например, при написании драйверов). Модули могли добавляться к операционной системе непосредственно во время работы. На самом деле вся операционная система представляла собой набор модулей без выделенного ядра [kernel] или подобного объекта. Модули могли компилироваться и загружаться раздельно, причем обеспечивалась полная проверка типов и версий их интерфейсов.

Успех Модулы-2 был наиболее значителен в задачах с высокими требованиями на надежность, таких как системы управления движением.

Simula, Smalltalk и Cedar

Однако Вирт продолжал интересоваться прежде всего настольными компьютерами, и опять важный импульс пришел из центра PARC компании Xerox. В этом центре были изобретены рабочая станция, лазерный принтер, локальная сеть, графический дисплей и многие другие технологии, расширяющие возможности использования компьютеров человеком. Кроме того, в центре PARC были популяризированы некоторые более старые и малоизвестные технологии, такие как мышь, интерактивная графика и, наконец, объектно ориентированное программирование. Эта последняя концепция (хотя и не сам термин) была впервые использована в языке высокого уровня Simula (1966) — еще одним из семейства алголоподобных языков. Как и предполагает имя, язык Simula использовал объектные технологии прежде всего для целей моделирования [simulation]. Однако язык Smalltalk (1983), разработанный в центре PARC компании Xerox, использовал объектные технологии как универсальное средство. Проект Smalltalk был также пионерским в плане дизайна пользовательского интерфейса: графический пользовательский интерфейс, каким мы его теперь знаем, был разработан для системы Smalltalk.

В центре PARC эти идеи повлияли на другие проекты, например, паскалеподобный язык Cedar. Как и Smalltalk и позднее Оберон, Cedar представлял собой не только язык программирования, но и операционную систему. Операционная система Cedar была весьма впечатляющей и мощной, однако сложной и нестабильной.

Оберон [Oberon]

Проект Оберон был начат в 1985 в ETH Виртом и его коллегой Юргом Гуткнехтом [Jurg Gutknecht]. Это была попытка выделить все существенное из системы Cedar в виде универсальной, но все же обозримой операционной системы для рабочих станций. Получившаяся система оказалась очень маленькой и эффективной, прекрасно работала в оперативной памяти размером всего 2 MB и требовала при этом лишь 10 MB пространства на диске. Важной причиной малого размера системы Оберон был ее компонентный дизайн: вместо интеграции всех желаемых средств в один монолитный программный колосс, менее часто используемые программные компоненты (модули) могли быть реализованы как расширение ядра системы. Такие компоненты загружались, только когда они были действительно нужны, и они могли совместно использоваться всеми приложениями.

Вирт понял, что компонентно-ориентированне программирование требовало некоторых средств объектно-ориентированного программирования, таких как упрятывание информации [information hiding], позднее связывание [late binding] и полиморфизм [polymorphism].

Упрятывание информации было сильной чертой Модулы-2. Позднее связывание поддерживалось в Модуле-2 посредством процедурных переменных. Однако там не было полиморфизма. По этой причине Вирт добавил расширенное переопределение типов [type extension]: тип записей мог быть объявлен как расширение <*потомок*> другого типа записей <*предка*>. Тип-потомок можно было использовать всюду вместо его предков.

Но компонентно-ориентированное программирование выходит за рамки объектно-ориентированного. В системе, построенной из компонент, компонента может разделять свои структуры данных с произвольным числом других компонент, о которых она ничего не знает. Эти компоненты обычно также не знают о существовании друг друга. Такое взаимное незнание делает управление динамическими структурами данных, и в частности правильное освобождение уже ненужной памяти, принципиально более трудной проблемой, чем в закрытых программных системах. Следовательно, необходимо оставить на долю реализации языка всю работу по определению момента, когда какая-то область памяти более не нужна, чтобы повторно использовать ее без ущерба для безопасности системы. Системный сервис, выполняющий такую автоматическую утилизацию памяти, называется сборщик мусора [garbage collector]. Сбор мусора предотвращает две из числа наиболее труднонаходимых и попросту опасных ошибок в программах: утечки памяти [memory leaks] (когда более не используемая память не освобождается) и висячие ссылки [dangling pointers] (преждевременное освобождение памяти). Висячие ссылки позволяют одной компоненте разрушить структуры данных, принадлежащие другим. Такое нарушение защиты по типам [type safety] должно быть предотвращено, т.к. компонентные системы могут содержать много независимо написанных компонент неизвестного качества (например, полученных из Интернета).

Хотя алголоподобные языки всегда имели высокую репутацию в отношении безопасности, введение полной защиты типов (и, следовательно, сбора мусора) было квантовым прыжком. Именно по этой причине полная совместимость с Модулой-2 оказалась невозможной. Получившаяся модификация Модулы-2 была названа как и вся система — Оберон.

Система модулей в Обероне, как и в Модуле-2, обеспечивала упрятывание информации для целых семейств типов, а не только для отдельных объектов. Это позволило определять и гарантировать инварианты для нескольких взаимодействующих объектов. Другими словами, разработчики получили возможность разрабатывать механизмы защиты более высокого уровня, отталкиваясь от базовых средств защиты на уровне модулей [module safety] и защиты по типам, обеспечиваемых хорошей реализацией Оберона.

Такие ортодоксальные объектно-ориентированные языка, как Smalltalk, пренебрегали как типизацией переменных (там вообще нет понятия тип переменной), так и упрятыванием информации (ограничивая ее объектами и классами), что было большим шагом назад в технологии программирования. Оберон примирил миры объектно-ориентированного и модульного программирования.

Последнее требование компонентно-ориентированного программирования — возможность динамически загружать новые компоненты. В Обероне единица загрузки та же, что и единица компиляции — модуль.

Компонентный Паскаль

В 1992 г. сотрудничество с профессором Х.П. Мёссенбёком (H.P. Mo:ssenbo:ck) привело к нескольким добавлениям к первоначальному языку Оберон ("Оберон-2"). Так возник фактический стандарт языка.

В 1997 г. компания Oberon microsystems, Inc., отпочковавшаяся [spin-off] от ETH (с Виртом в составе совета директоров), сделала некоторые небольшие добавления к Оберону-2 и назвала его Компонентный Паскаль, чтобы четче выразить как его нацеленность (компонентно-ориентированное программирование), так и его происхождение (Паскаль). Это промышленная версия Оберона, являющаяся наследницей первоначального Паскаля и Модулы-2.

Главная идея уточнений по сравнению с Обероном-2 была в том, чтобы дать проектировщику компонентного каркаса [component framework] (т.е. интерфейсов модулей, определяющих абстрактные классы для конкретной проблемной области) более полный контроль над ее проектируемыми свойствами в плане безопасности. Положительным результатом стало то, что теперь легче обеспечить целостность больших компонентных систем, что особенно важно во время итеративных циклов проектирования, когда библитека разрабатывается, и позднее, когда архитектура системы должна быть переработана, чтобы обеспечить дальнейшую эволюцию и поддержку.

BlackBox

Компания Oberon microsystems разрабатывала компонентную библиотеку BlackBox Component Framework начиная с 1992 г. (сначала библиотека называлась Oberon/F). Эта библиотека написана на Компонентном Паскале и упрощает разработку компонент графического пользовательского интерфейса. Она поставляется с несколькими компонентами, включая текстовый редактор, систему визуального проектирования, средство доступа к базам данных SQL, интегрированную среду разработки, а также систему поддержки выполнения программ на Компонентном Паскале. Весь пакет представляет собой развитый, но весьма нетребовательный к системным ресурсам инструмент быстрой разработки компонентных приложений, названный BlackBox Component Builder. Он нетребователен к системным ресурсам, т.к. полностью построен из модулей Компонентного Паскаля — включая ядро со сборщиком мусора, а также самого компилятора для языка Компонентный Паскаль. Это — иллюстрация как мощи концепции компонентного программного обеспечения вообще, так и адекватности языка Компонентный Паскаль в частности.

Недавно диапазон приложений системы BlackBox Component Builder был значительно расширен за счет среды кросс-программирования *<т.е. программирование для процессоров, отличных от того, на котором работает система; обычно это специализированные процессоры для встроенных систем — прим. перев.>* Denia, которая является компонентой, расширающей BlackBox. Denia позволяет выполнять кросс-программирование на Компонентном Паскале для новой операционной системы реального времени JBed, которая тоже полностью реализована на Компонентном Паскале. JBed предназначен для встроенных систем и приложений с жесткими требованиями реального времени [hard real-time requirements], например, в робототехнике и промышленной автоматизации. *<О сложности этого класса приложений свидетельствует тот факт, что компания Майкрософт отказалась от продвижения своих операционных систем в этом сегменте рынка — прим. перев.>*