**Развитие языков, методов и технологий программирования**

**Предисловие**

Из ранее изученных курсов известно, что языки программирования делят на 5 поколений.

Первое поколение - машинные языки, языки программирования на уровне команд процессора конкретной машины. Они не имели транслятора, а выполнялись как очередь команд. Позже сюда стали относить и языки ассемблера.

Второе поколение – языки требовавшие преобразования в машинный код транслятором или компилятором. Это такие языки как Фортран и Алгол. Программы на этих языках были аппаратно зависимыми и не были переносимыми.

Третье поколение – аппаратно независимые и универсальные языки, такие как Си, Паскаль. Аппаратная независимость была достигнута благодаря интерпретатору, который в свою очередь, создан под определенную платформу.

Четвертое поколение – языки ориентированные на специализированные области применения и большие проекты, и используют не универсальные, а проблемно-ориентированные языки. Это значит появление ООП и сред разработки.

Пятое поколение – системы автоматического создания программ и автоматное (визуальное) программирование без специальной подготовки.

**Начало**

В пятидесятые годы двадцатого века с появлением компьютеров на электронных лампах началось бурное развитие языков программирования. Изначально использовались низкоуровневые языки программирования. В середине 50-х годов был разработан алгоритмический язык программирования FORTRAN для фирмы IBM. Язык FORTRAN (FORmula TRANslator) – первый язык программирования высокого уровня, имеющий транслятор. Он предоставляет возможность записи алгоритма вычислений с использованием условных операторов и операторов ввода/вывода, это стало точкой отсчета эры алгоритмических языков программирования.

К языку FORTRAN предъявлялись требования cоздания высокоэффективного кода. Поэтому многие конструкции языка первоначально разрабатывались с учетом архитектуры IBM 407. Успех разработки этого языка привел к тому, что производители других вычислительных систем стали создавать свои версии трансляторов. Для этого языка были разработаны первые стандарты.

Как альтернатива языку FORTRAN, первоначально ориентированному на архитектуру IBM, в конце 50-х годов был разработан язык ALGOL (ALGOrithmic Language). Основной целью, преследуемой разработчиками этого языка, была независимость от конкретной архитектуры вычислительной системы. Кроме того, создатели языка ALGOL стремились разработать язык, удобный для описания алгоритмов и применяющий систему обозначений, близкую к той, что принята в математике.

Языки FORTRAN и ALGOL были первыми языками, ориентированными на программирование вычислений.

**Немного о древностях в нынешнее время**

Фортран исторически использовался в инженерных или научных расчетах, а потому со временем обрастал множеством готовых библиотек и кодами программ решения той или иной задачи.

Код в буквальном смысле передается из поколения в поколение, да еще и хорошо документируется. Можно найти множество готовых решений уравнений математической физики, линейной алгебры (здесь следует отметить удачную реализацию работы с матрицами), интегральных и дифференциальных уравнений и многого-многого другого. Наверное, тяжело найти задачу из области физмат наук, для которой не был бы реализован алгоритм на языке Фортран. А если учесть отличную оптимизацию интеловских компиляторов под интеловские же процессоры, поддержку параллельных вычислений на высокопроизводительных кластерах, то становится понятно, почему в научной среде этот язык занимает заслуженное первое место.

Большинство серьезных моделей, по крайней мере, из области физики атмосферы, написаны именно на Фортране. Например, прогноз погоды, которым каждый интересуется время от времени, получается в ходе расчетов моделей, написанных на этом языке. Более того, язык не находится в стагнации, а постоянно совершенствуется. Так, после стандартов 77 и 90, появились новые редакции 95, 2003, 2008, поддержка которых внедрена в актуальные компиляторы. Последние версии Фортрана несколько освежили старый проверенный временем язык, превнеся поддержку современного стиля, добавив объектно-ориентированное программирование, отсутствие которого было чуть ли не самым главным козырем противников этого языка. К тому же избавили его от наследия ассмблера – избытка GOTO. Более того, The Portland Group выпустила PGI CUDA Fortran Compiler, позволяющий проводить высокопараллельные расчеты на видеокартах. Таким образом, пациент более чем жив.

А вот алголу не так повезло, он стал прямым коммерческим конкурентом IBM. В то время IBM правила рынком и не дала распространиться ему в США. Однако, в Европе алгол быстро завоевал популярность в академической среде, повсеместно шла разработка компиляторов, многие из которых, несмотря на сложности реализации, оказались весьма успешными. Алгол распространился от Великобритании до Дальнего востока СССР, став как универсальным языком описания алгоритмов в научных публикациях, так и средством реального программирования.

Даже когда язык Алгол почти перестал использоваться для программирования, он ещё долго оставался официальным языком для публикации алгоритмов.

**Печать специализации**

Когда языки программирования стали жить не только для физтеха, но и для бизнеса и госучреждений, начали появляться такие монстры как специализированный САПР, выполняющий одну задачу для одного завода, бизнес-приложения узкой специализации, всевозможные электронные картотеки. В этой сфере также закрепились свои языки, предназначаемые для обработки деловой информации, самым популярным языком бизнес приложений начала 60-х стал COBOL, в 90-е годы его популярность достигла пика. COBOL имел прекрасные для своего времени средства для работы со структурами данных и файлами, он стал первым языком, в котором появился тип данных «запись».Это обеспечило ему долгую жизнь в бизнес-приложениях, по крайней мере, в США.

В пресс-релизе, на веб-сайте Micro Focus (ну очень большая фирма, предоставляет ПО и консультации бизнес-клиентам по обновлению унаследованных систем), Кевин Брерли, отвечающий за управление программными продуктами в компании Micro Focus, сказал:

«В настоящее время COBOL поддерживает 90% бизнес-систем, используемых корпорациями из списка Fortune 500 и используется в 85% всех ежедневных финансовых (бизнес) транзакций. Отсюда вывод — существует востребованность программистов на языке COBOL. Без них, риски для бизнеса, использующего COBOL, будут повышаться.»

В бизнес-системах сложно взять и все сразу переписать, потому приходится очень долго поддерживать старое. Тут можно вспомнить болезненный переход с СУБД на FoxPro.

Но также резонно заметить, что имеет место такой подход как создание оберток, но речь не об этом.

**Венец творения**

Самой заметной вехой в развитиии языков программирования можно считать разработку языка C. Универсальный язык программирования С был разработан к середине 70-х годов Денисом Ритчи и Кеном Томпсоном. Этот язык стал популярным языком системного программирования и в свое время использовался для написания ядра операционной системы UNIX. Это сделало ОС переносимой на другие платформы.

В 1978 году Брайан Керниган и Деннис Ритчи опубликовали первую редакцию книги «Язык программирования Си». Эта книга служила многие годы неформальной спецификацией языка. Вторая редакция этой книги посвящена более позднему стандарту ANSI C. Стандарт языка С начал разрабатываться рабочей группой института стандартов ANSI в 1982 году. Международный стандарт языка С принят в 1990 году. Язык С лег в основу разработки целого класса С-подобных языков программирования, таких как C++, C#, Java, PHP, Objective-C, Perl, JavaScript, Scala, Verilog, да тысячи их.

После освобождения от оков платформы появился большой простор для развития архитектуры программ.

**Методы программирования**

Методом или методикой (методологией) программирования следует считать совокупность способов, средств и технологий создания программ.

«Cтихийное» программирование. Пер­вые программы имели простейшую структуру. Они состояли из собственно программы на машинном языке и обрабатываемых ею данных.

Типичная программа того време­ни состояла из основной программы, области глобальных данных и набора подпрограмм (в основном библиотеч­ных), выполняющих обработку всех данных или их части. Слабым местом такой архитектуры было то, что при увеличении коли­чества подпрограмм возрастала вероятность искажения части глобальных данных какой-либо подпрограммой.

Cтруктурное программирование. В основе структурного подхода лежит декомпозиция сложных си­стем с целью последующей реализации в виде отдельных небольших подпрограмм.

Поддержка принципов структурного программирования была заложена в основу процедурных языков программирования.

Объектно-ориентированное программирование. ООП определяется как технология создания сложного программного обес­печения, основанная на представлении программы в виде совокупности объ­ектов, каждый из которых является экземпляром определенного клас­са, а классы образуют иерархию с наследованием свойств. Взаи­модействие программных объектов в такой системе осуществляется путем передачи сообщений.

Объектный подход предлагает способы организации про­грамм, основанные на механизмах наследования, полиморфизма, компози­ции, наполнения. Эти механизмы позволяют конструировать сложные объек­ты из сравнительно простых.

Компонентный подход и CASE-технологии. Компонентный подход предпо­лагает построение программного обеспечения из отдельных компонентов - физически отдельно существующих частей программного обеспечения, ко­торые взаимодействуют между собой через стандартизованные интерфейсы.

В отличие от обычных объектов объекты-компоненты можно собрать в динамически вызываемые библиотеки или исполняемые файлы, распространять в двоичном виде (без исходных текстов) и использовать в любом языке программирования, поддерживающем соответствующую тех­нологию.

**Технологии программирования**

Технология – это массовый способ производства какого-то продукта, гарантирующий устойчивый результат. На этом принципе основаны и все технологии программирования.

Основные известные технологии также были рассмотрены в предыдущих курсах. Выделим из них группы технологических подходов и укажем подходы для каждой из них.

1. Подходы со слабой формализацией.

Эти подходы не используют явных технологий и их можно применять только для очень маленьких проектов, как правило, завершающихся созданием демонстрационного прототипа. К подходам со слабой формализацией относятся так называемые ранние технологические подходы, например подход "кодирование и исправление".

1. Строгие подходы.

Данную группу подходов рекомендуется применять для средних, крупномасштабных и гигантских проектов с фиксированным объемом работ. Одно из основных требований к таким проектам - предсказуемость. В эту группу входят:

* Каскадные технологические подходы.
  + Классический каскадный подход.
  + Каскадно-возвратный подход.
  + Каскадно-итерационный подход.
  + Каскадный подход с перекрывающимися процессами.
  + Каскадный подход с подпроцессами.
  + Спиральная модель.
* Генетические подходы.
  + Синтезирующее программирование (синтезирующее программирование предполагает синтез программы по ее спецификации, документ на языке спецификаций является базисом для последующей реализации).
  + Сборочное (расширяемое) программирование.
  + Конкретизирующее программирование.
* Подходы на основе формальных преобразований.
  + Технология стерильного цеха (технология складывается из следующих частей:
* разработка функциональных и пользовательских спецификаций;
* инкрементальное планирование разработки;
* формальная верификация;
* статистическое тестирование).
  + Формальные генетические подходы (такие как:
* Формальное синтезирующее программирование, которое использует математическую спецификацию - совокупность логических формул.
* Формальное сборочное программирование использует спецификацию как композицию уже известных фрагментов.
* Формальное конкретизирующее программирование использует такие подходы, как смешанные вычисления и конкретизацию по аннотациям.)

### Гибкие (адаптивные, легкие) подходы

Подходы этой группы рекомендуется применять для небольших или средних проектов в случае неясных или изменяющихся требований к системе. Команда разработчиков должна быть ответственной и квалифицированной, а заказчики должны быть согласны принимать участие в разработке. В данную группу входят подходы, перечисленные ниже.

* Ранние технологические подходы быстрой разработки.
  + Эволюционное прототипирование.
  + Итеративная разработка.
  + Постадийная разработка.
* Адаптивные подходы.
  + Экстремальное программирование.
  + Адаптивная разработка.
* Подходы исследовательского программирования.
  + Компьютерный дарвинизм.