Министерство образования и науки РК
Казахский национальный университет им. аль- Фараби

Доклад

На тему:
ОБЪЕКТНО- ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА ADA

Выполнила: Жусипбекова З. ИС-6Б
Проверил: к.т.н. К.А. Хайдаров

Алматы 2014

Содержание

ОБЪЕКТНО- ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА ADA

Ада - мощнейший объектно-ориентированный язык общего назначения, ориентированный на разработку надежного программного обеспечения. В язык включены механизмы поддержки параллельного исполнения, обработки исключений, настраиваемых модулей, поддержки распределенных вычислений, стандартные интерфейсы с другими языками и библиотеками. Ада имеет компиляторы под практически любую операционную систему плюс Java байткод.
Ада это современный язык программирования, предназначенный для больших, долгосрочных приложений - и встраиваемых систем, в частности - где очень важными являются безопасность и сохранность. Идеология Ada обеспечивает легкость чтения, позволяет избегать ошибок, предусматривает многократное использование и коллективную разработку ПО.

Изначально Ada был создан для департамента обороны США (DoD) для встроенных систем реального времени. Ada - язык, наиболее широко используемый при создании и модернизации американских военных систем.
В то же время, язык Ada распространился гораздо дальше американского департамента обороны и сейчас используется как в крупномасштабных информационных системах, распределенных системах, так и для научных вычислений. Среди основных применений языка - аэрокосмическая область и область систем с повышенной безопасностью.

Ада язык является результатом самых обширных и самых дорогих проектных усилий когда-либо проводившихся. До 1974 половина приложений в Министерстве обороны не было встроенных систем. Более 450 языков программирования были использованы для реализации различных DoD проектов, и ни один из них не был стандартизирован. По этим причинам, армия, флот, ВВС и предложили разработать язык высокого уровня для встраиваемых систем.

Результаты IBM по 6 Категориям

В середине 1980-х годов, Федеральное управление гражданской авиации заключило контракт с IBM, чтобы оценить языки высокого уровня для использования в своей программе Advanced Automation System.

Категория	Макс.балл	Ада	C	Паскаль	JOVIAL	Фортран
Возможность	16,7	16,1	9.6	10,4	7.6	3.9
Эффективность	16,4	8	11,8	10,8	11	11,1
Наличие / Надежность	22,6	21,5	11,6	14,5	15,6	10,3
Ремонтопригодность / расширяемость	17,4	14	10,2	12,2	6,8	8.3
Жизненный цикл Стоимость	11,3	8.2	7.4	7.8	4.9	5.2
Риск	15,6	8.8	8.9	7.6	9.6	8.2
ИТОГО	100	76,6	59,6	63,3	55,5	47

Это исследование пришло к заключению, что использование Ады было “в окончательных интересах..., пожинать значительные преимущества/выплаты за длительный срок”.

Результаты сравнения языка Ада с C++ по 6 категориям

Институт Программирования провел исследование в 1991, чтобы сравнить Аду и C ++ для применения в информационных системах и Команде, Контроле и Коммуникациях. Хотя C ++ улучшенный C, Ада продолжала выигрывать.

Доказанный победитель, Ада - правильный выбор для этих систем.

Категория	Максимальный балл	Ада	С ++
Возможность	16,7	15,3	11,3
Эффективность	16,4	10,7	10,9
Наличие / Надежность	22,6	19,1	12,6
Ремонтопригодность / расширяемость	17,4	13,6	11,4
Жизненный цикл Стоимость	11,3	8.4	8
Риск	15,6	11,7	9,8
ИТОГО	100	**78,8**	63,9

```
-- Печать простого сообщения для демонстрации простейшей программы на Ada.
with Ada.Text_IO;
procedure Hello is
begin
  Ada.Text_IO.Put_Line("Hello, world!");
end Hello;
```

Первая строка представляет собой комментарий; В Аде комментарий начинается с «--» и заканчивается концом строки (комментарии в C++, которые начинаются с // действуют точно также).
Вторая строка - это спецификатор with, который указывает, какие из библиотечных модулей нам необходимы. В данном случае with указывает, что нам необходим модуль Ada.Text_IO. Модуль Ada.Text_IO является стандартным и предоставляет средства для осуществления основных операций текстового ввода/вывода.
Третья строка означает объявление новой процедуры с именем Hello. Необходимо отметить, что в языке Ada не придается особого значения названию основной программы (в C и C++, основная программа должна называться main, а в Pascal названию программы должно предшествовать ключевое слово program).
Четвертая строка содержит лишь ключевое слово begin, с которого начинается процедура Hello.
Пятая строка вызывает процедуру Put_Line из модуля Ada.Text_IO.

Процедура Put_Line печатает строку на текущее устройство вывода (обычно на экран) и переводит курсор в начало следующей строки. Стандартный синтаксис для вызова процедуры включает в себя имя модуля, точку, имя процедуры и далее, если это необходимо, список параметров, заключенный в круглые скобки . В Ada строки заключены в двойные кавычки (точно также, как в C и C++; в Pascal используются одинарные кавычки).
Последняя строка заканчивает процедуру.

Объектная ориентация в Ада

Объектно-ориентированное программирование заключается в создании программного обеспечения с точки зрения "объектов". "Объект" содержит данные и пресущее ему поведение.Данные, как правило, заключаются в константах и переменных, но можуг также, предположительно, находиться полностью вне программы , то есть на диске или в сети. Поведение прописывается в подпрограммах, которые работают с данными. То, что объект делает объектное Ориентирование уникальным, по сравнению с процедурным программированием, не единственный признак, но и сочетание нескольких функций:

- *инкапсуляции* , то есть возможность отделить реализацию объекта от его интерфейса; это, в свою очередь отделяет «клиентов» объекта, которые могут использовать объект только в некоторых предопределенных путях.

- *наследование* , способность одного типа объекта, наследовать данные и поведения (подпрограммы) другого.

- *расширение типа* , способность объекта, добавлять новые компоненты данных и новые подпрограммы на вершине унаследованных единиц и *заменить* некоторые унаследованные подпрограммы собственными версиями;

- *полиморфизм* , способность "клиента", воспользоваться услугами объекта, не зная точный тип объекта, то есть абстрактно. Фактический тип объекта действительно может измениться во время выполнения от одного вызова к другому.

В Аде, каждое из этих понятий имеет соответствующую конструкцию; Именно поэтому Ада непосредственно поддерживает объектно-ориентированное программирование .

- Пакеты обеспечивают инкапсуляцию;
- Производные типы обеспечивают наследование;
- Расширения Записи, предусматривают расширения типа;
- Надклассовые типы, предусматривают полиморфизм.

Объекты создаются во время выполнения и содержат значение данного типа. Объект может быть создан и инициализирован в рамках подготовки объявления, оценки распределителя, агрегата, или вызов-функции .

Например, предположим, типичная иерархия объектно-ориентированных типов: тип верхнего уровня Person(Человек) , а тип Программист , производный от Person(Человек) , и, возможно, несколько видов людей. У каждого человека есть имя; Предположим, Person , имеет компонент Name(Имя). Точно так же, каждый Человек имеет компонент Пол . Тип Программист наследует компоненты и операции Person типа, так объекты Программист также имеют компоненты Имя и Пол .Объекты Программист могут иметь дополнительные компоненты, пресущие для программистов.

Объекты тегированного типа создаются так же, как объектов любого типа.

```
declare
    P: Person;
begin
    Text_IO.Put_Line("The name is " & P.Name);
end;
```

Результатом является объект с именем P типа Person.Тем не менее, P имеет только значения компонентов Наименование по умолчанию и пол . Один из способов дать начальные значения компонентам объекта является присвоение агрегат.

```
declare
    P: Person := (Name => "Scorsese", Gender => Male);
begin
    Text_IO.Put_Line("The name is " & P.Name);
end;
```

Выражение в скобках после: = называется агрегат.

Еще один способ создания объекта, является вызов функции. Объект будет создан в качестве возвращаемого значения вызова функции.Поэтому, вместо того, чтобы, использовать агрегат начальных значений, мы могли бы вызвать функцию, возвращающую объект.

```
package Persons is

    type Person is tagged private;

    function Make (Name: String; Sex: Gender_Type) return Person;

    function Name (P: Person) return String;
private
    type Person is tagged
        record
            Name   : String (1 .. 10);
            Gender : Gender_Type;
        end record;
end Persons;
```

Значительные языковые особенности:

1. Пакеты - Типы данных, объекты данных и спецификации процедуры могут быть инкапсулированны в пакет.
2. Обработка исключений - Ада имеет очень хорошие возможности обработки исключений, которые позволяют программе обрабатывать свои ошибки во время выполнения.
3. Общие Единицы программы - Можно написать процедуру (например, процедуру сортировки), которая не требует тип данных
4. Параллельная обработка - Ада поддерживает параллельное и одновременное выполнение задач.

Наиболее характерная черта языка Ада - главный акцент на структурное программирование.

Язык позволяет писать программы в виде пакетов, Т.е. самостоятельных модулей, которые разрабатываются отдельными программистами, а затем собираются вместе.

Пакет может быть разработан, отлажен, оттестирован и сохранен в библиотеке модулей для дальнейшего использования в программе, как если бы это была часть стандартного программного обеспечения.

Такая модульная схема, утверждали сторонники Ады, позволяет создавать надежные, легко читаемые и удобные в сопровождении программы, что экономит тысячи часов и сотни миллионов долларов.

Пример пакета

```
                        -- Интерфейс    AdderPkg
package AdderPkg is
    type MY_ARRAY is array(INTEGER range <>) of FLOAT;
    procedure Add_Em_Up(In_Dat : in         MY_ARRAY;
                                Sum       :       out FLOAT);
end AdderPkg;
                        -- Реализация    AdderPkg
package body AdderPkg is
    procedure Add_Em_Up(In_Dat : in         MY_ARRAY;
                                Sum       :       out FLOAT) is
    Total : FLOAT;
    begin
        Total := 0.0;
        for Index in In_Dat'FIRST..In_Dat'LAST loop
            Total := Total + In_Dat(Index);
        end loop;
        Sum := Total;
    end Add_Em_Up;
end AdderPkg;
```

Пакет - это средство, которое позволяет сгруппировать логически связанные вычислительные ресурсы и выделить их в единый самостоятельный программный модуль. Под вычислительными

ресурсами в этом случае подразумеваются данные (типы данных, переменные, константы...) и подпрограммы которые манипулируют этими данными. Характерной особенностью данного подхода является разделение самого пакета на две части: спецификацию пакета и тело пакета. Причем, спецификацию имеет каждый пакет, а тело могут иметь не все пакеты.

Спецификация определяет интерфейс к вычислительным ресурсам (сервисам) пакета доступным для использования во внешней, по отношению к пакету, среде. Другими словами - спецификация показывает "что" доступно при использовании этого пакета.

Тело является приватной частью пакета и скрывает в себе все детали реализации предоставляемых для внешней среды ресурсов, то есть, тело хранит информацию о том "как" эти ресурсы устроены.

Необходимо заметить, что разбиение пакета на спецификацию и тело не случайно, и имеет очень важное значение. Это дает возможность по-разному взглянуть на пакет. Действительно, для использования ресурсов пакета достаточно знать только его спецификацию, в ней содержится вся необходимая информация о том как использовать ресурсы пакета. Необходимость в теле пакета возникает только тогда, когда нужно узнать или изменить реализацию чего-либо внутри самого пакета.

В Аде подпрограммы подразделяются на две категории: процедуры и функции. Процедуры вызова не возвращает никакого значения, в то время как функция возвращает значение и поэтому должно быть частью выражения.

Общий вид описания процедуры

procedure имя_процедуры [(формальные_параметры)] ;

спецификация процедуры, определяющая имя процедуры и профиль ее формальных параметров (если они есть)

Общий вид тела процедуры:

procedure имя_процедуры [(формальные_параметры)] is

спецификация процедуры, определяющая имя процедуры и профиль ее формальных параметров (если они есть)

. . . описательная (или декларативная) часть, которая может содержать локальные описания типов, переменных, констант, подпрограмм...

begin

. . . исполняемая часть процедуры, которая описывает алгоритм работы процедуры; обязана содержать хотя бы одну инструкцию

end [имя_процедуры]; здесь, указание имени процедуры опционально

Таким образом, описание содержит только спецификацию процедуры и определяет правила ее вызова (иначе - интерфейс), а тело содержит спецификацию и последовательность инструкций, которые выполняются при вызове процедуры.

Примечательно требование Ады, чтобы исполняемая часть процедуры содержала хотя бы одну инструкцию. Поэтому, как правило на этапе проектирования, при написании процедур-заглушек используется пустая инструкция, например:

```
procedure Demo(X: Integer; Y: Float) is begin
      null; -- пустая инструкция    end Demo;
```

Вызов процедуры производится также как и в языке Паскаль, например:

```
Demo(4, 5.0);
```

Необходимо также заметить, что Ада предоставляет программисту возможность, при необходимости, помещать в любых местах внутри исполнительной части процедуры инструкцию возврата из процедуры - **return**.

Общий вид описания функции:

function имя_функции [(формальные_параметры)]
return тип_возвращаемого_значения ;

спецификация функции, определяющая имя функции, профиль ее формальных параметров (если они есть) и тип возвращаемого значения

Общий вид тела функции:

function имя_функции [(формальные_параметры)]
return тип_возвращаемого_значения is

спецификация функции, определяюща я имя функции, профиль ее формальных параметров (если они есть) и тип возвращаемого значения

. . . описательная (или декларативная) часть, которая может содержать локальные описания типов, переменных, констант, подпрограмм...

begin

исполнительная часть функции, которая описывает алгоритм

. . . работы функции; обязана содержать хотя бы одну инструкцию возврата значения - return

end [имя функции]; здесь, указание имени функции опционально

Сферы применения:

- Системы организации воздушного движения
- Настольные и веб-приложений
- Гражданская авиация
- Банковские и финансовые системы
- Железнодорожный транспорт
- Информационные системы
- Коммерческие Ракеты
- Судовые коммерческие системы управления
- Телевидение / Индустрия развлечений
- Связь и навигационных спутников и приемники
- Медицинская промышленность
- Передача данных
- Общая промышленность
- Научные космические аппараты
- Военные Приложения

Преимущества и особенности языка:

Экономная разработка

Для наиболее крупных, долгоживущих систем основные усилия возникает не столько в начальной стадии кодирования, а во время тестирования / контроля качества, модернизации функциональности, переноса на новые платформы. Ада была специально разработана для решения этих проблем, и делает это более эффективно, чем другие языки. Его многовстроенные проверки позволяют компилятору обнаружить ошибки, что в языке С будут обнаружены только во время отладки, когда это обойдется гораздо дороже, чтобы разыскать их.

Зрелость языка и реализации

Первоначально он был разработан в начале 1980-х (Ada 83) На всех этапах разработки особое внимание было уделено практическим вопросам, как влияние предлагаемых возможностей языка на эффективность во время выполнения.

Технология реализация языка Ада имеет большой послужной список в крупных и долгоживущих критических системах. Высококачественные компиляторы, обширные наборы инструментов, обширные библиотеки, современные интегрированные среды разработки, и интерфейсы с помощью обычных инструментов и технологий сторонних (и с другими языками программирования) доступны в широком диапазоне платформ.

Статус международного стандарта
Ада язык поддерживается специальной и квалифицированной технической рабочей группой в соответствии с ИСО (Международная организация по стандартизации). Процесс стандартизации ISO гарантирует нейтралитет поставщиков, тщательный обзор и стабильность языка в то же время позволяя периодические обновления и дополнения.
В отличие от других языков, которые достигли стандартизации только после реализации, Ада была стандартизирована вначале, а реализована позже. Этот подход помог избежать технические и политические проблемы, пытаясь определить синтаксис и семантику функций, которые были осуществлены несовместимым образом.

Взаимодействие с другими языками
Это редкое, особенно в больших системах, для программного обеспечения, быть разработанным исключительно на одном языке программирования. Часто возникает необходимость для подпрограмм низкого уровня, написанных на C или ассемблере, элементов графического интерфейса, которые могут быть написаны на C ++ или Java, или численных данных библиотек, которые могут быть написаны на языке Фортран. Ада является уникальным в том, стандартные функции взаимодействуют с другими языками. Это облегчает разработку многоязычных систем - например, C может быть вызван из Ада, или наоборот, определяется стандартом Ada.

Простота подготовки программистов
Ада, интуитивно понятный и простой в освоении язык. Ада гораздо проще в освоении, чем C ++, сложного языка со многими синтаксических и семантических тонкостях. Ада также легче учиться, чем язык Java, чья «чистая» объектная ориентация может сделать простые программы на удивление сложным и чей параллелизм функций весьма подвержены ошибкам.
Несколько видов высококачественных ресурсов доступны для лиц, желающих изучать Аду, в том числе учебники, электронные учебники и справочные материалы, а также инструкции. Опыт Ада педагогов на протяжении многих лет показал, что с 5-дневной практический курс, программист, знакомый с языком, таким как C может стать специалистами в Аде.

Успешное использование на практике
Ада всегда был привлекательным выбором в прикладных областях, где надежность (по сравнению с, скажем, быстротой выхода на рынок) был доминирующим требованием. Исторически это было наиболее очевидно в оборонной и аэрокосмической промышленности, со многими миллионами строк кода в использовании на операционных системах. Ада используется в промышленных масштабах во многих областях, включая авионику, судовые системы, контроль ядерных реакторов, поезда и систем метро, и коммуникации.

Программное обеспечение переносимости
Реальные системы зачастую должны работать на нескольких платформах, например, из маркетинговых соображений или из-за обновления в аппаратном обеспечении или операционной системе. Легкость или трудность таких усилий по переносу напрямую зависит от выбора языка программирования, и Ада имеет много свойств, которые решают эту задачу. Во-первых, язык был разработан, чтобы минимизировать зависимость от платформы. Во-вторых, определенный международный стандарт; обойдены неясности, которые могут помешать переносимости. . В-третьих, обширный и открытый люкс тест соответствия используется поставщиками компиляторов для обеспечения уверенности, что особенности языка успешно реализованы. Действительно, опыт за эти годы с переносимости кода в Аде в целом положительные.

Компиляторы языка Ada

На просторах InterNet, если постараться, можно найти множество информации по компиляторам языка Ada. Многие из них на сегодняшний день уже не сопровождаются, поэтому имеют слабый интерес для общественности. На этой странице приведён список только тех компиляторов, которые по состоянию на декабрь 2011 года продолжают активно сопровождаться.

GNAT от AdaCore

Этот компилятор представляет наибольший интерес для широкой общественности в виду того, что его исходные тексты, равно как и исходные тексты библиотеки времени исполнения, являются открытыми и публично доступными. Сам компилятор построен на основе генератора кода GCC, и как бы является неотъемлемой частью проекта GCC. В то же время, он активно поддерживается компанией AdaCore.

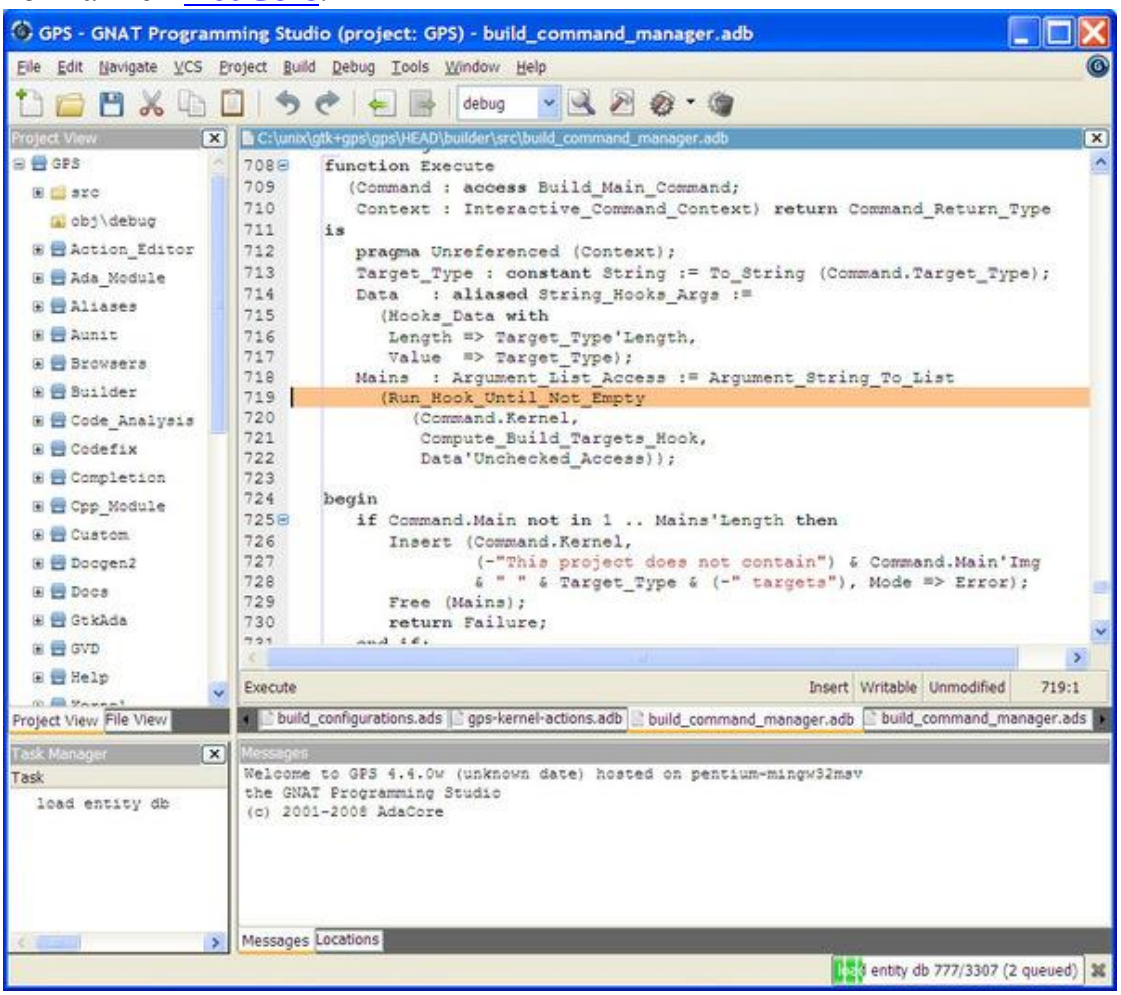

Рис. 1 GNAT GPL 2011

ICC от Irvine

Автором этого компилятора является компания Irvine Compilers Corporation. Не смотря на весьма неприглядный вид сайта работы над компилятором ведуться достаточно активно в том числе в тестировании находится реализация стандартра Ada 2005.

ObjectAda от Atego

Являясь приемником Aonix, компания Atego продолжает некоторое развитие компилятора ObjectAda.

AdaMULTI от GHS

Ещё одни закрытый, но активный продукт предлагается компанией Green Hills Software и называется AdaMULTI.

Список использованной литературы:

1. http://www.adacore.com/adaanswers/benefits-and-features#sthash.sHWQJPkq.dpuf
2. http://en.wikipedia.org/wiki/Ada_(programming_language)
3. http://groups.engin.umd.umich.edu/CIS/course.des/cis400/ada/ada.html
4. www.ada-ru.org/V-0.4w/part_1/ch_06.html
5. http://archive.adaic.com/docs/present/engle/whyada/tsld047.htm#weiderman
6. http://ada-ru.qtada.com/resources/compiler