Интерфейс ASIS

Василий Александрович Фофанов
Сергей Игоревич Рыбин

Содержание

Предназначение....1
История разработки....2
Архитектура и строение интерфейса....3
Основные типы данных ASIS....3
Иерархия пакетов ASIS....4
Использование ASIS....5
Понятие ASIS-приложения (ASIS Application)....5
Понятие и назначение Контекста ASIS (ASIS Context); работа с Контекстами....5
Единицы компиляции (Compilation_Units); работа с Единицами компиляции....7
Структурная декомпозиция Единиц компиляции. Элементы....7
Семантические зависимости между Единицами компиляции и между Элементами....7
Текстовые представления Элементов ASIS.....7
Обход дерева синтаксических Элементов (element traversal). Процедура Traverse_Element....8
Представление неявных элементов и результатов конкретизации настраиваемых модулей....9
Реализации ASIS....9
ASIS-for-GNAT....10
Текущий статус интерфейса....10
Литература....10

Предназначение

Спецификация Семантического Интерфейса к языку Ада (Ada Semantic Interface Specification, ASIS) [ASIS] - это новый стандарт Международной Организации по Стандартизации, область Информационных технологий, раздел Языков программирования, их окружений и программных интерфейсов к системам (ISO/IEC 15291:1999).

В соответствии с официальным определением, ASIS является интерфейсом между программным окружением языка Ада (см. [RM95]) и инструментами, нуждающимися в информации из этого окружения. ASIS является открытой спецификацией интерфейсной библиотеки, предоставляющей доступ к такой информации средствам поддержки программной инженерии (CASE) и разработчикам приложений. Интерфейс ASIS был разработан независимым от деталей реализаций программного окружения Ада, обеспечивая тем самым переносимость инструментов программной инженерии и освобождая их разработчиков от необходимости понимания деталей конкретной реализации программного окружения Ада.

Этот интерфейс позволяет упростить подготовку средств анализа текста приложений на языке Ада, таких как трансляторы, программы навигации по тексту, средства контроля стиля кодирования, инструменты для реструктуризации исходного текста, генерации документации, обратной инженерии, вычисления различных метрик, генерации тестовых наборов, автоматического доказательства корректности, и многих других. Несмотря на наличие многочисленных универ-

сальных инструментов такого рода, нередко возникает необходимость разработки аналогичных инструментов для узкоспециализированных нужд. Все эти инструменты объединяет тот факт, что в ходе своей работы они используют совокупность лексической, синтаксической и семантической информации, которую компилятор языка получает в процессе разбора программы. В результате разработчики этих инструментов вынуждены реализовывать значительную часть компилятора, что приводит к значительному росту временных и финансовых затрат на разработку инструментов и резкому снижению их надежности. Кроме того, такие инструменты, как правило, не переносимы на другие платформы.

Область применения ASIS становится более понятной с учетом следующих его свойств:

- ASIS - интерфейс "только для чтения" (read-only); иными словами, пользователь может получать информацию из ASIS-окружения, но не может изменять содержимое компонентов окружения.
- ASIS предоставляет только статически определимую информацию о программах на языке Ада.
- ASIS является полной библиотекой запросов по отношению к синтактико-семантической информации языка Ада: ASIS предоставляет всю синтаксическую и базовую семантическую информацию о программах; любая семантическая зависимость, не доступная прямым вызовом некоторого запроса ASIS, вычислима некоторой комбинацией запросов.
- За исключением некоторых зависимостей от реализаций, ASIS предоставляет информацию о программах в высокоуровневых терминах, находящихся в хорошем соответствии с [RM95] и являющихся по своей природе не зависимыми от реализаций транслятора с Ады и Интерфейса ASIS.

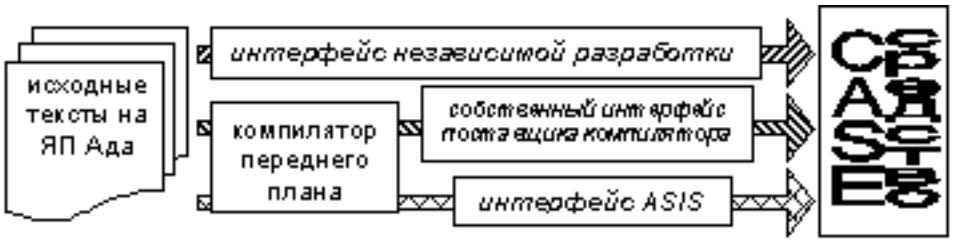

Рис. 1. Роль интерфейса ASIS в разработке средств программной инженерии языка Ада.

Исходя из этих свойств Интерфейса становится понятным, что ASIS является привлекательной платформой для реализации вышеперечисленных средств анализа текста программ. ASIS компенсирует трудоемкость разработки подобных инструментов и может значительно ускорить и упростить их разработку благодаря тому, что разработчик может целиком сосредоточиться на содержательной части программы. Появление подобного Стандарта характеризует новый шаг в направлении популярной в последнее время модели разработки инструментов на базе промежуточных представлений программы (рис. 1).

История разработки

В начале декабря 1998 года технический комитет Всемирной организации по стандартизации провел финальное голосование по проекту стандарта ASIS. Это - долгожданное событие в области программирования на языка Ада, так как работа над стандартизацией ASIS ведется с

1993 г.

В НИВЦ МГУ с 1994 года разрабатывается реализация ASIS для транслятора GNAT с языка Ада. Система программирования GNAT доступна практически для всех современных платформ. Существует свободно распространяемая версия системы GNAT, которая периодически обновляется. На сегодня текущая реализация ASIS для системы GNAT, представляющая собой наиболее полную реализацию этого интерфейса, включена в набор средств, поставляемых в составе дистрибутива СП GNAT. Группа разработчиков из НИВЦ МГУ приняла активное участие в подготовке стандарта ASIS.

Архитектура и строение интерфейса

Интерфейс ASIS, определенный как набор пакетов на языке Ада, представляет собой библиотеку абстрактных типов данных и операций над ними, обеспечивающих высокоуровневый доступ к синтаксической и семантической информации, содержащейся в анализируемой программе. Программа, нуждающаяся в такой информации, компилируется в контексте библиотеки ASIS и в результате получает документированный, стандартный доступ к синтаксической и семантической структуре. При этом приложения, созданные на базе ASIS, будут переносимы не только между платформами, на которых существует ASIS-совместимый компилятор, но и между компиляторами, несмотря на то, что возможны огромные различия во внутренних представлениях данных в этих компиляторах.

Основные типы данных ASIS

Основные абстрактные типы данных Интерфейса, обеспечивающие отображение таких понятий языка Ада, как программное окружение, элемент компиляции, синтаксический элемент, исходный текст, реализованы как приватные типы языка Ада. Важнейшими из них являются:

- Context (Контекст) - ограниченный приватный тип, служащий для задания множества анализируемых модулей из программного окружения.
- Compilation_Unit (Единица Компиляции) - приватный тип, являющийся абстракцией элементов компиляции.
- Unit Kinds (Типы Единиц Компиляции) - набор перечислимых типов, описывающих различные типы модулей компиляции.
- Element (Элемент) - приватный тип, являющийся абстракцией сущностей внутри синтаксического дерева. Множество всех Элементов, соответствующих различным синтаксическим конструкциям данной Единицы Компиляции, является непосредственной абстракцией самого синтаксического дерева и называется *деревом Элементов ASIS* или просто *деревом ASIS*.
- Element Kinds (Типы Элемента) - набор перечислимых типов, отображающих классификацию синтаксических конструктов языка Ада.
- Program_Text (Текст программы) - тип данных, представляющий исходный текст программы; отображается на стандартный тип Wide_String.

Ниже по тексту, говоря об абстрактных типах, описанных в Интерфейсе, будем использовать названия, указанные в скобках.

Классификация Элементов.

В Интерфейсе ASIS описана сложная многоуровневая иерархия перечислимых типов *_Kinds, называемых *классификационными.* Отдельные классификационные типы в этой иерархии соответствуют различным категориям синтаксических элементов языка Ада, а литералы этих перечислимых типов соответствуют различным метапеременным БНФ языка Ада, таким как Объявление_Пакета, Оператор_Присваивания, Вызов_Функции и т.п.

Для любого Элемента применим набор классификационных запросов *_Kind, при помощи которых можно узнать, какую синтаксическую конструкцию анализируемой программы содержит данный Элемент, то есть какова его позиция в классификационной иерархии. Как правило, необходимо вызвать несколько таких запросов, последовательно конкретизирующих его позицию. Например, для определения, что Элемент содержит описание ссылочного типа, необходимо вначале применить к нему запрос Element_Kind, который вернет значение A_Definition типа Element_Kinds, обозначающее, что Элемент содержит некоторое описание. Затем необходимо применить запрос Definition_Kind, возвращающий значение A_Type_Definition, указывающее, что Элемент содержит описание типа. Далее придется применить запрос Type_Kind, чтобы узнать, какого именно типа это описание, и наконец, Access_Type_Kind, чтобы узнать, какого именно ссылочного типа.

Иерархия пакетов ASIS

Интерфейс ASIS определен как иерархия спецификаций дочерних пакетов Ада. Ниже дано краткое описание различных пакетов Интерфейса, взятое из [ASIS].

Asis. - верхний пакет иерархии. Определяет основные абстракции ASIS - Контекст, Единица Компиляции и Элемент. Также вводится набор перечислимых типов, определяющих классификационную иерархию Элементов и Единиц Компиляции. Пакет не содержит запросов.

Asis.Implementation. - содержит запросы для работы с реализацией интерфейса: инициализация и финализация, получение и сброс диагностической информации. Его дочерний пакет Asis.Implementation.Permissions содержит булевские запросы, определяющие, как ряд реализационно-зависимых запросов реализован в данной реализации.

Asis.Ada_Environments. - содержит запросы для работы с Контекстами ASIS: связывание и обрыв связывания Контекста с Ада-окружением, открытие и закрытие Контекста.

Asis.Compilation_Units. - содержит запросы для работы с Единицами Компиляции: получение Единиц из Контекста, внешние свойства Единиц, некоторые семантические зависимости между Единицами, напр. переход от спецификации к телу и обратно.

Asis.Compilation_Units.Relations. - содержит запросы, возвращающие семантические зависимости между Единицами, напр. все Единицы, используемые данной.

Asis.Iterator. - содержит механизм Traverse_Element для итерационного обхода дерева Элементов ASIS. Этот механизм является фундаментальным компонентом большинства приложений ASIS. Так как он также составляет важную часть инструментов разработанной в рамках настоящей работы системы тестирования реализаций ASIS, о нем будет подробнее рассказано ниже.

Asis.Elements. - содержит запросы для работы с Элементами и реализации их основных свойств: шлюзовые запросы для перехода от абстракции Единицы к абстракции Элемента, за-

просы, определяющие положение Элемента в классификационной иерархии, запросы, возвращающие для данного Элемента объемлющие его Элемент и Единица Компиляции. Пакет также содержит запросы для работы с директивами компилятору (т.наз. прагмами, pragmas).

Asis.Declarations, Asis.Definitions, Asis.Statements, Asis.Expressions и ASIS.Clauses . - каждый из этих пакетов содержит запросы для работы с соответствующим типом Элементов, т.е. с Элементами, представляющими объявления, описания, операторы, выражения и указания, соответственно.

Asis.Text. - содержит запросы, возвращающие информацию об исходном текстовом представлении Единиц Компиляции и Элементов ASIS.

Asis.Exceptions. - описывает исключения, возбуждаемые Интерфейсом ASIS.

Asis.Errors. - описывает возможные ошибочные состояния ASIS.

Asis.Ids. - содержит запросы для работы с абстракцией Идентификатора Элемента.

Кроме того, в состав иерархии пакетов Интерфейса входят два необязательных пакета, Asis.Data_Decomposition и Asis.Data_Decomposition. Portable_Transfer.

Использование ASIS

Понятие ASIS-приложения (ASIS Application)

ASIS-приложением называется всякая программная система или набор программных компонентов, которая использует запросы Интерфейса ASIS для получения информации о любом наборе программных компонентов на языке Ада.

Понятие и назначение Контекста ASIS (ASIS Context); работа с Контекстами

В соответствии с [RM95], компиляционное окружение языка Ада определяется так (п.10.1.4(1)): "Каждый модуль компиляции, переданный компилятору, компилируется в контексте раздела описаний окружения (или просто окружения), который является концептуальным разделом описаний, формирующим самую внешнюю область описаний любого компиляционного контекста." Однако, механизмы создания компиляционного окружения и добавления/удаления единиц компиляции внутри окружения зависят от реализации компилятора. Соответственно Интерфейс ASIS нуждается в задании пользователем (т.е. ASIS-приложением) компиляционного окружения.

Для эмуляции компиляционного окружения в рамках ASIS служит Контекст. Контекст определяет набор единиц компиляции и конфигурационных директив компилятору, подлежащих обработке ASIS-приложением. Фактически Контекст представляет из себя "точку зрения" на Ада-окружение. ASIS требует от пользователя задания этой "точки зрения" посредством операции связывания Контекста с Ада-окружением, выполняемой запросом Asis.Ada_Environments. Associate (см. рис. 2). Способ установления связи между Контекстом и Ада-окружением является реализационно-зависимым, но после этого с Контекстом можно обращаться переносимым образом. После выполнения операции связывания Контекст должен быть "открыт" посредством выполнения запроса Asis.Ada_Environments.Open. После этого могут использоваться запросы ASIS, предоставляющие информацию о содержимом Контекста. Существуют также команды

закрытия Контекста и обрыва связывания (Close и Dissociate).

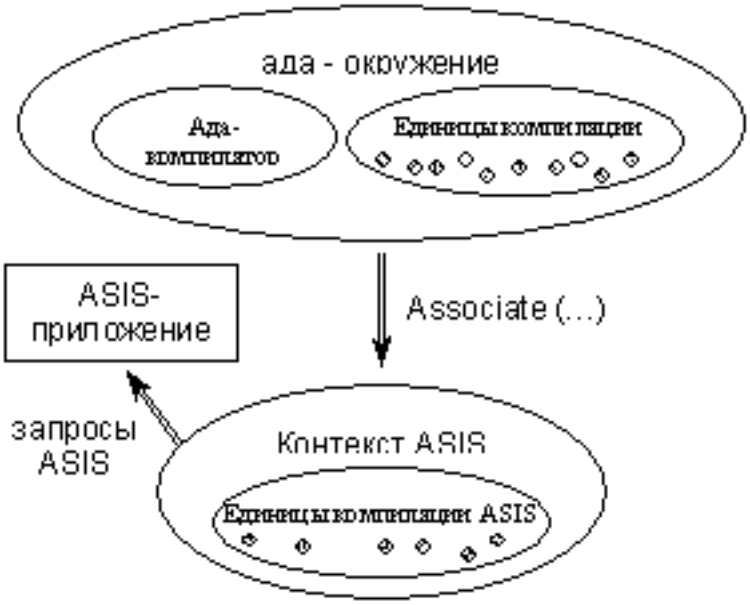

Рис. 2. Взаимосвязь Ада-окружения и Контекста ASIS.

Таким образом, за исключением экзотических случаев, основной цикл ASIS-приложения является следующим:

```
with Asis.Ada_Environments;

...

My_Context : Asis.Context;

...

Asis.Ada_Environments.Associate

   (My_Context, "My Context", "<параметры связывания>");

Asis.Ada_Environments.Open (My_Context);

<выполнить действия с Контекстом>

Asis.Ada_Environments.Close (My_Context);

Asis.Ada_Environments.Dissociate (My_Context);
```

Приложение может работать с несколькими Контекстами (т.е. точками зрения на Ада-окружение) одновременно.

Единицы компиляции (Compilation_Units); работа с Единицами компиляции

Тип Compilation_Unit является логическим эквивалентом понятия единицы компиляции языка Ада. Он отображает практически все свойства единиц компиляции, описанные в [RM95], а также предоставляет доступ к некоторым свойствам физических единиц компиляции данной Ада-реализации (таким как время последнего изменения, имя модуля исходного текста, соответствующего единице компиляции, и т.п.).

Compilation_Unit представляет единицу компиляции как единое целое, "черный ящик". Возможна декомпозиция Единиц компиляции через абстракцию Элементов.

Структурная декомпозиция Единиц компиляции. Элементы

Для представления структурных компонентов Единиц компиляции служит тип Element. Он является логическим эквивалентом понятия синтаксической конструкции языка. К дереву Элементов, составляющих конкретную Единицу компиляции, можно получить доступ при помощи шлюзового запроса Asis.Elements.Unit_Declaration, позволяющего от Единицы компиляции перейти к корневому узлу дерева.

Для перехода вниз по ветвям дерева используются 125 запросов ASIS, приблизительно соответствующих выражениям БНФ, описывающей язык Ада [RM95 секция P.1]. Например, запрос Loop_Statements позволяет перейти от Элемента, представляющего оператор цикла, к списку операторов тела цикла.

Кроме запросов, осуществляющих переход по дереву сверху вниз, имеется также запрос Enclosing_Element, переходящий в обратном направлении.

Эти запросы называются *запросами структурной декомпозиции*, или просто *структурными запросами*.

Семантические зависимости между Единицами компиляции и между Элементами

Рассмотренные выше запросы структурной декомпозиции предназначены для разбора синтаксической структуры Ада-программы. В дополнение к ним, ASIS предоставляет набор запросов, позволяющих переходить от Элемента к Элементу по семантическим связям. Так, например, запрос Corresponding_Name_Declaration позволяет перейти от использования некоторого идентификатора к его объявлению.

Такие запросы называются *семантическими*.

Текстовые представления Элементов ASIS.

Взаимосвязь между иерархией Элементов ASIS и их представлением в виде программы на языке Ада устанавливается типами Program_Text, Span, Line и запросами, находящимися в пакетах Asis.Text и Asis.Text.Lines. Существует возможность получить исходный текст всей Единицы Компиляции (запрос Compilation_Unit_Image), произвольного Элемента ASIS (запрос Element_Image), а также любых содержательных терминальных элементов

(Defining_Name_Image, Name_Image, Value_Image и т.п.)

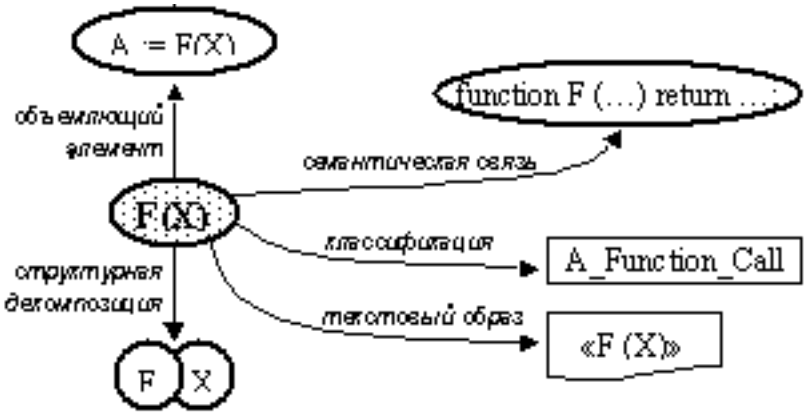

Рис. 3. Основные типы операций со значением типа Элемент, описанные Интерфейсом

Основные действия, описанные Интерфейсом ASIS для значений типа Элемент, показаны на рис.3.

Обход дерева синтаксических Элементов (element traversal). Процедура Traverse_Element

Запросы структурной декомпозиции каждый по отдельности имеют очень небольшое множество типов Элементов, для которых они применимы. Практически невозможно устраивать в ASIS-приложениях сложные многоуровневые цепочки структурных запросов и операторов условного перехода для всего множества возможных в рамках данного ASIS-приложения случаев. Естественнее было бы воспользоваться механизмом автоматического обхода дерева Элементов, проверяя в каждом узле, интересен ли данный Элемент для приложения, и предоставляя соответствующую обработку.

Такой механизм предоставляется Интерфейсом ASIS в виде пакета Asis.Iterator, который содержит настраиваемую процедуру Traverse_Element, принимающую Элемент ASIS в качестве параметра и выполняющую рекурсивный обход дерева Элементов, укорененного в данном Элементе. Принципиальная схема ее работы заключается в последовательном применении алгоритма обхода к компонентам данного Элемента (т.е. Элементов, возвращаемых всеми запросами структурной декомпозиции, применимых к Элементу данного типа). Очередной такой шаг обозначает погружение обхода на более глубокий уровень дерева. В конечном итоге, все Элементы оказываются посещенными ровно один раз, сверху вниз слева направо.

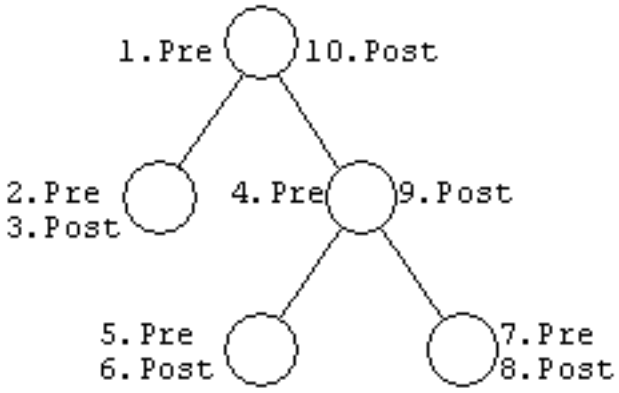

Рис.4. Порядок обхода Элементов ASIS и вызова пользовательских процедур настраиваемой процедурой Traverse_Element (цифра около операции указывает номер шага, на котором она осуществляется)

Процедура Traverse_Element вызывает создаваемые пользователем функции Pre_Operation и Post_Operation, вызываемые соответственно в начале и в конце обхода синтаксического поддерева, укорененного в текущем Элементе (рис.4). Внутрь этих операций пользователь и должен поместить код, обеспечивающий обработку посещаемых Элементов. Пользовательские процедуры могут также дать указание процедуре обхода не погружаться в текущее поддерево (например, если оно заведомо не содержит подлежащих обработке элементов) или полностью прекратить рекурсивный обход.

Представление неявных элементов и результатов конкретизации настраиваемых модулей

В процессе обычной декомпозиции программы возможен доступ только к так называемым явным" Элементам, то есть Элементам, соответствующим синтаксическим конструкциям, явно присутствующим в тексте анализируемой программы. Этим, однако, не исчерпывается все множество Элементов, которое может быть получено для данного модуля при помощи Интерфейса ASIS.

Кроме явных Элементов, ASIS предоставляет возможность доступа, во-первых, к неявным синтаксическим конструкциям, то есть конструкциям, которые отсутствуют в тексте программы, но неявно находятся в нем в силу семантики других конструкций, и, во-вторых, к Элементам, соответствующим расширениям настраиваемых модулей.

Доступ к соответствующим участкам синтаксического дерева возможен через семантические запросы, применяемые к явным синтаксическим конструкциям, приводящим к появлению неявных или расширенных. Так, например, для перехода к спецификации пакета, полученного при расширении настраиваемого пакета, необходимо воспользоваться запросом Corresponding_ Declaration, применяемый к Элементу, соответствующему синтаксической конструкции расширения (т.е. имеющему тип A_XXX_Instantiation). Элемент, полученный при этом, является корневым в дереве, соответствующем спецификации расширенного пакета. Начиная с этого Элемента, возможна обычная декомпозиция этого поддерева стандартными средствами ASIS.

Реализации ASIS

Несмотря на молодость стандарта, первые его реализации (или реализации его промежуточных ревизий) уже существуют. Другие находятся в стадии разработки.

Первый прототип реализации ASIS-for-GNAT был представлен на Международной конференции Tri-Ada'96. Фирма Rational предлагает реализацию ASIS для СП Apex, причем обещается также обратная совместимость с ASIS 83. Фирма Aonix предлагает реализацию ASIS для своей популярной визуальной СП ObjectAda.

ASIS-for-GNAT

ASIS-for-GNAT является реализацией Интерфейса ASIS для СП GNAT. Начавшись как совместный проект между Московским Государственным Университетом и Федеральным Политехническим Институтом в г. Лозанне, Швейцария, сейчас она является составной частью СП GNAT и сопровождается разработчиком СП, фирмой Ada Core Technologies.

В настоящий момент это - единственная публично доступная реализация ASIS и единственная реализация, полностью поддерживающая Стандарт Интерфейса, включая необязательные пакеты.

Текущий статус интерфейса

В настоящее время реализации Интерфейса ASIS, в том числе и ASIS-for-GNAT, используются в работе отделов ПО таких индустриальных гигантов, как Boeing, McDonnell-Douglas, Lockheed-Martin, Thales, Raytheon и многих других. География ASIS-приложений охватывает многие страны мира и важнейшие области человеческой деятельности, включая морские, авиационные и космические программы.

Таким образом, можно говорить об индустриальном характере Интерфейса и о большой практической важности стандарта ASIS.

Литература

[ASIS] Information technology - Programming languages . *Ada Semantic Interface Specification (ASIS)* . International standard ISO/IEC 15291:1999 .

[RM95] S. Tucker Taft. Robert A. Duff. *Ada 95 Reference Manual: Language and Standard Libraries* . International Standard ISO/IEC 8652:1995(E) . Springer-Verlag. 1997.

Более подробную информацию по ASIS (спецификацию, уроки, примеры, программы в исходных текстах, перечень реализаций и пр.) можно получить на сайте ASIS Working Group [http://www.acm.org/sigada/wg/asiswg/asiswg.html].