Ada Compiler
CS355:Compiler Design

Anshu Avinash
anshuavi@iitk.ac.in

Pranjal Singh
spranjal@iitk.ac.in

Atique Firoz
atiquef@iitk.ac.in

Parth Tripathi
partht@iitk.ac.in

January 18, 2013

Ada is a strongly typed, modular, object oriented, concurrent, readable and expressible high-level computer programming language [1]. In this project we would be creating a compiler for a subset of Ada language. We would be implementing following features:

- Type System: Ada's type system is governed by four principles: Strong typing, Static typing, Abstraction and Name equivalence.
  - type and subtypes: Creation of new types and subtypes with following features:
    - constrained and unconstrained types
    - dynamic types

  Following types would be in standard package:
  - Signed Integers
  - Unsigned Integers
  - Enumerations:
    - Operators: $<, <=, =, /=, >=, >$
    - Attributes: Pos, Val, Image, Value
    - Enumeration Literals: Character and Boolean as enumeration literals
    - subtype
  - Floating Point
  - Ordinary and Decimal Fixed Point
  - Array:
    - Allow creation of arrays with:
      - with known subrange
      - with unknown subranges
      - with aliased elements
    - Multi-Dimensional Arrays
    - Operations with Arrays: Assignment, Concatenate
    - Attributes: First, Last, Length, Range
    - Null Arrays
  - Record: A record is a composite type that groups one or more fields.
    Support for Null Record, Record with Values, Discriminated Record, Variant Record, Union, Tagged, Abstract Tagged, with Aliased Elements and Limited.
  - Access Type: Access types in Ada are what other languages call pointers. There are following Access types:
    - Pool Access

    - General Access: Access to Variable and Access to Constant
    - Anonymous Access
    - Access to subprogram

- Input/Output: ADA has 5 independent libraries for Input and Output operations.
  - Text I/O: It provides support for line and page layout but the standard is free form text.
  - Direct I/O: It is used for random access files which contain only elements of one specific type. It can also be used to position the file pointer to any element of that type, however one cannot freely choose the element type, the element type needs to be a definite subtype.
  - Sequential I/O: It can be used to choose between definite and indefinite element types but one has to read and write the elements one after the other.
  - Storage I/O: It allows to store one element inside a memory buffer. The element needs to be a definite subtype. Storage I/O is useful in *Concurrent programming* where it can be used to move elements from one task to another.
  - Stream I/O: It allows to mix objects from different element types in one sequential file. In order to read/write from/to a stream each type provides a Read and Write attribute as well as an Input and Output attribute. These attributes are automatically generated for each type one declares.

- Exception: Ada has modules which raise an error when certain conditions are not satisfied and another module which does corresponding error-handling.
  - Predefined: They are included in *Standard* package. Some of them are: **Constraint_Error, Program_Error, Storage_Error, Tasking_Error**
  - Input/Output: These exceptions raised by subprograms of the predefined package *Ada.Text_IO*. Some of them are **End_Error, Data_Error,Mode_Error,Layout_Error**
  - Raising exceptions: The *raise* statement explicitly raises a specified exception.

References

[1] *Ada Information ClearingHouse: News and resources for Ada programming language.* URL: `http://www.adaic.org`.