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].

1 Detailed Specification

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.
- Conditionals: These are blocks of code that will only execute if a particular expression (the condition) is true. They are categorized into:
  - **if-else:** When the program arrives at an "if" statement during its execution, control will "branch" off into one of two or more "directions". If the original condition is met, then all the code within the first statement is executed. The optional else section specifies an alternative statement that will be executed if the condition is false. Only one statement in the entire block will be executed. This statement will be the first one with a condition which evaluates to be true.
  - **case:** The case statement is used to compare one specific variable against several constant expressions. If one of the several cases are not satisfied then the statement given in *others* is executed.
- Unconditionals: They let you change the flow of the program without a condition. They are categorized into:
  - **goto:** It transfers the control to the statement after the specified *label*.
  - **return:** It ends a function and returns to the calling procedure or function. For *procedures* just return is used whereas for *functions* return is followed by a *value*.
- Loops: They allow you to have a set of statements repeated over and over again. They are categorized into:
  - **for loop:** It counts a specific variable from a given start value up or down to a specific end value and keeps on repeating the statements enclosed until the final value is reached.
  - **while loop:** This loop has a condition at the beginning. The statements are repeated as long as the condition is met. If the condition is not met at the very beginning then the statements inside the loop are never executed.
  - **until loop:** This loop has a condition at the end and the statements are repeated until the condition is met. Since the check is at the end the statements are at least executed once.
  - **exit loop:** It is used when one wants to do first calculation and exit the loop when a certain criterion is met. However when the criterion is not met there is something else to be done. Hence for this purpose *exit* condition is in the middle.
  - **array loop:** This loop iterates over every element of the array specified.

- 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.

2 Implementation

- Lexical Analyser: It is typically preferable to have a parser be fed a token-stream as input, rather than having it consume the input character-stream directly. Lex is often used to produce such a token-stream. So we will be using **Lex** as our Lexical Analyzer.
- Parser: YaCC
- Langauge: C

References

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