A.C.E.

A Compiler for Eiffel

Functional Specifications

This document describes the functional specifications of A.C.E.- A Compiler for Eiffel

Supported Data types and Classes

- *INTEGER, REAL, BOOLEAN, ARRAY, CHARACTER*
  Each of the above data types are defined by default as expanded classes in Eiffel, different from what it is in contemporary OOP languages
- *ARRAY* -- This has been implemented as a class in Eiffel. We are going to implement only *single dimensional* arrays
- *BASIC_IO* – This is a Eiffel class which facilitates basic I/O functions

Operators

Operator	Meaning
.(dot)	Routine call
not	Relational operator
+ -	Unary plus and minus
^	Power (exponentiation)
* / //	Multiplication, division, integer division, \\ integer remainder
+ -	Binary addition and subtraction
= /=	Equal and not-equal
< > <= >=	Relational operators
and and then	Relational operators
or or else	Relational operators
implies	Relational operators
:=	Assignment
;	Separator

Keywords

- **Boolean and selection**-*and, not, or, true, xor, implies, if, then, else, elseif, true, false, inspect, when*
- **Iteration**-*loop, until, from*
- **Class specific**-*class, creation, do, end, expanded, feature, indexing, is, like, local, current*
- **Design by contract**-*check, require, ensure, variant, invariant*

Functions

We are going to implement only procedures among all the routine calls. We will also implement attributes and our compiler would support identifiers and variables. Procedures in Eiffel do not return values but accept arguments. But we can return values through attributes, which would be instantiation of basic data types.

Object Oriented features

- **Classes**- We will implement basic class structures for objects with attributes like *ANY* and *NONE*
- **Encapsulation**- By implementing the class structure we would implement encapsulation

Language Specific features

Eiffel has a very unique feature namely *design by contract*. We will implement the basic methods of design by contract which include boundary checking and loop checking over variants. The keywords associated have been mentioned earlier.

Tentative features

- **Object oriented**- *Inheritance* (single parent and single level)
- **Language specific**- Advanced techniques of design by contract, like *Debug* and *Trace*.
- **Functions**-Implementing return values for functions through the keyword *Result.* Recursion is also desirable.

Out of Scope

We won’t be implementing Exception Handling. We won’t be handling classes like strings and files and would not implement I/O functions related to strings. We won’t be implementing other data structures like list, trees etc. We won’t use keywords other than those mentioned above in the list of keywords and tentative features.