UVA

DOCUMENTATION FOR OBERON INTERPRETER

THIS DOCUMENTATION WILL CONTAIN SOME BASIC INFORMATION ON THE OBERON INTERPRETER

BEN KWINT
6385559

USAGE OF THE INTERPRETER

COMMANDLINE OPTIONS

The Oberon parser has several command line options that can be used. The following options are available.

-print
The print option is used to return an textual representation of the source code after parsing.

-memory
The memory option is used to dump the entire memory to the screen after execution. This means that it will print the memory as it is after execution. This means only the MODULE scope will be shown, the other parts of the memory will be removed.

-type
The type option will dump all the type definitions that are in the MODULE, type definitions in procedures will not be shown because they are not in the TypeDefinitionManager after execution

-procedure
The procedure option dumps all procedure declarations to the screen after execution, again only the Procedures that belong in the scope of the MODULE are shown

The last parameter you can give on the command line is the Oberon source file that has to be interpreted.

PACKAGES

The Oberon parser depends on 4 different packages and the ANTLR.jar file. Here I will give a short description of what each package does.

oberon0:
This package is the execution package, meaning it contains the main procedure. Which is in the oberon0.java file. This package also contains the CommandLineOptions class which handles the input from the command line. The classes ParserInput and ParserOutput are used to interact with the ANTLR functions.

management:
The management package contains all actions that the interpreter should be able to do except the evaluation function which rests in the AST. This means that it contains the systemcalls that oberon supports Write( ), WriteLn() & Read( ). It also contains the memory, memory management, type management, procedure management and the parameter container which is used during procedurecalls.

ASTnodes:
This package contains all the Nodes that can be in the abstract syntax tree, these nodes also contain all functionality to execute the Oberon source code.

parseErrorLog:
This package only contains the errorlog functionality, you can write to the errorlog if something goes wrong, currently it is only used to show parse errors and whether or not procedures & modules are ended properly.

VARIABLES

Variables need some explanation, to show how the implementation of the Array and Record types have been done.

The oberon parser doesn't use any other type then INTEGER, even BOOLEAN TYPES have been implemented as INTEGER types. Also the Record and Array types have been altered to be INTEGER types.

The way that this has been done is that the selector node is converted to a string and pasted after the variable name. This way for example "a : ARRAY 2 OF INTEGER" is converted to "a[0], a[1] : INTEGER" where "a[0]" and "a[1]" are the complete names as they are in the memory. You can see this if you do a memory dump with array variables declared. The selector node thus only returns the string of the selector, this appended with the variable name gives you a valid location in the memory.

WHY ANTLR

ANTLR has been used to implement the Oberon language because ANTLR has the most complete documentation. Which helps rapid development of your application.

For more information send a email to: 6385559@student.uva.nl