SLE LANGUAGE DOCUMENTAION

Structures and ENUM

`typedef enum { FALSE,TRUE} bool;`	Boolean enum FALSE = 0,TRUE = 1
`typedef enum {INTEGER ,FLOAT ,PROGNAME ,ERROR} Type;`	The types in the sle language.

`typedef struct{`
`  int intNum;`
`  float floatNum;`
`  Type type;`
`} number;`
	This struct represent a number int the language. Its contain int value and float value but the type tell us what is the type of the number.

`typedef struct node {`
`  char id[20];`
`  number num;`
`  struct node* next;`
`}node ;`
	Node of a linked list, each node store a symbol id and value.

`typedef struct{`
`  node* head;`
`}List;`
	The head of the symbols linked list.

`typedef struct undefined{`
`  char id[20];`
`  struct undefined* next;`
`}undefined;`
	Node of a linked list that store save the id's of variables that don't declared in the program.

`typedef struct{`
`  undefined* head;`
`}udefined_list;`
	The head of the undefined linked list.

Global variable

`List s_list;`	Linked list of all the symbols in the program and their values.
`udefined_list u_list;`	Linked list of all the symbols that don't declared in the program.

SLE LANGUAGE DOCUMENTAION

Functions

`int sleTypeCheck(const char* str)`	Str it's a ending of file name, and the function check if the ending is "sle" or "SLE" if it does then return 1.
`void* allocation(unsigned int size)`	Check if allocation of node was fine or failed.
`node* searchSymbol(const char* id)`	Search symbol in the symbols linked list by his. If the symbol found then return him other return null.
`void insertSymbol(const char* id, number insert_num)`	Insert a symbol to the symbols linked list with his id and value, if the value already exist in the linked list then we update his value.
`void insertUndefinedSymbol(const char* id)`	Insert a symbol to the undeclared linked list.
`int searchInUndefined(const char* id)`	Search a symbol in the undeclared linked list.
`void insertAssignOp(const char* id, number insert_num)`	Search the symbol with this id in the symbols linked list and then update his value to the new value.
`void printNumber(number print_num)`	Print the value of a symbol with a given id.
`number getSymbolValue(const char* id)`	Get the value of a symbol by a given id.
`int booleanExpration(number left_num, char sign, number right_num)`	Return the result of a Boolean expression.
`number addOperator(number left_num, char sign, number right_num)`	Return the result of a expression with add operator.
`number mulOperator(number left_num, char sign, number right_num)`	Return the result of a expression with multiplication operator.

SLE LANGUAGE DOCUMENTAION

Files and their uses.

sle.l	Input file for the flex lexical analyzer. The file contains regular expressions for all the tokens.
lps.y	Input file for the bison parser generator. The file contains the grammar rules for the language.
main.c	Run the actual program and execute the exe file, contain the functions implementation.
lex.yy.c	Generated from flex.
Lps_tab.c , lps_tab.h	Generated form bison.