Explanations: TAC Generation for miniMatlab

Grammar Augmentations

Auxiliary symbols M,N and K have been used in Augmentations :
M-> Used to store the next instruction in the quad list when it reduces to epsilon. It is used in backpatching.
N-> Used to generate goto quads whenever required. These are mainly required after statements to transfer control.
K-> Used when we have detected that the current declaration is of a function to change the current symbol table to a new table for the function and save the old one. This is needed as arguments are stored in the symbol table for that function.

Apart from these certain grammar rules have been changed to aid in handling matrices:

- Postfix_expression -> postfix_expression [expression] has been changed to Postfix_expression -> primary_expression [expression] [expression] since only 2-D matrices are allowed and for a function first the call takes place and then the Matrix is referenced.
- direct_declarator -> direct_declarator [assignment_expression_opt] has been replaced by :
  direct_declarator -> IDENTIFIER[assignment_expression] [assignment_expression] for the same reason.
- Because of the above change assignment_expression_opt becomes a useless non-terminal so all rules associated to it are removed.

Design of Attributes for Grammar Symbols :

GRAMMAR SYMBOL	ATTRIBUTE	EXPLANATION/STRUCTURE

IDENTIFER	id_attr	{
		Name
		Loc : Symbol table entry
		}

INT_CONSTANT ZERO_CONSTANT	intVal(int)	-
CHAR_CONST unary_operator assignment_operator	charVal(char)	-
FLOAT_CONST	dbVal(double)	-
STRING_LITERAL	strVal(char*)	-

expression_opt
parameter_type_list
parameter_list
parameter_declaration
primary_expression
postfix_expression
unary_expression
cast_expression
multiplicative_expression
additive_expression
shift_expression
relational_expression
equality_expression
AND_expression
exclusive_OR_expression
inclusive_OR_expression
logical_AND_expression
logical_OR_expression
conditional_expression
assignment_expression
expression
constant_expression
direct_declarator
init_declarator
init_declarator_list
init_declarator_list_opt
declarator
	exp(exp_attr)	{
		
		addr : address of the corresponding symbol table entry if present
		
		t_exp : type of expression / declarator
		
		mat : symbol table entry of the matrix in case we need to dereference before use
		
		truelist : used for boolean expression to backpatch
		
		falselist : used in backpatching
		
		isMattype(bool) : to know if we need to deference the matrix before use
		
		isPtrtype(bool) : to know if we need to deference the poiner before use
		
		isConst(bool) : to know if it is a static constant that is derived from primary_expression without passing through any expression where a new temp or quad gets generated
		}

N statement compound_statement selection_statement iteration_statement jump_statement block_item_list_opt block_item_list block_item	nextlist(pointer to a vector<int>)	Pointer to vector of integers which is used for backpatching

Type_specifier
K
	dec(dec_attr)	{
		t_dec : type of the declaration
		
		size : size in bytes of the declaration type
		}

argument_expression_list argument_expression_list_opt initializer_row initializer_row_list	arglist (pointer to a vector<exp_attr>)	Used for arguments when handling function calls or when initializing a matrix

initializer	inr(init)	{
		E(exp_attr) : if it is an assignment expression then this is used
		
		matlist(pointer to vector<exp_attr>) : when initializing a matrix, this is used to store the values
		}

M	instr(int)	To store a quad instruction index

Design and Implementation of Symbol Table and supporting data structures:

The ass4_15CS30043_translator.h file and ass4_15CS30043_translator.cxx have the design and implementation of the Symbol table and supporting data structures along with appropriate comments as explanations

Design and Implementation of Quad Array

The ass4_15CS30043_translator.h file and ass4_15CS30043_translator.cxx have the design and implementation of the Quad Array along with appropriate comments as explanations

Design and Implementation of Global Functions:

The ass4_15CS30043_translator.h file and ass4_15CS30043_translator.cxx have the design and implementation of the global functions along with appropriate comments as explanations

Limitations and Shortcomings:

1. A matrix cannot be dynamically allocated. Matrix mat[a][b]; is not allowed. Neither is Matrix[2+2][3+3]. Only Matrix mat[Integer][Integer] will work
2. Please provide return statements to functions for proper code generation especially for functions of type Matrix
3. Only 2D matrices are allowed
4. A function once declared cannot be defined because we are checking whether that name is already present in the symbol table. But we only have to support function definitions so that is anyway not required
5. Double to char conversions not supported
6. A lot of redundant gotos while backpatching
7. Pointer to matrices have some bugs especially when passed to functions as a parameter.
8. Simple pointers work fine but multi pointers are not supported on the LHS of an expression or with ++ and -- operations.
9. (*p)++ or ++(*p) do not work