PAF-KIET NORTH CAMPUS

COMPILER CONSTRUCTION

CLASS ID: (103411)

PROJECT REPORT

GROUP MEMBERS:

UMNA ALEEM	61638
AMNA ARSHAD	60041
SIDRA SAEED	60714

PROJECT DESCRIPTION:

We aim to construct a compiler for a subset of a programming language Mini C in 3 phases. Code Optimization and Machine Code generation is not the scope of the project. These phases include:

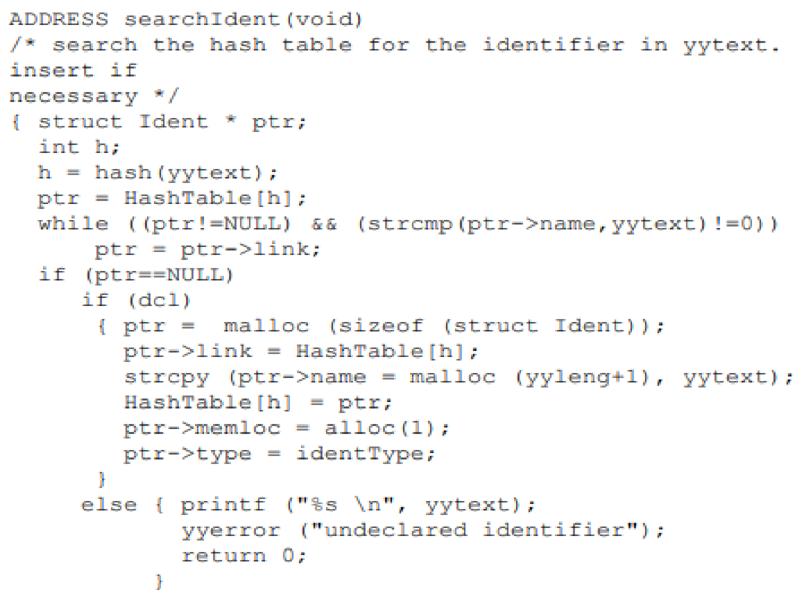
- Phase I: Lexical Analyzer (Laxer)
- Phase II: Syntax Analyzer (Parser)
- Phase III: Semantic Analyzer

SAMPLE LANGUAGE:

MINI C:

Mini C is a simple subset of the standard C language. It does not include arrays, structs, unions, files, sets, switch statements, do statements, or many of the low level operators. The only data types permitted are int and float.

SAMPLE CODE:



```
ADDRESS searchIdent(void)
/* search the hash table for the identifier in yytext.
insert if
necessary */
{ struct Ident * ptr;
  int h;
  h = hash(yytext);
  ptr = HashTable[h];
  while ((ptr!=NULL) && (strcmp(ptr->name,yytext)!=0))
      ptr = ptr->link;
  if (ptr==NULL)
     if (dcl)
      { ptr =  malloc (sizeof (struct Ident));
        ptr->link = HashTable[h];
        strcpy (ptr->name = malloc (yyleng+1), yytext);
        HashTable[h] = ptr;
        ptr->memloc = alloc(1);
        ptr->type = identType;
      }
     else { printf ("%s \n", yytext);
            yyerror ("undeclared identifier");
            return 0;
          }
```

LEXICAL SPECIFICATION:

```
INT    [0-9]+
EXP    ([eE][+-]?{INT})
NUM    ({INT}\.?{INT}?|\.{INT}){EXP}?
%{
#include <stdlib.h>
ADDRESS searchIdent(void);
ADDRESS searchNums(void);
ADDRESS alloc(int size);
%}
%start COMMENT1 COMMENT2
%%
<COMMENT1>.+       ;
<COMMENT1>\n       BEGIN 0;          /* end comment */
<COMMENT2>[^*]+    ;
<COMMENT2>\*[^/]   ;
<COMMENT2>\*\/     BEGIN 0;          /* end comment */
"//"         BEGIN COMMENT1;
"/*"         BEGIN COMMENT2;
int          {yylval.code = 2; return INT;}
float        {yylval.code = 3; return FLOAT;}
for          return FOR;
while        return WHILE;
if           return IF;
else         return ELSE;
"=="         {yylval.code = 1; return COMPARISON;}
\<           {yylval.code = 2; return COMPARISON;}
>            {yylval.code = 3; return COMPARISON;}
"<="         {yylval.code = 4; return COMPARISON;}
```

```
">="         {yylval.code = 5; return COMPARISON;}
"!="         {yylval.code = 6; return COMPARISON;}
[a-zA-Z][a-zA-Z0-9_]*    {yylval.address = searchIdent();
                         return IDENTIFIER;}
{NUM}        {yylval.address = searchNums(); return NUM;}
[ \t]        ;  /* white space */
\n           lineno++;         /* free format */
.            return yytext[0]; /* any other char */
%%
yywrap ()
{return 1;  /* terminate when reaching end of stdin */}
```

GRAMMER:

```
Function     →      Type identifier ( ArgList ) CompoundStmt
ArgList      →      Arg
                    | ArgList , Arg
Arg          →      Type identifier
Declaration  →      Type IdentList ;
Type         →      int
                    | float
IdentList    →      identifier , IdentList
                    identifier
Stmt         →      ForStmt
                    | WhileStmt
                    | Expr ;
                    | IfStmt
                    | CompoundStmt
                    | Declaration
                    | ;
ForStmt      →      for ( Expr ; OptExpr ; OptExpr ) Stmt
```

```
OptExpr        →      Expr
                      | ε
WhileStmt      →      while ( Expr ) Stmt

IfStmt         →      if ( Expr ) Stmt ElsePart
ElsePart       →      else Stmt
                      | ε
CompoundStmt→         { StmtList }
StmtList       →      StmtList  Stmt
                      | ε
Expr           →      identifier = Expr
                      | Rvalue
Rvalue         →      Rvalue Compare Mag
                      | Mag
Compare        →      == | < | > | <= | >= | !=
Mag            →      Mag + Term
                      | Mag - Term
                      | Term
Term           →      Term * Factor
                      | Term / Factor
                      | Factor
Factor         →      ( Expr )
                      | - Factor
                      | + Factor
                      | identifier
                      | number
```

<u>PROBLEMS FACED IN PROJECT:</u>

PROBLEM 1: NEW TO THE LANGUAGE

The language was new for all of us. It takes too much time to understand but we did our best to learn it and implement it.

PROBLEM 2: WINDOW CORRUPTED

When I installed virtual machine on my PC, windows got corrupted.

PROBLEM 3: PROJECT LOAD & LOCKDOWN

We all have project load on ourselves as well as FYP is also going on.