ΕΘΝΙΚΟ & ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ &
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΘΠ06 Μεταγλωττιστές
Εργασία Εξαμήνου:
Υλοποίηση ενός Μεταγλωττιστή για τη γλώσσα προγραμματισμού Robin

ΣΑΚΕΛΛΑΡΗ ΕΛΙΣΑΒΕΤ 1115200600152

ΠΕΡΙΕΧΟΜΕΝΑ

Λεκτικός αναλυτής

Ο λεκτικός αναλυτής που έχουμε υλοποιήσει αναγνωρίζει τα παρακάτω στοιχεία της γλώσσας

1. δεσμευμένα αναγνωριστικά char, int, float, else, if, main, return, void, while,record

2. απλά αναγνωριστικά που απαρτίζονται από γράμματα (A..Z, a..z),
ψηφία (0..9) και τον ειδικό χαρακτήρα _ (underscore).
Σαν λάθος αναγνωρίζεται η είσοδος id που αρχίζει από αριθμούς πχ int 33x = 33;

3. βασικούς τύπους δεδομένων char, int και float.
Σαν char αναγνωρίζονται οι απλοί χαρακτήρες πλην των ',",\
και οι ειδικοί χαρακτήρες \n \" \' \0 \t \\

4. ακέραιες σταθερές και σταθερές κινητής υποδιαστολής.

5. τα σύμβολα (,),[,],{,},;,&,=, ","

6. αριθμητικούς τελεστές

7. σχεσιακούς τελεστές

8. λογικούς τελεστές

9. οι πίνακες αναγνωρίζονται απο τα στοιχείο που τους απαρτίζουν. Δηλαδή όταν δίνεται array[20] αυτό αναγνωρίζεται
στα tokens, array,[,20,].

10. Συμβολοσειρές στις οποίες γίνεται έλεγχος για unterminated string όπως και για string πολλαπλών γραμμών. Μεσα στις συμβολοσειρές σε αυτό το επίπεδο δεχόμαστε ότι και αν δοθεί και ελέγχουμε μόνο για multiline strings
ή unterminated.

11. Η οδηγία προς τον λεκτικό αναλυτή #include
Το αρχείο που γίνεται include δεν μπορεί να κάνει άλλο include

12. Όπως και για τους πίνακες , οι σύνθετοι τύποι δεδομένων αναλύονται επίσης στα σύμβολα που τους απαρτίζουν. Πχ.

```
record point {
        int x;
        int y;

        };
```

αναλύεται στα record ,point, {,int,x,;,y,;,},;

Ο λεκτικός αναλυτής αναγνωρίζει τους χαρακτήρες και στέλνει στον συντακτικό αναλυτή τις αντίστοιχες λεκτικές μονάδες. Επίσης στέλνει και την τιμή της λεκτικής μονάδας που αναγνώρισε.

Κώδικας στο flex

```
letter  [a-zA-Z]
digit   [0-9]
underscore _
id      {letter}({letter}|{digit}|{underscore})*
integer {digit}+
float {digit}+\.{digit}+([eE][+-]?{digit}+)?

simple_char [^\\\'\"]
special_char \\[n\"\'0t\\]
non_printable_char [^{simple_char}]
char \'({simple_char}|{special_char})\'
invalid_special_char \'\\[^n\"\'0t\\]\'
empty_char      \'\'
non_printable  \'[^{simple_char}]\'
multibyte_char \'(.)+\'
unterminated_char \'([^(\')])*

illid ({integer}|{float})+(\.)*{id}+

%x IN_COMMENTS
%x INCLUDE
%x IN_STRING
%%


"int"           { return TK_INT_TYPE; }
"float"         { return TK_FLOAT_TYPE; }
"char"          { return TK_CHAR_TYPE; }
"else"          { return TK_ELSE;}
"if"            { return TK_IF; }
"main"          { return TK_MAIN; }
"return"        { return TK_RETURN; }
"void"          { return TK_VOID; }
"while"         { return TK_WHILE; }
"record"        { return TK_RECORD; }

"("             { return TK_LPAR; }
")"             { return TK_RPAR; }
"["             { return TK_LBRACKET; }
"]"             { return TK_RBRACKET; }
"{"             { return TK_LBRACE; }
"}"             { return TK_RBRACE; }

"+"             { return TK_PLUS; }
"-"             { return TK_MINUS; }
"*"             { return TK_MULT; }
"/"             { return TK_DIV; }
"%"             { return TK_MOD; }
```

```
"=="           { return TK_EQ; }
"!="           { return TK_DIFF; }
">"            { return TK_GR; }
"<"            { return TK_LESS; }
">="           { return TK_GOE; }
"<="           { return TK_LOE; }

"\|\|"         { return TK_OR; }
"&&"           { return TK_AND; }
"!"            { return TK_NOT; }

"&"            { return TK_AMP; }
";"            { return TK_SEMI; }
"\."           { return TK_SDOT; }
"="            { return TK_MAKE_EQ;}
","            { return TK_COMMA; }
"//"([^\n])*   { /* trwei ta // comments*/ }

{integer}      { return TK_INT; }
{float}        { return TK_FLOAT; }
{id}           { return TK_ID; }
{illid}        {
               cout<<"Illegal Id found"<<endl;
               return TK_ILLEGAL_ID;
}
{char}  {
       yyStr = yytext;
       cBuf = yyStr.substr(1,yyleng-2); // bgazoume ta ' '
       strcpy(yytext,cBuf.c_str());
       return TK_CHAR;
}
{invalid_special_char} {
       cout<<"Error:This special char is not valid:"<<yytext<<endl;
        return TK_ILLEGAL_CHAR;

}
{non_printable}       {
       cout<<"Error:This  character is not valid(non-printable):"<<yytext<<endl;
        return TK_ILLEGAL_CHAR;
}

{multibyte_char}      {
       cout<<"Error:Multibyte char constand found:"<<yytext<<endl;
       return TK_ILLEGAL_CHAR;
}

{empty_char}   {
       cout<<"Error:Empty char constand found:"<<yytext<<endl;
        return TK_ILLEGAL_CHAR;
}

{unterminated_char}     {
        // tha mpei edw mono ean exoyme ' mpla mpla mexri to eof
        cout<<"Unterminated or not valid char constant found, exiting"<<endl;
        yyterminate();
}
<IN_STRING>
{

       ([^\"])*\" { //telos tou string
              yyStr = yytext;
              sBuf = yyStr.substr(0,yyleng-1);
              strcpy(yytext,sBuf.c_str());
              BEGIN(INITIAL);
              yylval.stringValue = new char[strlen(yytext)+1];
              strcpy(yylval.stringValue, yytext);
              return TK_STRING;
       }
}

<IN_STRING>\\[^tn0\'\"\\] {

        cout<<"unknown sequence, exiting"<<endl;
        BEGIN(INITIAL);
       // yyterminate();
}
```

```
<IN_STRING>\n {
        cout<<"Multiline string error, exiting"<<endl;
        BEGIN(INITIAL);
        // yyterminate();
}

<IN_STRING><<EOF>> {
        cout<<"Unterminated string, exiting"<<endl;
        BEGIN(INITIAL);
         // yyterminate();
}

[ \t\n]  /*ignores spaces,tabs and newlines */

<INITIAL>
{
        "/*"           BEGIN(IN_COMMENTS);
        "\""           BEGIN(IN_STRING);
}

<IN_COMMENTS>{
        "*"+"/"      BEGIN(INITIAL); // an brethei * akolouthoumeno apo /
        [^*]+       // otidipote brethei ektos apo * to trwei
        "*"+[^*/]*   // an brethei * to opoio den akoloutheitai apo / i (*/) to trwei
}

<IN_COMMENTS><<EOF>> {
        cout<<"Unterminated comments, exiting"<<endl;
        yyterminate();
}



"#include" {BEGIN(INCLUDE);}

<INCLUDE>[ \t]  /* eat the whitespaces & tabs*/

<INCLUDE>
{
        \"(.)*\" {
                if ( include_stack_ptr >= MAX_INCLUDE_DEPTH ) {
                        cout<<"Includes nested too deeply"<<endl;
                        num_errors++;
                        exit( 1 );
                }

                 yyStr = yytext;
                 sBuf = path+yyStr.substr(1,yyleng-2);

                FILE *yyinb;
                yyinb = fopen(sBuf.c_str(),"r");
                 if ( yyinb == NULL ) {
                          cout<<"Error opening file:"<<sBuf<<endl;
                        num_errors++;
                        BEGIN(INITIAL);
                }

                else {
                        include_stack[include_stack_ptr++] = YY_CURRENT_BUFFER; /*saving
last file*/
                        yyin = yyinb;
                        yy_switch_to_buffer(yy_create_buffer( yyin, YY_BUF_SIZE ));
                        BEGIN(INITIAL);
                }
        }
}
<INCLUDE>
{
        [^\"$] {
                 cout<<"Error in included file name ,propably unterminated file name
given"<<endl;
                // yyterminate();
        }
}
```

```
<<EOF>> {
        if ( --include_stack_ptr < 0 ) {
                yyterminate();
        }
        else {
                yy_delete_buffer( YY_CURRENT_BUFFER );
                yy_switch_to_buffer( include_stack[include_stack_ptr] );
        }
}

%%
```

Τα λάθη που αναγνωρίζονται απο τον λεκτικό αναλυτή είναι

1. λάθος μορφής αναγνωριστικά , πχ αναγνωριστικά που ξεκινάνε απο αριθμό 111a.

2. μη τερματισμός σχολίων

3. μη τερματισμός συμβολοσειράς

4. συμβολοσειρά πολλαπλών γραμμών

5. Σαν λάθος chars πιάνονται τα εξής
   1) multibyte character πχ 'aaaa'
   2) invalid_special_character πχ '\q'
   3) unterminated_character πχ 'a
   4) empty character πχ ''
   5) και invalid απλός χαρακτήρας όπως ''' ή '"'

Συντακτικός αναλυτής

Γραμματική στον bison

```
program : function_prototype_plus program_header compound_statement
          function_definition_star
        | record_definition_plus program_header compound_statement
          function_definition_star
        | function_definition_plus program_header compound_statement
          function_definition_star
        | variable_definition_plus program_header compound_statement
          function_definition_star
        | program_header compound_statement function_definition_star
        ;


function_definition_star : function_definition
function_definition_star
                         | /*empty*/
                         ;

function_prototype_plus : function_prototype
                         | function_prototype_plus function_prototype
                         ;

record_definition_plus : record_definition
                        | record_definition_plus record_definition
                        ;

function_definition_plus : function_definition
                          | function_definition_plus
function_definition
                          ;

variable_definition_plus : variable_definition
                          | variable_definition_plus
variable_definition
                          ;

program_header  : TK_VOID TK_MAIN TK_LPAR TK_RPAR
                | TK_VOID TK_MAIN error TK_RPAR
        | TK_VOID TK_MAIN TK_LPAR error TK_RPAR
```

```
               ;

function_prototype : function_header TK_SEMI;
                   | function_header error TK_SEMI
                   ;

record_definition : composite_type TK_LBRACE primitive_type_def_plus
                    TK_RBRACE TK_SEMI
                  | composite_type TK_LBRACE error TK_RBRACE TK_SEMI
                  | composite_type error TK_RBRACE TK_SEMI
                  | composite_type TK_LBRACE error TK_SEMI
                  ;
function_definition : function_header compound_statement
                    ;
variable_definition : primitive_type_def
                    | composite_type_def
                    ;

function_header : data_type TK_ID TK_LPAR TK_RPAR
                | TK_VOID TK_ID TK_LPAR TK_RPAR
                | composite_type TK_ID TK_LPAR TK_RPAR
                | data_type TK_ID TK_LPAR formal_params TK_RPAR
                | TK_VOID TK_ID TK_LPAR formal_params TK_RPAR
                | composite_type TK_ID TK_LPAR formal_params TK_RPAR
                | data_type TK_ID TK_LPAR error TK_RPAR
                | TK_VOID TK_ID TK_LPAR error  TK_RPAR
                | composite_type TK_ID TK_LPAR error TK_RPAR
                ;

composite_type :TK_RECORD TK_ID;

primitive_type_def_plus : primitive_type_def
                        | primitive_type_def_plus primitive_type_def
                        ;

primitive_type_def : data_type def_onevar_plus TK_SEMI
                   | data_type error TK_SEMI
                    ;

composite_type_def : composite_type def_onevar_plus TK_SEMI
                   | composite_type error TK_SEMI
                    ;

def_onevar_plus : def_one_variable
                | def_one_variable TK_COMMA def_onevar_plus
                ;

def_one_variable : TK_ID
                 | TK_ID TK_LBRACKET TK_INT TK_RBRACKET
                 | TK_ID TK_LBRACKET error TK_RBRACKET
                  ;


formal_params : formal_parameter
              | formal_parameter TK_COMMA formal_params
              ;

formal_parameter : data_type TK_ID
                 | data_type TK_AMP TK_ID
                 | data_type TK_AMP TK_ID TK_LBRACKET TK_RBRACKET
                 | data_type TK_ID TK_LBRACKET TK_RBRACKET
```

```
                  | composite_type TK_ID
                  | composite_type TK_AMP TK_ID
                  | composite_type TK_AMP TK_ID TK_LBRACKET
                        TK_RBRACKET
                  | composite_type TK_ID TK_LBRACKET TK_RBRACKET
                 ;

variable_definition_star : /*empty*/
                         | variable_definition_star
variable_definition
                         ;

statement_star : /*empty*/
               | statement_star statement
               ;


statement : assignment
          | if_statement
          | while_statement
          | return_statement
          | compound_statement
          | TK_SEMI /*empty statement*/
          | void_function_call
          ;


void_function_call : function_call TK_SEMI
                   ;

function_call : TK_ID TK_LPAR TK_RPAR
              | TK_ID TK_LPAR actual_params TK_RPAR
              | TK_ID TK_LPAR error TK_RPAR
              ;

actual_params : actual_parameter
              | actual_parameter TK_COMMA actual_params
              ;

actual_parameter : expression
                 | TK_STRING
                 ;

if_statement : TK_IF TK_LPAR b_expression TK_RPAR statement
             | TK_IF TK_LPAR b_expression TK_RPAR statement TK_ELSE
                    statement
             | TK_IF TK_LPAR error TK_RPAR statement
             | TK_IF TK_LPAR error TK_RPAR statement TK_ELSE
                    statement
             | TK_IF TK_LPAR b_expression error statement
             | TK_IF TK_LPAR b_expression error statement TK_ELSE
                    statement
             | TK_IF error TK_RPAR statement
             | TK_IF error TK_RPAR statement TK_ELSE statement
             ;

while_statement : TK_WHILE TK_LPAR b_expression TK_RPAR statement
                | TK_WHILE TK_LPAR error TK_RPAR statement
                | TK_WHILE  error TK_RPAR statement
                | TK_WHILE TK_LPAR error statement
                ;
```

```
return_statement : TK_RETURN TK_SEMI
                 | TK_RETURN expression TK_SEMI
                | TK_RETURN error TK_SEMI
                 ;

assignment :  comp_l_value TK_MAKE_EQ expression TK_SEMI
           |  comp_l_value TK_MAKE_EQ error TK_SEMI
           |  comp_l_value error expression TK_SEMI
           ;

comp_l_value : TK_ID TK_SDOT l_value
             | l_value
             | TK_ID TK_SDOT error TK_SEMI
             ;


l_value : TK_ID
        | TK_ID TK_LBRACKET  expression TK_RBRACKET
        | TK_ID TK_LBRACKET error TK_RBRACKET
        ;

compound_statement : TK_LBRACE variable_definition_star
statement_star TK_RBRACE;
                   | TK_LBRACE error TK_RBRACE
                   ;

data_type : TK_INT_TYPE
          | TK_FLOAT_TYPE
          | TK_CHAR_TYPE
          ;

b_expression : b_expression TK_OR b_expression
             | b_expression TK_AND b_expression
             | TK_NOT b_expression
             | expression TK_EQ expression
             | expression TK_DIFF expression
             | expression TK_GR expression
             | expression TK_LESS expression
             | expression TK_GOE expression
             | expression TK_LOE expression
             ;



expression : expression TK_ADD expression
           | expression TK_SUB expression
           | expression TK_MULT expression
           | expression TK_DIV expression
           | expression TK_MOD expression
           | TK_ADD expression %prec TK_PLUS
           | TK_SUB expression %prec TK_MINUS
           | TK_INT
           | TK_CHAR
           | comp_l_value
           | function_call
           ;
%%
```

Συγκρούσεις της γραμματικής που διορθώσαμε στο bison

<function-header> ::= <return-type> <id>
“**(**” [<formal-params>] “**)**”

Διορθώθηκε σε

```
function_header : data_type TK_ID TK_LPAR TK_RPAR
                | TK_VOID TK_ID TK_LPAR TK_RPAR
                | composite_type TK_ID TK_LPAR TK_RPAR
                | data_type TK_ID TK_LPAR formal_params TK_RPAR
                | TK_VOID TK_ID TK_LPAR formal_params TK_RPAR
                        | composite_type TK_ID TK_LPAR
formal_params TK_RPAR
```

(το return-type δεν το βάλαμε γιατί προκαλούσε συγκρούσεις.)

και όπου στη γραμματική είχαμε [κάτι] έπρεπε να γραφτούν δυο διαφορετικά δεξιά μέλη του κανόνα ένα χωρίς αυτά που είχε το [ ] και έναν μαζί με αυτά πχ.

```
<formal-parameter> ::= <data-type> [“&”] <id> [“[” “]”]
| <composite-type> [“&”] <id> [“[” “]”]
```

έγινε
```
formal_parameter : data_type TK_ID
          | data_type TK_AMP TK_ID
          | data_type TK_AMP TK_ID TK_LBRACKET TK_RBRACKET
          | data_type TK_ID TK_LBRACKET TK_RBRACKET
          | composite_type TK_ID
          | composite_type TK_AMP TK_ID
          | composite_type TK_AMP TK_ID TK_LBRACKET TK_RBRACKET
          | composite_type TK_ID TK_LBRACKET TK_RBRACKET
                  ;
```

Επίσης συγκρούσεις που υπήρχαν στο expression | b_expression λόγω προτεραιότητας διορθώθηκαν με τις εξής προτεραιότητες

```
%left TK_COMMA
%right TK_MAKE_EQ
%left TK_OR
%left TK_AND
%left TK_EQ TK_DIFF
%left TK_GR TK_LESS TK_GOE TK_LOE
%left TK_ADD TK_SUB
%left TK_MULT TK_DIV TK_MOD
%right TK_NOT TK_PLUS TK_MINUS TK_AMP
%left TK_LPAR TK_RPAR TK_LBRACKET TK_RBRACKET TK_SDOT
```

Επίσης το συμβολο + - όταν χρησιμοποιείται σαν πρόσημο παίρνει την προτεραιότητα του TK_ADD | TK_SUB με την χρήση του %prec
Τέλος συγκρούσεις προκάλεσε και το if_else statement οι οποίες θεωρήσαμε ότι δεν επηρεάζουν την γραμματική μας οπότε χρησιμοποιήσαμε την λειτουργία %expect 4 /*if -else kai gia ta error-recovery tous */

<u>Συντακτικά λάθη που αναγνωρίζονται</u>

- λάθος program header: void main) {
  void main (asda) {..

- λάθος μεταβλητές: int 111y

- λάθη σε συνθήκες : while (y z) { , if (a a) ..
- ξεχασμένες παρενθέσεις while (1 ==2 { .., if 3aa) {

- λάθη σε function call : suball(glob\,glob2,glob3,res);

- λάθη σε δηλωσεις μεταβλητων : int glob;

- λάθη σε αναθέσεις τιμών : y = y 1, y a;
- λάθη σε πίνακες int x[a0];
- λάθη στα records

Πίνακας συμβόλων

Αρχεία SymbolTable.h, SymbolTable.cpp
Ο πίνακας συμβόλων είναι μία κλάση, η οποία κρατάει πληροφορίες όπως το όνομα της εμβέλειας και το αν είναι η global εμβέλεια ή όχι.
Περιέχει επίσης maps για τις μεταβλητές, τις συναρτήσεις και τα records της εμβέλειάς του. Επίσης περιέχει συναρτήσεις εισαγωγής και αναζήτησης στον πίνακα συμβόλων μεταβλητών, συναρτήσεων και records.

Αρχεία Node.h, Node.cpp
Το θέμα της εμβέλειας αντιμετωπίζεται με τη βοήθεια της δενδρικής δομής, όπου κάθε κόμβος του δένδρου αναπαριστά μία εμβέλεια.
Κάθε κόμβος περιέχει ένα πίνακα συμβόλων όπου κρατούνται τα ονόματα και όλες οι απαραίτητες πληροφορίες σχετικά με αυτά.
Όποτε συναντάται μία δήλωση ονόματος, αυτό εισάγεται στην εμβέλεια που βρίσκεται (πίνακας συμβόλων του συγκεκριμένου κόμβου), και όποτε ξανά εντοπίζεται, για χρήση, γίνεται αναζήτηση μέσα σε αυτή την εμβέλεια.

Αρχεία Object.h , Object.cpp
Οι μεταβλητές , οι συναρτήσεις και τα records υλοποιούνται από κλάσεις τύπου Object , και κάθε μία από αυτές κρατάει τις απαραίτητες πληροφορίες ανάλογα με τον τύπο της.
Μεταβλητές
- Τύπος

- Όνομα
- Αν είναι πίνακας
- Μέγεθος πίνακα
- Και αν είναι κατ’ αναφορά

Συναρτήσεις

- Παράμετροι
- Τύπος
- Όνομα
- Αν είναι τύπου record το όνομα του record

Records

- Μέλη
- Όνομα
- Όνομα του record της δήλωσης
- Αν είναι πίνακας
- Μέγεθος πίνακα
- Αν είναι κατ’ αναφορά
- Τα offset των μελών
- Ένα καθολικό offset

Σημασιολογικά λάθη που εντοπίζονται

1.Δήλωση πίνακα με αρνητικό μέγεθος ή 0 int ar[-100];

2.Δήλωση πίνακα με μέγεθος μεταβλητο ή πράξη int ar[3+100 -x];

3.Μη δήλωση της συνάρτησης main

4.Απαγορέυονται όλες οι αναθέσεις και οι πράξεις μεταξύ διαφορετικών τύπων πχ
int x = 'a', int x = 3.3; float f1 = 'a' char c1 = 3; char c2 = 3.3;

5.Επιτρέπεται η ανάθεση int σε float πχ int x , float f = 3 + 5 * x;

6.Απαγορεύεται να αναφερθείς σε στοιχείο πίνακα με αρνητικό index int x; int ar[100]; x = ar[-1];

7.Αναφορά σε στοιχείο πίνακα το οποίο είναι out of bounds , πχ int ar[100]; int x; x = ar[100]+ar[101]; 2 out of bounds

8.Η δήλωση μιας μεταβλητής που είχε δηλωθεί ξανά στο ίδιο scope πχ int x; char x;

9.Αν δηλώση κάποιος μια μεταβλητή το όνομα της οποίας είχε δηλωθεί σαν record , και το αντίστροφο πχ
record Point {int x;}; void main() { record Point x; int x;}

10. Η δήλωση ενός "αντικειμένου" ενός record Που δεν έχει δηλωθεί (το record definition) πχ record Undefined x;

11. Η χρήση μιας μεταβλητής που δεν είναι array με index πχ , int x; x = x[10];

12. Η χρήση μιας μεταβλητής που δεν έχει δηλωθεί int x; x = y;

11. Η χρήση σε πράξεις του ονόματος ενός record πχ, record Point x; int y; y=x;

12. Οτι ισχύει για πράξεις-αναθέσεις μεταξύ μεταβλητών ισχυεί και για τα στοιχεία ενός record.

13. Η χρήση ενός μέλους ενός record που δεν υπάρχει πχ.
record Point{ int x; }; record Point x; int y; y = x.aa;

14. Binary operations == , != , >= , <= , < , > μεταξύ τύπων που δεν μπορούν να συγκριθούν px if (3 >= 'a')
57:Error:Greater or Equal check operation between a int And a char operand is not allowed!
ή με record names
63:Error:Equal check operation between a record Point And a int operand is not allowed!

15. Διαιρέσεις με το 0 ή το 0.0, χρήση float σε mod calculation.

16. Απαγορευονται πράξεις μεταξύ διαφορετικών τύπων εκτός κι αν πρόκειται για int-float και το result είναι float.
int x; float y; char c; x = 3 *x *y (error), y = 3*x*y (ok), c = x + y (error)

(Σημείωση. Επιτρέπεται η δήλωση πίνακα με χρήση προσήμου + int ar[+100])

Δηλώσεις ονομάτων

Δήλωση μεταβλητής

Όταν συναντήσουμε μια δήλωση μεταβλητής (`primitive_type_def`) ελέγχεται αρχικά αν έχει ξανά δηλωθεί κάποια με το ίδιο όνομα ή κάποιο record με το ίδιο όνομα, στην ίδια

εμβέλεια. Αν ναι, εκτυπώνεται μήνυμα λάθους. Αν όχι, εισάγεται στον πίνακα συμβόλων της τρέχουσας εμβέλειας.

Ο αντίστοιχος κώδικας φαίνεται παρακάτω.

```
primitive_type_def :

        data_type def_onevar_plus TK_SEMI {

              vector<Variable *>::iterator it;
              Variable *ptr;

              for (it = $2->begin(); it != $2->end(); it++) {
                     string name = (*it)->getName();
                     if ((ptr = curNode->lookupVariable(DECLARATION, name)) == NULL) {

                            /* psaxnw na dw an exei dilwthei sitn idia embeleia record me
idio name*/
                            Record *tmpr;
                            if ((tmpr = curNode->lookupRecord(DECLARATION, name) )==
NULL) {
                                   (*it)->setType($1);
                                   curNode->insertVariable(name, *it);
                            }
                            else {
                                   Error = name+" is previously declared as record
"+tmpr->getRecordName();
                                   delete (*it);
                                   $2->erase(it);
                                   it--;
                                   yyerror(Error);
                                    sem_errors++;
                                    yyerrok;
                            }
                     }
                     else {
                            Error = "Variable "+name+" has been already declared as ";
                            int type = ptr->getType();
                            if( type == CHAR )
                                   Error += "char";
                            else if (type == INT)
                                   Error += "int";
                            else if (type == FLOAT)
                                   Error += "float";
                            yyerror(Error);
                            sem_errors++;
                            yyerrok;
                     }
              }
              delete $2;
        }
```

```
composite_type_def :

       composite_type def_onevar_plus TK_SEMI  {
              Record *Re = NULL;
              string recName = $1.Val;
              if( (Re = root->lookupRecord(DECLARATION, recName) ) != NULL)
              {
                     for(int i = 0; i < $2->size(); i++)
                     {
                            string name = (*$2)[i]->getName();
                            if( curNode->lookupRecord(DECLARATION,name) == NULL)
                            {
                              if(curNode->lookupVariable(DECLARATION, name) == NULL) {
                                     Record *temp = new Record(name,recName,Re-
>getMembers(),(*$2)[i]->getArraySize(), Re->getOffset(), Re->getOffsets());
                                     curNode->insertRecord(name,temp);
                              }
                              else {
                                     Error = name+" is previously declared as a variable";
                                     yyerror(Error);
                                     sem_errors++;
                                     yyerrok;
                              }
                            }

                            else {
                                   Error = name+" has already been declared as a record";
                                   yyerror(Error);
                                     sem_errors++;
                                     yyerrok;
                            }
                     }
              }
              else {
                     Error = "Record "+recName+" has not been declared";
                     yyerror(Error);
                      sem_errors++;
                      yyerrok;
              }

              delete $1.Val;
              for (int i = 0; i < $2->size(); i++)
                     delete (*$2)[i];
              delete $2;
       }
```

Δηλώσεις εγγραφών

Η ίδια διαδικασία περίπου ακολουθείται και κατά την δήλωση εγγραφών. Σε αυτή την περίπτωση ελέγχεται αν έχει ξαναδηλωθεί στην ίδια εμβέλεια κάποια άλλη εγγραφή ή μεταβλητή με το ίδιο όνομα. Επίσης ελέγχεται αν το όνομα εγγραφής έχει οριστεί πριν γίνει δήλωση κάποιας εγγραφής αυτού του τύπου.

Κώδικας

Δηλώσεις συναρτήσεων

Όταν δηλώνεται μια συνάρτηση αρχικά ελέγχεται αν εχει ξαναδηλωθεί κάποια με το ίδιο όνομα. Αν δεν έχει ξαναδηλωθεί κάποια με το ίδιο όνομα, αυτή η συνάρτηση εισάγεται στην εμβέλεια.

Κώδικας

```
function_prototype :
        function_header TK_SEMI{
                if ($1.type != ERROR_TYPE){
                Function *f;
                if((f = root->lookupFunction($1.name)) == NULL){
                Function *temp;
                if ($1.type == RECORD)
                        temp  = new Function($1.type,$1.Val,$1.name,
 $1.tempvars);
                else
                        temp =  new Function($1.type, " ",$1.name,
 $1.tempvars);

                root->insertFunction($1.name,temp);
                function_protos.push_back($1.name);
                }else{
                string name = $1.name;
                Error = "Function "+name+" has already been defined as
 "+f->TakePrototypeString();
                yyerror(Error);
                yyerrok;
                sem_errors++;
                }
        }
        if ($1.name != NULL) delete $1.name;
        if ($1.Val != NULL)  delete $1.Val;
}
| function_header error TK_SEMI  {yyerror("Error in function prototype,
 function not declared"); num_errors++; yyerrok;
                if ($1.name != NULL) delete $1.name;
                if ($1.Val != NULL)  delete $1.Val;
}
```

Συνολικά γίνεται έλεγχος για το αν όλες οι συναρτήσεις που έχουν δηλωθεί είναι και ορισμένες. Κατά τον ορισμό της συνάρτησης ελέγχεται αν έχει δηλωθεί, αν ο αριθμός και ο τύπος των παραμέτρων είναι σωστός, και αν ο τύπος επιστροφής είναι σωστός. Αυτά ελέγχονται στον κανόνα function_front. Μερικά κομμάτια του κώδικα φαίνονται παρακάτω

Κώδικας
Έλεγχος ανάλογα με τον τύπο της συνάρτησης

```
if((f = root->lookupFunction(name)) != NULL){
                        if( f->getType() != $1.type){
                                string tp = getType($1.type);
                                Error = "Conflicting return types of
function "+name+" , previous declaration as "+f->TakePrototypeString();
                                yyerror(Error);
                                sem_errors++;
                                yyerrok;
                                $$.type = ERROR_TYPE;
                                correct = false;
                        }else if (f->getType() == RECORD) {
                                string vl = " ";
                                if ($1.Val != NULL) vl = $1.Val;
                                if (f->getRecName() != vl) {
                        Error = "Conflicting return types of function
"+name+" as record "+vl+" , previous declaration as "+f-
>TakePrototypeString();
                                yyerror(Error);
                                sem_errors++;
                                yyerrok;
                                $$.type = ERROR_TYPE;
                                correct = false;
                        }
                }
```

Εντοπισμός λαθών σχετικά με τις δηλώσεις (συνέχεια)

17. Δήλωση ενός empty record record Point { } - Δήλωση ενός record που ήδη υπάρχει

10:Error:Cannot declare an empty record!

24:Error:Record Point has already been defined!

18. Μη ύπαρξη ορισμού σε συνάρτηση που δηλώθηκε το πρωτοτυπό της. πχ int fun(); void main(){}

19. Δήλωση συνάρτησης και ορισμός της με διαφορετικά λιγότερα περισότερα η διαφορετικού τύπου ορίσματα.

```
int fun2( int ax[], char ac[], float &ay[]);
void main() {}
 int fun2(int x) { return x;}
```

19:Error:Wrong number of parameters for function fun2 , previous declaration as
(Prototype) : int fun2(int[], char[], float &[])!

20. Η Δήλωση μιας συνάρτησης που έχει ξαναδηλωθεί int fun3(); void fun3();

3:Error:Function fun3 has already been defined as (Prototype) : int fun3()!

21. Warnings εαν έχουμε δηλώσει μια συνάρτηση με τύπο != VOID και δεν επιστρέφει τίποτα

```
int fun() { } just warning
```

22. Error εάν έχω δηλώσει μια συνάρτηση που έχει τύπο πχ int και γυρνάω κάτι διαφορετικό πχ return 'a' ή return 3.3;

(!) Ελέγχουμε όλα τα return και αυτά σε compounts statements δηλαδή όλα όσα έχει μια συνάρτηση και πρέπει να συμφωνούν με το data type της.

```
πχ      char fun4() {
```

πιάνει και τα 3 false return statements

```
                int ar[10];
                if ( 3 == 3) return 3;
                else return 'c';
                return 3.3;
                return ar;
        }
}
```

23. Δήλωση συνάρτησης που επιστρέφει πίνακα. int []fun() (simeiwsi , επιτρέπεται η χρήση record Point σαν return typ;

```
record Point {  int x;};
record Point funnn(){
        record Point x;
                return x;}
```

24.Error output for wrong function definition

```
int fun2( int ax[], char ac[], float &ay[]);
int fun2(int x, int y, int z) {}
```

41:Error:Conflicting type for parameter no:3 of function fun2 previous declared as (float & ay[])!

41:Error:Conflicting types (array) for parameter no:3 was declared as (float & ay[])!

41:Error:Conflicting types (ref) for parameter no:3 was declared as (float & ay[])!

41:Error:Conflicting type for parameter no:2 of function fun2 previous declared as (char ac[])!
41:Error:Conflicting types (array) for parameter no:2 was declared as (char ac[])!
41:Error:Conflicting types (array) for parameter no:1 was declared as (int ax[])!

25. Η χρήση στο πρωτότυπο ή στη δήλωση μιας συνάρτησης παραμέτρου με το ίδιο όνομα
int fun(int x, int x); int fun (record Point x, int x) { }

26. if /while statemetns πρέπει να έχουν μόνο binary types σαν condition (apo prin)

27. Χρήση array σαν παράμετρο συνάρτησης με fixed size ; int fun(char ar[3])

28. Χρήση μιας συνάρτησης που δεν έχει δηλωθεί στο παραπάνω scope πχ.
void main() { fun10(); } void fun10() { } απαγορεύεται
void fun3(); void main() { fun3();} void fun4() { }; void fun3() { fun4();} επιτρέπεται

29. Χρήση ενός ονόματος που δηλωθηκε σαν συνάρτηση αλλά πχ ξαναδηλώθηκε μέσα στην main σαν variable
void fun3(){ }; void main() { int fun3; fun3(); }
7:Error:called object fun5 is not a function,is a variable!

30. Χρήση συνάρτησης που έχει παραμέτρους και την καλούμε χωρίς πχ void fun(int x) {} void main() { fun();}

31. Χρήση συνάρτησης που παίρνει ορίσματα πχ 1 κ την καλέσαμε με λάθος αριθμό ορισμάτων πχ
void fun(int x) {} void main() { fun(3,2); }

32. Χρήση συνάρτησης με ίδιο αριθμό ορισμάτων αλλά με λάθος τύπο πχ
void fun(int x, char &y){ } void main() { fun(3,3.3); }
19:Error:Wrong type of parameter y for function fun!
void errfun(int ar[], int x, char ac[], float y){ }
void main() { char ac[100];
errfun(ac, "sadasda" , " asdasd", "asda");
}
28:Error:Wrong type of parameter y for function errfun!
28:Error:Wrong type of parameter x for function errfun!
28:Error:Wrong type of parameter ar for function errfun!

33.Ότι ισχύει για πράξεις μεταξύ μεταβλητών και μελών ενός record ισχυεί και για την επιστροφή τύπων συνάρτησης

πχ,

int fun(){} void main() { int x; x = 3*fun();} /* ok*/

float fun() {} void main() { int x; x = fun()*2; /*not ok*/

34.Επιτρέπεται να περάσουμε σε function call με παράμετρο τύπο float ένα int ή μια μεταβλητή int με την προϋπόθεση ότι δεν είναι παράμετρος by reference γιατί τότε πρέπει να είναι αποκλειστικά ιδίου τύπου

Ενδιάμεσος κώδικας για τις υπόλοιπες εντολές

Για τη δημιουργία του ενδιάμεσου κώδικα έχουμε χρησιμοποιήσει και πάλι μία κλάση, η οποία είναι μια τετράδα. Αυτή κρατάει το σύμβολο της πράξης, τους τελεστές, την ετικέτα της, καθώς και τις λίστες truelist και falselist.

Ενδιάμεσος κώδικας δημιουργείται στις παρακάτω εντολές

1. Πράξεις

   Πχ κώδικας δημιουργίας τετράδας για πρόσθεση

```
if(correct){

                    fquad.emit("+",$1.place,
$3.place,result);
                    $$.place = new char[result.length()
+1];
                    strcpy($$.place,result.c_str());
             }
```

Ενδιάμεσος κώδικας

```
z = 2 + 3;      ⟶      9:  +, 2, 3, $7
                       10:  =, $7, -, z
```

2. Λογικές εκφράσεις

   Πχ κώδικας λογικής έκφρασης ‘==’

```
if(correct){

                  fquad.emit("+",$1.place,$3.place,result);
                  $$.place = new char[result.length()+1];
                  strcpy($$.place,result.c_str());
            }
```

3. Προσπέλαση στοιχείου πίνακα
Υπολογίζεται το offset του στοιχείου από την αρχή του πίνακα

```
stringstream str;
str << index;
string result = "$"; /*ask new temp*/
if ($$.type == INT)
       fquad.emit("*",str.str(),"4",result);
else if ($$.type == FLOAT)
       fquad.emit("*",str.str(),"4",result);
else if ($$.type == CHAR)
       fquad.emit("*",str.str(),"1",result);
string result2 = "$";
       fquad.emit("array", name, result, result2);
$$.place = new char[result2.length()+1];
strcpy($$.place,result2.c_str());
```

```
z = x[1] +4;

11:  *, 1, 4, $8
12:  array, x, $8, $9
```

4.Προσπέλαση μελών εγγραφής
Στην κλάση Record κρατάμε ένα vector από ακεραίους που αντιστοιχούν στα offset των μελών της εγγραφής. Δηλαδή έστω ότι έχουμε την εγγραφή

```
record point{
        int x;
        int y[4];
        char c;
};
```

Το x έχει offset 0 ο πίνακας y έχει offset 4 και το c έχει offset 20. Οπότε όταν θέλουμε να δημιουργήσουμε τις τετράδες για τα μέλη των εγγραφών, δουλεύουμε όπως περίπου με τα μέλη του πίνακα.

Στην περίπτωση που το μέλος στο οποίο αναφερόμαστε δεν είναι πίνακας, παίρνουμε το offset και βάζουμε το περιεχόμενο σε έναν καταχωρητή έχοντας ως καταχωρητή βάσης την εγγραφή.

```
int offset = temp->searchOffset(name);
stringstream ss;
 ss << offset;
 ss.str();
string result = "$";
fquad.emit("array", $1,ss.str(), result);
$$.place = new char[result.length()+1];
strcpy($$.place,result.c_str());
```

Αν το μέλος είναι πίνακας βρίσκουμε πρώτα την μετατόπιση από την αρχή του πίνακα και αφού την προσθέσουμε στην μετατόπιση

```
int offset = temp->searchOffset(name);
stringstream ss;
ss << offset;
string result = "$";

if ($$.type == INT)
       fquad.emit("*",$3.indexPlace,"4",result);
else if ($$.type == FLOAT)
       fquad.emit("*",$3.indexPlace,"4",result);
else if ($$.type == CHAR)
       fquad.emit("*",$3.indexPlace,"1",result);

string result2 = "$";
fquad.emit("+",result, ss.str(),result2);
string result3 ="$";
fquad.emit("array", result2, $1, result3);

$$.place = new char[result3.length()+1];
strcpy($$.place,result3.c_str());
```

Παράδειγμα

```
record point{
        int x,y,z;
        int a[40];
        int b[40];
        char c;
};
void main() {

record point y;

int i;

y.a[i] = y.a[i] +1;
}
```

Ενδιάμεσος κώδικας

```
1:  unit, main, -, -
2:  *, i, 4, $1
3:  +, $1, 12, $2
4:  array, $2, y, $3
5:  *, i, 4, $4
6:  +, $4, 12, $5
7:  array, $5, y, $6
8:  +, $6, 1, $7
9:  =, $7, -, $3
10:  endu, main, -, -
```

4. Δομή if-else

Χρησιμοποιούμε την τεχνική του backpatching για να συμπληρώσουμε τις τετράδες με τις κατάλληλες ετικέτες, όταν πρόκειται για κάποια δομή if-else. Στον κανόνα if_statement τα MQ και MQ2 κρατάνε τις ετικέτες των εντολών που πρέπει να συμπληρωθούν σε κάθε περίπτωση.

Κώδικας

If

```
if_statement :
       TK_IF TK_LPAR b_expression TK_RPAR MQ statement MQ {
              if ($3.type != BOOL) {
                     Error = "Invalid type of if's condition
expression, must be binary";
                     yyerror(Error);
                     yyerrok;
                     sem_errors++;
              }
              else {
                     if ($3.trueList != NULL)
       fquad.backpatch($3.trueList, $5);
                     if ($3.falseList != NULL)
       fquad.backpatch($3.falseList, $7);
              }
              if ($3.trueList != NULL) delete $3.trueList;
              if ($3.falseList != NULL) delete $3.falseList;
```

If else

```
TK_IF TK_LPAR b_expression TK_RPAR MQ statement  TK_ELSE MQ2 statement MQ {
              if ($3.type != BOOL) {
                     Error = "Invalid type of if's condition expression, must be
binary";
                     yyerror(Error);
                     yyerrok;
                     sem_errors++;
              }
              else{

                     vector<int>* nextlist =  new vector<int>();
                     nextlist->push_back($8-1);
                     fquad.backpatch(nextlist,$10);


                     if($3.trueList != NULL)           fquad.backpatch($3.trueList,
$5);
                     if($3.falseList != NULL)   fquad.backpatch($3.falseList,$8);
                     delete nextlist;
              }

              if ($3.trueList != NULL) delete $3.trueList;
              if ($3.falseList != NULL) delete $3.falseList;
```

Η ίδια τακτική χρησιμοποιείται και στη δομή while.

Παραδείγματα επισυνάπτονται στο mail και στο φάκελο του Project