PES UNIVERSITY ONLINE

Compiler Design

Preet Kanwal

Department of Computer Science & Engineering

Compiler Design

Building a Mini Compiler - Expression Evaluation

Preet Kanwal

Department of Computer Science & Engineering

Compiler Design

Implementation tasks

- Updating the symbol table with values, line number and scope.
- Expression evaluations. In case the input file has statements like a=10, b=20, c= a+b then your code must be able to update in the symbol table, value of c=30.
- The code needs to take care of checks for variables that haven't been declared and mismatched type variables in expressions. Display appropriate error message.

Compiler Design

Expected Results

Sample Input:

```
int main()
{
   int a=5;
   float b=4.6;
   double c=6.9845;
   char d="c";
   a=2;
   int b;
}
```

Output

```
Variable b already declared
Error :b at 8
Valid syntax
Name  size  type  lineno   scope value
a     2     2     7        1     2
b     4     3     4        1     4.6
c     4     3     5        1     6.9845
d     1     1     6        1     "c"
```

Compiler Design

Expected Results

Sample Input:

```
int main()
{
    float x;
    int y;
    x = 3.4;
    y = 45.4;
    if(5>6)
    {
        x=4.5;
    }
    else
    {
        int z=5*6+5;
    }
}
```

Output

```
Mismatch type
Error :45.4 at 6
Valid syntax
Name   size   type   lineno scope  value
x      4      3      9      1      4.5
y      2      2      6      1      45.4
z      2      2      13     2      35
```

Compiler Design

Expected Results

Sample Input:

```
int main()
{
    int a=5;
    b=3;
    float c=4.5;
    c=6.5;
    double d=5.44;
    double e=d+9.0-4.0/2.0;
}
```

Output

```
Variable b not declared
Error :b at 4
Valid syntax
Name   size   type   lineno scope  value
a   2   2   3   1   5
c   4   3   6   1   6.5
d   4   3   7   1   5.44
e   4   3   8   1   12.440000
```

Symbol table implementation: yacc file

- Updating of the value and line number of a variable can happen during assignment or during expression evaluation.
- Variable not declared can be checked when a variable is being assigned but isn't in the symbol table.

```
//find type of T_ID for type checking
ASSGN : T_ID '=' EXPR   {
            /*
                Check if variable is declared in the table
                insert value
            */
            }
    ;
```

Compiler Design

Mini-Compiler

```
DECLR : TYPE LISTVAR
LISTVAR : LISTVAR ',' VAR
        | VAR
        ;
VAR: T_ID '=' EXPR  {
                    /*
                    check if symbol is in table
                    if it is then error for redeclared variable
                    else make entry and insert into table
                    insert value coming from EXPR
                    revert variables to default values:value,type
                    */
          }
   | T_ID           {
                    /*
                    check if symbol is in table
                    if it is then error for redeclared variable
                    else make entry and insert into table
                    revert variables to default values:value,type
                    */
          }
```

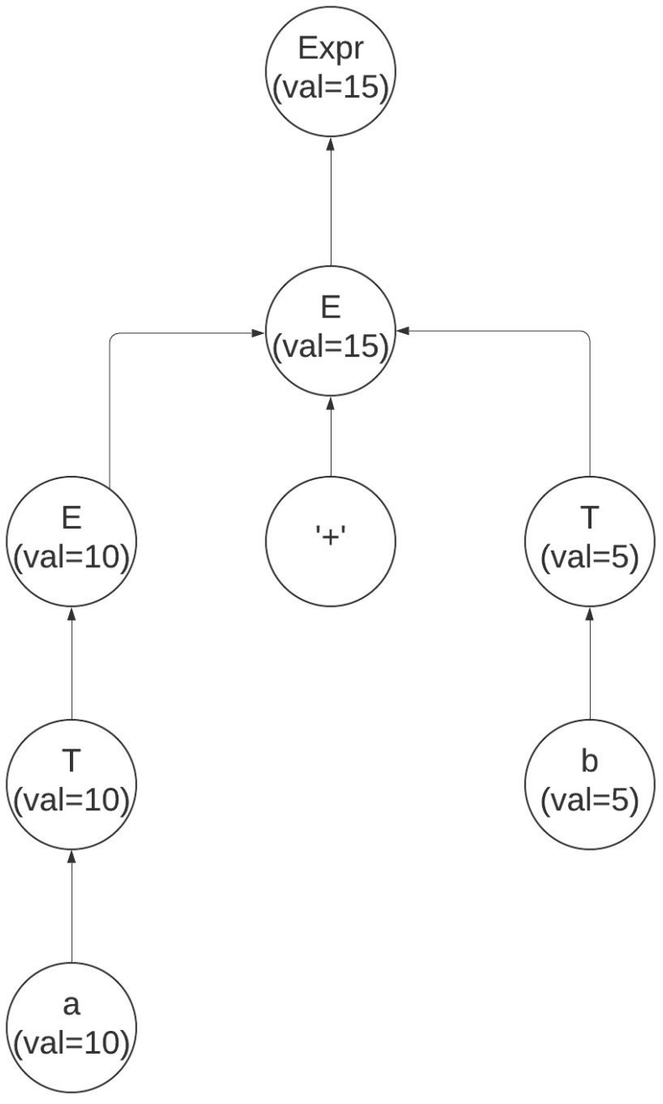
Expression Evaluation implementation: yacc file

- Expression grammar can extend to multiple rules, to keep track of the value you need to perform the calculation and pass it to to the rule.
- It is very similar to a tree structure passing the value all the way up to the root.
- Let's take an example:
  a=10
  b=5
  c=a+b
- Let the grammar be:
  ASSGN: T_ID '=' EXPR
  EXPR:E
  E: E '+' T | T
  T: T_NUM | T_ID

PES UNIVERSITY ONLINE

Important points:

- You will have to create helper functions for the symbol table.(example: extracting value from name)
- You will also have to keep track of certain variables during expression evaluation.(type, value, etc)
- Take care of mismatch type using the above variables.



Here we see how the values travel from the variables a and b all the way to the expr grammar and from there we can just update the new value to c, which matches token T_ID

To do this in YACC we need to use the $$,$1 notations to hold the semantic value
Ex: T : T_ID {$$=$1}
Here it means the value of token 1($1) is stored in T($$).

So our grammar would look like this
ASSGN: T_ID ‘=’ EXPR { update($1,$3)} //update(name,value)

EXPR: E {$$=$1}

E: E ‘+’ T {$$=$1+$3}
| T {$$=$1}

T: T_NUM {$$=$1}
| T_ID {$$=$1}

Note: The calculation may not be the same, if YYSTYPE is defined as char *, then you need to convert the string to int/float do the calculation and convert it back.

Compiler Design

Mini-Compiler

Scope tracking can be implemented in different ways, one way could be to have a variable keeping track of the block you are in.

```
//increment and decrement at particular points in the grammar to keep track of the scope.
MAIN : TYPE T_MAIN '(' EMPTY_LISTVAR ')' '{' STMT '}';

BLOCK : '{' STMT '}';
```

THANK YOU

Preet Kanwal

Department of Computer Science & Engineering

preetkanwal@pes.edu