```
Script started on Sat Sep 23 11:25:08 2017
[?1034hbash-3.2$ cat main.cpp
//
//  main.cpp
//  FresnoF17
//  a mini-language named FresnoF17 that supports variable declaration
with type and two
//  primitive statements; i.e., assign_statement and print_statement.
//
//  This program gets the input from file and analyse it with words
and characters.
//  It checks with words equal poosiblities to perform next operation.
//  It calculate the experssion and get back output which it will
store in table.
//  It will analyse each expression and shows the output
//  This program also included error checking
//  Exit code 0 : Error in file
//  Exit code 1 : End program
//  Exit code 2 : Syntax error
//  Exit code 3 : Lexical error
//
//
//
//  Created by Harkaranjeet Singh on 9/14/17.
//  Copyright © 2017 Harkaranjeet Singh. All rights reserved.
//

#include <iostream>
#include <cstdlib>
#include <fstream>
#include <cmath>  //for power
using namespace std;

ifstream fin;
string prog, s1;
int indexx = 0;


void declarations();
void assign_Statement(string id);
void print_Statement();
void statements();
void declarations2(string type);
void statement2(string word);
int Exp(), Term(), Exp2(int), Term2(int), Fact(), Fact2(int inp),
Num();


// Main function reads from file
// it reads one word into a string and analyse
// it call the other function as string is compatible to it
// it shows error if any problem in opening file with exit code 0
int main(int argc, const char * argv[])
{

    string path1 = argv[1];
    fin.open(path1.c_str());


    fin>>s1;
    while(!fin.eof())
    {
        if (s1 == "program")
        {
            declarations();
            statements();

        }else
        {
            cout<<"Error in opening the file"<<endl;
            exit(0);
        }

    }
    return 0;

}



// Declarations function fetch the word from file and check all the
possibilties
// if the possibilties are equal it send the program to next step or
next function to perform
// if there are any syntax error it shows with exit code 2

void declarations() //extension to the language
{


    string word;
    fin >>word;
    if (word =="begin")
    {
        return;
    }
    else if (word == "int"||word == "double")
    {
        declarations2(word);
    }
    else{
        cout<<"Syntax error program does not contain exact
Words"<<endl;
        exit(2);
    }

    declarations(); //recursion on the function declarations
}

// Statements function fetch the word from file and check all the
possibilties
// if the possibilties are equal it send the program to next step or
next function to perform
// when program possibilty equal to end it quit the program with exit
code 1

void statements(){

    string word;
    fin>>word;
    if (word == "end")
    {
        exit(1);
    }else{
        statement2(word);
        statements();//recursion on statements
    }

}

// Struct for storing ID, Type , Values

struct table
{
    char id;

    string type;
    double value;

};



const int tableSize = 100;     //table size
table symbolTable[tableSize];  // table
int tableIndexx = 0;           // table indexx



// Declarations2 function with parameter of string fetch char from
```

```
file and check all the possibilties
// if the possibilties are equal it and store the values in table

void declarations2(string type){

    char id;
    fin >>id;
    while(id !=';')
    {
        if (id == ',')
        {
            fin >> id;
        }
        else
        {
            symbolTable[tableIndexx].id = id;
            symbolTable[tableIndexx].type = type;
            tableIndexx++;
            fin>>id;

        }
        // Table

        //        cout<<"ID"<<"  "<<"Type"<<"    "<<"value"<<endl;
        //        cout<<symbolTable[0].id<< "
"<<symbolTable[0].type<< "    "<<symbolTable[0].value<<endl;
        //        cout<<symbolTable[1].id<< "
"<<symbolTable[1].type<< "    "<<symbolTable[1].value<<endl;
        //        cout<<symbolTable[2].id<< "
"<<symbolTable[2].type<< "    "<<symbolTable[2].value<<endl;
        //        cout<<symbolTable[3].id<< "
"<<symbolTable[3].type<< "    "<<symbolTable[3].value<<endl;
        //        cout<<symbolTable[4].id<< "
"<<symbolTable[4].type<< "    "<<symbolTable[4].value<<endl;
        //
    }
}



// Statement2 function with parameter of string check all the
possibilties
// if the possibilties are equal it send the program to next step or
next function to perform

void statement2(string word)
{
    if (word == "print")
    {
        print_Statement(); // callin the print_Statement
    }else{
        assign_Statement(word);
    }


}


// assign_Statement function with parameter of string check all the
possibilties
// it fetch the char from file
// if the possibilties are equal it and store the values in table
// if there are any spelling error it will show lexical error with
exit code 3
// if there is any syntax error it will show syntax error with exit
code 3

void assign_Statement(string id)
{
    char word;
    fin >> word;
    if(word == '=')
    {
        indexx = 0;

        getline (fin,prog);

        int temp = Exp();

        tableIndexx = 0;

        for(tableIndexx = 0; tableIndexx < tableSize; tableIndexx++)
        {
            if(id[0]==symbolTable[tableIndexx].id)  //id
            {

                symbolTable[tableIndexx].value=temp;  //update the
table
                if (id[0]=='c')
                {
                    symbolTable[tableIndexx].value =
symbolTable[0].value;
                }
                else if (!isalpha(id[0]))
                {
                    cout<<"Lexical error"<<endl;
                    exit(3);
                }

            }
            else if (!isalpha(id[0]))
            {
                cout<<"Lexical error"<<endl;
                exit(3);
            }


        }


    }
    else
    {
        cout<<"Syntax Error"<<endl;
        exit(2);
    }



    //    cout<<"ID"<<"  "<<"Type"<<"    "<<"Value"<<endl;
    //    cout<<symbolTable[0].id<< "    "<<symbolTable[0].type<< "
"<<symbolTable[0].value<<endl;
    //    cout<<symbolTable[1].id<< "    "<<symbolTable[1].type<< "
"<<symbolTable[1].value<<endl;
    //    cout<<symbolTable[2].id<< "    "<<symbolTable[2].type<< "
"<<symbolTable[2].value<<endl;
    //    cout<<symbolTable[3].id<< "    "<<symbolTable[3].type<< "
"<<symbolTable[3].value<<endl;
    //    cout<<symbolTable[4].id<< "    "<<symbolTable[4].type<< "
"<<symbolTable[4].value<<endl;
    //
    //

}


// print_Statement function shows the output of program
// it fetch the char from file
// if the possibilties are equal it and shows the values of table
// if the expression are directly get here it will handle it and show
the output


void print_Statement()
{
    char id;
    fin>>id;
    if (isalpha(id))
    {
        tableIndexx = 0;
        for (tableIndexx= 0; tableIndexx <tableSize; tableIndexx++)
        {
            if(id == symbolTable[tableIndexx].id)
            {
                cout<<symbolTable[tableIndexx].value<<endl;
            }

        }
    }
    else
    {
        fin.putback(id);
        getline (fin,prog);  // get one line from file
        indexx = 0;         //  indexx initialize to 0
        cout<<Exp()<<endl; // for expression

    }

    id = fin.get();
    if (id == 'e')
    {
        fin.putback(id);
    }

}




//Calling the function as grammer indicates
//returns the Exp2 funtion with a parameter of Term
int Exp(){
    return Exp2(Term());
}

//Calling the function as grammer indicates
//returns the Term2 funtion with a parameter of Fact

int Term(){
    return Term2(Fact());
}


// Read the string char by char
// Handle the space by skiping
// Perform the grammer
// Handles T+T and T-T
int Exp2(int inp)
{
    int result = inp;
    if (indexx < prog.length())   //if not the end of program string
    {
        char a = prog.at(indexx++); //get one chr from program string
        while (a == ' ' && indexx < prog.length()) // if space skip
and read until a char
        {
            a = prog.at(indexx++);
        }
        if (a == '+')
            result = Exp2(result + Term());  //handles T+T
        else if (a == '-')
            result = Exp2(result - Term());  //handles T-T
    }
    return result;
}

// Read the string char by char
// Handle the space by skiping
// Perform the grammer
// Handles "*" , "/" , "+" , "-" , ")"

int Term2(int inp)
{

    int result = inp;
    if (indexx < prog.length())   //if not the end of program string
    {
        char a = prog.at(indexx++); //get one chr from program string
        while (a == ' ' && indexx < prog.length()) // if space skip
and read until a char
        {
            a = prog.at(indexx++);
        }
        if (a == '*')
        {
            result = Term2(result * Fact()); //handles consecutive *
operators
        }

        else if (a == '/')
        {
            result = Term2(result / Fact()); //handles consecutive /
operators
        }
        else if (a == '+' || a == '-'|| ')')     //if " + , -,or )"
get back one position
        {
            indexx--;
        }
    }
    return result;
}

//Calling the function as grammer indicates
//returns the Fact2 funtion with a parameter of Num


int Fact()
{

    return Fact2(Num());
}

// Read the string char by char
// Handle the space by skiping
// Perform the grammer
// Handles power ^

int Fact2(int inp)
{

    int result = inp;
    if (indexx<prog.length())
    {

        char a = prog.at(indexx++); //get one chr from program string
        while (a == ' ' && indexx < prog.length()) // if space skip
and read until a char
        {
            a = prog.at(indexx++);
        }
        if (a == '^') //handles consecutive / operators
            result = Fact2(pow(result, Fact()));
        else
            indexx--;
    }
    return result;
}

// Read the string char by char
// Handle the space by skiping
// Perform the grammer
// Handles "("
// return by converts a char to a numeric number and return

int Num()
{
    if (indexx<prog.length())
    {
        char a = prog.at(indexx++); //get one chr from program string
        while (a == ' ' && indexx < prog.length()) // if space skip
and read until a char
        {
            a = prog.at(indexx++);
        }
        if (a == '(') // handles the "("
            return Exp();
        else
            return atoi(&a); //converts a char to a numeric number and
return
    }
    return Num();
}
```

```
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$ cat datafilecopy.txt
program
int a,b,c;
double d;
begin
a = 3*(5+2);
b = (3 + 4) * 5;
c = a;
print a;
print b;
print c;
print (2+3)*7 + 2^3;
end

bash-3.2$
bash-3.2$
```

```
bash-3.2$
bash-3.2$ g++ main.cpp
bash-3.2$ ./a.out datafilecopy.txt
21
35
21
43
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
```