

**Project Report**

**Project Title:** Mini Parser

**Course Code:** CSE207

**Course Title:** Data Structure

**Section:** 03

**Semester:** Fall’19

**Submitted by:**

Mahmud Jamil

**ID:** 2017-2-60-147

Farhat Jebin

**ID:** 2017-2-60-059

Md Fahimul Islam

ID:2017-1-62-022

**Submitted to:**

Dr. Maheen Islam

Assistant Professor

Department of CSE

EAST WEST UNIVERSITY

**Date of Submission: 11/12/2019**

**Description:**

A parser is a software component that takes input data and build structure.

In this project, the parser manages to check syntactical error of a source code.

It is made by using file, stack and linked list.

The file contains a small program containing mathematical expression and branching statements.

**Project Introduction:**

This is a mini parser. In this parser we can able to find an error massage from the source code, if we can put any wrong parenthesis or variable in the source code.

**Source Code:**

#include<bits/stdc++.h>

using namespace std;

string lines[100];

int linelen=0;

struct node

{

char data;

struct node \*next;

}\*topfp=NULL,\*topsp=NULL,\*toptp=NULL;

struct errorlist

{

int data;

struct errorlist \*next;

}\*head=NULL,\*tail=NULL;

void checklist(int val);

void createlist(int val);

void print();

void pushfp(char x);

void popfp(int x);

void pushsp(char x);

void popsp(int x);

void pushtp(char x);

void poptp(int x);

map<string,int>mp;

map<string,int>mp1;

int fun(string st)

{

int i,j,flag=0,flag1=1;

vector<string>res;

if(st.size()<3)flag=0;

else if(flag1==1)

{

for(i=0; i<st.size()-2; i++)

{

if(st[i]=='i' &&st[i+1]=='n' &&st[i+2]=='t')

{

flag=1;

break;

}

}

}

if(st.size()>5)

{

for(i=0; i<st.size()-5; i++)

{

if(st[i]=='r' &&st[i+1]=='e' &&st[i+2]=='t' &&st[i+3]=='u' &&st[i+4]=='r' &&st[i+5]=='n')

{ return 1;

}

if(st[i]=='m' &&st[i+1]=='a' &&st[i+2]=='i' &&st[i+3]=='n' )return 1;

}

}

if(flag==1)

{

string st1;

flag1=0;

for(i=0; i<st.size(); i++)

{

if(st[i]==' ')flag1=1;

else if(flag1==1)

{

st1.push\_back(st[i]);

}

}

string st3;

for(i=0; i<st1.size(); i++)

{

if(st1[i]==',' || st1[i]==';')

{

mp[st3]=1;

st3.clear();

}

else st3.push\_back(st1[i]);

}

}

if(flag==0)

{

string st2;

int i;

for(i=0; i<st.size(); i++)

{

st2.push\_back(st[i]);

}

string st3;

for(i=0; i<st2.size(); i++)

{

char ch=st2[i];

if(ch=='=' || ch=='\*' || ch=='/' || ch==';' || ch=='+' || ch=='-' || ch == '>' || ch== '<' || ch=='(' || ch==')' || ch=='[' || ch==']')

{

mp1[st3]=1;

st3.clear();

}

else st3.push\_back(st2[i]);

}

}

return flag;

}

int cheaker\_code(string st)

{

string st3;

vector<string>v;

int i;

for(i=0; i<st.size(); i++)

{

char ch=st[i];

if(ch=='=' || ch=='\*' || ch=='/' || ch==';' || ch=='+' || ch=='-' || ch == '>' || ch== '<' || ch=='(' || ch=='[' || ch==')' || ch==']'||ch=='1'||ch=='2'||ch=='3'||ch=='4')

{

v.push\_back(st3);

st3.clear();

}

else

{

st3.push\_back(st[i]);

if(st3=="if")

{

st3.clear();

}

}

}

for(i=0; i<v.size(); i++)

{

if(mp[v[i]]!=mp1[v[i]])

{

cout<< v[i] << " is not defined\n";

return 0;

}

}

return 1;

}

int main()

{

ifstream file("dataset.txt",ios::in);

string templine;

while(!file.eof())

{

getline(file,templine);

lines[linelen++] = templine;

int res=fun(templine);

if(res==0)

{

int res1=cheaker\_code(templine);

if(res1==0)cout<<linelen<< " th line error\n";

}

}

for(int i=0;i<linelen;i++)

{

for(int k=0;k<lines[i].length();k++)

{

if(lines[i][k]== '(')

{

pushfp(lines[i][k]);

}

if(lines[i][k]==')')

{

popfp(i+1);

}

if(lines[i][k]== ';' || lines[i][k] == '{' || lines[i][k] == '}')

{

if(topfp != NULL)

{

checklist(i+1);

while(topfp != NULL)

{

popfp(i+1);

}

}

}

if(lines[i][k]== '[')

pushtp(lines[i][k]);

if(lines[i][k]==']')

poptp(i+1);

if(lines[i][k] == ')' || lines[i][k] == ';' || lines[i][k] == '{' || lines[i][k] == '}')

if(toptp != NULL)

{

checklist(i+1);

while(toptp != NULL)

{

poptp(i+1);

}

}

if(lines[i][k]== '{')

pushsp(lines[i][k]);

if(lines[i][k]=='}')

popsp(i+1);

}

}

if(topsp != NULL)

checklist(linelen-1);

print();

return 0;

}

void checklist(int val)

{

int flag = 1 ;

struct errorlist \*temp;

temp=head;

while(1)

{

if(temp==NULL) break;

else if(temp->data==val)

{

flag = -1;

break;

}

else

temp = temp->next;

}

if(flag == 1)

createlist(val);

}

void createlist(int val)

{

struct errorlist \*curr;

curr=(struct errorlist \*)malloc(sizeof(struct errorlist));

curr->data = val;

curr->next = NULL;

if(head==NULL)

{

head= curr;

tail=curr;

}

else

{

tail->next = curr;

tail=curr;

}

}

void print()

{

struct errorlist \*temp;

temp = head;

while(1)

{

if(temp == NULL) break;

else

{

cout<<"Error in line number: "<<temp->data<<endl;

temp = temp->next;

}

}

delete(temp);

}

void pushfp(char x)

{

struct node \*newnode;

newnode=(struct node \*)malloc(sizeof(struct node));

newnode->data = x;

newnode->next = NULL;

if(topfp==NULL)

topfp= newnode;

else

{

newnode->next = topfp;

topfp=newnode;

}

}

void popfp(int x)

{

struct node \*temp;

if(topfp==NULL)

checklist(x);

else

{

temp = topfp;

topfp=topfp->next;

delete(temp);

}

}

void pushsp(char x)

{

struct node \*newnode;

newnode=(struct node \*)malloc(sizeof(struct node));

newnode->data = x;

newnode->next = NULL;

if(topsp==NULL)

topsp= newnode;

else

{

newnode->next = topsp;

topsp=newnode;

}

}

void popsp(int x)

{

struct node \*temp;

if(topsp==NULL)

checklist(x);

else

{

temp = topsp;

topsp=topsp->next;

delete(temp);

}

}

void pushtp(char x)

{

struct node \*newnode;

newnode=(struct node \*)malloc(sizeof(struct node));

newnode->data = x;

newnode->next = NULL;

if(toptp==NULL)

toptp= newnode;

else

{

newnode->next = toptp;

toptp=newnode;

}

}

void poptp(int x)

{

struct node \*temp;

if(toptp==NULL)

checklist(x);

else

{

temp = toptp;

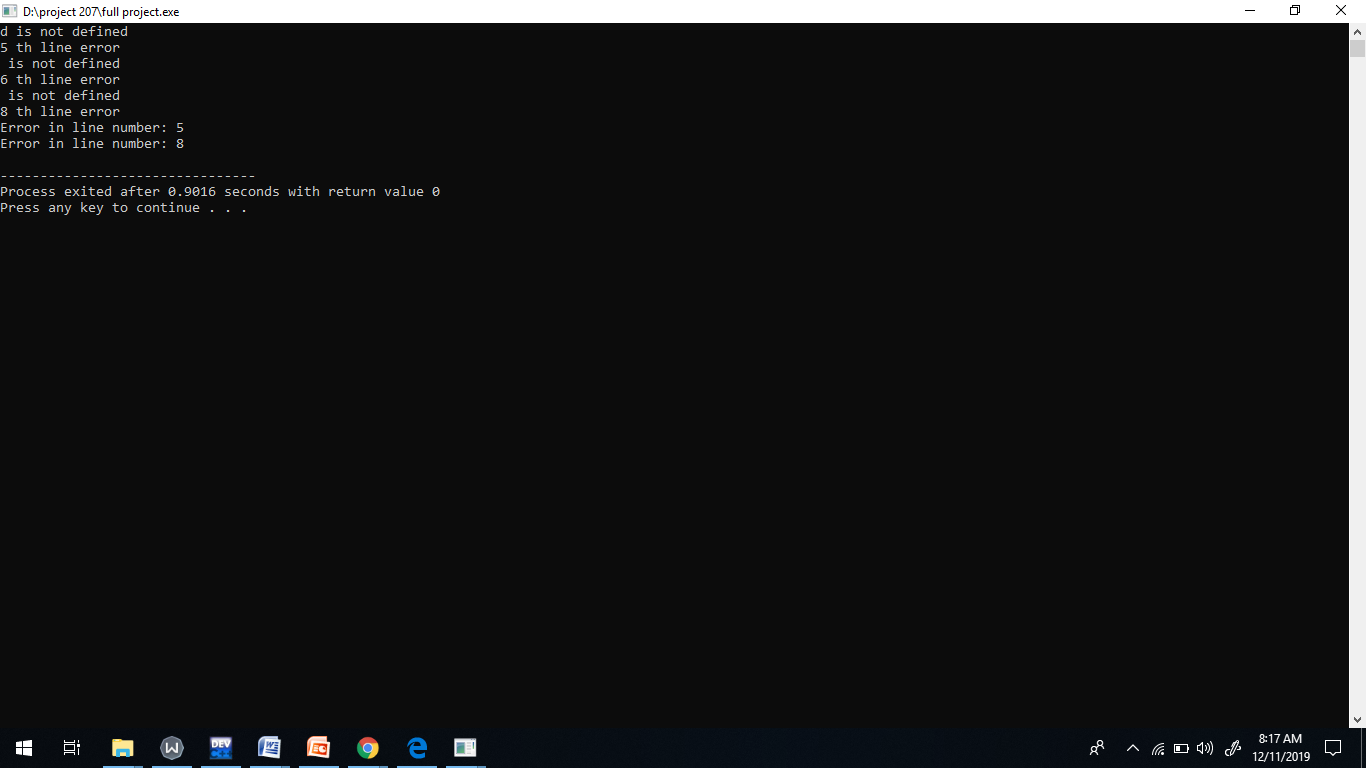
toptp=toptp->next;

delete(temp);

}

}

**Output:**

****

**Limitations:**

Parser is case sensitive. So there are too many limitations in the project. We can only find bracket error and undefined variable error in this project.