



# EAST WEST UNIVERSITY

FALL-2018

Project Report

**Project Name: Mini Parser**

Course code: CSE207

Course title: Data structures

Section: 3

**Submitted To:**

Tanni Mittra (TM)

Lecturer, Dept. of CSE

**Submitted by:**

Jannatul Naim (ID: 2017-2-60-029)

Dept. of CSE

Date of submission: 09/12/2018

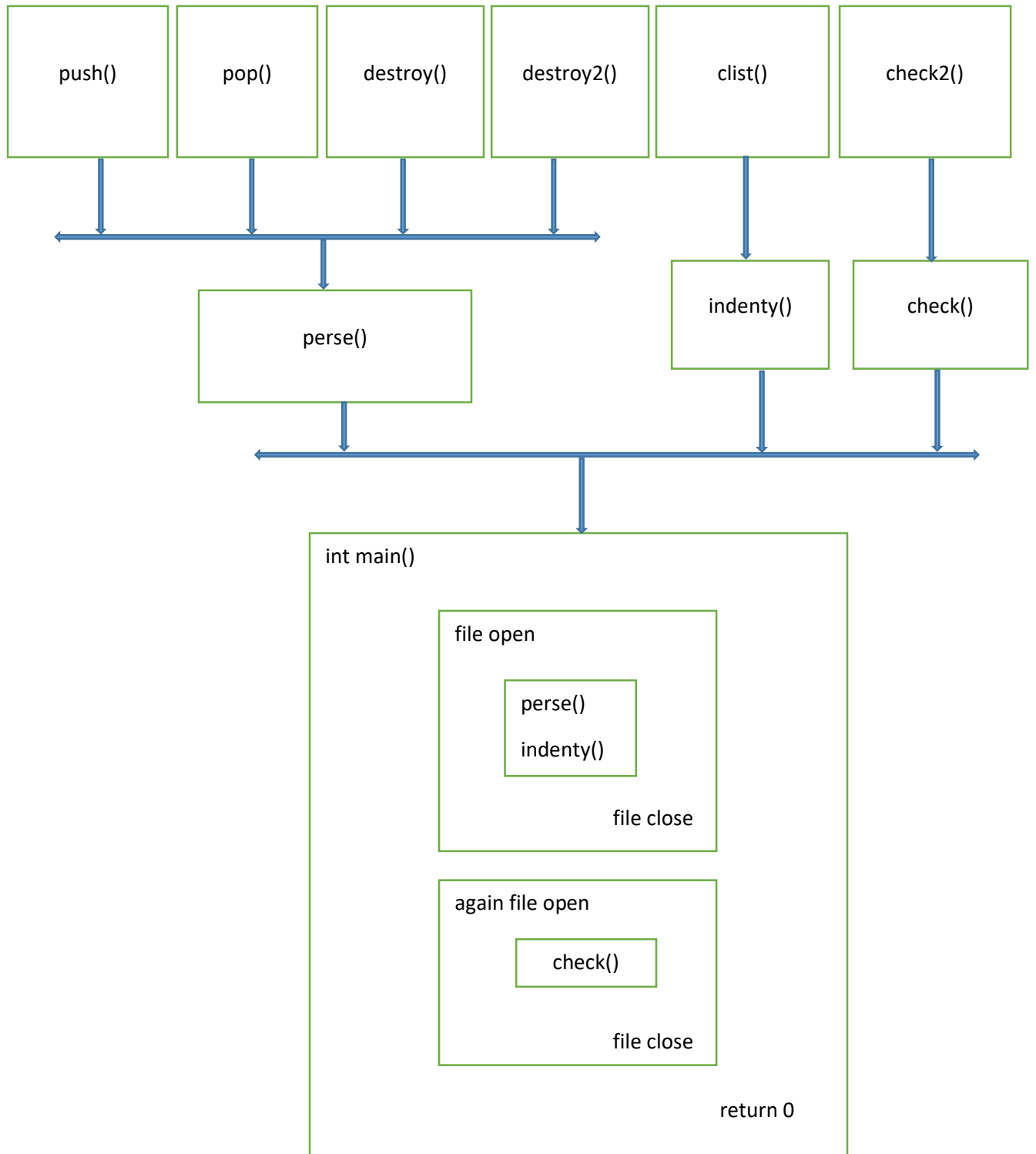
**Introduction:** Our project name is “Mini Parser”. We complete the project using C++ language.

**Requirement:** The following requirements for the project were given to us:

1. Where are 1<sup>st</sup>, 2<sup>nd</sup> and 3<sup>rd</sup> bracket missing?
2. What variables were not declared?

We complete all requirement.

## System Design:



**Feature Descriptions:**

Our given project is Mini Parser, here we use structure, stack, file, pointer, built in function, array, loop, if-else condition and different user-define function. It will help to understand the code easily and will help to make changes if needed.

Our user define functions are-

- a) push( )
- b) pop( )
- c) destroy( )
- d) destroy2( )
- e) clist()
- f) check2( )
- g) check( )
- h) indenty()
- i) perse()

**Built in function:** Library functions are the built in function in C++ programming. Every valid C++ program has at least one function, that is main() function.

**main ():** Manually code execution starts in this function. We also use switch in this function as the approach of choosing option.

**User defined function:** C++ allows programmer to define their own function. A user defined function groups code to perform a specific task and that group of code is given a name (identifier). When a program beings running , the system calls the main() function, that is, the system starts executing codes from main() function.

**push( ):** push function is used to insert variables in stack.

**pop ( ):** pop function is used to delete variables one by one from stack.

**destroy ( ):** destroy function is used to delete all variable from stack.

**destroy2( ):** destroy2 function is used to delete all variable from stack2.

**clist():** We used this function to insert variables from file to stack, those variable's type is integer and float.

**check2 ( ):** We used this function to search a variable where was declared.

**check ( ):** We used this function to send those variable from file after and before operator to cheack2 function .

**indenty():**We used this function to identify those variables are integer and float.

**perse():**We used this function to check where bracket were missing.

**Structure:** Structure is a user-defined data type in C language which allows us to combine data of different types together. Structure helps to construct a complex data type which is more meaningful. It is somewhat similar to an Array, but an array holds data of similar type only. But structure on the other hand, can store data of any type, which is practical more useful.

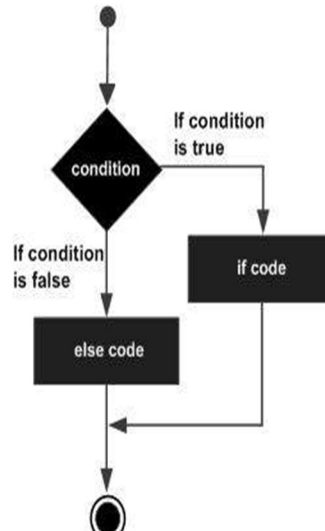
**Stack:** Stack is an ordered list of similar data type. Stack is a LIFO (Last in First out) structure or we can say FILO (First in Last out). push() function is used to insert new elements into the stack and pop() function is used to remove an element from the stack. Both insertion and removal are allowed at only one end of stack called Top.

**Pointer:** A pointer is a variable whose value is the address of another variable. Like any variable or constant, you must declare a pointer before you can work with it. The general form of a pointer variable declaration is – type \*var-name; Here, type is the pointer's base type.

**One-dimensional array:** A one-dimensional array is an array where each element in the array points to a specific value of the type specified by the array (all values must be of the same type). For example, we can store integer values in an integer array, character values in a character array and strings in a string array.

**Loop:** Loops are used in programming to repeat a specific block until some end condition is met. There are three type of loops in C++ programming. One type of loop is for loop. Here we used for loop.

**if-else:** An if statement can be followed by an optional else statement, which executes when the Boolean expression is false. If the Boolean expression evaluates to true, then the if block of code will be execute, otherwise else block of code will be execute. Flow the diagram:



## Implementation:

By completing this project, we learned how do work compiler. And this project's ideas will help us to complete the next time project. So we think this project is important in our futures life will play the role.

## C++ Code for Project:

```
#include <iostream>
#include <string>
#include<cstdio>
#include<cstdlib>
#include <fstream>
using namespace std;
int c=1;
char a[100],a1[100];
struct node
{ char data;
  int line;
  struct node *next;
};
struct node *top=NULL,*top1=NULL,*top2=NULL,*head=NULL;

char push( char ch)
{ if(ch=='('){
  struct node *curr;
  curr=(struct node*) malloc (sizeof (struct node));
  curr->next=top;
  top=curr;
}
else if(ch=='[')
{
```

```

    struct node *cur;
    cur=(struct node*) malloc (sizeof (struct node));
    cur->next=top1;
    top1=cur;
}
else if(ch=='{')
{
    struct node *curl;
    curl=(struct node*) malloc (sizeof (struct node));
    curl->line=c;
    if(top2==NULL)
    {
        curl->next=NULL;
        top2=curl;
    }
    else{
        curl->next=top2;
        top2=curl;
    }
}
}

void pop(char p)
{
    if(p==' '){
        struct node *temp;
        temp=top;
        top=top->next;
        free(temp);
    }
    else if(p=='[')

```

```

{
    struct node *temp1;
temp1=top1;
top1=top1->next;
free(temp1);

}
else if(p=='}')
{
    struct node *temp2;
temp2=top2;
top2=top2->next;
free(temp2);
}
}
void destroy()
{

    struct node *temp,*l;
while(top!=NULL)
{
    l=top->next;
    free(top);
    top=l;
}
}
void destroy2()
{

    struct node *temp3,*l2;

```



```

while(top1!=NULL)
{
    l2=top1->next;
    free(top1);
    top1=l2;
}
}

```

```

void clist(char ti)
{
    struct node *sta;
    sta=(struct node*) malloc (sizeof (struct node));
    sta->data=ti;
    if(head==NULL)
    {
        sta->next=NULL;
        head=sta;
    }
    else{
        sta->next=head;
        head=sta;
    }
}

```

```

int check2(char r)
{
    int y=0;
    struct node *temp;
    temp=head;
    while(temp!=NULL)

```

```

{
    if(temp->data==r)
    {
        y=1;
    }

    temp=temp->next;
}
return y;
}

```

```

void check(char a1[],int n1)
{
    for(int i=0;i<n1;i++)
    {
        int d9=0,f9=0;

        if(a1[i]=='>'||a1[i]=='<'||a1[i]=='+'||a1[i]=='-'||a1[i]=='*'||a1[i]=='/'||a1[i]=='='||a1[i]=='%')
        {

            d9=check2(a1[i+1]);

            if(a1[i+1]>='a' && a1[i+1]<='z'){
                if(d9!=1)
                {
                    cout<<a1[i+1]<<" is not declare in the scope "<<endl;
                }
            }
        }
    }
}

```

```

    }

    f9=check2(a1[i-1]);
    if(a1[i-1]>='a' && a1[i-1]<='z'){
        if(f9!=1)
        {
            cout<<a1[i-1]<<" is not declare in the scope "<<endl;
        }
    }
}
}

```

```

void perse(char a[],int n){

    for(int i=0; i<n; i++){

        if((a[i]=='(' || a[i]==')') && a[n-1]!=';')
        {
            if(''==a[i])
            {
                push(a[i]);
            }

            else if(')'==a[i])
            {
                if(top==NULL)

```

```

    {
        cout<<"opening ( missing in line number "<<c<<endl;
    }
    else
    {
        pop(a[i]);
    }
}
}

```

```

else if((a[i]=='(' || a[i]=='))' ) && a[n-1]==';' )

```

```

{
    if('('==a[i])
    {
        push(a[i]);
    }
}

```

```

else if(')'==a[i])
{
    if(top==NULL)
    {
        cout<<"opening ( missing in line number "<<c<<endl;
    }
    else
    {
        pop(a[i]);
    }
}
}
}

```

```

else if((a[i]=='['||a[i]==']') && a[n-1]==';')
{
    if('['==a[i])
    {
        push(a[i]);
    }

    else if(']'==a[i])
    {
        if(top1==NULL)
        {
            cout<<"opening [ missing in line number "<<c<<endl;
        }
        else
        {
            pop(a[i]);
        }
    }
}

else if((a[i]=='{'||a[i]=='}') && a[n-1]!=';')
{
    if('{')==a[i])
    {
        push(a[i]);
    }

    else
    {if(top2==NULL)

```

```

    {
        cout<<"Opening { missing in line "<<c<<<endl;
    }
    else{
pop(a[i]);
    }
}
}
}

}

```

```

if(top!=NULL)
{
    cout<<"closing ) missing in line "<<c<<<endl;

}
if(top1!=NULL)
{
    cout<<"closing ] missing in line "<<c<<<endl;

}
destroy();
destroy2();

}

```

```

void indenty(char a[],int n)
{
    if(a[0]=='i'&&a[1]=='n'&&a[2]=='t' && a[n-1]==';')

```

```

{int l=4;
while(a[l]!='\0'){
    if(a[l]>='a' && a[l]<='z' )
    {
        clist(a[l]);
    }
    l++;
}
}

if(a[0]=='f' && a[1]=='l'&& a[2]=='o' && a[3]=='a' && a[4]=='t' && a[n-1]==';'){
    int s=6;
while(a[s]!='\0'){
    if(a[s]>='a' && a[s]<='z' )
    {

        clist(a[s]);
    }
    s++;
}
}
}
}

```

```

int main()
{
    int i,j;
    ifstream infile;
    infile.open("mathExpression.txt");
    string line;
    string vari;

```

```

while (getline(infile, line))
{
    i=0;
    while(line[i]!='\0')
    {
        a[i]=line[i];
        i++;
    }
    perse(a,i);
    indenty(a,i);
    cout<<endl;

    c++;
}
if(top2!=NULL)
{
    while(top2!=NULL)
    {
        cout<<" Closing } missing in line "<<top2->line<<endl;
        top2=top2->next;
    }
}
infile.close();

```

```

ifstream inf;
inf.open("mathExpression.txt");
    while (getline(inf,vari))

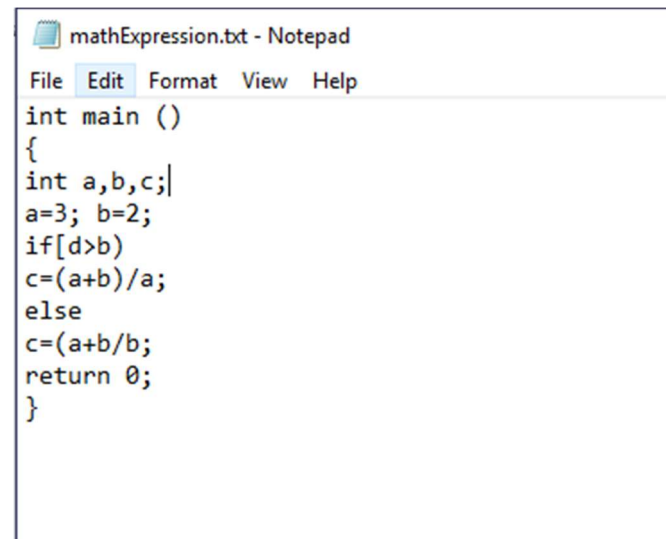
```



```
{
    j=0;
    while(vari[j]!='\0')
    {
        a1[j]=vari[j];
        j++;
    }
    check(a1,j);
}

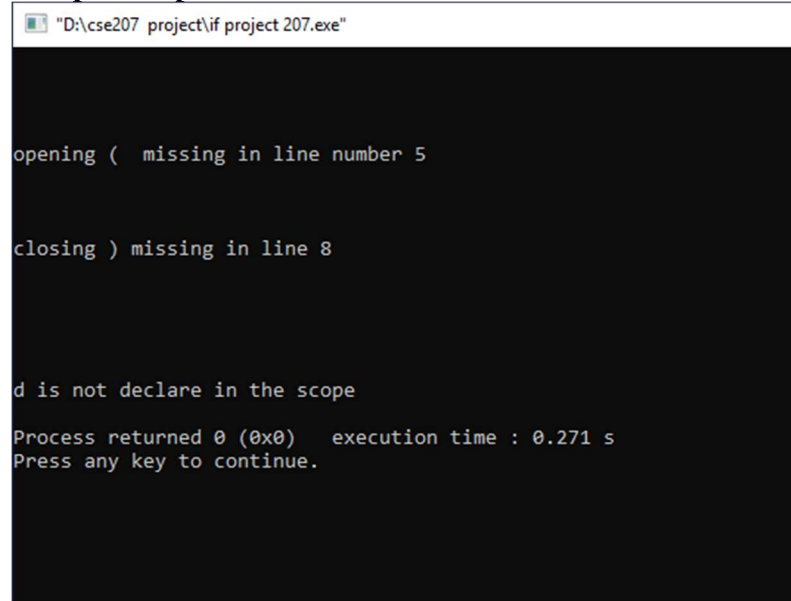
inf.close();
return 0;
}
```

### Sample Input:



```
mathExpression.txt - Notepad
File Edit Format View Help
int main ()
{
    int a,b,c;
    a=3; b=2;
    if(d>b)
        c=(a+b)/a;
    else
        c=(a+b)/b;
    return 0;
}
```

### Sample Output:



```
"D:\cse207 project\if project 207.exe"

opening ( missing in line number 5

closing ) missing in line 8

d is not declare in the scope

Process returned 0 (0x0) execution time : 0.271 s
Press any key to continue.
```

### Limitation:

If we get more time, we could better.

### Conclusion:

There can be other types of syntax errors in a program like missing semicolon, return type error etc. But in our program, it is limited to only missing parenthesis and variable declaration in integer and float.

Without any doubt this project has made us more comfortable with coding and also enhanced our capacity to code more efficiently. This mini parser can be developed by adding some advanced feature in future.

**Reference:**

1. Data structures a pseudocode approach with c -Book by Richard F. Gilberg.
2. <https://stackoverflow.com/questions/7868936/read-file-line-by-line-using-ifstream-in-c>.