# Compiler Design Lab Final

CSE-0302 Summer 2021

Tarin Ahmed
UG02-50-19-020

*Department of Computer Science and Engineering*

*State University of Bangladesh (SUB)*
Dhaka, Bangladesh

tarinshara911@gmail.com

***Abstract*—A compiler translates the code written in one language to some other language without changing the meaning of the program. Compiler design covers basic translation mechanism and error detection & recovery.**

***Index Terms*—C++**

## I. Session 4: Detecting Simple Syntax Errors

```
#include <stdio.h>
#include <stdlib.h>

FILE *fp , *fp2;

void check_comment(char a)
{
    char x;

    if( a == '/')   //checking if the character starts with '/', it will be a comment
    {
        if((x=fgetc(fp))=='*')
         check_block_comment();

        else if( x == '/')   // else if the next character '/', it is the beginning of single
        {
          check_single_comment();

        }
        else
        {
            // when both the cases fail then it is not a comment
            fputc(a,fp2);
            fputc(x,fp2);
        }
    }
    // when all the conditions are false, add the character as it is in the new file.
    else
        fputc(a,fp2);
}

// function for block comments
void check_block_comment()
{

 char x,y;

    while((x=fgetc(fp))!=EOF)   // the block comme
    {

        if(x=='*')
        {
            y=fgetc(fp);  // check if it ends

            if(y=='/')
                return;
        }
    }

// function for single line comments
void check_single_comment()
{
 char x,y;

    while((x=fgetc(fp))!=EOF)

        if(x=='\n')
            return;  // if the comment ends return

    }
}
```

```
int main(void)
{
    char c;

    fp = fopen ("testfile.txt","r") ;   // first file in read mode
    fp2 = fopen ("solved.txt","w") ;    // second file in write mode

    while(
          (c=fgetc(fp))!=EOF)
        check_comment(c);   // checking for the beginning of a comment

     //  closing both files
    fclose(fp);
    fclose(fp2);

    return 0;
}
```

## II. Session 5: Use of CFGs for Parsing

```
#include<stdio.h>
#include<string.h>

int i,j,k,l,m,n=0,o,p,nv,z=0,t,x=0;
char str[10],temp[20],temp2[20],temp3[20];

struct prod
{
    char lhs[10],rhs[10][10];
    int n;
}pro[10];

void findter()
{
    for(k=0;k<n;k++)
    {
        if(temp[i]==pro[k].lhs[0])
        {
            for(t=0;t<pro[k].n;t++)
            {
                for(l=0;l<20;l++)
                    temp2[l]='\0';
                for(l=i+1;l<strlen(temp);l++)
                    temp2[l-i-1]=temp[l];
                for(l=i;l<20;l++)
                    temp[l]='\0';
                for(l=0;l<strlen(pro[k].rhs[t]);l++)
                    temp[i+l]=pro[k].rhs[t][l];
                strcat(temp,temp2);
                if(str[i]==temp[i])
                    return;
                else if(str[i]!=temp[i] && temp[i]>=65 && temp[i]<=90)
                    break;
            }
            break;
        }
        if(temp[i]>=65 && temp[i]<=90)
            findter();
    }
}

int main()
{
   FILE *f;
//    clrscr();
    for(i=0;i<10;i++)
        pro[i].n=0;

    f=fopen("in.txt","r");
    while(!feof(f))
    {
        fscanf(f,"%s",pro[n].lhs);
        if(n>0)
        {
            if( strcmp(pro[n].lhs,pro[n-1].lhs) ==
            {
                pro[n].lhs[0]='\0';
                fscanf(f,"%s",pro[n-1].rhs[pro[n-1
                pro[n-1].n++;
                continue;
            }
        }
        fscanf(f,"%s",pro[n].rhs[pro[n].n]);
        pro[n].n++;
        n++;
    }
    n--;

    printf("\n\nTHE GRAMMAR IS AS FOLLOWS\n\n");
    for(i=0;i<n;i++)
        for(j=0;j<pro[i].n;j++)
            printf("%s -> %s\n",pro[i].lhs,pro[i].

    while(1)
    {
        for(l=0;l<10;l++)
            str[0]=NULL;

        printf("\n\nENTER ANY STRING ( 0 for EXIT
        scanf("%s",str);
        if(str[0]=='0')
            break;

        for(j=0;j<pro[0].n;j++)
        {
            for(l=0;l<20;l++)
                temp[l]=NULL;
            strcpy(temp,pro[0].rhs[j]);

            m=0;
            for(i=0;i<strlen(str);i++)
```

```
            {
                if(str[i]==temp[i])
                    m++;
                else if(str[i]!=temp[i] && temp[i]>=65 && temp[i]<=90)
                {
                    findter();
                    if(str[i]==temp[i])
                        m++;
                }
                else if( str[i]!=temp[i] && (temp[i]<65 || temp[i]>90) )
                    break;
            }

            if(m==strlen(str) && strlen(str)==strlen(temp))
            {
                printf("\n\nTHE STRING can be PARSED !!!");
                break;
            }
        }

        if(j==pro[0].n)
            printf("\n\nTHE STRING can NOT be PARSED !!!");
    }

//    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}
```

## III. Session 6: Predictive Parsing

```
#include<bits/stdc++.h>
using namespace std;

vector<string>sp,ke,ri;
map<string,string>mp,mpp;
string ans;

bool isTERMINAL(char a){
    if(a>='A' && a<='Z') return true;
    return false;
}

void FIRST(string key){

    string val = mp[key];

    if(isTERMINAL(val[0])){
        string p = "";
        p += val[0];
        FIRST(p);
    }
    else{
        ans += val[0];
        ans += ",";
        int flag = 0;
        for(int i=0;i<val.size();i++){
            if(val[i]=='|'){
                flag = 1;
                continue;
            }
            if(flag){
                ans += val[i];
            }
        }
    }

void FOLLOW(string key,int z){
    int flag = 0;

    for(int i=0;i<ri.size();i++){
        if (ri[i].find(key) != string::npos) {
            if(key.size()==1){
                for(int j=0;j<ri[i].size();j++){
                    if(ri[i][j]==key[0]){
                        if(j+1<ri.size() && ri[i][
                            flag = 1;
                            if(isTERMINAL(ri[i][j+
                                if(z==0)ans += "$,
                                ans += ri[i][j+1];
                            }
                            else{
                                string g = ri[i];
                                g.erase(0,1);
                                FIRST(g);
                                if(z==0)ans += "$,
                                FOLLOW(mpp[ri[i]],

                            }

                            break;
                        }
                    }
                }
            }
            else{
                flag = 1;

                for(int j=0;j+1<ri[i].size();j++){
                    if(ri[i][j]==key[0] && ri[i][j+
                        if(j+2>=ri[i].size()){
                            FOLLOW(mpp[ri[i]],1);
                            if(z==0)ans += ",$";
                        }
                        else{

                        }
                    }
                }
```

```
                break;
            }
        }
        if(flag) break;
    }


}



string remove_space(string s){

    string p="";

    for(int i=0;i<s.size();i++){
        if(s[i]!=' ') p = p + s[i];
    }

    return p;

}



int main(){

    freopen("input.txt","r",stdin);
    freopen("out.txt","w",stdout);

    string s;

    while(getline(cin,s)){
        sp.push_back(remove_space(s));
    }

    for(int i=0;i<sp.size();i++){
        int flag = 0;

        string key="",val="";

        for(int j=0;j<sp[i].size();j++){
            if(sp[i][j]=='='){
                flag = 1;
                continue;
            }

            if(flag==0) key += sp[i][j];
            else val += sp[i][j];
        }

        mp[key] = val;
        ke.push_back(key);
    }

    cerr<<"FIRST: \n\n";

    cout<<"FIRST: \n\n";

    for(int i=0;i<ke.size();i++){
        ans = "";
        FIRST(ke[i]);
        cerr<<"FIRST("<<ke[i]<<")"<<" = {"<<ans<<"
        cout<<"FIRST("<<ke[i]<<")"<<" = {"<<ans<<"
    }

    for(int i=0;i<ke.size();i++){

        string val = mp[ke[i]];
        string v = "";

        for(int j=0;j<val.size();j++){
            if(val[j]=='|') break;
            v += val[j];
        }

        mp[ke[i]] = v;
        mpp[v] = ke[i];
        ri.push_back(v);
    }

    cerr<<"\nFOLLOW: \n\n";
    cout<<"\nFOLLOW: \n\n";


    for(int i=0;i<ke.size();i++){
        ans = "";

        FOLLOW(ke[i],0);
        cerr<<"FOLLOW("<<ke[i]<<")"<<" = {"<<ans<<
        cout<<"FOLLOW("<<ke[i]<<")"<<" = {"<<ans<<
    }


}
```


## ACKNOWLEDGMENT

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this final lab work.