PROJECT REPORT

Syntax Checker Program In C

Prepared By: Rajdip Pal
Computer Science and Engineering Dept.

Date: 10th October 2020

Table of Contents

1. Abstract

For this project, I chose to create a syntax checker program. The purpose of the program was to read another .cpp file with its source code and give specific information and errors in the code. By adapting this functionality I would provide the information about the functions, variables, brackets used in the program.

2. Introduction

It is an individual programming project which requires significant effort and knowledge in C programming skills including file handling, pointers and functions. This report aims to provide a detailed look at the resulting output and the key features of the program.

2.1 Background

At the beginning of the project, I had some previous experience with C programming, but I invested a large portion of time to understand and apply its functionality, as well as looking at alternative implementations to figure out what I thought worked best. Ultimately, I would end up having to teach myself C programming, mostly relying on my ability to apply the knowledge I accumulated over the last two years at Swami Vivekananda Institute of Science and Technology to different scenarios. I have tried to keep a record for this in the code itself by including links to the appropriate documentation.

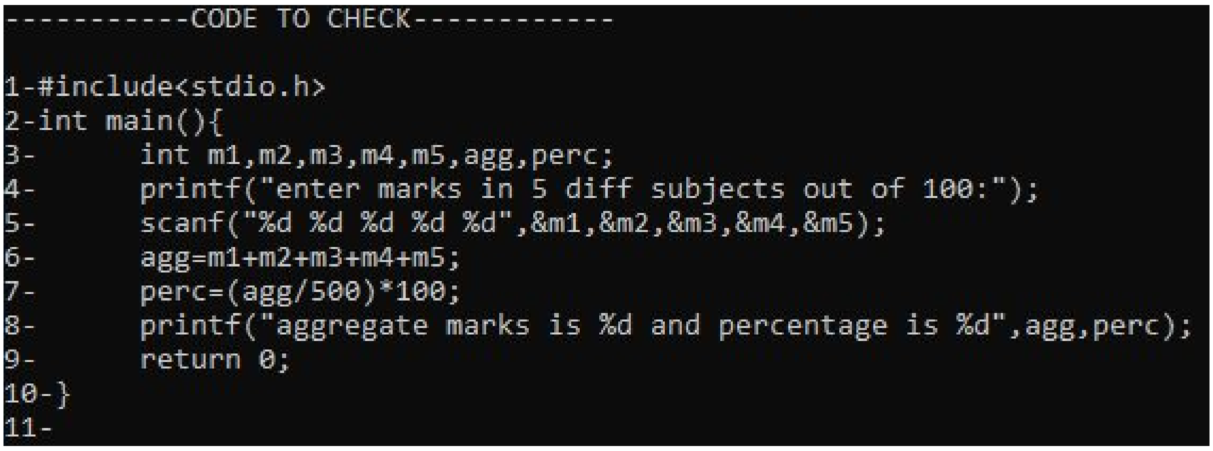
2.2 Project Brief

The main goal of this project was to create a Syntax Checker C program that would read another .cpp file and its respective source code line by line to bring out some basic information, functions used, variables used, and errors in brackets. Though while executing a program the errors are shown, but this project can work as a compact information sheet of another long code. This can help the programmer to find out errors and whoever is working on the code can easily understand the basics and functionality of it.

3. Project Details

3.1 Overview

The output first shows the code which is to be checked stored in the project folder and file named “checkprogram”. However, this file name can be changed in the line 71 of the source code.



```
-----------CODE TO CHECK------------

1-#include<stdio.h>
2-int main(){
3-      int m1,m2,m3,m4,m5,agg,perc;
4-      printf("enter marks in 5 diff subjects out of 100:");
5-      scanf("%d %d %d %d %d",&m1,&m2,&m3,&m4,&m5);
6-      agg=m1+m2+m3+m4+m5;
7-      perc=(agg/500)*100;
8-      printf("aggregate marks is %d and percentage is %d",agg,perc);
9-      return 0;
10-}
11-
```

Following it comes the output which shows the information and errors in the code which is to be checked.
These include:

1. **Basic Information**
2. **Information about functions used**
3. **Information about variables used**
4. **Information about the brackets**

which is discussed in the functionality section in this report.

3.2 Source Code

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

typedef struct{
                    char l[1000];
                    int ln;
}s1;

void file_read(s1 file[],int* lcount);
void check(s1 file[],int *intf,int *curve, int*prntf ,int*scnf,int*fpts,int*fgts ,int*ffloat, int*fscnf, int*curly,
int*straight,int*flot, int*nt, int*chr, int*pts,int*gts, int* vod );

main(){
s1 file[1000];
int lc=0;
char d[901];
printf("-----------CODE TO CHECK------------\n\n");
file_read(file, &lc);
int
i=0,k=0,j=1,m,intf=0,prntf=0,scnf=0,fpts=0,fgts=0,ffloat=0,fscnf=0,curly=0,curve=0,straight=0,flot=0,nt
=0,chr=0,pts=0,gts=0,vod=0;
check(file,&intf,&curve, &prntf ,&scnf,&fpts,&fgts ,&ffloat, &fscnf, &curly, &straight,&flot, &nt, &chr,
&pts,&gts, & vod );


printf("\n\n\n\t\tBASIC
INFORMATION\n______________________________________________________\n");
printf("\n\tTotal number of lines ->\t       %d",lc);
printf("\n\tTotal number of variables ->\t\t%d",chr+flot+nt-intf-ffloat);
printf("\n\tTotal built in functions ->\t\t%d",prntf+scnf+fscnf+fgts+fpts);


printf("\n\n\n\tINFORMATION ABOUT FUNCTIONS
USED\n______________________________________________________\n");
if(vod>0)printf("\n\tNumber of \"void\" fuctions-->         %d",vod);
if(intf>0)printf("\n\tNumber of \"integer\" fuctions-->      %d",intf);
if(ffloat>0)printf("\n\tNumber of \"float\" fuctions-->      %d",ffloat);
if(prntf>0)printf("\n\tNumber of \"printf\" function-->       %d",prntf);
if(scnf>0)printf("\n\tNumber of \"scanf\" function-->        %d",scnf);
if(fscnf>0)printf("\n\tNumber of \"fscanf\" function-->       %d",fscnf);
if(pts>0)printf("\n\tNumber of \"puts\" function-->         %d",pts);
if(gts>0)printf("\n\tNumber of \"gets\" function-->         %d",gts);
if(fgts>0)printf("\n\tNumber of \"fgets\" function-->        %d",fgts);
if(fpts>0)printf("\n\tNumber of \"fputs\" function-->        %d",fpts);


printf("\n\n\n\tINFORMATION ABOUT VARIABLES
USED\n______________________________________________________\n");
if(nt-intf>0)printf("\n\tNumber of int variables used -> %d",nt-intf);
else printf("\n\tNO int variables are used.");

if(flot>0)printf("\n\tNumber of floats variables used -> %d",flot-ffloat);
else printf("\n\tNO float variables are used.");

if(chr>0)printf("\n\tNumber of char variables used -> %d",chr);
else printf("\n\tNO char variables used.");


printf("\n\n\n\tINFORMATION ABOUT THE
BRACKETS\n_______________________________________________\n");
if(curly>0)printf("\n\t%d ERROR! Closing curly bracket is missing. ",curly);
else if(curly<0)printf("\n\t%d ERROR! Opening curly bracket is missing.",-curly);
else printf("\n\tCurly brackets are perfectly closed.");

if(curve>0)printf("\n\t%d Closing curve bracket is missing. ",curve);
else if(curve<0)printf("\n\t%d Opening curve bracket is missing.",-curve);
else printf("\n\tCurve brackets are perfectly closed.");

if(straight>0)printf("\n\t%d Closing long bracket is missing. ",straight);
else if(straight<0)printf("\n\t%d Opening long bracket is missing.",-straight);
else printf("\n\tLong brackets are perfectly closed.");

}


void file_read(s1 file[],int* linecount){
        FILE *p;
        p=fopen("checkprogram.cpp","r");
        int i=0,k=0;
  while(!feof(p))
{
                fgets(file[i].l,1000,p);
                if((file[i].l[1]==' '||file[i].l[0]=='\n')||(file[i].l[0]=='/'&&file[i].l[1]=='/'))
                continue;
                *linecount+=1;
                file[i].ln=strlen(file[i].l);
                for(k=0;k<file[i].ln;k++)
                {
                        if(file[i].l[k]=='/'&&file[i].l[k+1]=='/')
                        {
                          file[i].l[k]='\n';
                          file[i].l[k+1]='\0';
                  }
                file[i].ln=strlen(file[i].l);
                }
                        printf("%d",i+1);
                        printf("-%s",file[i].l);
                        i++;
}
}
```

```
void check(s1 file[],int*intf,int *curve, int*prntf ,int*scnf,int*fpts,int*fgts ,int*ffloat, int*fscnf, int*curly,
int*straight,int*flot, int*nt, int*chr, int*pts,int*gts, int* vod ){
        int l,i,k,m;
        for(i=0;i<500;i++){
                for(k=0;k<file[i].ln;k++){
/*scanf*/
if(file[i].l[k]=='s'&&file[i].l[k+1]=='c'&&file[i].l[k+2]=='a'&&file[i].l[k+3]=='n'&&file[i].l[k+4]=='f'&&file[i].l[k+5]
=='('){*scnf+=1;}
/*printf*/
if(file[i].l[k]=='p'&&file[i].l[k+1]=='r'&&file[i].l[k+2]=='i'&&file[i].l[k+3]=='n'&&file[i].l[k+4]=='t'&&file[i].l[k+5]
=='f'&&file[i].l[k+6]=='('){*prntf+=1;}
/*fscanf*/
if(file[i].l[k]=='f'&&file[i].l[k+1]=='s'&&file[i].l[k+2]=='c'&&file[i].l[k+3]=='a'&&file[i].l[k+4]=='n'&&file[i].l[k+5]
=='f'&&file[i].l[k+6]=='('){*fscnf+=1;}
/*gets*/
if(file[i].l[k]=='g'&&file[i].l[k+1]=='e'&&file[i].l[k+2]=='t'&&file[i].l[k+3]=='s'&&file[i].l[k+4]=='('){*gts+=1;}
/*puts*/
if(file[i].l[k]=='p'&&file[i].l[k+1]=='u'&&file[i].l[k+2]=='t'&&file[i].l[k+3]=='s'&&file[i].l[k+4]=='('){*pts+=1;}
/*fgets*/
if(file[i].l[k]=='f'&&file[i].l[k+1]=='g'&&file[i].l[k+2]=='e'&&file[i].l[k+3]=='t'&&file[i].l[k+4]=='s'&&file[i].l[k+5]
=='('){*fgts+=1;}
/*fputs*/
if(file[i].l[k]=='f'&&file[i].l[k+1]=='p'&&file[i].l[k+2]=='u'&&file[i].l[k+3]=='t'&&file[i].l[k+4]=='s'&&file[i].l[k+5]
=='('){*fpts+=1;}


        if(file[i].l[k]=='v'&&file[i].l[k+1]=='o'&&file[i].l[k+2]=='i'&&file[i].l[k+3]=='d'&&file[i].l[k+4]==' '){
                        for(m=0;m<file[i].ln;m++){
                                if(file[i].l[m]=='('){
                                        *vod+=1;
                                        break;
                                }
                        }
                }


if(file[i].l[k]=='f'&&file[i].l[k+1]=='l'&&file[i].l[k+2]=='o'&&file[i].l[k+3]=='a'&&file[i].l[k+4]=='t'&&file[i].l[k+5]=
=' '){
                        for(m=0;m<file[i].ln;m++){
                                if(file[i].l[m]=='('){
                                        *ffloat+=1;
                                        break;
                                }
                        }
                }
                        if(file[i].l[k]=='{'){*curly++;}
                        if(file[i].l[k]=='}'){*curly--;}
                        if(file[i].l[k]=='('){*curve++;}
                        if(file[i].l[k]==')'){*curve--;}
                        if(file[i].l[k]=='['){*straight++;}
```

```
                                if(file[i].l[k]==']'){*straight--;}

if(file[i].l[k]=='f'&&file[i].l[k+1]=='l'&&file[i].l[k+2]=='o'&&file[i].l[k+3]=='a'&&file[i].l[k+4]=='t'&&file[i].l[k+5]=
=' '){
                                        *flot+=1;
                                        for(k=k+6;k<=file[i].ln;k++){
                        if(file[i].l[k]==',')*flot+=1;
                                                        if(file[i].l[k]==';')break;
                                                else continue;
                                        }
                                }
                                if(file[i].l[k]=='i'&&file[i].l[k+1]=='n'&&file[i].l[k+2]=='t'&&file[i].l[k+3]==' '){
                                        *nt+=1;
                                        for(k=k+4;k<=file[i].ln;k++){
                    if(file[i].l[k]==',')*nt+=1;
                                          if(file[i].l[k]==';')break;
                                          else continue;
                                        }
                                }

if(file[i].l[k]=='c'&&file[i].l[k+1]=='h'&&file[i].l[k+2]=='a'&&file[i].l[k+3]=='r'&&file[i].l[k+4]==' '){
                                        *chr+=1;
                                        for(k=k+5;k<=file[i].ln;k++){
                    if(file[i].l[k]==',')*chr+=1;
                                          if(file[i].l[k]==';')break;
                                          else continue;
                                        }
                                }
                        }
                }
                for(i=0;i<500;i++){
                        for(k=0;k<file[i].ln;k++){
                                if(file[i].l[k]=='i'&&file[i].l[k+1]=='n'&&file[i].l[k+2]=='t'&&file[i].l[k+3]==' '){
                                  for(k=k+4;k<=file[i].ln;k++){
                                          if(file[i].l[k]=='('){
                                                  *intf+=1;
                                                  break;
                                        }
                                }
                        }
                        }
                }
}
```

3.3 Functions Used

A. file_read

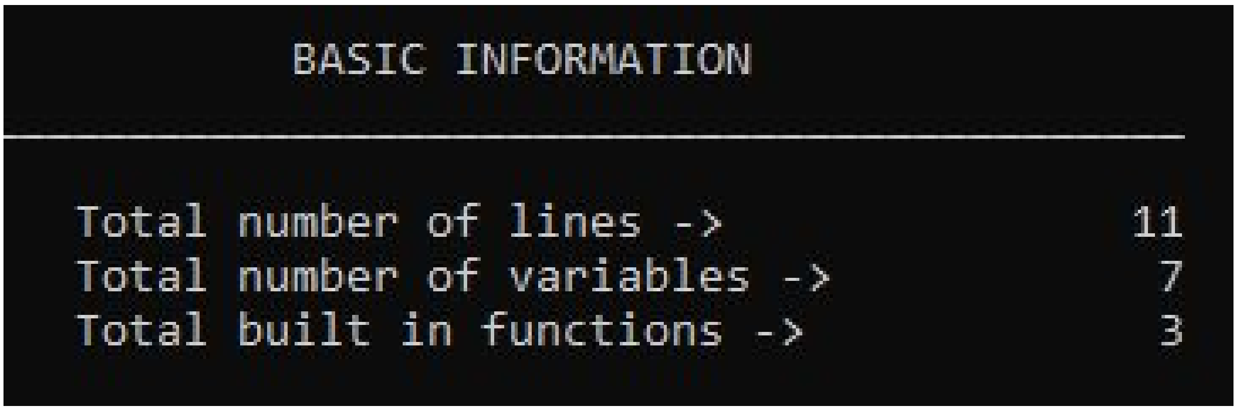
This function was used to read the file which is to be checked named as "checkprogram.cpp". Thus, the file is opened and read. After that, it is printed in the output screen line by line in the section named 'Code To Check'.

B. Check

This function was used to check the code and it is called in the main function to bring out the basic information, functions, variables and brackets used in the code. This is divided into 4 sub-parts in which the basic information part counts the lines, variables and build-in functions used. Similarly, the functions information is displayed and counted how many times it is used in the code. Then, the variables are displayed if it is used in the program. At the end, the curly braces are checked if they are perfectly closed and for any error in it.

4. Functionality

4.1 Basic Information

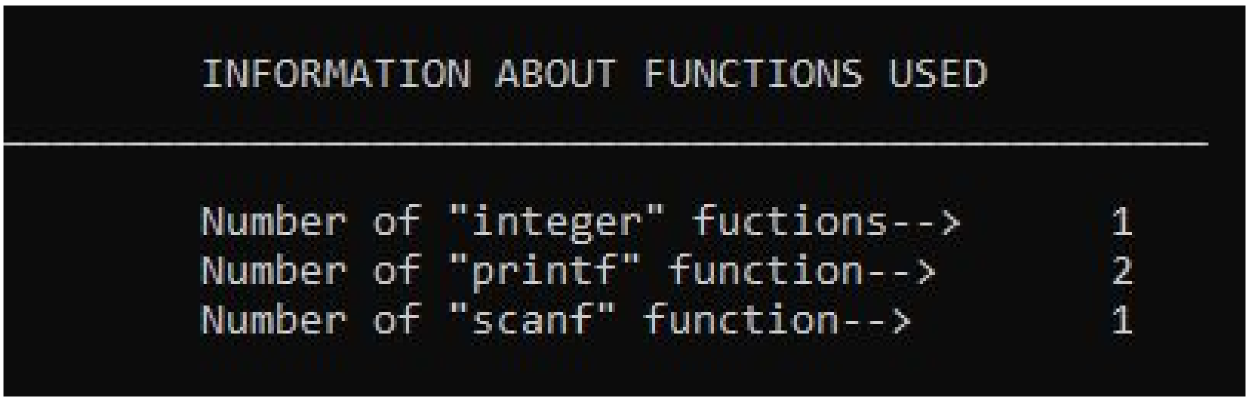


```
          BASIC INFORMATION
____________________________________________

  Total number of lines ->                11
  Total number of variables ->             7
  Total built in functions ->              3
```

This portion shows the total number of lines in the code, the number of variables used and the total build in functions used in the total program which is to be checked.

4.2 Information about functions used

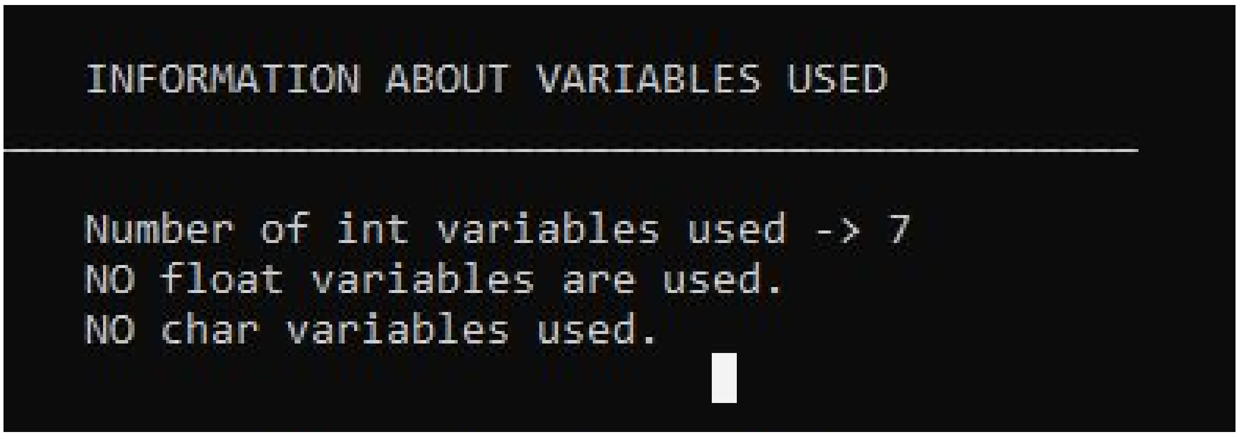


```
        INFORMATION ABOUT FUNCTIONS USED
__________________________________________________

        Number of "integer" fuctions-->      1
        Number of "printf" function-->       2
        Number of "scanf" function-->        1
```

The output shows the number of scanf, printf, fscanf, gets, puts, fgets, fputs functions used in the code.

4.3 Information about variables used

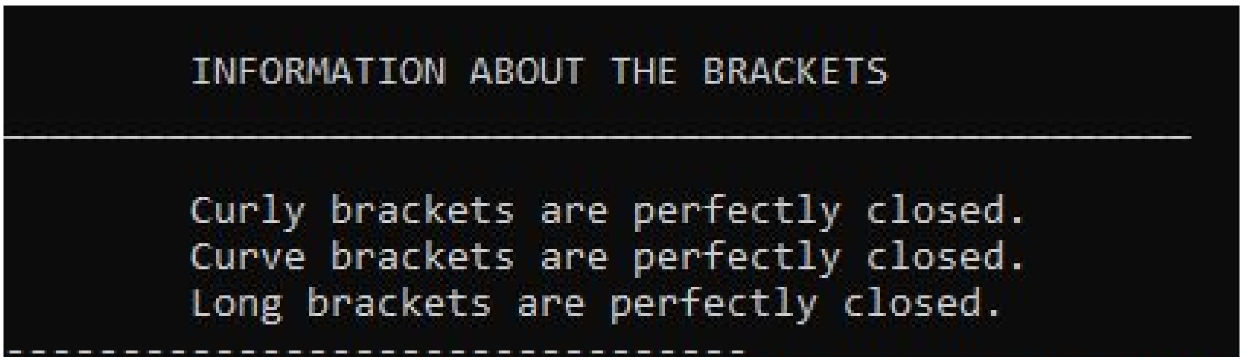


```
   INFORMATION ABOUT VARIABLES USED
________________________________________________

   Number of int variables used -> 7
   NO float variables are used.
   NO char variables used.
```

This shows the number of int, float and char variables used and also if they are used in the program or not.

4.4 Information about the brackets



```
        INFORMATION ABOUT THE BRACKETS
_____________________________________________________

        Curly brackets are perfectly closed.
        Curve brackets are perfectly closed.
        Long brackets are perfectly closed.
--------------------------------
```

This shows the brackets i.e. curly, curved and long brackets are perfectly closed or not or the position where it is missing so that it can be solved.

5. Future Concerns

5.1 Errors/Bugs

This program has a bug when the program which is to be checked is not provided. Thus it displays garbage value when executed.

6. Conclusion

It was an exciting project to work on and there is a lot I learnt from it, above and beyond its original scope. I was able to study and train myself on development in an environment which was new to me, which I believe I have been reasonably successful with. Although I was not able to complete some of the functionality, I believe there is still a lot of potential for this program, and will continue development in the future.