SPL-1 Project Report, 2019

Program Slicing

SE 305: Software Project Lab 1

Submitted by

Md. Mehedi Hasan

BSSE Roll No. : 1037

BSSE Session: 2016-17

Supervised by

Abdus Sattar

Designation: Lecturer

Institute of Information Technology

University of Dhaka
[29-05-2019]

Table of Contents

1. Introduction

Program slicing is a methodology that omits program modules that are irrelevant to a particular computation process based on a criterion known as the slicing criterion. Using this methodology we can automatically determine the relevance of a module in a particular computation. Once such modules are found, the testing process takes considerably less effort and time because testing phase generally accounts for more than one third of time during the software development cycle.

In my project, I have implemented static slicing methodology to find the relevant parts of slicing criterion & then used tree data structure to store the hierarchy of relevant statements and finally printing them into an output file.

1.1. Background Study

Program slicing was originally introduced by Mark Weiser.
• Finding all statements in a program that directly or indirectly affect the value of a variable occurrence is referred to as Program Slicing .

Slicing Criterion

• The pair <s,v> is known as Slicing Criterion where ‘s’ is a program point of interest and ‘v’ is a variable used or defined

Types of program slicing

Depending on the run-time environment, it can be
• Static slicing
• Dynamic slicing

Depending on graph traversal, it can be
• Backward slicing
• Forward slicing

Static Slicing

Static slicing may be used to identify these parts of the program that potentially contribute to the computation of the selected function for all possible programs inputs. Static slicing is helpful to gain a general understanding of these parts of the program that contribute to the computation to the selected function. Although static slicing has many advantages in the process of program understanding, static slices are frequently still large subprograms because of the imprecise computation of these slices. In addition, static slices cannot be used in the process of understanding of program execution.

Dynamic Slicing

Dynamic slicing is used to identify these parts of the program that contribute to the computation of the selected function for a given program execution (program input). Dynamic slicing may help to narrow down this part of the program that contributes to the computation of the function of interest for particular program input. Dynamic slices are frequently much smaller than static slices. Moreover, dynamic slicing may be used to understand program execution.

Forward Slicing

Forward slices contain all parts of the program that might be influenced by the variable.

Backward Slicing

Backward slices contain all parts of the program that might have influenced the variable at the statement under consideration.

1.2. Challenges

During implementing this project, I have faced so many challenges and some of them are mentioned here:

- Understanding & implementing the algorithm for the very first time.
- Parsing each statement and then comparing the data to find dependencies.
- Storing & handling a large size of data.
- Finally, making the sliced program executable.

2. Project Overview

The whole project is the combination of all of the following steps that I have done :

2.1 . Reading the input file

At first, I have opened the input file in read mode and placed my file pointer at the specified statement mentioned in the slicing criterion. Then I have started to read each statement and parse them into words to find out the dependent variables, functions and therefore the statements And store them in a tree for further use.

2.2. Finding dependencies

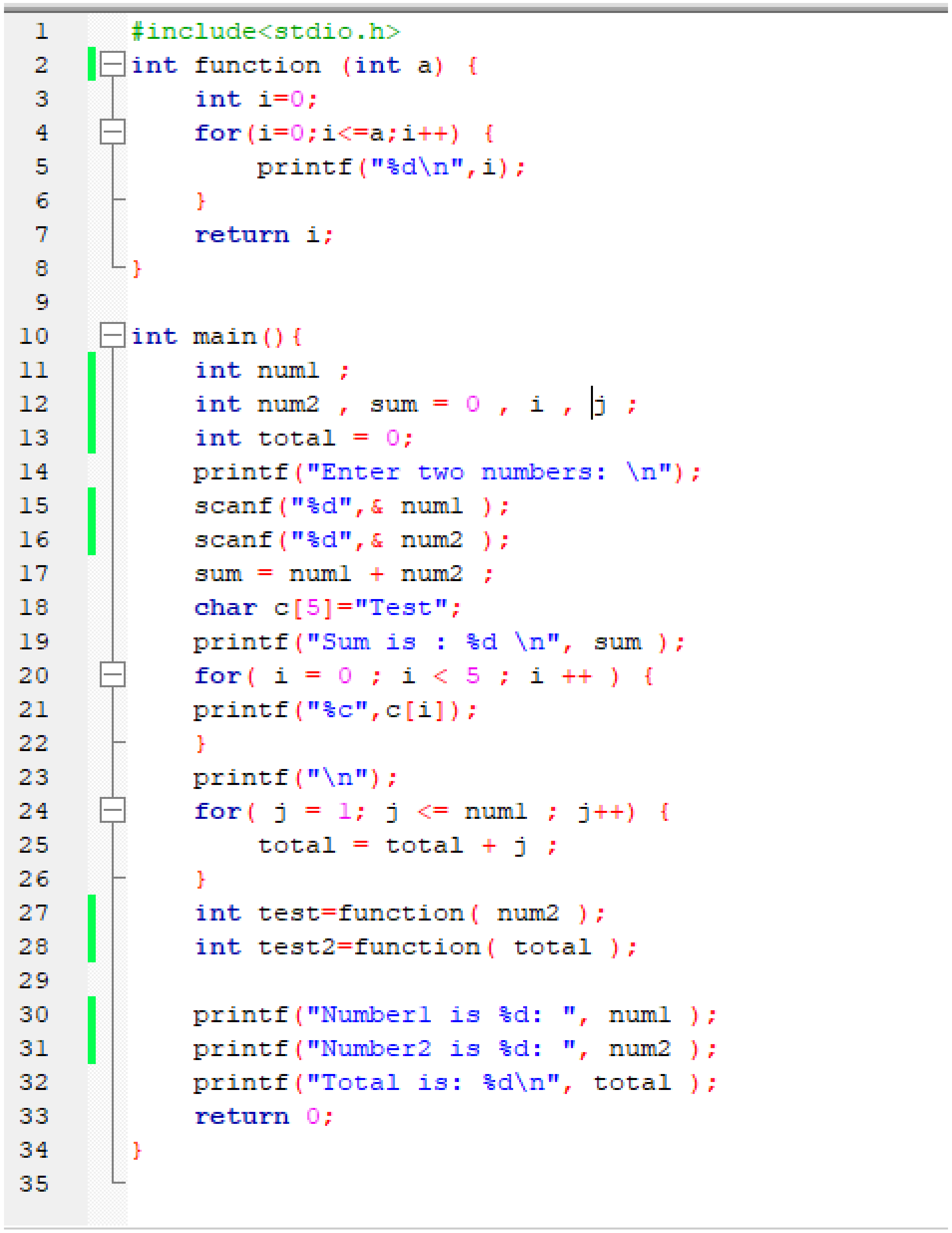
Using some operators and comparing, I have found directly dependent statements and then using these, I have found out the indirectly dependent statements.

2.3. Printing into output file

Using tree data structure, all the relevant statements and functions are printed in an output file which is executable also.

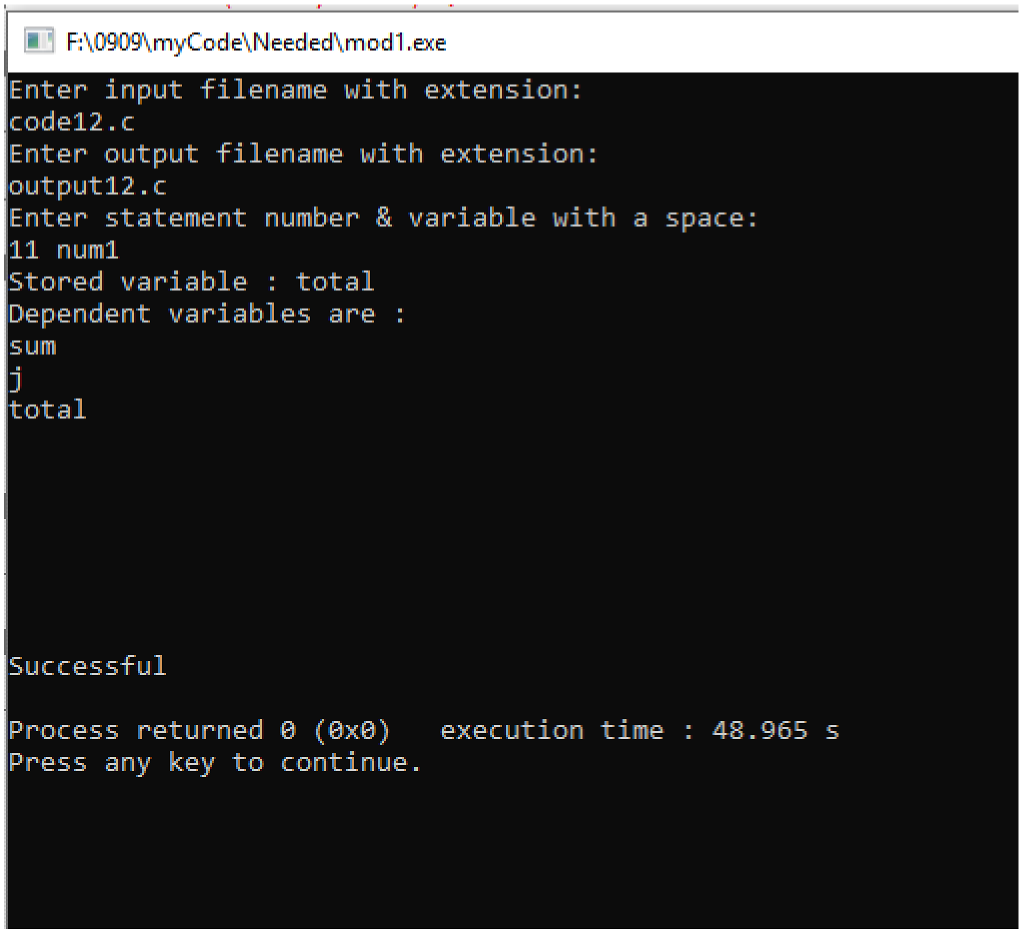
3. User Manual

I am going to slice this program on slicing criterion <11,num1>



```
1   #include<stdio.h>
2   int function (int a) {
3       int i=0;
4       for(i=0;i<=a;i++) {
5           printf("%d\n",i);
6       }
7       return i;
8   }
9
10  int main(){
11      int num1 ;
12      int num2 , sum = 0 , i , j ;
13      int total = 0;
14      printf("Enter two numbers: \n");
15      scanf("%d",& num1 );
16      scanf("%d",& num2 );
17      sum = num1 + num2 ;
18      char c[5]="Test";
19      printf("Sum is : %d \n", sum );
20      for( i = 0 ; i < 5 ; i ++ ) {
21      printf("%c",c[i]);
22      }
23      printf("\n");
24      for( j = 1; j <= num1 ; j++) {
25          total = total + j ;
26      }
27      int test=function( num2 );
28      int test2=function( total );
29
30      printf("Number1 is %d: ", num1 );
31      printf("Number2 is %d: ", num2 );
32      printf("Total is: %d\n", total );
33      return 0;
34  }
35
```

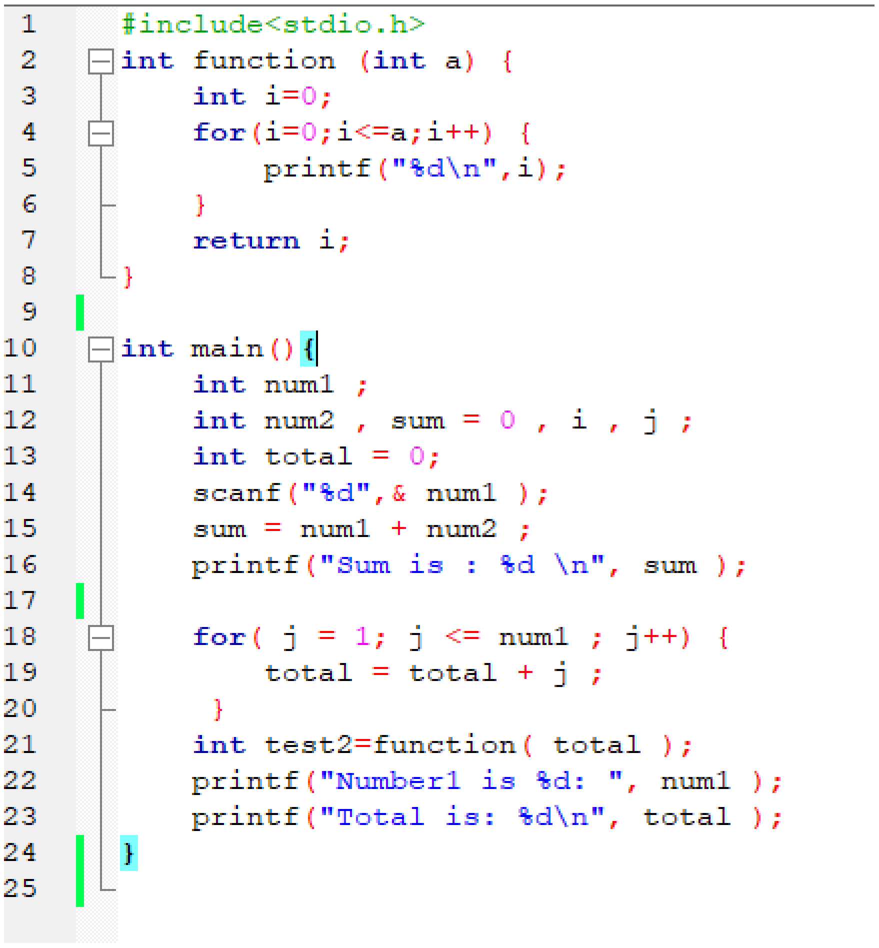
Now I am running my program and taking input :



F:\0909\myCode\Needed\mod1.exe

```
Enter input filename with extension:
code12.c
Enter output filename with extension:
output12.c
Enter statement number & variable with a space:
11 num1
Stored variable : total
Dependent variables are :
sum
j
total







Successful

Process returned 0 (0x0)   execution time : 48.965 s
Press any key to continue.
```

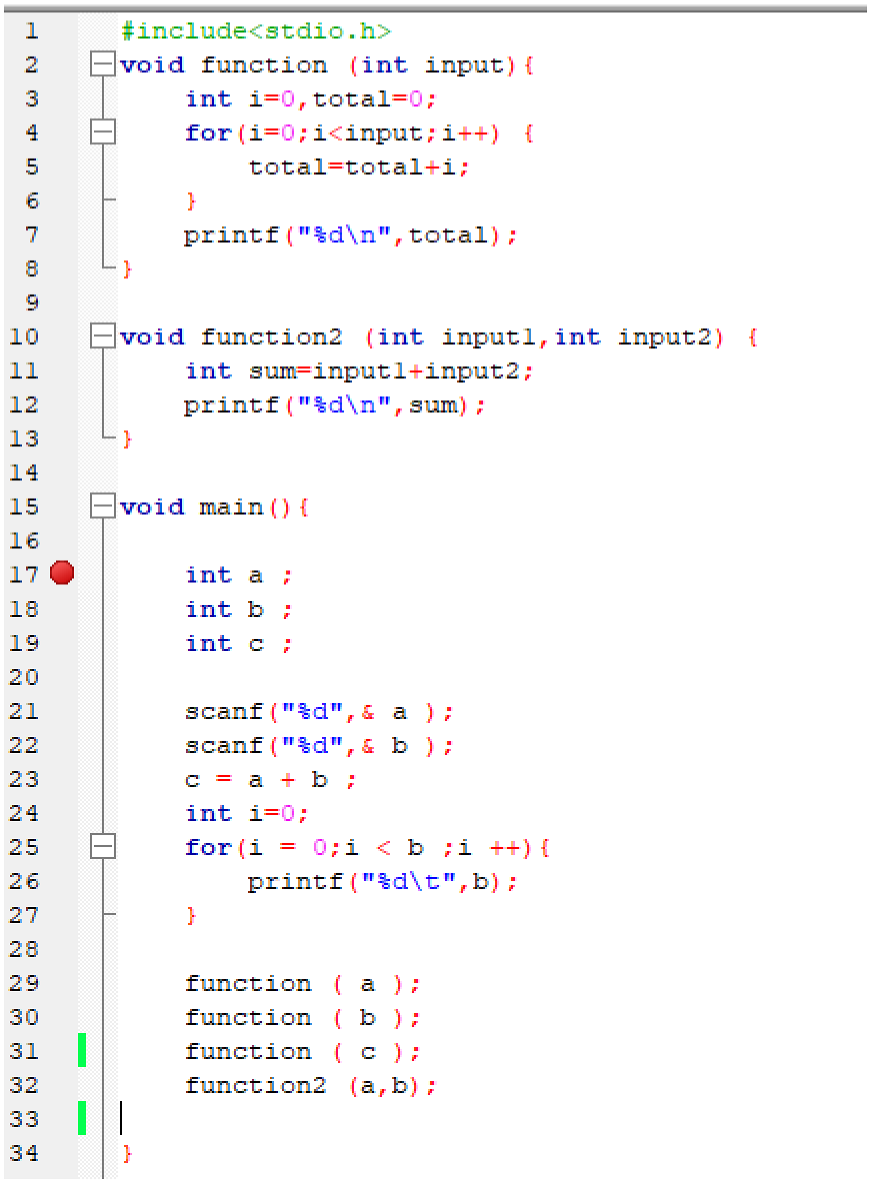
And this is my sliced output file :



```
#include<stdio.h>
int function (int a) {
    int i=0;
    for(i=0;i<=a;i++) {
        printf("%d\n",i);
    }
    return i;
}

int main(){
    int num1 ;
    int num2 , sum = 0 , i , j ;
    int total = 0;
    scanf("%d",& num1 );
    sum = num1 + num2 ;
    printf("Sum is : %d \n", sum );

    for( j = 1; j <= num1 ; j++) {
        total = total + j ;
     }
    int test2=function( total );
    printf("Number1 is %d: ", num1 );
    printf("Total is: %d\n", total );
}
```

Here is another example :

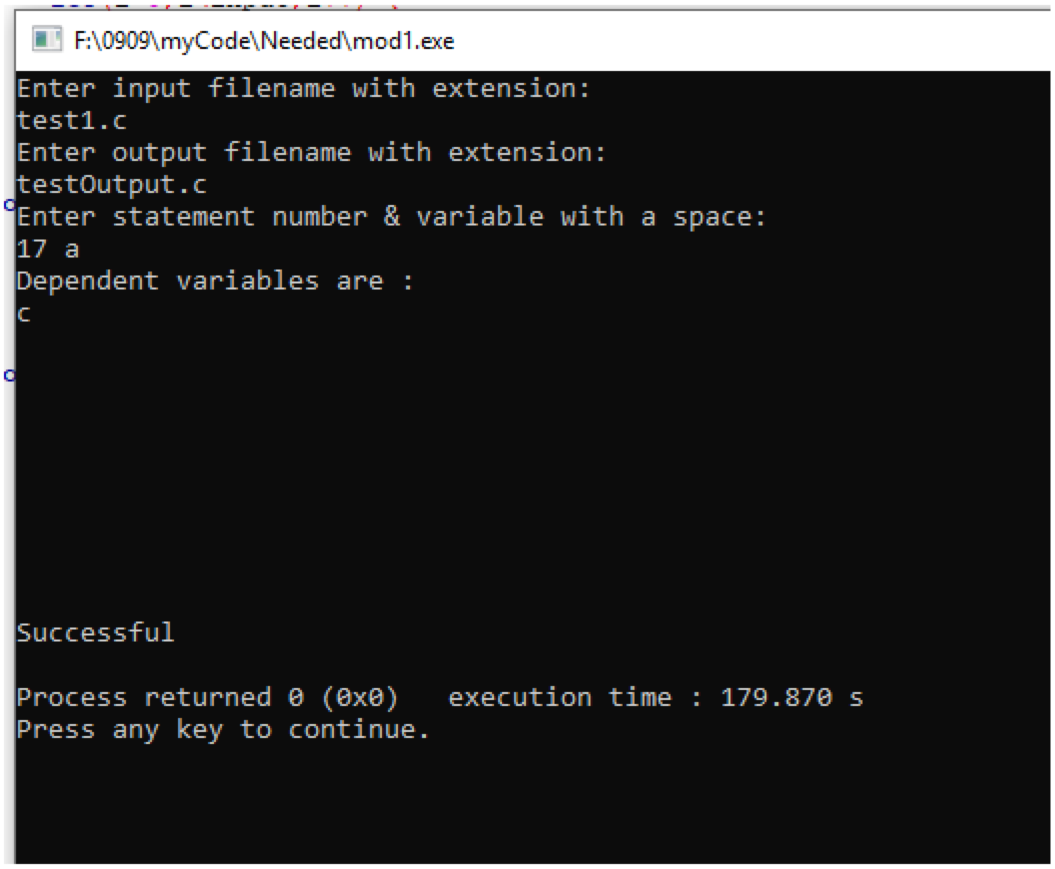
This is sample code as input :



```
#include<stdio.h>
void function (int input){
    int i=0,total=0;
    for(i=0;i<input;i++) {
        total=total+i;
    }
    printf("%d\n",total);
}

void function2 (int input1,int input2) {
    int sum=input1+input2;
    printf("%d\n",sum);
}

void main(){

    int a ;
    int b ;
    int c ;

    scanf("%d",& a );
    scanf("%d",& b );
    c = a + b ;
    int i=0;
    for(i = 0;i < b ;i ++){
        printf("%d\t",b);
    }

    function ( a );
    function ( b );
    function ( c );
    function2 (a,b);

}
```

This step is for taking input :

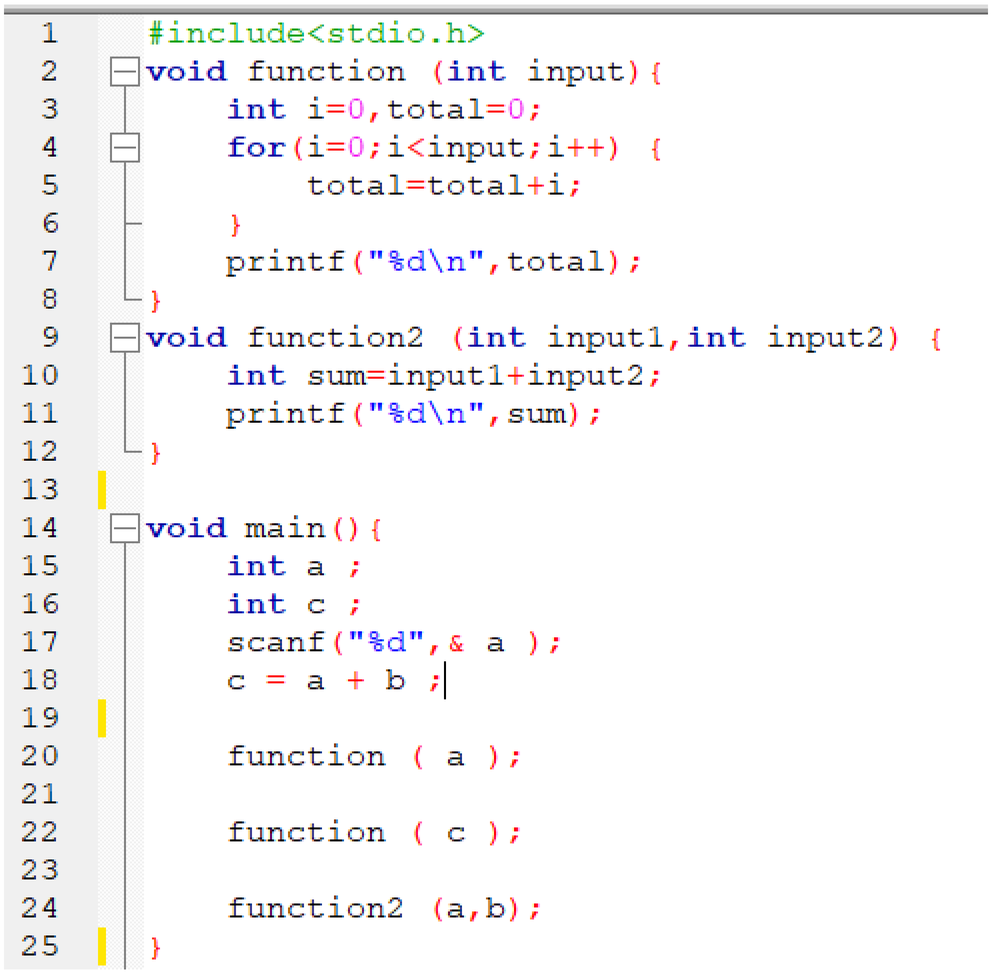
Slicing criterion <17,a>



F:\0909\myCode\Needed\mod1.exe

```
Enter input filename with extension:
test1.c
Enter output filename with extension:
testOutput.c
Enter statement number & variable with a space:
17 a
Dependent variables are :
c









Successful

Process returned 0 (0x0)   execution time : 179.870 s
Press any key to continue.
```

And this is sliced output file :



```
#include<stdio.h>
void function (int input){
     int i=0,total=0;
     for(i=0;i<input;i++)  {
          total=total+i;
     }
     printf("%d\n",total);
}
void function2 (int input1,int input2)  {
     int sum=input1+input2;
     printf("%d\n",sum);
}

void main(){
     int a ;
     int c ;
     scanf("%d",& a );
     c = a + b ;

     function ( a );

     function ( c );

     function2 (a,b);
}
```

5.Conclusion

I have developed an algorithm for computing forward static slices of simple programs which derives the slices which are affected by the variable. This slicing technique has various uses like analyzing and reusing the code which makes the task of the programmer easier. While implementing this project, I have learnt how to handle large size of code, different operations on strings, how to extract or parse data, how to compare, how to find out direct or indirect dependencies, how to implement tree data structure and such more topics which has improved my programming skill and I hope that this will help me in future works.

6.References

[1] https://en.wikipedia.org/wiki/Program_slicing
[2] http://www0.cs.ucl.ac.uk/staff/M.Harman/exe1.html
[3]
https://people.eecs.ku.edu/~hossein/Teaching/Fa14/814/.../program-slicing-purvi.ppt

[4]
https://www.researchgate.net/publication/2390528_An_Overview_of_Program_Slicing
[5]
www.cs.loyola.edu/~binkley/papers/fosm08-slicing.pdf

[6] A survey of program slicing techniques, Frank Tip, IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598, USA