EXPERIMENT 2
C PROGRAMMING AND DEBUGGING

Aims

1. Learning how to use Microsoft Visual Studio compiler.
2. Compilation and execution of programs with examples.
3. Debugging C programs using Microsoft Visual Studio

In this experiment we will be using Microsoft Visual Studio 2008; however, with small differences, usage of most integrated development environments are similar.

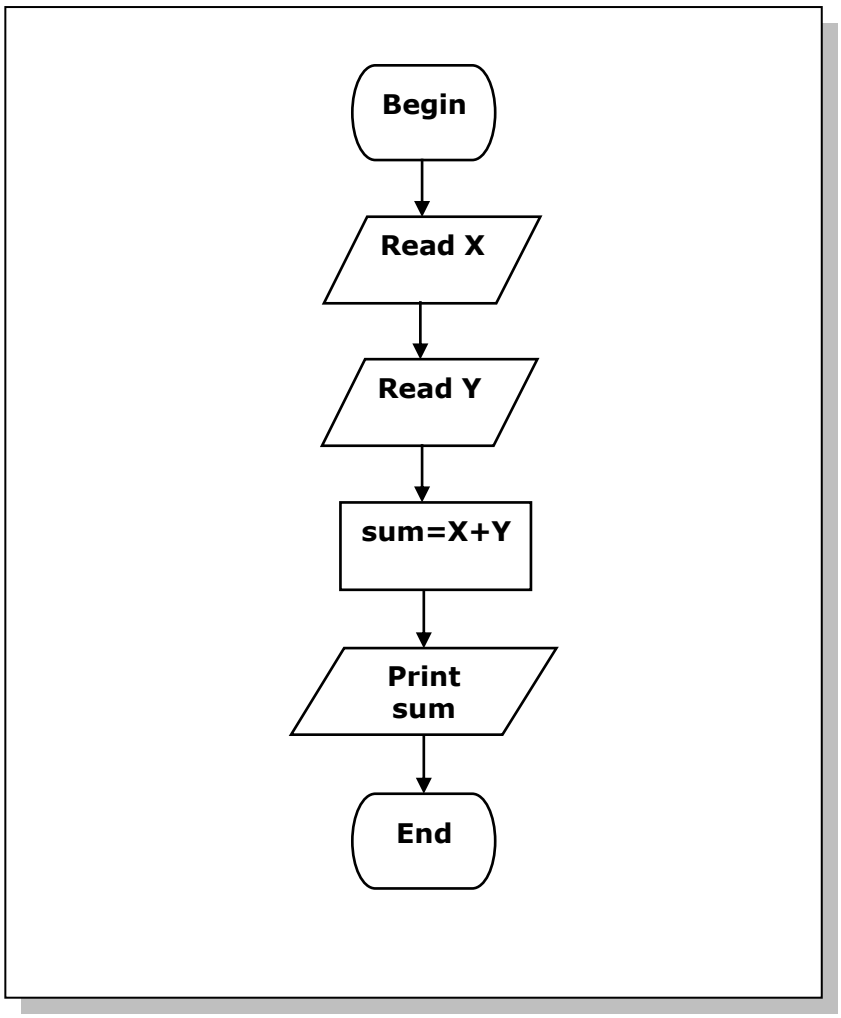
LABORATORY WORK:

1. In this laboratory work, we will start writing C programs. The following algorithm is for addition of two numbers entered by the user. The aim is to prompt the user for the first number and then the second number and print the sum of these numbers. The algorithm and flowchart for this problem is as follows:

Algorithm:

```
1. Begin
2. Read X
3. Read Y
4. sum=X+Y
5. Print sum
6. End
```

Flowchart:



The C program for this algorithm is:

```
/* This program prints the sum of two numbers given by the
user */
#include <stdio.h> /* library files declaration */
int main()
{
        /* variable declaration */
        int x, y, sum;
        x=0;
        y=0;
        printf("My First C program \n");

        /* Ask to user to enter a value for the first variable */
        printf("Please enter the first number \n");

        /* Read the value entered by the user and assigns it to
        the variable x */
        scanf("%d", &x);

        //Ask to user to enter a value for the second variable

        printf("Please enter the second number \n");

        /* Read the value entered by the user and assigns it to
        the variable y */
        scanf("%d",&y);

        /* Adds the values of x and y and assigns the result to
        the variable sum */
        sum = x + y;

        /* Display the value of the variable sum on the
        screen*/
        printf("The sum is %d \n", sum);
        return 0;
} /* end main */
```

You should type program given above into the editor as it can be seen in figure 1.

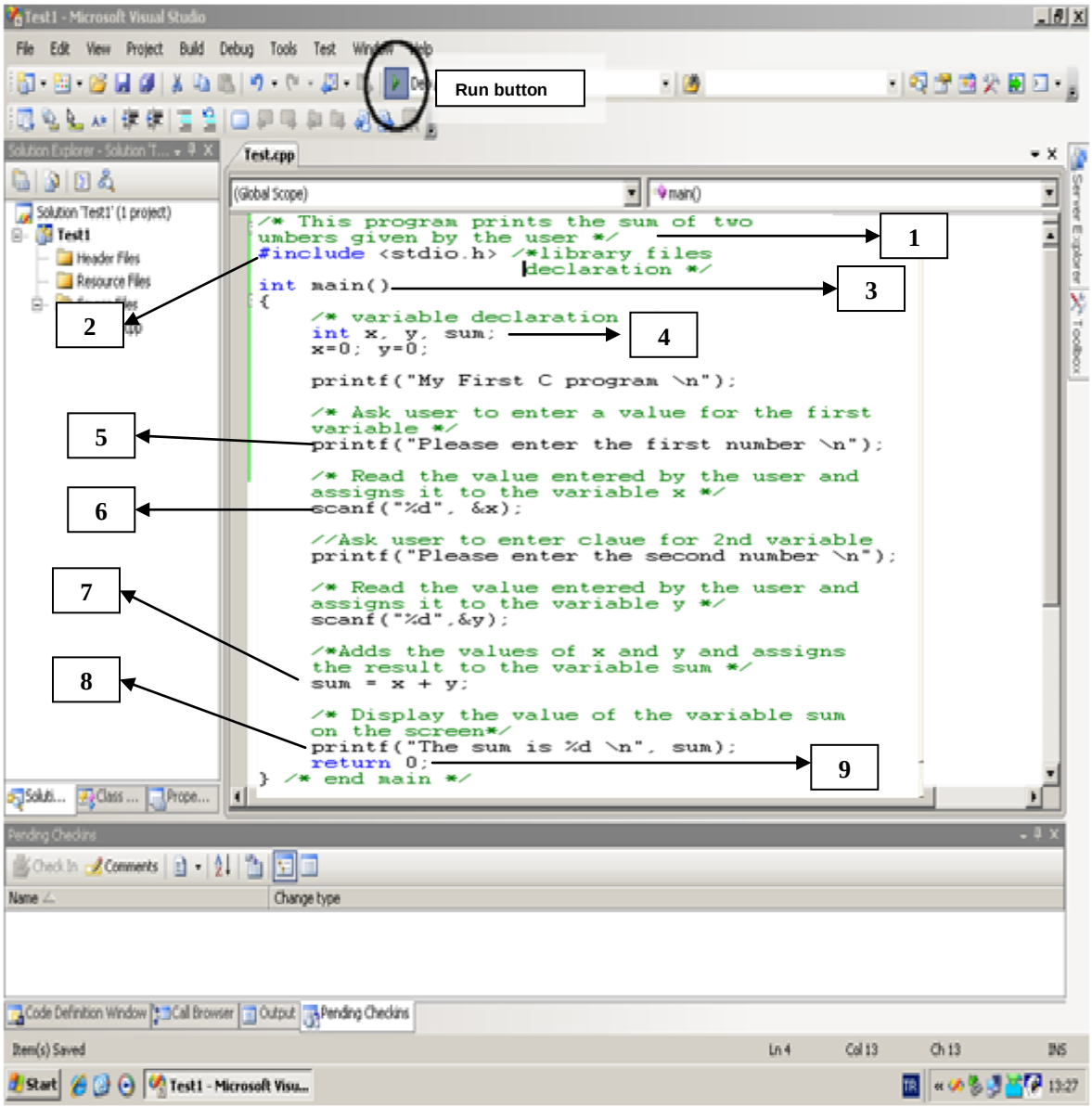


Figure 1. Example program

Let's see how each part of the algorithm is written as a part of the C code. /* and */ symbols [1] can be used to provide comments in a program and anything written in between these symbols is not part of the C code and not compiled.

To begin writing a C program, one must include the necessary library files for the predefined functions and system operations such as printing and reading output from the user or for example mathematical functions.

These include files are defined in the following format:

- **#include <stdio.h>** [2]

The pound sign (#) must come before the keyword "include" and the name of the library file must be written inside < and > for this definition. The blue color of the #include indicates that this is a keyword. "stdio.h" is the standard input and output header file that must be included if we want to use functions such as "*printf"* and "*scanf".*

The beginning of the program also includes the declaration of the main function. Each C program has a function called main which is the actual program. In our example,

int main() [3] is where this definition takes place. The format of the main function declaration is:

- "return type" main (arguments).

The "return type" is the result that the program returns once it is finished. The blue colored int keyword indicates that this is an integer. The keyword "void" can be used as an argument to indicate that there is no return value. Arguments are parameters that are sent to the program from the command prompt when the program is started. This declaration must be completed by a return statement at the end of the program for an error free program. In our example this is as follows:

- **return 0;** [9]

The curly braces, { and }, indicate the beginning and the end of the main program or any other fragment in a C program such as a IF statement, WHILE loop or FOR loop. These fragments are part of the code that is meant to be executed together based on a condition. In our example, the braces show the beginning and the end of the program, corresponding to "begin" and "end" steps of the algorithm.

When we want to output a message to the user, meaning that we want to display something to the screen, the "*printf"* keyword can be used.

From our example program we can see the following uses of "*printf":*

- **printf("My First C program \n");** [5] ➔ this full C program statement will print the message "My First C program" to the screen and move the cursor to the next line. Anything you want to print to the user must be put in quotes (i.e. " "). \n means that we want a new line after this is printed.

  Note that the semicolon (;) must be put after each C program statement to indicate that that statement is completed.

- **printf("The sum is %d \n:", sum);** [8]➔ We can also print variables stored in memory that we are using in our program to the screen. In our case, since we added two numbers, we should show the result. To print a variable, (an integer in this example) the percent sign (%) followed by the variable type ("d" means integer) will be placed where we want the variable to be printed. And the name of the variable must be written after " and , so that we can indicate which variable we want to print.

The variables that will be used in the program must be declared in the following format:

int x,y; [4]

The blue colored "*int"* keyword reserves an integer with the given variable name, and multiple variables can be declared by separating them with a comma. To initialize these variables we can use a simple assignment such as *x=0;*

If we don't initialize a variable, it's value is undefined. You should initialize the variable before using it in the program. You can also assign a value to a variable and initialize it through user input (reading). For this purpose, the "*scanf"* keyword is used. For example:

scanf("%d", &x); [6] **/* Reads the value entered by the user and assigns it to the variable x */**

Similar to *"printf", "%d"* indicates that we are dealing with an integer. This is followed by a comma (,) and the & sign indicating where the user input will be stored provided with the variable name. The above *scanf* statement will put the value given by the user into the variable x. Note that you should use & for all types of variables (int, float, double, char) except the string types .

Any mathematical operation can be performed in C programming by typing the operation in a left aligned format. That is, we can say

- **sum = x + y;** [7]

which will add the value of "x" and the value of "y" and put the result into the variable sum.

Once a C program is written, you must compile, build and execute the program. The compile phase is for finding errors in your program. For example, if you do not include a ";" at the end of a statement you will receive an error. The build phase is for linking the library files and other predefined properties to your program. If the program compiles and builds successfully, then it can be executed. To run your program you can press the button shown in Figure 5 or "F5" key.

Fixing Problems

If there is an error in your program while compiling, Visual studio will ask if you want to run last working version of your program. Since we need to fix the problems we will answer "No" to this question and abort the running our program. After closing this dialog, a list of errors will be displayed. This window is shown in Figure 6. If you cannot see this window in your screen, you can open it using "View" menu. On the top of this list you will find "Errors", "Warnings" and "Messages" buttons. These buttons control the visibility of these types of items. We are mostly interested in "Errors" and "Warnings", so make sure that these two will be displayed. If you double click on an error or warning in this list, Visual Studio will take you to the line that is causing the error or warning. Some errors are caused by the general settings of your program; therefore, the line number information might not be available.

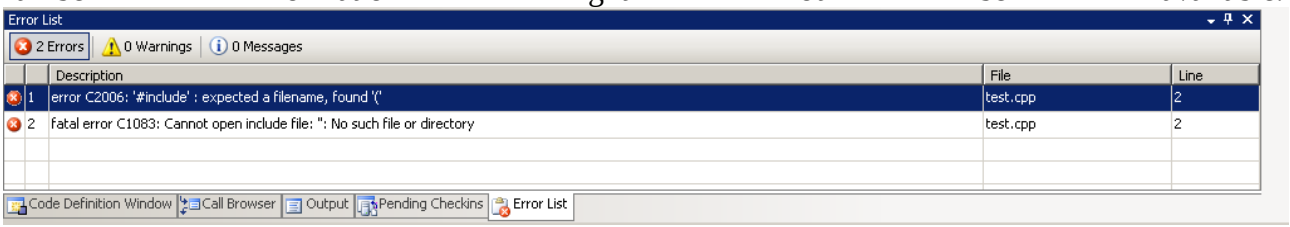


Figure 6: Error list

It is important to know that an error might be caused by previous line. If you are getting too many errors or if you are sure that the line displayed in the error list does not contain any problems, remember to check quotes, parentheses or semicolons (";") in the previous line. Most of the time missing semicolon error is given to the next line.

2. The goal of this task is to see what kind of errors can be made during writing of a C program. find and fix the errors in the code provided below.

```
/* this programs prints the sum of two
numbers given by the user */

include <stdio.h>  /  library files declaration */

int main2(){
        /* variable declaration */
        integer x, y, sum;
        x=0
        y=0;
        printf("My First C program \n);

         /* Ask the user to enter a value for the first variable */
        printf("Please enter the first number \n");

         /* Read the value entered by the user and
        assigns it to the variable x */
        scanf("%d", x);

         /* Ask the user to enter a value for the
        second variable */
        printf("Please enter the second number \n");

         /* Read the value entered by the user
        and assigns it to the variable y */
        scanf("d",&y);

         /* Adds the values of x and y and assigns
        the result to the variable sum */
        sum == x + y;

         /* Display the value of the variable sum on the screen*/
        printf("The sum is %d \n", &sum);
        return ;
} /* end main */
```