

Lab 1: Introduction to Dev-C++

Date: 5 June 2009

Number of Problems: 2

Introduction to Labs

Algorithm can be understood purely from mathematical point of view. However, most frequent of time, it is much better if you have hands on experience in realizing the algorithms into the actual computer program. This year, we push this aspect forward such that Lab is integrated into the lecturing hours. Our goal is to have you, our dear students, be able to convert your algorithmic idea into working source code. Moreover, you will have a chance to experiment with your code, to actually see how each line of code work and how much you can tinker it.

We faithfully hope that, with the series of labs, the learning of Algorithm would be funnier and more interesting to all of you

How do the Labs Work?

Each week, you will receive a list of problems to be solved. The problems will somehow relate to the topic we discussed in the class prior to the lab. All you need to do is to solve the problem using the things we talked in the class. Most of the time, we will provide a starting code for you and you will be required to write down some more code or to fix something in the code. The problem will be simple enough to be done within one and a half hour. Each problem comes with the definition of input and output. It also contains the introduction section which describes the detail of the problem. You need to read the problem definition and try to accomplish what it asks.

Nonetheless, you are not on your own. There will be a plethora of Teaching Assistants (TA), all equipped with years experience in algorithm and code punching, ready to serve you. And if it would make any difference, all lecturers will be there providing any help you so required as well. So, use us, don't hesitate to ask any question.

TIPS

- All Labs worth 10 points
- You will gain half of the points just by visiting the lab (yes, we check your attendance)
- The remaining points will be gained by completing the lab
- You will be graded by TA. They have the right to judge whether your code is correct
- You can discuss with your friends but you are not allowed to copy their code. It must be your hands that type anything into your computer. DO NOT COPY YOUR FRIENDS!!! Once done, you will get zero point for that lab
- Don't forget to ask your TA for any question

Problem 0: Introduction to Labs Environment (Dev-C++)

Input: nothing

Output: the string “Hello, CP!”

Objective

- Must be able to create a C project using Dev-C++
- Must be able to compile C code and be able to run it
- Must be able to write C code to display a string on the screen

Introduction

The development environment for the Labs is Dev-C++. This introductory problem shows you, step-by-step, how to use Dev-C++ to develop software using C language. For this problem, all you need to do is to write a C code that print the word “Hello, CP!” on the Screen.

***** Your task is to write a program that display “Hello, CP!” on the screen*****

Using Dev-C++

1. We first need to create a folder to store all of our work. Since you have your own storage mapped to each machine as drive Z:\, let us create a folder for our lab there. Create A folder called “Z:\AlgoLab” in drive Z:\.
2. Launch Dev-C++ by clicking the shortcut on your desktop shown in Fig. 1. The Dev-C++ main screen will be displayed as in Fig. 2. It will show the tip of the day. Read it, if you like, and then click the “close” button.



Figure 1: The Dev-C++ shortcut icon

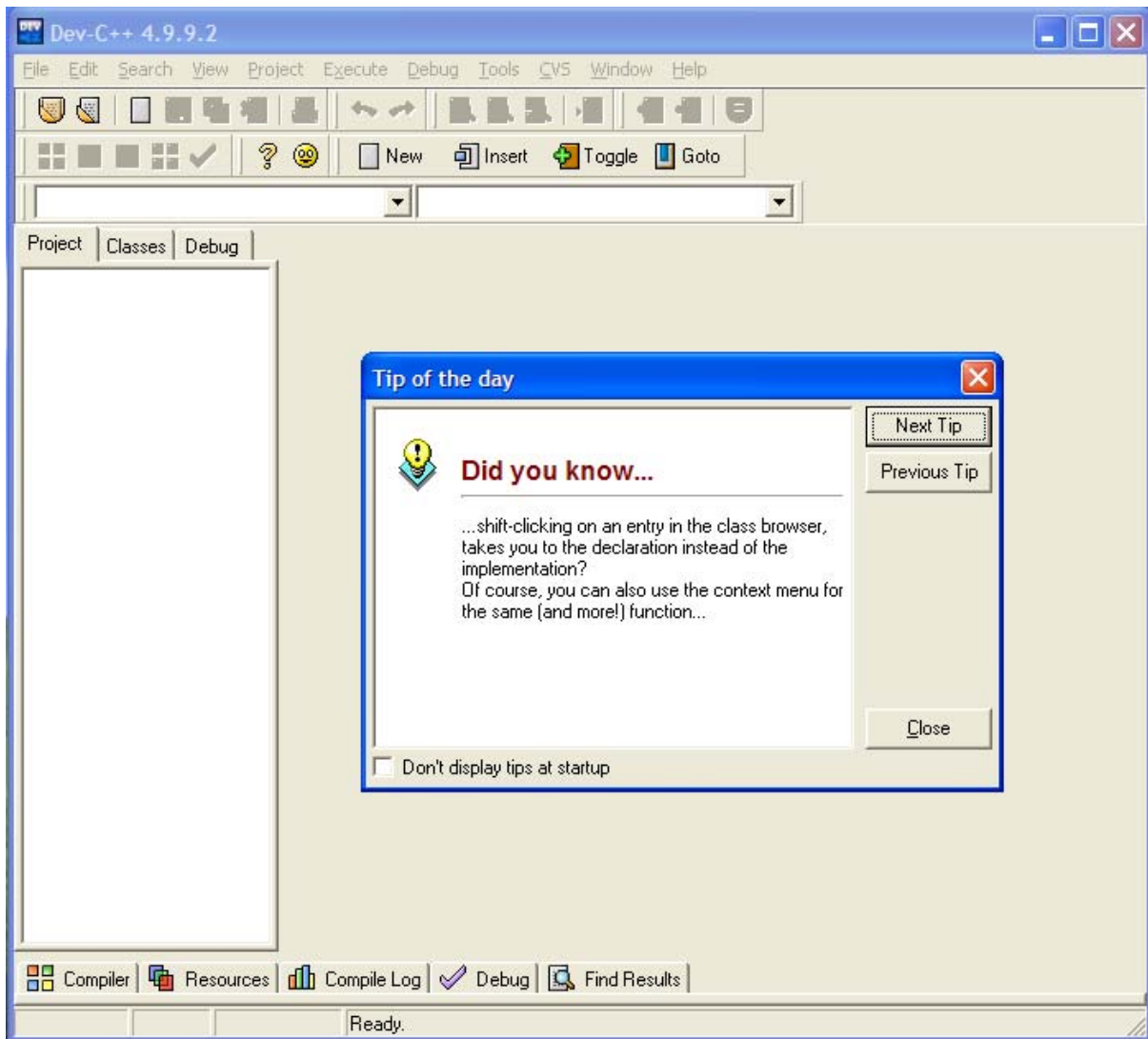


Figure 2: Dev-C++ main screen

3. Dev-C++ manages source code as projects. Let us create one project. Click on the menu item **File / New / Project** as shown in Fig. 3. The New Project window will pop up as in Fig. 4.

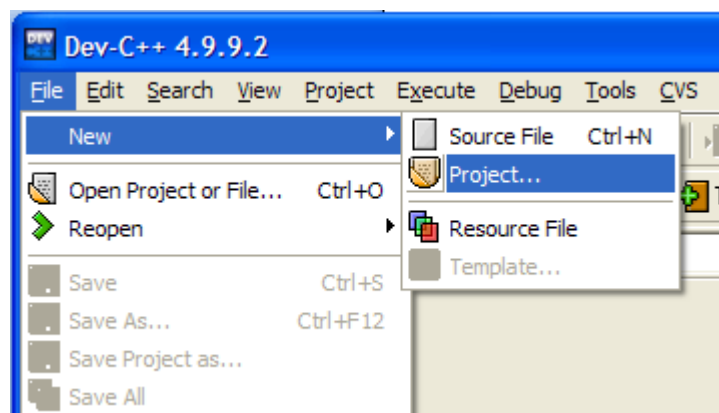


Figure 3: New Project Menu.

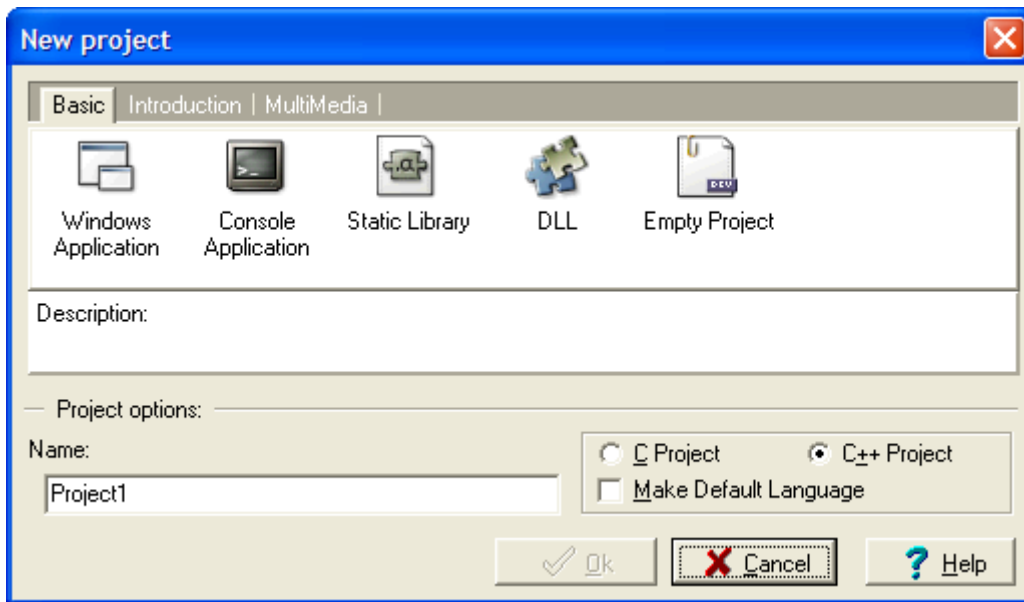


Figure 4: New Project Windows.

4. All of our labs will be windows console applications using simple C language. So, we need to tell Dev-C++ to create such project. Click on “Console Application” button and select “C Project” radio button. You will need to input a project name as well. Please enter “Lab1-Prob0” as the project name. Fig. 5 shows what should be done.

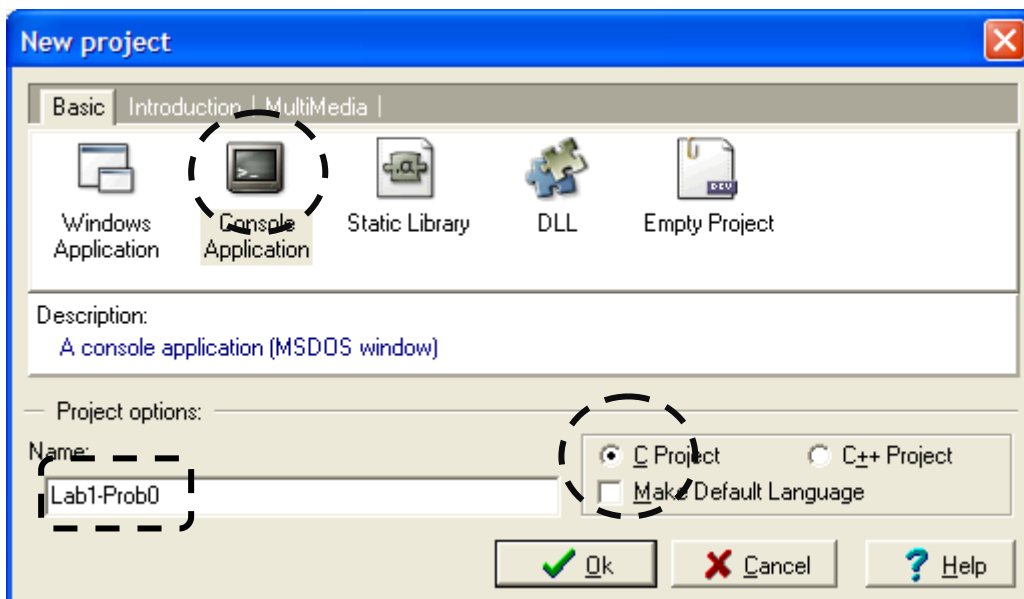


Figure 5: Configuring a Project.

5. Dev-C++ will ask you where to save the project file, save it at “Z:\AlgoLab\Lab1”. You need to create the folder “Z:\AlgoLab\Lab1” as well.

- The screen will show the main Dev-C++ screen and Dev-C++ will create the starting file (main.c) for you as shown in Fig. 6. Save this file as well, using the name "main.c". This can be done by the menu item **File / Save** or you can press the short cut key **Ctrl+S**

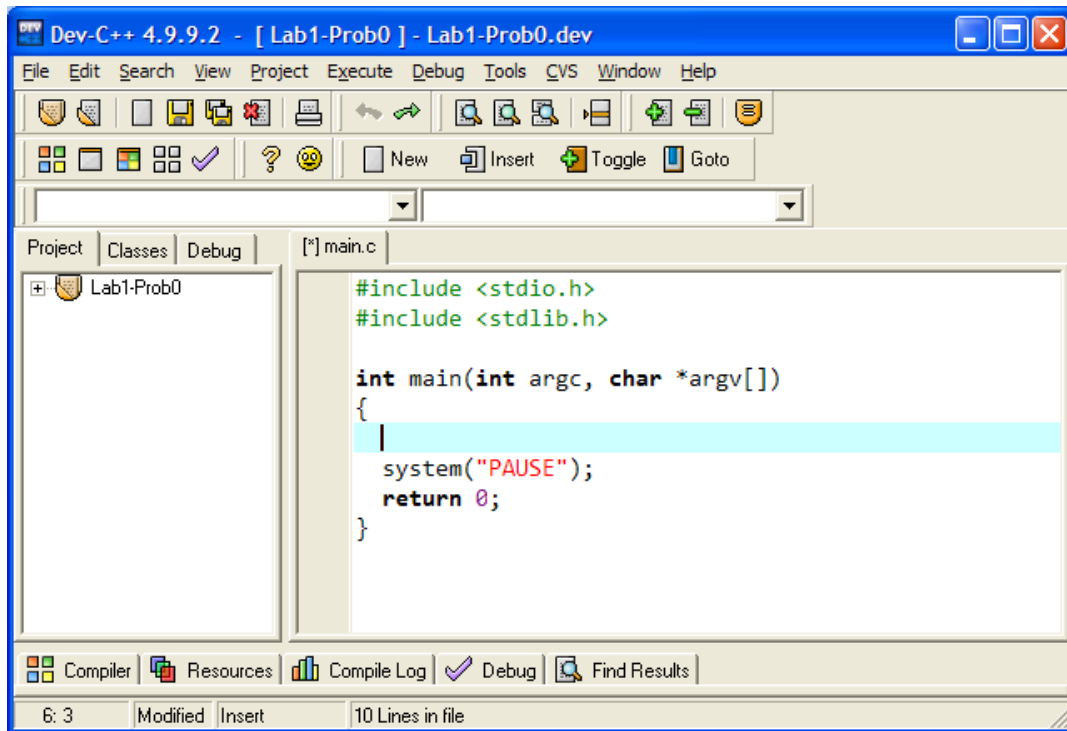


Figure 6: Main File.

- Now, we are ready to write any code. This is your first lab, so we are going to do it for you. What you need to do is to type a line saying `printf("Hello, CP!\n");` just above the line saying `system("PAUSE");` as shown in Fig. 7.

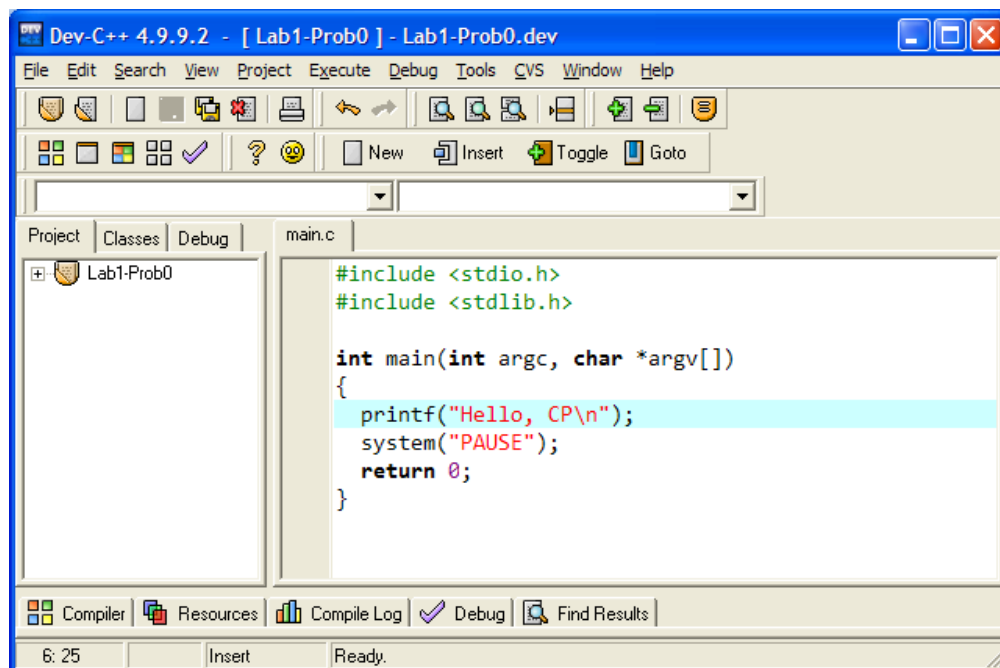


Figure 7: The Code.

8. The code is ready to be used. Save it again before we are going to compile and run it. To compile and run this source code, we need to click the run button located near the top left corner of the screen, please see Fig. 8. Click it, or use the short cut key "F9".

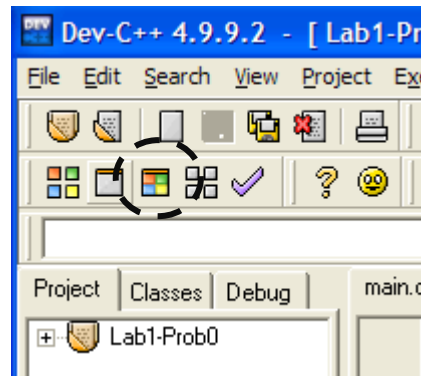


Figure 8: The run button.

9. Dev-C++ will compile and run the code. If nothing's wrong, you can see the result of your code displayed as a console window as in Fig. 9.

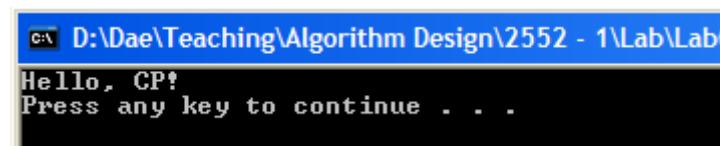


Figure 9: The result of the code.

10. This completes the requirement of this problem. However, we are going to do something more. Let's try tinker with the code. Start by removing the semi-colon after the `printf` line. The result should be as in Figure 10;

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

int main(int argc, char *argv[])
{
    printf("Hello, CP!\n")
    system("PAUSE");
    return 0;
}
```

Figure 10: Tinkering with the code.

11. Let us try to compile the code again. You will see that Dev-C++ complain something about our doing as in Fig. 11. Since C statement has to end with a semi-colon, omitting one makes the code become wrong and Dev-C++ detects and complains about it. You can see that, at the bottom of Dev-C++ window, there are lines saying something wrong about the code. If the code is not correct, this panel will tell you anytime you compile the source code.

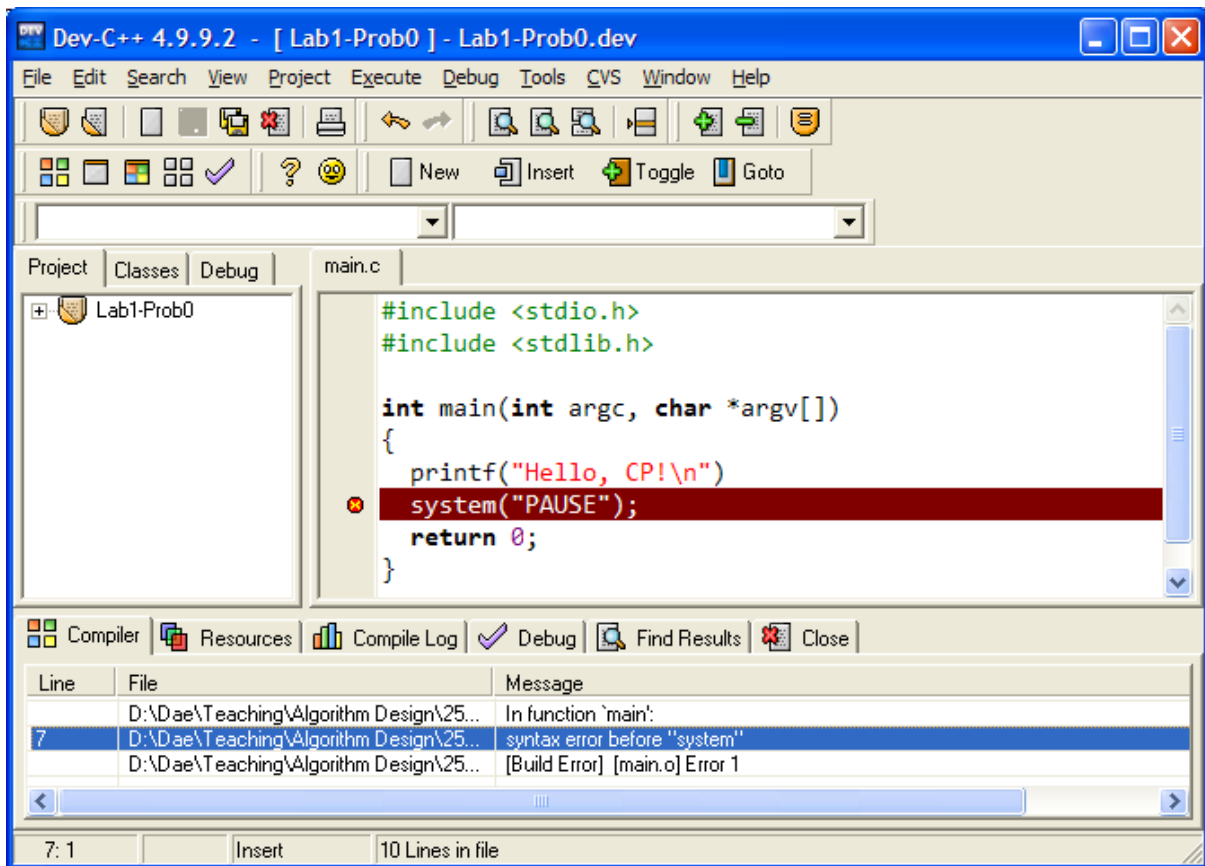


Figure 11: Error Reporting.

- Let us fix it by bring back the semi-colon and try to compile and run it again. This time, everything should be fine.

Remark

At this point, you might wondering what are the rest of the code, e.g., what is the meaning of `#include` on the first line, what is `int main(int argc, char *argv[])` and why we need `system("PAUSE")` (you can try removing it). Why we have to `return 0` on the last line. These questions are not the intended content of this lab, or even this class. You will do fine knowing nothing about it. Remember that the main content of this class is algorithms, not C language.

Anyway, it is a good time to try the TA! You should ask them about anything. Maybe you should start by asking what is the meaning of `\n` near the end of the `printf` line, for example.

Problem 1: Computation of Fibonacci number

Input: N, the index of the Fibonacci number, from Keyboard

Output: the Nth Fibonacci Number, on screen

Objective

- Must be able to take an integer input from the keyboard
- Must be able to display an integer on the screen
- Must be able to write simple for-loop structure

Introduction

Fibonacci number is a sequence of number shown as follow

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

The first two numbers are 0 and 1 and it is called the 0th Fibonacci number and the 1st Fibonacci number. The remaining number is calculated by adding the previous two numbers together. For example the 2nd Fibonacci number is $1 = 0 + 1$, the 3rd Fibonacci number is $2 = 1 + 1$, the 4th Fibonacci number is $3 = 2 + 1$.

The Fibonacci number can be written as a recurrent relation $F_i = F_{i-1} + F_{(i-2)}$.

***** Your task is to write a program that computer the Nth Fibonacci number. *****

Supplementary Explanation

Reading Input from the Keyboard

In C language, taking an input from the keyboard is usually done by `scanf` function. The function prototype of scanf is

```
int scanf (char *format, ...);
```

The first argument is the format of data to be read from the keyboard. The format is written as a "format string", a conversion used by C programming language. In the starting code, you can see that, to read an integer, the format string as "%d" is used. The second argument is the variable to store the data read from the keyboard. It has to be preceded by a character "&". For example, To read an integer from the keyboard and store it into a variable N, one must write

```
scanf("%d", &N);
```

Showing Value of Variables to the Screen

To display something to the screen, we usually use printf. printf works very similar to the scanf, we need to supply the format string indicating the type of variable to be display as the first

argument. The remaining argument will be the variable to be printed out. For example to write a value of an integer N on the screen, one must write

```
printf("%d", &N);
```

Please notice that the format string used in printf is the same as scanf. In fact, they use almost the same format. Some of the most commonly used format string are

- **“%d”** : Scan an integer as a signed decimal number.
- **“%f”** : Scan a floating-point number in normal (fixed-point) notation.
- **“%s”** : Scan a character string. The scan terminates at whitespace. A null character is stored at the end of the string, which means that the buffer supplied must be at least one character longer than the specified input length.
- **“%c”** : Scan a character (char).

Starting Code

This is the starting code for this problem. It shows you how to read the input of the user from the keyboard and display the output to the screen.

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

int main(int argc, char *argv[])
{
    //-- declare variables
    int N;           // the index of Fibonacci
    int answer;     // the output variable which will
                   // the N-th Fibonacci

    //-- get user input
    printf("Please Enter N: ");
    scanf("%d",&N);

    //-- computer F_n
    // -- begin of edit zone
    // *** fill your code here ***
    // -- end of edit zone

    // -- display output
    printf("The value of %d-th Fibonacci number is\n",N);
    printf(" %d \n",answer);

    // pause
    system("PAUSE");
    return 0;
}
```

Solution

Since this is the first lab, the solution is provided as an introductory benefit. For the later lab, you have to try solving them yourself.

The solution for this problem is to replace the line saying `// *** fill your code here` with the following code

```
int Fi_1 = 0;    // value of F_(i-1)
int Fi_2 = 0;    // value of F_(i-2)
int F_i  = 1;    // the current Fibonacci number
int i;
for (i = 2; i <= N; i++) {
    // copy the previous values back to the appropriate variable
    Fi_2 = Fi_1;
    Fi_1 = F_i;

    // computer the current Fibon number
    F_i = Fi_1 + Fi_2;
}
answer = F_i;
```

Example

This should be the result of a correct code when N is 10 and 20

Ex 1

N = 10

Answer = 6765

```
Please Enter N: 10
The value of 10-th Fibonacci number is
55
Press any key to continue . . . _
```

Ex. 2

N = 20

Answer = 6765

```
Please Enter N: 20
The value of 20-th Fibonacci number is
6765
Press any key to continue . . .
```