# Lab Tutorial for Week 1 Session 1: Simple Programs in C++

## Contents

1. Tutorial 1a: Using Dev-C++
2. Tutorial 1b: MyActivity
3. Tutorial 1c: A Simple Searching
4. Tutorial 1d: A Simple Calculator

## Tutorial 1a: Using Dev-C++

### Learning outcomes:

1. Install Dev-C++
2. Create C++ sources files
3. Understand the basic structure of C++ program
4. Compile and Run a C++ program

### Tasks:

1. Download the Dev-C++ in their website and install in your PC or laptop
2. Create a folder to contain the C++ source files (from your Windows Explorer)
3. In Dev-C++, create a new source file (File, New) or simple press CTRL-N. When you save the file, make sure it has .cpp extension.
4. Type the following codes:

#include <iostream>

using namespace std;

int main(){

cout << "Hello World..." << endl;

}

1. Save the source file as "helloworld.cpp" (File, Save) or CTRL-S
2. Compile the source file (Execute, Compile) or F9
3. Run the program (Execute, Run) or F10
4. You can also close the file without saving it (File, Close) or CTRL-W

### Summary:

In this tutorial, you have created a basic C++ program which prints "Hello world..." to the screen. You're able to create a new source file, write a simple C++ program, compile and run it. You organized the source code in a basic structure of a C++ program (include, namespace, main function) You used cout to prompt a message (text) to the screen. You can work much easier and faster by using short-cuts (CTRL-N, CTRL-S, CTRL-W, F9 and F10).

|  |
| --- |
| **Notes for Tutorial 1a:**   * Dev-C++ is a small, simple and fast C++ compiler * It comes with the useful editor and IDE * For the exercises, do not create Project.   + A project is supposed to have multiple files of C++ programs.   + These files are supposed to be linked.   + This will generate compilation errors if your multiple files is not linked   + So, for the exercise and tutorials, just create New File and put the files in the designated folder. * Make sure the name of the source file does not contain any blank space, for example "hello world.cpp" (wrong), "helloworld.cpp" (right). * Useful shortcuts in Dev-C++:   + Open a new file: CTRL-N   + Compile: F9   + Run: F10   + Compile and Run: F11   + Close a file: CTRL-W   There are tons of editors and IDE out there: Windows-based IDE such as Microsoft Visual Studio or Visual Studio Code, Mac-based IDE such as CLion or xCode and Platform-independent editor such as Sublime or Atom. It’s up to you to choose which one that works best for you. |

## Tutorial 1b: My Activity

### Learning outcomes:

1. Create and use an array
2. Use if-else statement to iterate over an array

### Tasks:

1. Create a new file (CTRL-N).
2. Solve the following programming problem.

Problem: your activity in week day and week end.

Programming task: Create a program to print “I’m studying” if today is a week day; otherwise “I go fishing” if today is a week end.

Input: The program prompts “What day is today?” then waiting for an input.

Output: print “I’m studying” or “I go fishing”

1. You can start from the below code:

#include <iostream>

using namespace std;

int main(){

// declare variables

char day;

// print message to the console

cout << "What day is it today?" << endl;

// get input data

cin >> day;

}

1. Complete the program.
2. Save the source file, compile and run it.
3. What new C++ programming keywords that you learned from this exercise?
4. What programming constructs (see the below notes) that you learned from this exercise?
5. What programming components (see the below notes) that you learned from this exercise?

### Summary:

In this tutorial, you have learned using a simple decision (if-else) statement to decide certain actions upon particular input data. You also created variables and using cin to get input from the command prompt (keyboard).

|  |
| --- |
| **Notes for Tutorial 1b:**   * Logically, a program is made up of SEQUENCE, SELECTION or DECISION, and ITERATION or LOOP statements. We denote these as programming constructs, for example,   + SEQUENCE: any single statement which ended up with semi colon   + SELECTION: if, if-else, if-else if-else, nested if, nested if-else, switch   + LOOP: for, while, do * Syntactically, a program is made up of keywords, identifers and data, for example:   + keywords: include, using, namespace, main, int, char, float, double, string, if, else, switch, for, while, do, class, private, public, protected, virtual, friend, template, etc.   + identifiers: variable names, function names, class names, etc.   + data: the literals, for example 'c', "name", 4, 37.85, etc * As a program increases in size (number of lines), it is easier and more maintainable if the program is modularized (organized into modules). We denote these as programming components, for example:   + functions   + classes |

## Tutorial 1c: A Simple Searching

### Learning outcomes:

1. Create and use an array
2. Use the string data type
3. Use if-else statement to iterate over an array

### Tasks:

1. Create a new file (CTRL-N).
2. Solve the following programming problems (Task 1 and Task 2).

Problem : Searching integers and strings.

Task 1: Create a program to search for an integer over the following data:

{3,49,1,6,33,13,9,4,6};

Input: The program prompts “What data are you searching for?” then waiting for an input.

Output: Print “found” in index \_\_\_\_ or “not found” otherwise

Task 2: Create a program to search for a string over the following data:

{"Nicky","David","Guntur","Cindy","Rizqy","James"}

Input: The program prompts “What data are you searching for?” then waiting for an input.

Output: Print “found” in index \_\_\_\_ or “not found” otherwise

1. You can start from the below code:

#include <iostream>

using namespace std;

int main(){

// declare an array

int ROW = 9;

int data[ROW] = {3,49,1,6,33,13,9,4,6};

int key;

// prompt the search key

cout << "What data are you searching for?" << endl;

cin >> key;

}

1. Complete the program.
2. Save the source file, compile and run it.
3. What new C++ programming keywords that you learned from this exercise?
4. What programming constructs (see the notes in Tutorial 1b) that you learned from this exercise?
5. What programming components (see the notes in Tutorial 1b) that you learned from this exercise?

### Summary:

In this tutorial, you have learned using a for loop to iterate an array. You created and initialized array.

## Tutorial 1d: A Simple Calculator

### Learning outcomes:

1. Create functions and function prototype
2. Use switch statement to make decision

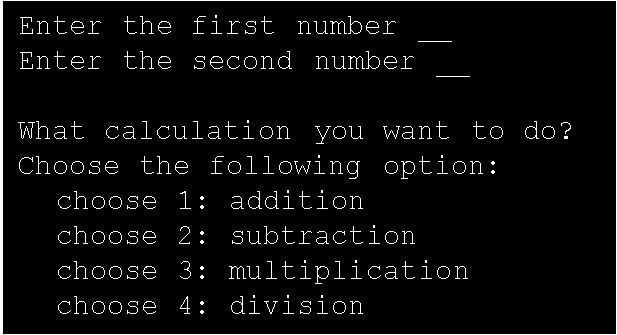
### Tasks:

1. Create a new file (CTRL-N).
2. Solve the following programming problem.

Problem: calculate two numbers.

Task: Create a program that calculate (add, subtract, multiply, divide) two numbers (integers and floating points altogether).

Input: The program prompts:



Output: the result of the calculation.

1. You can start from the below code:

#include <iostream>

using namespace std;

// declare function prototype

double add (double a, double b);

double subtract(double a, double b);

double multiply(double a, double b);

double divide(double a, double b);

// the main function

int main(){

//declare variables

double num1;

double num2;

int option;

double result;

// prompt for input

cout << "Enter the first number " << endl;

cin >> num1;

cout << "Enter the second number " << endl;

cin >> num2;

cout << "What calculation you want to do?" << endl;

cout << "Choose the following option:" << endl;

cout << "choose 1: addition" << endl;

cout << "choose 2: subtraction" << endl;

cout << "choose 3: multiplication" << endl;

cout << "choose 4: division" << endl;

cin >> option;

}

1. Complete the program.
2. Save the source file, compile and run it.
3. What new C++ programming keywords that you learned from this exercise?
4. What programming constructs (see the notes in Tutorial 1b) that you learned from this exercise?
5. What programming components (see the notes in Tutorial 1b) that you learned from this exercise?

### Summary:

In this tutorial, you have learned using a switch or if-else statement to make decision. You created functions and invoke them. There are two ways of writing a function. The first one is by specifying the function "prototype" before the main function, then implement (write up the body of) the function somewhere after the code of the main function. Otherwise, you can directly write the complete code of the functions before the main function code. Using function prototype is a better practice since in the actual delivery, you will split your program into header files (.h extension) and implementation files (.cpp extension).

---end of Tutorial Week 1---