Aug 13, 2015

Computer Programming

Fast National University of computer and emerging sciences |

Lab 1

abdul aziz

|  |  |
| --- | --- |
| **Lab Instructor** | **Mr. Abdul Aziz** |
| **Course** | **Computer Programming Lab** |
| **Duration** | **2hrs** |

**Objectives:**

* In this lab, following topics will be covered:
* Introduction to C++
* Getting started with Dev C++ IDE
* Best coding practices
* Basic constructs in C++
* Data Types in C++
* Conditional Statements
* Loops

# Introduction to C++

* C++ was developed by **Bjarne Stroustrup** of AT&T Bell Laboratories in the early 1980's.
* C++ is based on the C language. The name is a pun - "++" is a syntactic construct used in C (to increment a variable), and C++ is intended as an incremental improvement of C.
* Most of C is a subset of C++.The extra object-oriented facilities in C++ are partly included to overcome these shortcomings.

# Getting started with Dev C++ IDE

* Dev C++ is a full - featured Integrated Development Environment (IDE) for the C/C++
* As similar IDEs, it offers to the programmer a simple and unified tool to edit, compile, link, and debug programs.
* It also provides support for the management of the files of a program in “projects” containing all the elements required to produce a final executable programThe application development process encompasses the following steps:

**Create a project**

The type of application and the programming language to be used are specified.

**Write source code**

Write the program in C++ and save the source code file.

**Compile and link the code**

The source code is compiled and linked to generate a running program. Other files of the project may be created.

**Fix compilation errors, if any**

**Run the program**

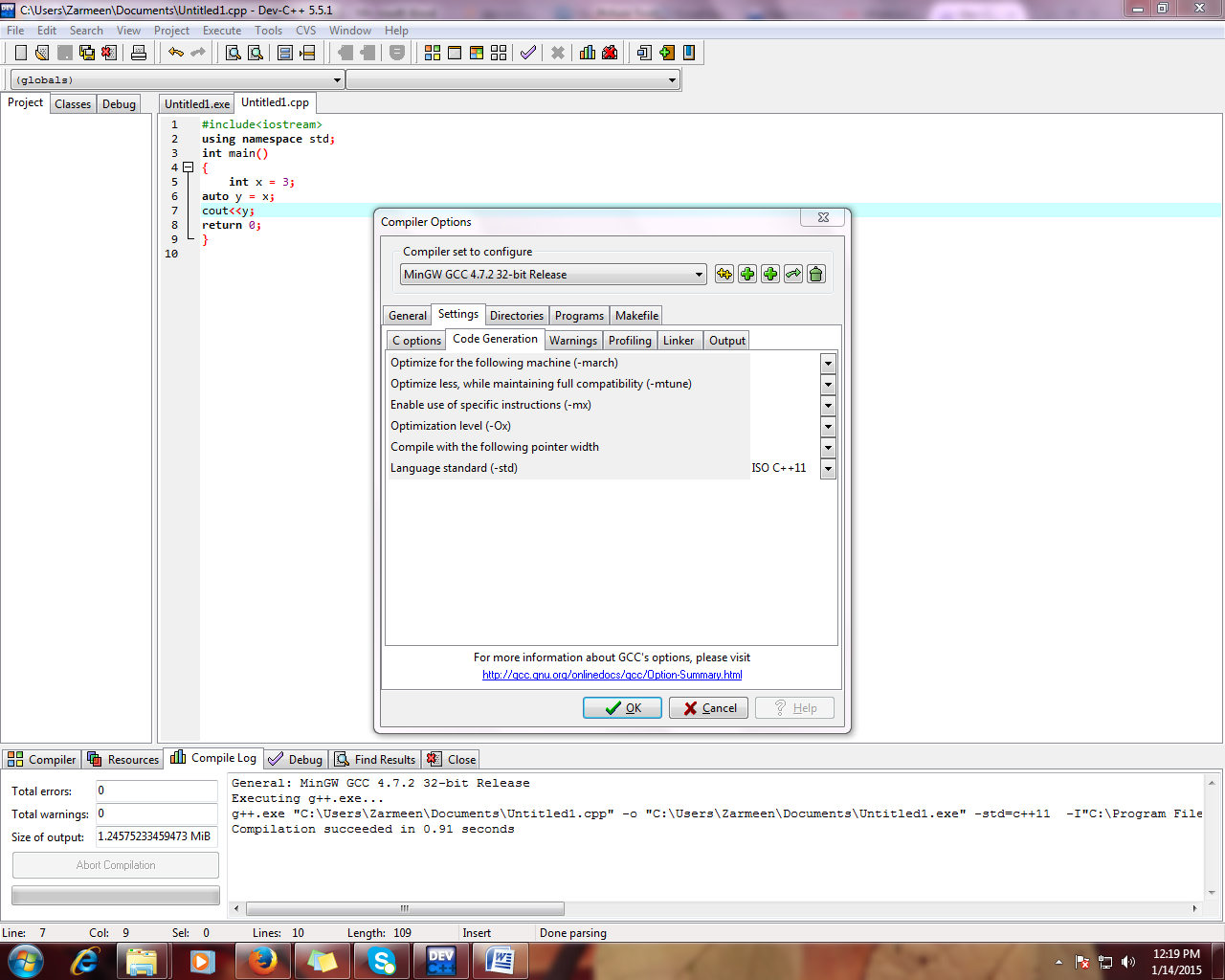
Run the program to validate the functioning.

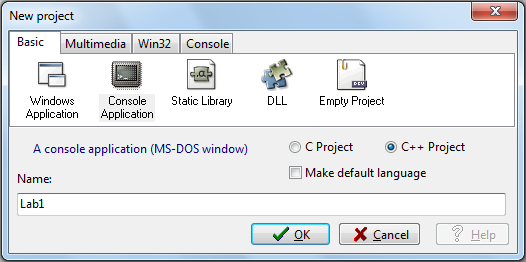
**Fix execution errors, if any**

## Project creation

**Steps:**

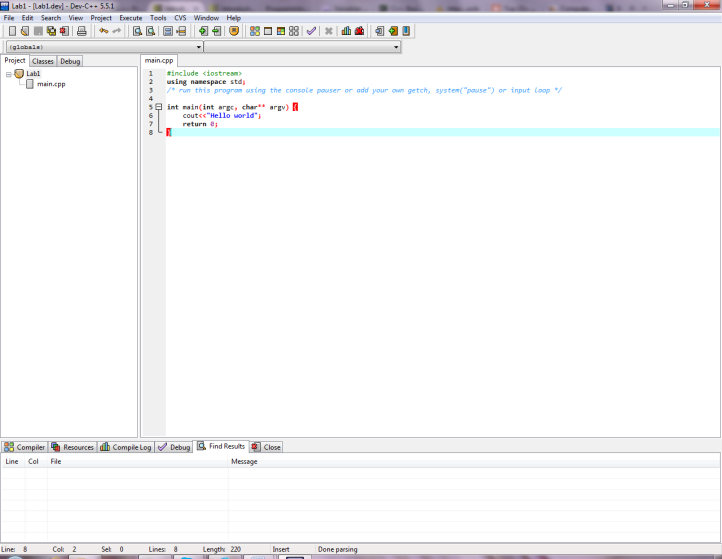
1. Start the IDE from the Program folder Dev-C++ or Bloodshed Dev-C++
2. Tools -> Compiler Options, configure compiler to support C++ 11. Select the "Settings" tab, and within it, the "Code Generation" tab. There, in "Language standard (-std)" select "ISO C++ 11":



****

**Fig 1: Creating Project**

## Writing source code

 Once the project has been created, we can start writing our C++ program.

File Explorer

Source Code Editor

**Fig 2: Writing Source code**

Result Window

## Compilation and link

To run a program, the source code must be compiled and linked. Dev- C++ performs the complete process by clicking the Compile button (or Ctrl + F9)

## Fixing Compiler Errors

Dev -C++ underlines in red the line of code where the compilation error has been detected. The Compile tab of the Results window provides a detailed description of the error. The Compile Log shows the error message issued by the compiler program.

## Running the program

As a result of the link process, an executable program is created. To run this program, we have to click the Run button (Ctrl + F10). Alternatively, we can find the executable program (.exe) in the project folder and double-Click on it.

# Coding Standards in brief:

When writing code, it is recommended to consider following practices:

1. Know what the code block must perform. Properly indent code blocks.
2. Indicate a brief description of what a variable is for (reference to commenting)
3. Correct errors as they occur.
4. Keep your code simple
5. Maintain naming conventions which are uniform throughout.

# An Example C++ Program

/\* Comments can also be written starting with a slash

followed by a star, and ending with a star followed by

a slash. As you can see, comments written in this way

can span more than one line. \*/

/\* Programs should ALWAYS include plenty of comments! \*/

/\* This program prompts the user for the two numbers from user and print the sum, product, difference and the quotient of the two numbers \*/

#include <iostream>

using namespace std;

int main()

{

**int num1, num2; // declare variables**

**cout << "Enter two integers: "; // prompt user**

**cin >> num1 >> num2; // read values from keyboard**

**// output the results**

**cout << "The sum is " << num1 + num2**

**<< "\nThe product is " << num1 \* num2**

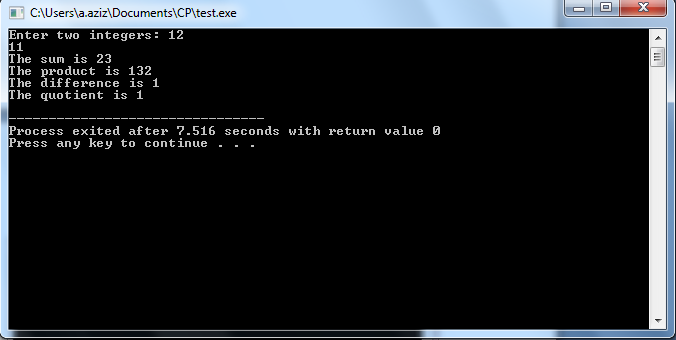
**<< "\nThe difference is " << num1 - num2**

**<< "\nThe quotient is " << num1 / num2 << endl;**

**return 0; // indicate successful termination**

}

**Output:**



**Fig 4: Output of Sample Program**

**Program Notes:**

**using namespace std;**

The names **cout** and **endl** belong to the **std namespace**. They can be referenced via fully qualified name**std::cout and std::endl**, or simply as **cout and endl with a "using namespace std;**" statement.

**return 0;**  
 The return value of 0 indicates normal termination; while non-zero (typically 1) indicates abnormal termination. C++ compiler will automatically insert a "return 0;" at the end of the themain () function, thus, it statement can be omitted.



**Common Escape Sequences:**

|  |  |  |
| --- | --- | --- |
| **Escape Sequence** | **Name** | **Description** |
| \n | Newline | Causes the cursor to go to the next line for subsequent printing |
| \t | Horizontal tab | Causes the cursor to skip over to the next tab stop |
| \b | Backspace | Causes the cursor to back up, or move left one position |
| \r | Return | Causes the cursor to go to the beginning of the current line, not the next line. |
| \\ | Backslash | Causes a backslash to be printed |
| \' | Single quote | Causes a single quotation mark to be printed |
| \" | Double quote | Causes a double quotation mark to be printed |

# **Basic Constructs in C++**

## The #include Directive

* The #include directive causes the contents of another file to be inserted into the program
* Preprocessor directives are not C++ statements and do not require semicolons at the end

## Variables and Constants

* Variables represent storage locations in the computer’s memory. Constants are data items whose values do not change while the program is running.
* Every variable must have a declaration.

#include <iostream.h>

void main(void)

{

int value;

value = 5;

cout<< “The value is “ << value <<endl;

}

**Program Output:**

The value is 5

## Constants

* A variable is called a “variable” because its value may be changed. A constant, on the other hand, is a data item whose value does not change during the program’s execution.

# DATATYPES:

There are many different types of data. Variables are classified according to their data type, which determines the kind of information that may be stored in them. Integer variables only hold whole numbers

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Size** | **Range** |
| short | 2 Bytes | -32,768 to +32.767 |
| unsigned short | 2 Bytes | 0 to +65,535 |
| Int | 4 Bytes | -2,147,4833,648 to +2,147,4833,647 |
| unsigned int | 4 Bytes | 0 to 4,294,967,295 |
| Long | 4 Bytes | -2,147,4833,648 to +2,147,4833,647 |
| unsigned long | 4 Bytes | 0 to 4,294,967,295 |

## Char Data Type

* Usually 1 byte long
* Internally stored as an integer
* ASCII character set shows integer representation for each character
* ‘A’ == 65, ‘B’ == 66, ‘C’ == 67, etc.
* Single quotes denote a character, double quotes denote a string

## The bool Data Type

* Boolean variables are set to either true or false

# OPERATORS:

There are many operators in C++ for manipulating data which include arithmetic Operators, Relational Operators, Logical operators and many more which will be discussed accordingly. Some of the fundamental arithmetic and relational operators are:

|  |  |
| --- | --- |
| **operator** | **description** |
| + | addition |
| - | subtraction |
| \* | multiplication |
| / | division |
| % | modulo |
| == | Equal to |
| != | Not equal to |
| < | Less than |
| <= | Less than or equal to |
| && | Logical AND Operator |
| || | Logical OR Operator |
| ! | NOT Operator |

## Increment and Decrement Operators:

C++ introduces increment and decrement operators which are ++ and – respectively. These operators increment/decrement 1 in the operand’s value. For example: x++ will be equivalent to x=x+1 or x+=1.

The special characteristic of these operators is that they can be used for pre-increment as well as post-increment. To understand, consider the following statements.

A=b++; //The statement will assign the contents of b to A and then increments the value of b by 1.

A=++b; //The statement will first increment the value of b by 1 and then assign the new value to a.

## Assignment statements:

**Value = 5;**//This line is an assignment statement.

* The assignment statement evaluates the expression on the right of the equal sign then stores it into the variable named on the left of the equal sign
* The data type of the variable was in integer, so the data type of the expression on the right should evaluate to an integer as well.

# Conditional Statements:

## IF Statement:

if (expression)

statement;

* Statement will be executed only if expression is true

//Sample Program

int x;

cin >> x;

if (x == 5)

cout << “x is 5 \n”;

//Output

x is 5

Often, we want to execute several statements if a condition is true. Use braces to indicate the block of statements to be executed.

if (expression)

{

statement1;

statement2;

}

## IF-Else Statement:

if (expression)

statement1;

else

statement2;

* Statement1 will be executed if expression is true
* Statement 2 will be executed if expression is false.
* Both statements will never be executed.

//Sample Program

int x;

cin >> x;

if (x == 5)

{

cout << “ x is equal to 5 \n”;

}

else

{

cout << “ x is not equal to 5 \n”;

}

//Output when x is 5:

x is 5

//Output when x is not 5

x is not 5

## Nested IF Statements:

* Possible to put one if or if-else statement inside another if or if-else statement

if (expression1)

{

statement1;

if (expression2)

{

statement2;

}

}

//Sample Program

int x;

cin >> x;

if (x < 6)

{

cout << “ Less than 6 \n”;

if (x == 5)

{

cout << “ x is 5 \n”;

}

}

//Output when x is 5

Less than 6

x is 5

//Output when x is less than 6 but not 5

Less than 6

## Else If Statements:

if (expression1)

statement 1;

else if (expression 2)

statement 2;

...

else

statement n;

//Sample Program

int x;

cin>>x;

if(x<60)

{

cout<<”Less than 60”;

}

else if(x>60)

{

cout<<”Greater than 60”

}

//Output when x is less than 60

Less than 60

//Output when xis greater than 60

Greater than 60

* Multiple conditions can be written by making several else-is clauses.
* Once a condition is true, control will never go to other else-if conditions.
* You can also add an else clause after else if statements.

//Sample Program

int x;

cin>>x;

if(x<60)

{

cout<<”Less than 60”;

}

else if(x>60)

{

cout<<”Greater than 60”

}

else

{

cout<<”Neither”

}

//Output when x is less than 60

Less than 60

//Output when x is greater than 60

Greater than 60

//Output when x is 60

Neither

# Iteration statements (loops):

Loops repeat a statement a certain number of times, or while a condition is fulfilled. They are introduced by the keywords while, do, and for.

1. **The while loop**

The simplest kind of loop is the while-loop. Its syntax is:  
 ***while (expression) statement***

The while-loop simply repeats statement while expression is true. If, after any execution of statement, expression is no longer true, the loop ends, and the program continues right after the loop. For example, let's have a look at a countdown using a while-loop:

#include <iostream>

using namespace std;

int main ()

{

int n = 10;

while (n>0) {

cout << n << ", ";

--n;

}

cout << "liftoff!\n";

}

#### The do-while loop

A very similar loop is the do-while loop, whose syntax is:

***do statement while (condition);***  
It behaves like a while-loop, except that condition is evaluated after the execution of statement instead of before, guaranteeing at least one execution of statement, even if condition is never fulfilled.

#include <iostream>

#include <string>

using namespace std;

int main ()

{

string str;

do {

cout << "Enter text: ";

getline (cin,str);

cout << "You entered: " << str << '\n';

} while (str != "goodbye");

}

#### The for loop

The for loop is designed to iterate a number of times. Its syntax is:  
  
***for (initialization; condition; increase) statement;***

#include <iostream>

using namespace std;

int main ()

{

for (int n=10; n>0; n--) {

cout << n << ", ";

}

cout << "liftoff!\n";

}

**Exercise**

1. Write a program that asks the user to enter two integers, obtains the numbers from the user, then prints the larger number followed by the words "**is larger**." If the numbers are equal, print the message “**These numbers are equal**.”
2. Write a program that inputs three integers from the keyboard and prints the sum, average, product, smallest and largest of these numbers.
3. Write a program to calculate the electricity bill. The rate of electricity per unit are as follow:

1. If the consumed units are equal or less than 300,then the cost is Rs. 3/-per unit.

2. If the consumed units are more than 300,then the cost is Rs. 3.5/-per unit.

1. Write a program to print the multiplication table of a number entered from the keyboard.