CC-301 Programming Fundamentals

Lecture 2

Engr. Waseem Ullah Khan

Programming Language Rules

Rules of **Syntax** which specify how valid instructions are written in the language

- Deal with the structure of an instruction

Rules of **Semantics** which determine the meaning of the instructions (what the computer will do in response to the given instructions)

- Deal with the content of an instruction

Programming Errors

Programming errors are called **bugs**

The process of tracking bugs and correcting them is called **debugging**

Three kinds of errors can occur in a program:

Syntax Errors

- Most languages including C++ can only execute a program if the program is syntactically correct; otherwise, the process fails and returns an error message

```
#include <iostream>
using namespace std

int main()
{
  cout << "Programming is fun << endl;

  return 0;
}
```

Programming Errors

Sematic Errors

- Semantic error is an error in the content of a code
- If there is a semantic error in your program, it will run successfully, i.e., the computer will not generate any error messages, but it will not do the right thing
- The problem is that the program you wrote is not the program you wanted to write. The meaning of the program (its semantics) is wrong

Runtime Errors

- The third type of error is a runtime error, so called because the error does not appear until you run the program
- These errors are also called exceptions because they usually indicate that something exceptional (and bad) has happened

Compiling Source Code

Source Program

- A program written in a human readable version, which you will write

Compiler

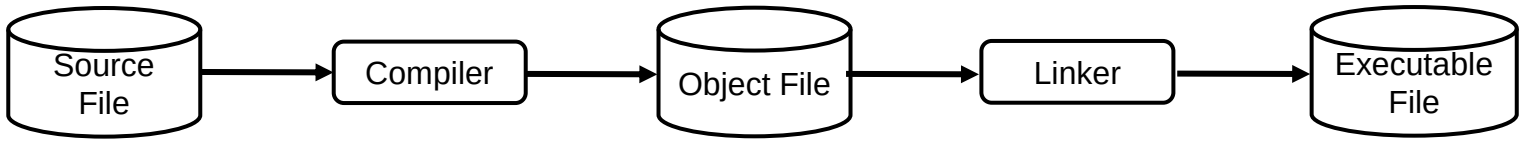
- Software use to translate source program into machine language program

Object Code

- The machine language version of a source program in 0s and 1s
- Files usually have extension '.obj'

EXE Code

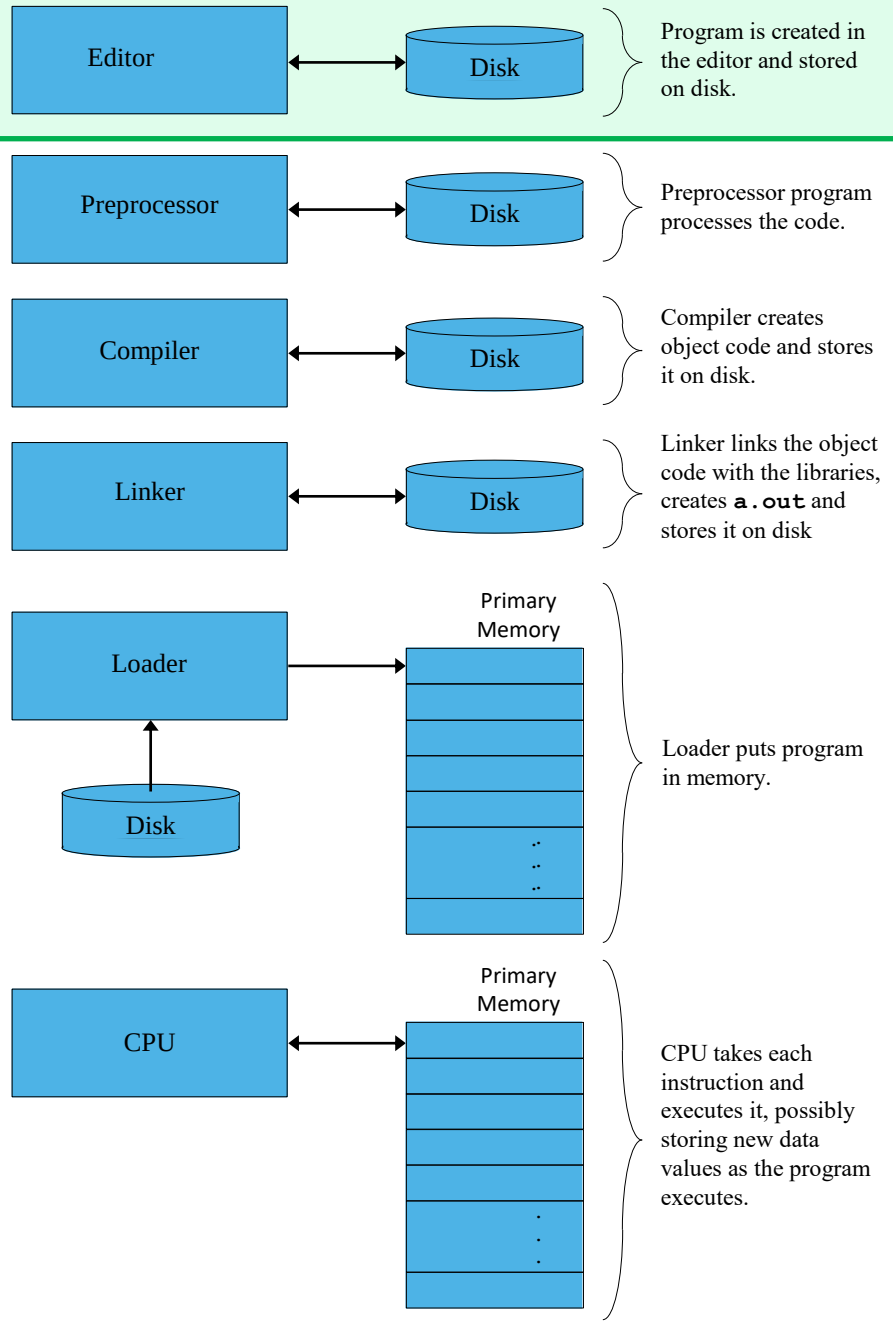
- It is an executable program
- Files usually have extension '.exe'



Basics of a C++ Environment

Phases of C++ Programs:

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute



Structure of C++ Program

C++ program consists of three main parts:

- Preprocessor Directives
  - Start with “#”
- The main( ) function
- C++ statements

C++ Program:

```
// A first Program in C++
/* Programming Fundamental Course*/

#include <iostream>
using namespace std;

int main()
{
    cout<<"Welcome to Programming Fundamental Course!";
    return 0;
}
```

Basic Structure of a C++ Program

C++ Program:

```
// A first Program in C++
/* Programming Fundamental Course*/

#include <iostream>
using namespace std;

int main()
{
    cout<<"Welcome to Progra
    return 0;
}
```

Comments

Written between ***/**** and ****/*** or following a ***//***

Improve program readability and do not cause the computer to perform any action

Preprocessor directive

Lines beginning with # are preprocessor directives

#include <iostream> tells the preprocessor to include the contents of the file **<iostream>**, which includes input/output library

Read: *"Hash-include"*

Basic Structure of a C++ Program (cont.)

C++ Program:

```
// A first Program
/* Programming Func

#include <iostream>
using namespace st

int main()
{
    cout<<"Welcome to Programming Fundamental Course!";
    return 0;
}
```

C++ programs contain one or more functions, exactly one of which must be main

Parenthesis used to indicate a function

int means that main "returns" an integer value

Left brace { begins the body of every function and a right brace } ends it

Basic Structure of a C++ Program (cont.)

C++ Program:

```
// A first Program in C++
/* Programming Fundamental Course*/

#include <iostream>
using namespace std;

int main()
{
    cout<<"Welcome to Programming Fundamental Course!";
    return 0;
}
```

Prints the string of characters contained between the quotation marks

The entire line, including *cout*, the << operator, the string *"Welcome to Programming Fundamental Course!"* and the semicolon (;), is called a statement

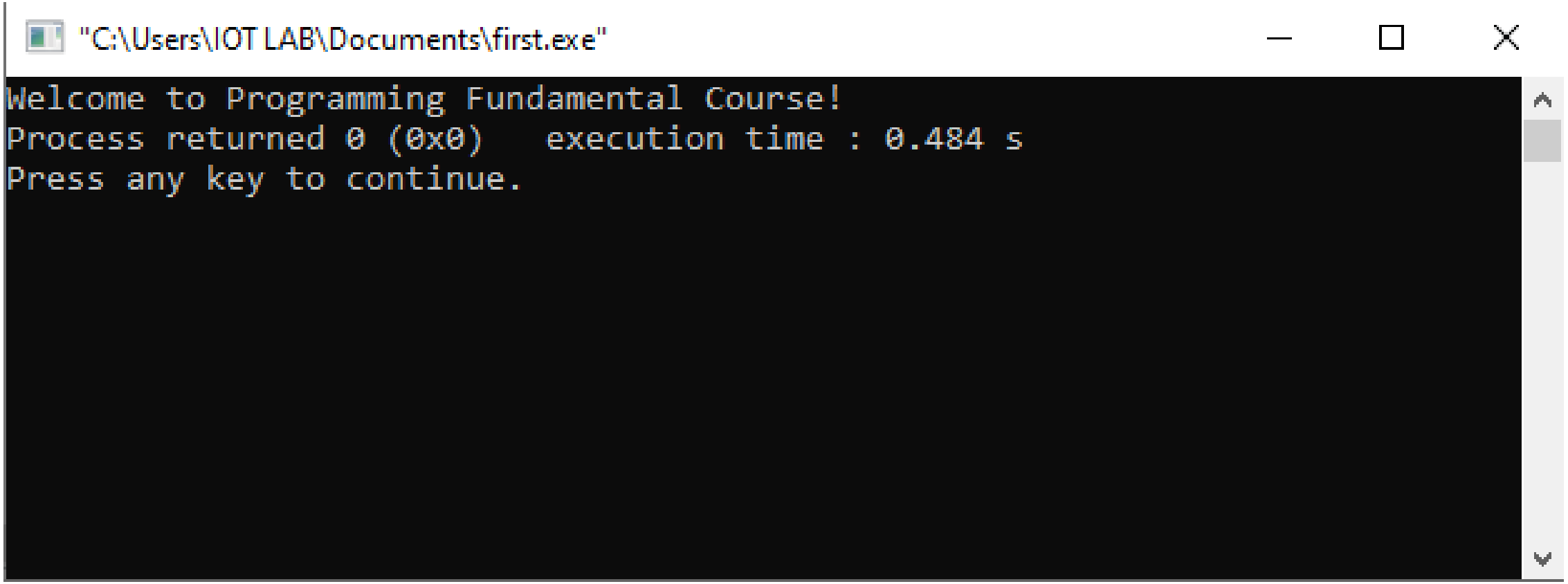
All statements must end with a semicolon

return is one a way to exit a function

return 0, in this case, means that the program terminated normally

Basic Structure of a C++ Program (cont.)

Output:



Input and Output

C++ treats input and output as a stream of characters

Stream : sequence of characters (printable or nonprintable)

The functions to allow standard I/O are in iostream header file or iostream

Thus, we start every program with

- #include <iostream>
- using namespace std;

Include Directives and Namespaces

include: directive copies that file into your program

namespace: a collection of names and their definitions. Allows different namespaces to use the same names without confusion

Insertion Operator (<<)

Variable **cout** is predefined to denote an output stream that goes to the standard output device (display screen)

The insertion operator **<<** called “put to” takes 2 operands

- The left operand is a stream expression, such as cout
- The right operand is an expression of simple type or a string constant

Extraction Operator (>>)

Variable **cin** is predefined to denote an input stream from the standard input device (the keyboard)

The extraction operator **>>** called “get from” takes 2 operands

- The left operand is a stream expression, such as cin
- The right operand is a variable of simple type

Output Statements

Syntax of cout is:

```
cout << variable ;
```

These examples yield the same output:

```
cout << x ;
cout << y ;
```

```
cout << x << y ;
```

cout01.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    int x, y, sum;
    x = 10;
    y = 20;
    sum = x + y;

    cout << "The first number is " << x << endl;
    cout << "The second number is " << y << endl;
    cout << "The Sum is: " << sum;

    return 0;
}
```

cout02.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    int x, y, sum;
    x = 10;
    y = 20;
    sum = x + y;

    cout << "The first number is " << x << endl << "The second number is " << y << endl;
    cout << "The Sum is: " << sum;

    return 0;
}
```

Input Statements

Syntax of cin is:

```
cin >> variable ;
```

cin statements can be linked together using >> operator

These examples yield the same output:

```
cin >> x ;
cin >> y ;
```

```
cin >> x >> y ;
```

cin01.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    int x, y, sum;
    cout << "Enter first number: ";
    cin >> x;
    cout << "Enter second number: ";
    cin >> y;
    sum = x + y;
    cout << "The Sum is: " << sum;
    return 0;
}
```

cin02.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    int x, y, sum;
    cout << "Enter the number 1 and number 2: ";
    cin >> x >> y;
    sum = x + y;
    cout << "The Sum is: " << sum;
    return 0;
}
```

How Extraction Operator works?

Input is not entered until user presses <ENTER> key

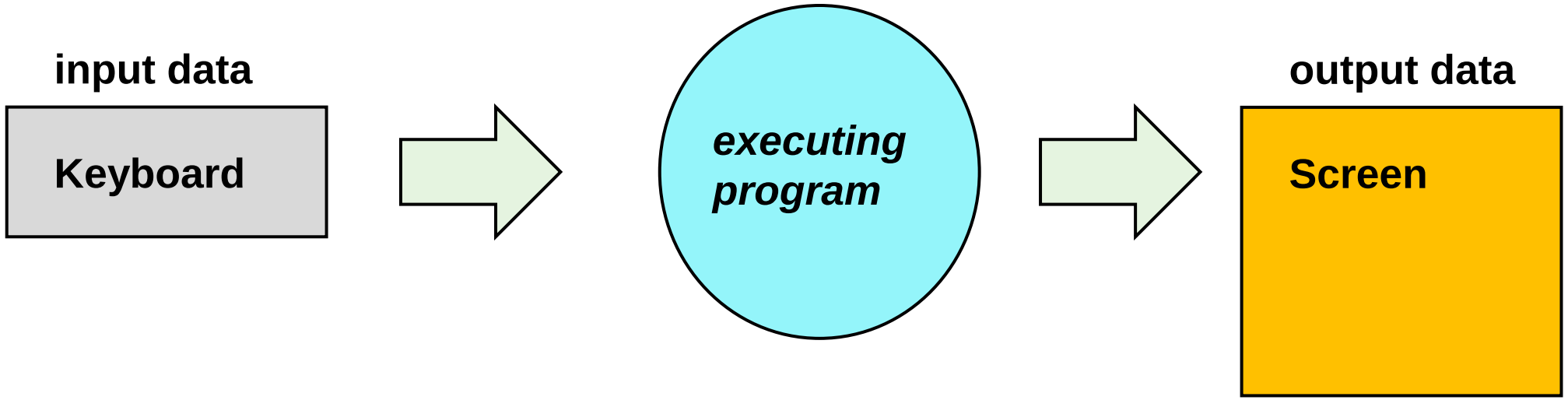
Allows backspacing to correct

Multiple inputs are stored in the order entered:

- cin>> num1 >> num2;
- User inputs: 3 4
- Assigns num1 = 3 and num2 = 4

Keyboard and Screen I/O

```
#include <iostream>
```



cin

(of type istream)

cout

(of type ostream)

Escape Sequences

Escape Sequence	Character
\a	Bell (beep)
\b	Backspace
\n	New line
\r	Return
\t	Tab
\\	Backslash
\’	Single quotation mark
\”	Double quotation mark

```
#include <iostream>
using namespace std;

int main()
{
    cout <<"\a"; //alarm or bell
    cout <<"Welcome\b"; //backspace
    cout <<"Welcome \n to"<<endl<<" Pakistan"; //New line
    cout <<"\\Welcome\\"; /* output: \Welcome\  */
    cout <<"\'Programming\'"; /* output: 'Programming' */
    cout <<"\"Programming\""; /* output: "Programming" */
    return 0;
}
```

Escape Sequence in C++ with Examples

Write a program that displays the following output:

‘D’ ‘I’ ‘g’ ‘e’ ‘s’ ‘t’

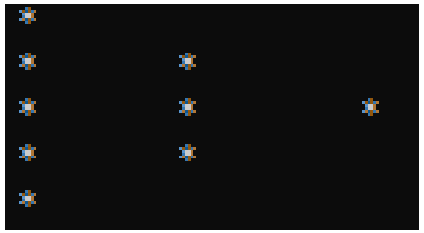
```
#include <iostream>
using namespace std;

int main()
{
    cout<<"\'D\'\t\'i\'\t\'g\'\t\'e\'\t\'s\'\t\'t\'";
    return 0;
}
```

Write a program that displays arrowhead pattern using a single output statement

```
#include <iostream>
using namespace std;

int main()
{
    cout<<" *\n *\t*\n *\t*\t*\n *\t*\n *";
    return 0;
}
```



https://sourceforge.net/projects/orwelldevcpp/

C and C++

Traditionally C programs use the file extension **.C** and C++ programs the extension **.CPP**

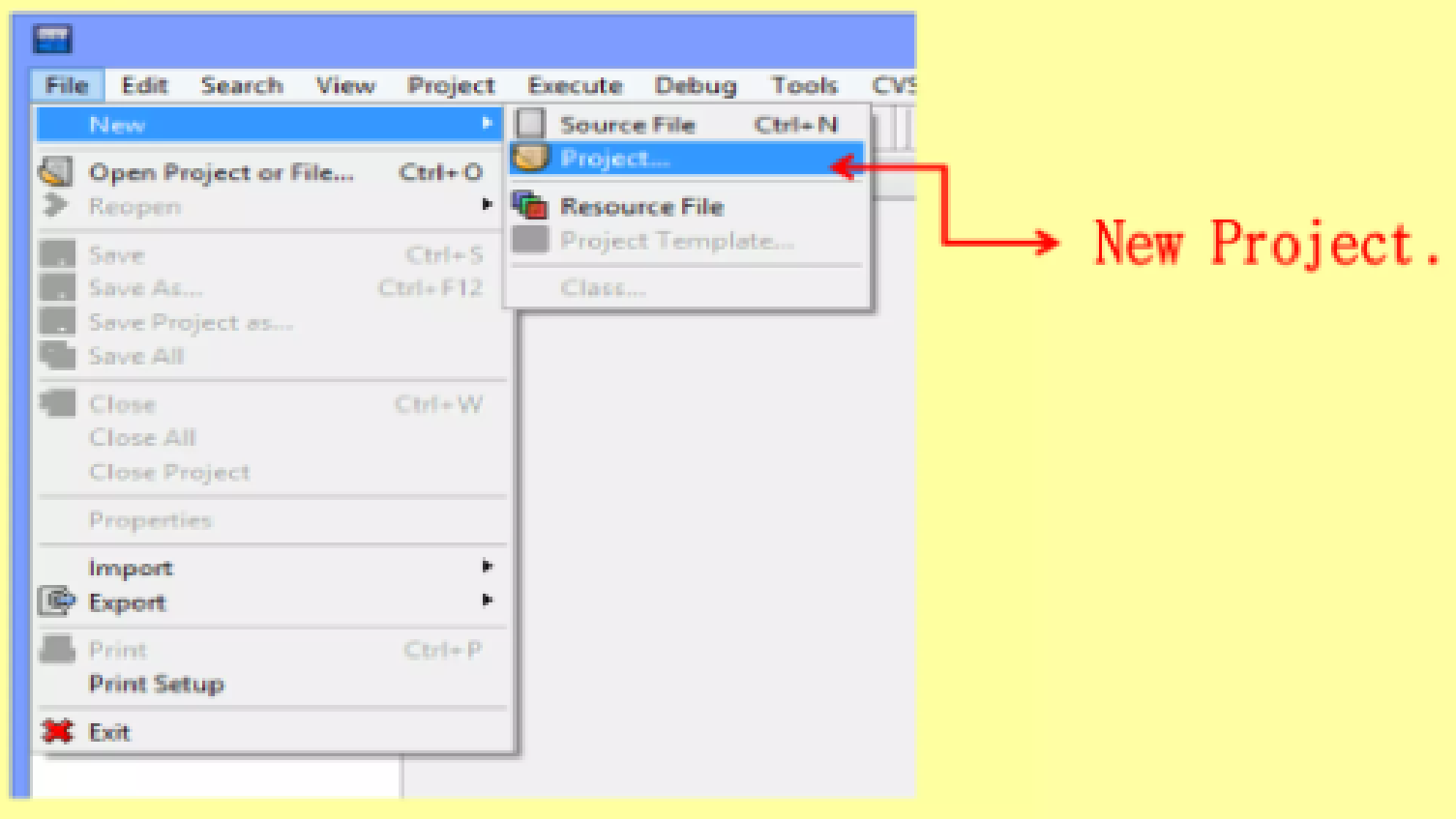
In the class we will be using **Dev-C++** software (Dev-Cpp 5.11 TDM-GCC 4.9.2 Setup.exe)

https://sourceforge.net/projects/orwelldevcpp/

Familiarization with IDE of Dev C++

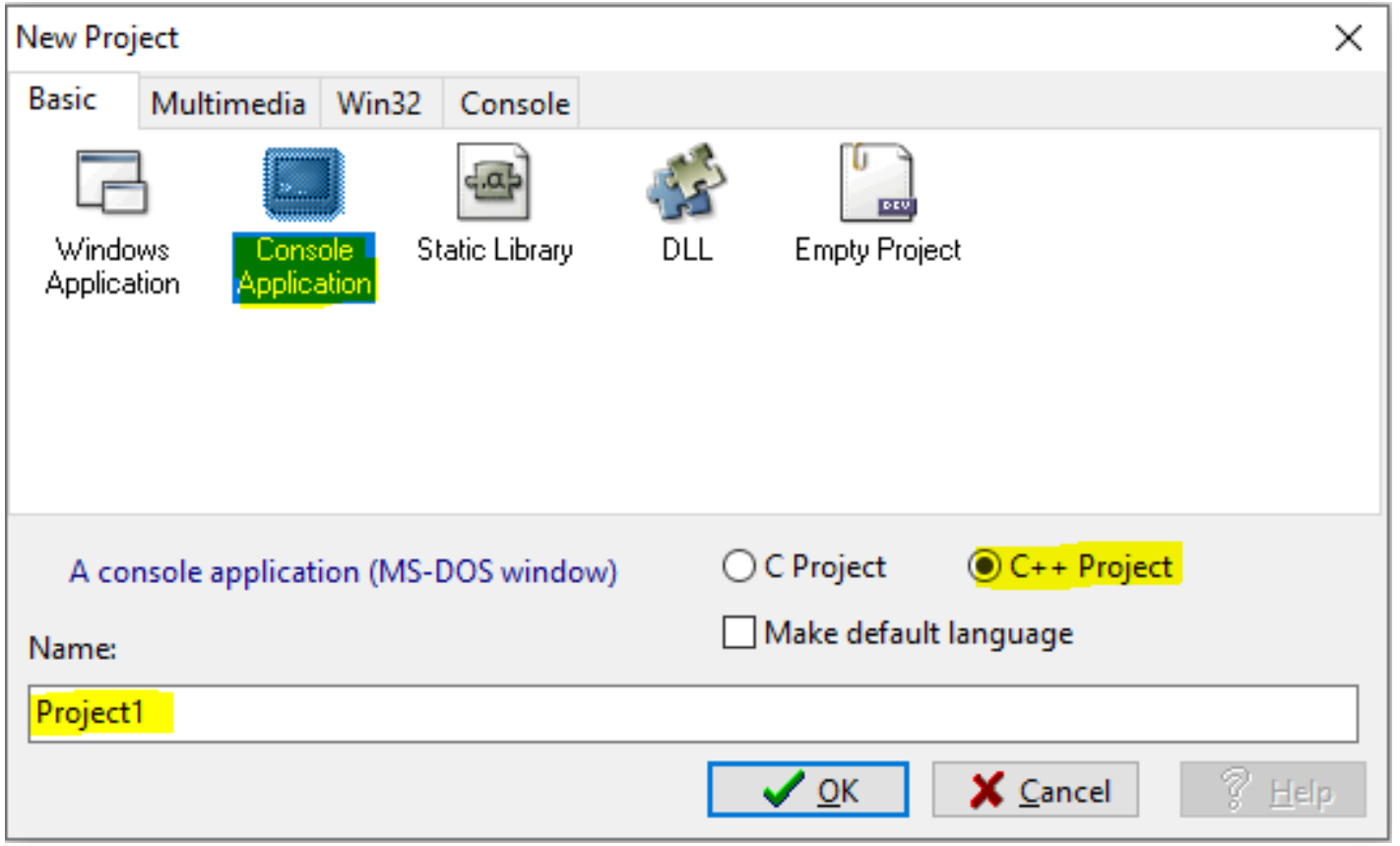
1. Launch the Dev-C++ software installed in your computer

2. Click on File → New → Project… in the menu bar



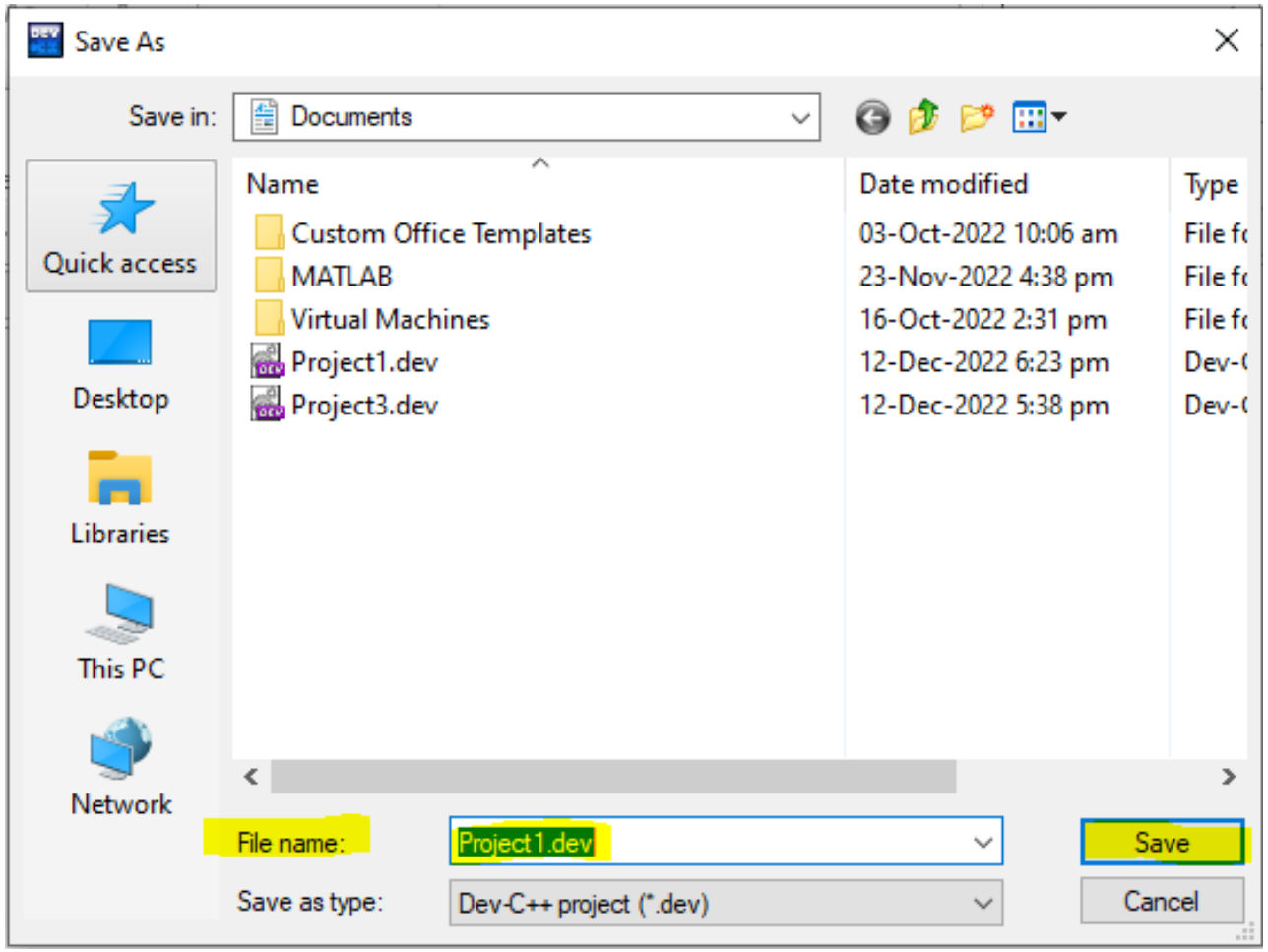
Familiarization with IDE of Dev C++

3. Now a new window will open, just select Console Application and choose your desired language C++ and Click Ok



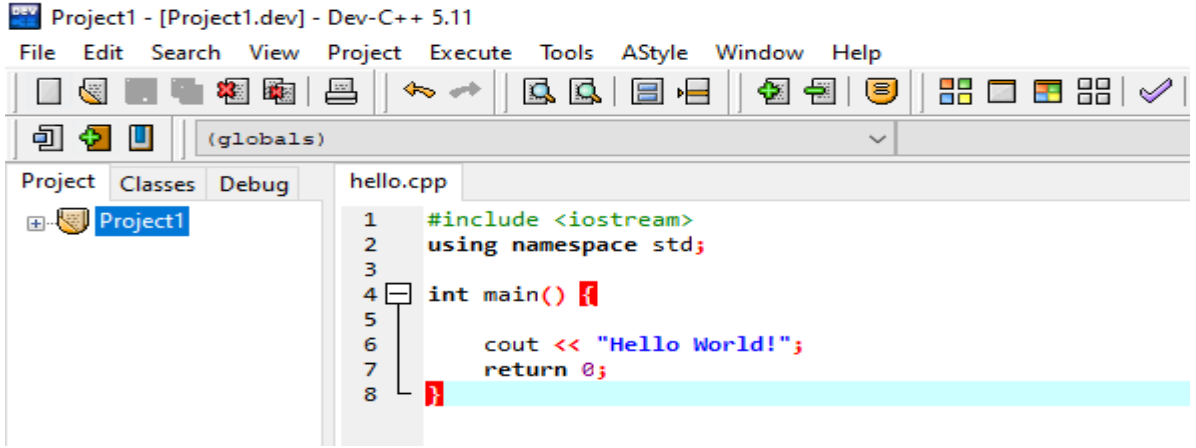
Familiarization with IDE of Dev C++

4. Save your Project.dev file into the desired directory. You may rename it if you wish

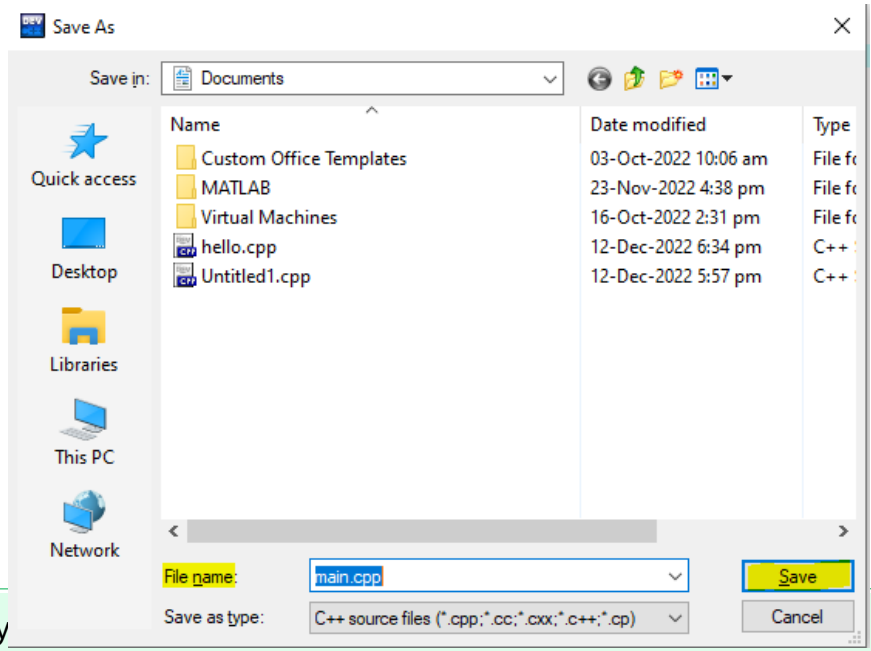


Familiarization with IDE of Dev C++

5. Now write your code in the editor console

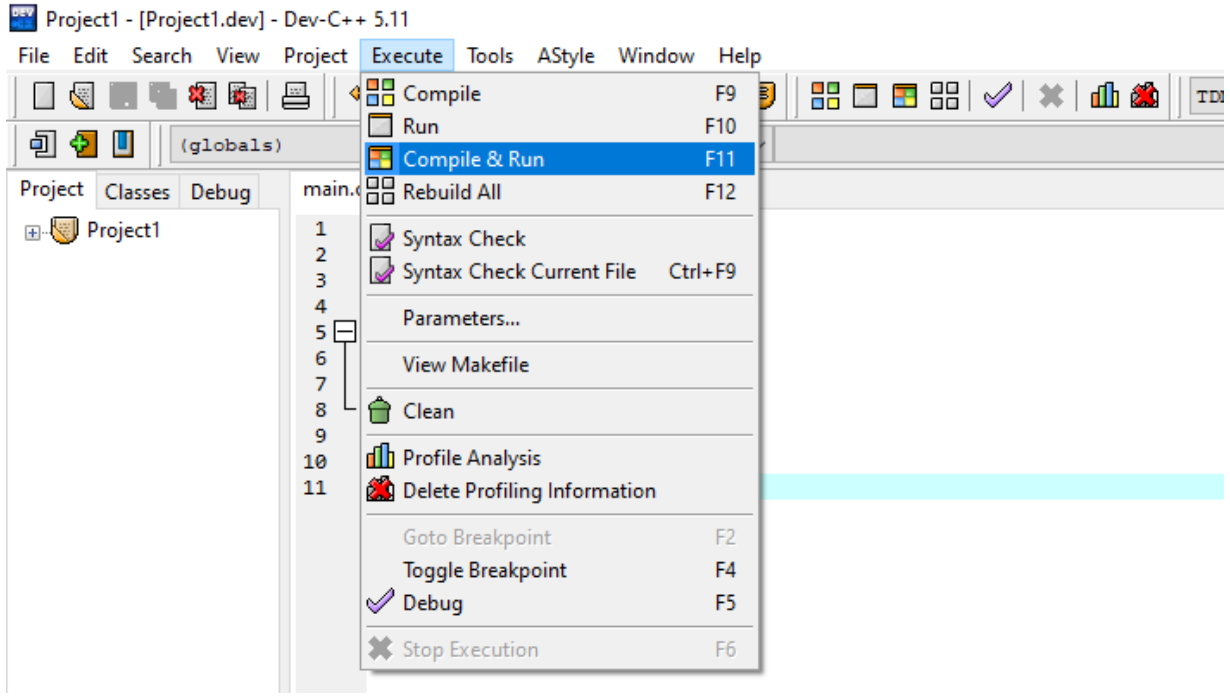


6. Save your code into the same directory

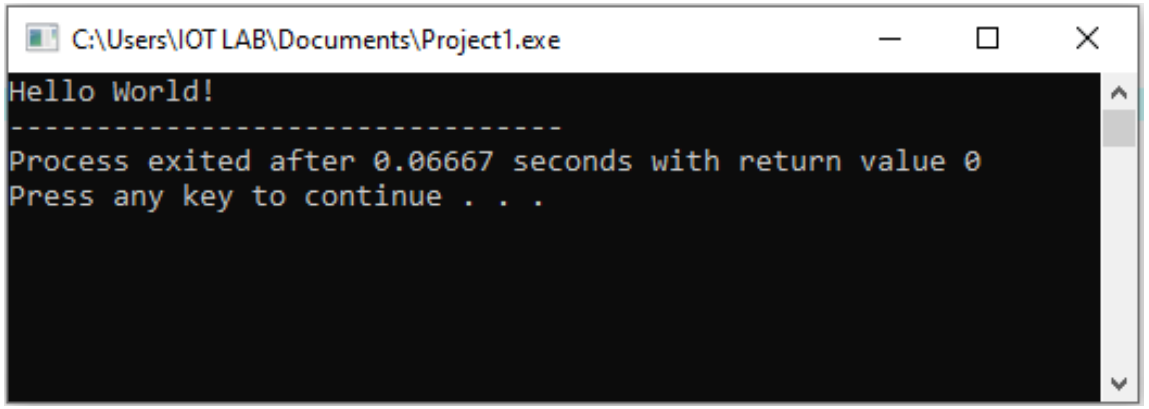


Familiarization with IDE of Dev C++

7. Now compile and run your code or just press F11 key to compile and run the program

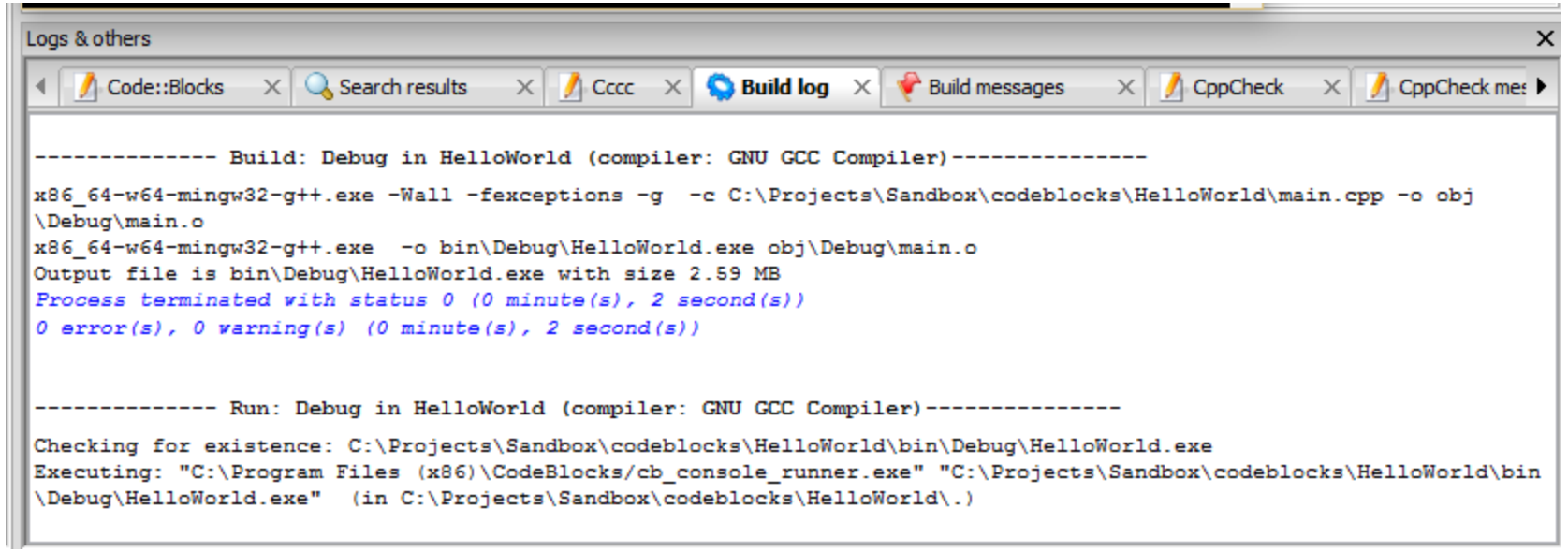


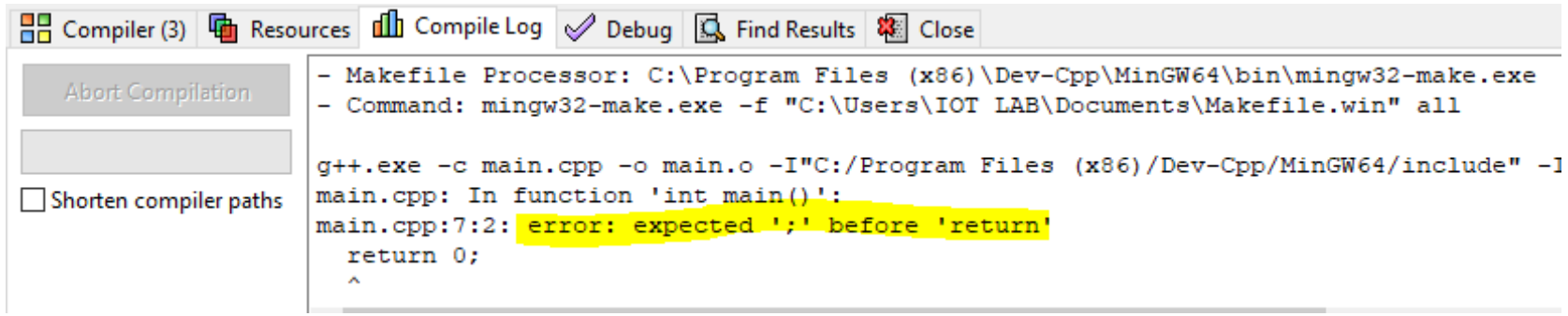
8. Console Window:



Familiarization with IDE of Dev C++

9. Build and compile messages





C++ IDE and Source Editors