Tishk International University
Cybersecurity Department
Course Code: CBS 113

Programming Fundamentals

Lecture 1

Introduction to the course & C++

Fall 2025

Hemin Ibrahim, PhD
hemin.ibrahim@tiu.edu.iq

Outline

- Introduction to the course
- What is programming?
- Programming in Cybersecurity
- Introduction to C++
- Special Characters

Objectives

- Provide an introduction to the concept of programming.
- Emphasize the importance and relevance of learning programming.
- Explore the various types of programming languages.
- Highlight the advantages of the C++ programming language.
- Explore into the basics of C++, covering comments and special characters.
- Foster a foundational understanding of programming concepts, with a specific focus on the practical applications of C++.

Concepts

- Print
- Variables
- Input & output
- Control statements (IF, switch)
- Loops

Ideas

- Sum
- Average
- Odd and Even numbers
- Random Numbers
- Prime numbers
- Positive and negative numbers

Course policy - Attendance

Don’t be Late

Be on time for the lecture

Remember, if class starts at 11:00AM, 11:06, means you are absent

What is programming?

- Raise your hand if you need to step out of the class.
- Ensure that you arrive on time for the second part of the lecture

Course policy - Attendance

- I have a big problem with attendance
  - Ahmmm
- I will fail if I don’t have 70%
  - Ahmmm
- Can you please help me
  - **NO**

There will be no attendance record if you are unavailable

What is programming?

Programming is the process of giving instructions to a computer to perform specific tasks.

- Instructions must be written in a way the computer can understand.
- We use programming languages to write these instructions.
- **In Cybersecurity**: Programming skills help identify, prevent, and ease security vulnerabilities in software.

What is programming?

- Cybersecurity professionals often need to
  - analyze code for vulnerabilities,
  - write secure scripts, or
  - develop custom tools

Why Learn Programming?

- **Core Skill for Cybersecurity**
  - Understand how software and systems work
- **Automating Tasks**
  - Write scripts to automate repetitive tasks like scanning and monitoring.
- **Building Security Tools**
  - Develop custom tools for penetration testing and threat analysis.
- **Analyzing Malware and Exploits**
  - Learn how attacks are executed to strengthen defenses.

Programmers in cybersecurity?

Programmers who understand secure coding principles can proactively include proper:

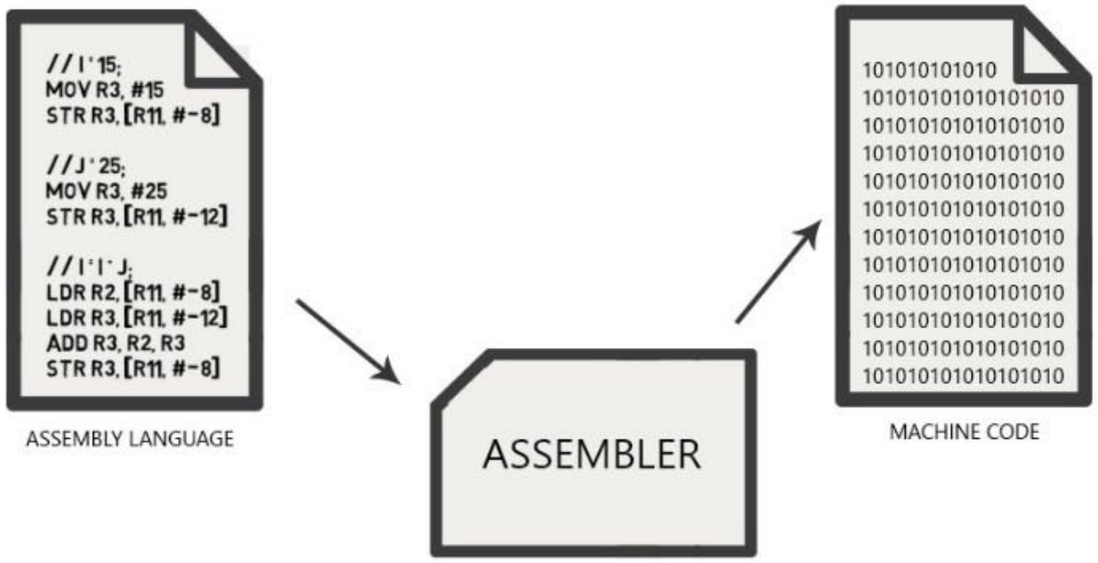
- validation checks,
- error handling, and
- memory management in their software,
- making it harder for attackers to exploit the system.

Types of Programming Languages

Programming languages come in various types, each serving specific purposes and catering to different needs.

1)Low-level Programming Languages

- Machine Language: Binary code (0s and 1s)
- Assembly Language: Human-readable code representing machine instructions

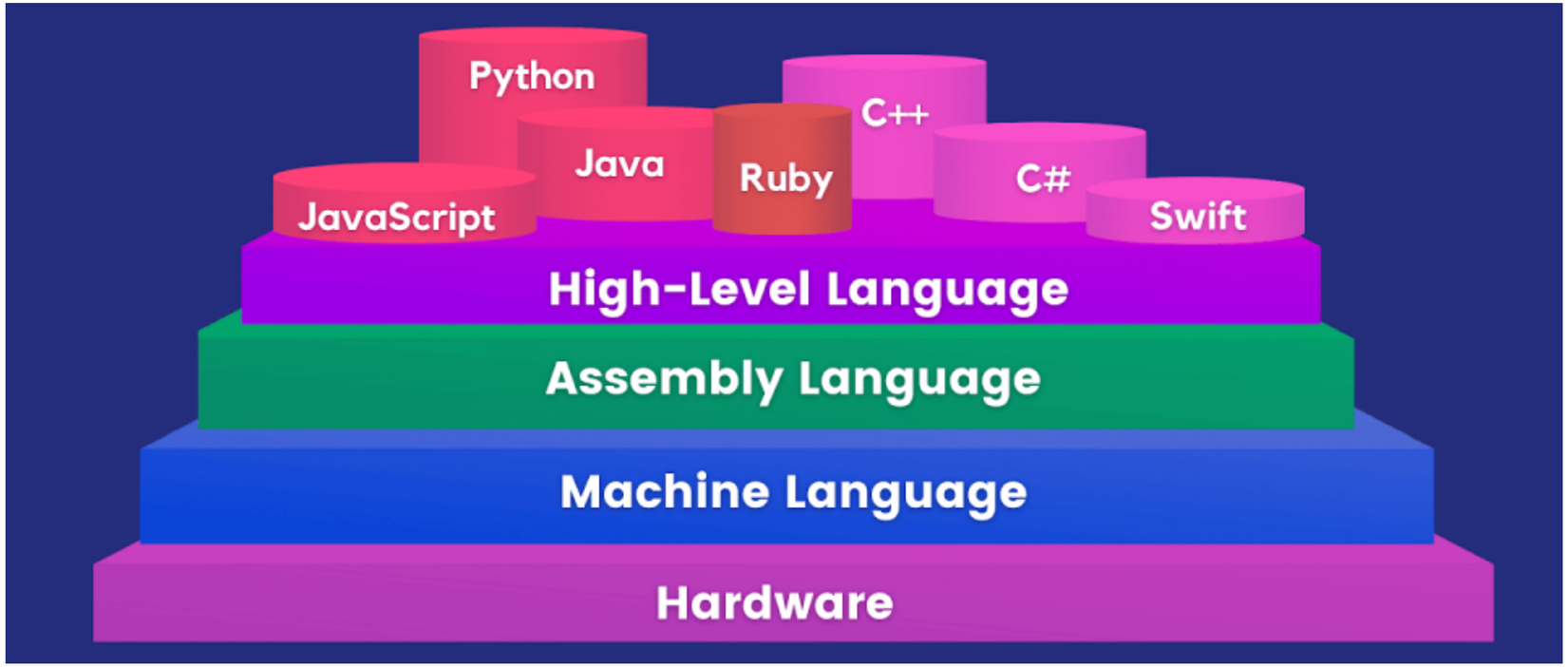


Types of Programming Languages

Programming languages come in various types, each serving specific purposes and catering to different needs.

2) High-level Programming Languages

- Easier to read, write, and understand compared to low-level languages.
- C, C++, Java, Dart, Python …



Types of Programming Languages

Programming languages come in various types, each serving specific purposes and catering to different needs.

3) Scripting Languages

- Specialized for task automation and web development.
- Examples: Python, JavaScript, PHP, Ruby.

4) Specialized Languages

Some languages are designed for specific purposes

- SQL: For managing relational databases.
- R: For statistical analysis.
- MATLAB: For mathematical computing.

Programming Objectives

A program should solve a problem and should be :

- **Correct** : It actually solves the problem
- **Efficient**: Without wasting time or space
- **Readable**: Understandable by another person
- **User-friendly** : In a way that is easy for its user to use

How to write a program?

Steps to writing a program:

Step 1. Think about it

Step 2. Organize your thoughts

Step 3. Write them down in a keywords

Step 4. Translate them into code

How to learn programming?

- Start by reading to understand the concepts and syntax, then **practice**, **practice** and **practice**

- Working till 3:00 am in the morning on a course work only to find that you typed "==" instead of "=" is a great learning experience

Recommended Programming Languages for Cybersecurity

- **C++**: Low-level languages that provide deep system-level understanding, useful for reverse engineering, malware analysis, and kernel exploitation.

- **Python**: Versatile, easy to learn, and widely used in cybersecurity for scripting, automation, and data analysis.

Who Developed C++?

- C++ an extension of C, was developed by Bjarne Stroustrup in the early 1979s at Bell Laboratories.
- The first edition of his book “The C++ Programming Language” was released in 1985.
- C++ is an extension of the C programming language with additional features, including classes, objects, and other features supporting object-oriented programming.

Advantages of Learning C++ in Cybersecurity

- C++ provides direct access to hardware and system resources
- C++ is highly efficient, making it ideal for writing performance
- Suitable for a wide range of cybersecurity applications (network scanning tools, cryptography libraries)
- Secure and analyze these systems effectively
- International standard
- Easy to move from C++ to other languages
- It is FAST

Advantages of Learning C++ in Cybersecurity

IDE (Integrated Development Environment)

IDE’s provide comprehensive tools for

- Writing and editing codes
- Adding and editing resources
- Building (compiling and linking)
- Debugging codes
- Deploying applications

IDE (Integrated Development Environment)

IDE’s for C++

- Microsoft Visual Studio (Community Ed.)
  - Windows OS

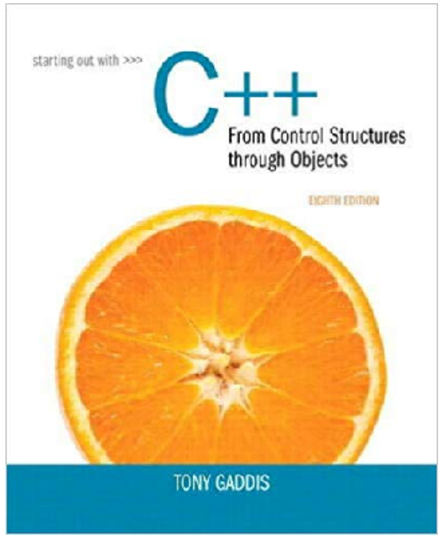
- Microsoft Visual Studio Code
  - Mac OS, Linux OS

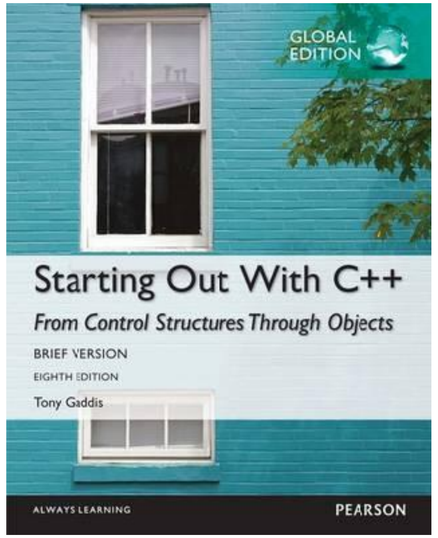
https://visualstudio.microsoft.com

Books

Tony Gaddis, Starting Out with C++: From Control Structures through Objects, 8th Edition

D.S. Malik, C++ PROGRAMMING: From Problem Analysis to Program Design, 5th Edition





Part #2

Introduction to C++

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6
7      return 0;
8  }
```

Introduction to C++

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6
7      return 0;
8  }
```

We are putting our code here

Every C++ program has the same essential format.

Comments

Single-line comments:

```
1  // The first C++ program
```

The // marks the beginning of a *comment*. The compiler ignores everything from the double slash to the end of the line. That means you can type anything you want on that line and the compiler will never complain! Although comments are not required, they are very important to programmers.

Comments

Multi-line comments:

```
/*
    This is a multi-line comment.
    It can put multiple lines.
*/
```

Multi-line comments begin with /* and end with */. Everything between these delimiters is treated as a comment, and it can span multiple lines.

Comments

- Heading
- Author
- Purpose
- Usage
- References
- File Formats
- Restrictions
- Revision History
- Error Handling
- Notes
- Anything else that’ s useful

Comments - Example

```
/* ****************************************************************
* Program: Assignment.cpp                                         *
* Programmer:  Hemin F.                                           *
* Purpose: This program for a simple game called seek and find *
* Time and Date: 30.04.2010  02:00 am                             *
*****************************************************************/


# include "stdafx.h"  //include a header file
# include <iostream> //include library
// Get the current calendar time I found it from http://www.cplusplus.com/reference/clibrary/ctime/time
#include <time.h>
using namespace std; //to look in the std library to find the class
int lucky_random(); //prototype of function lucky_random
int UnLucky_random();  //prototype of function Unlucky_random
int conv(int);  //prototype of function conv
char User_Letter; // Global variable

int main()  //main function and mandatory
{
    //generates a new random set each time
    srand(time(0));
    // define a Main Array it has lucky & Unlucky Number
    int Main_Array[5][5];
    //Define this array just to incorrect selection
    int Incorrect_Selection[5][5];
int i,j; //declare i,j as integer using in for loops
int b=65;// you need it to get Alphabet letter
//Start to Create 2-D array and print it
for (i=0;i<5;i++){ //number of rows from 0-4
```

Comments - Example

- You must always comment your programs.
- Comments help you to organize your thoughts.
- Comments help you to remember what you did .
- Comments help the other programmers to understand your program.

First Program in C++: Printing a Line of Text

```
1  // The first C++ program
2  #include <iostream>
3  using namespace std;
4  int main() {
5      cout<< "Welcome to TIU";
6      return 0;
7  }
```

First Program in C++: Printing a Line of Text

```
2  #include <iostream>
```

- The line begins with #, making it a preprocessor directive. The preprocessor reads your program before compilation.
- (#include) Instructs the preprocessor to include the contents of another file in the program.
- (iostream) Contains code for displaying output on the screen. Enables the program to read input from the keyboard.

Declares a set of functions for standard Input/Output

First Program in C++: Printing a Line of Text

```
3  using namespace std;
```

- Allows you to use elements from the C++ Standard Library (STL) without explicitly specifying the std:: namespace.
- Examples of Standard Library Components:
  - **cout**: Used for displaying output.
  - **cin**: Used for reading input.

First Program in C++: Printing a Line of Text

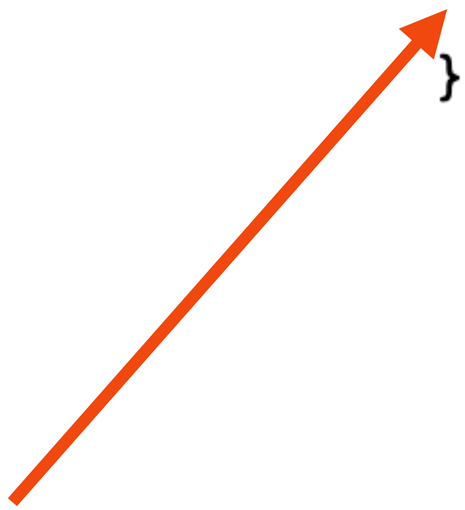
```
3  using namespace std;
```

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

int main() {
   cout << "Hello Tishk!";
   return 0;
}
```

```
#include <iostream>

int main() {
   std::cout << "Hello World!";
   return 0;
}
```



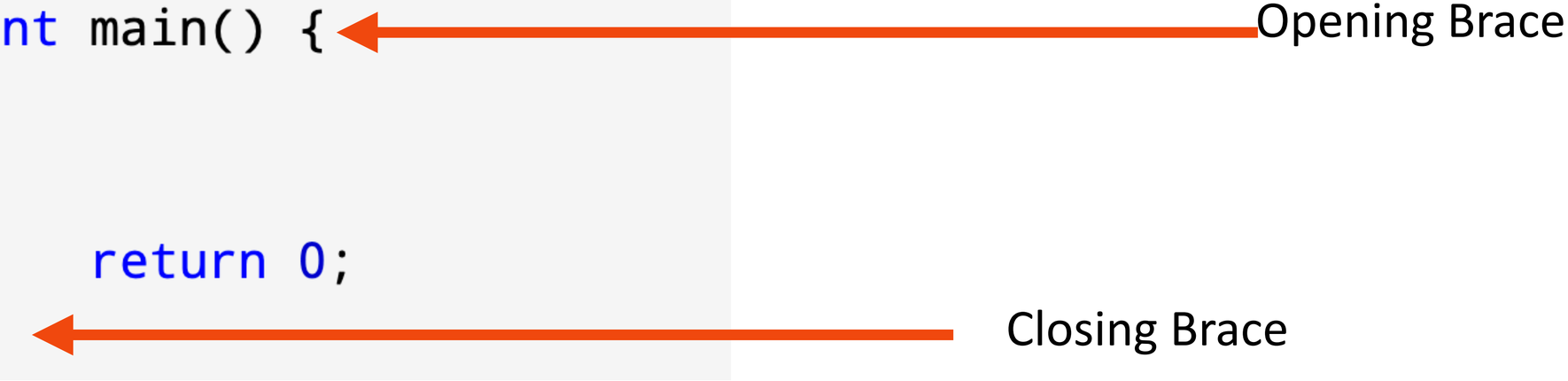
First Program in C++: Printing a Line of Text

```
4  int main() {
5
6
7      return 0;
8  }
```

- **Main Function:**
  - A required part of every C++ program.
  - Program execution starts from the main() function.
- **Return Type (int):**
  - The int before main indicates the function returns an integer.
  - Conventionally, returning 0 indicates successful execution.

First Program in C++: Printing a Line of Text

```
4  int main() {
5
6
7      return 0;
8  }
```



The Curly braces { and } are used to define blocks of code, which are often referred to as compound statements or blocks.

First Program in C++: Printing a Line of Text

```
5       cout<< "Welcome to TIU";
```

- **cout**:
  - Stands for "character output".
  - An instance of the ostream class from the C++ Standard Library.
  - Used to output data to the standard output stream (e.g., console).
- **<<**:
  - Stream Insertion Operator: Inserts data into the output stream for display.
- **;**: Semicolon:
  - Marks the end of a complete statement, similar to a period in a sentence.

Example

```
1 // A simple C++ program
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     cout << "Programming is " << "great fun!";
8     return 0;
9 }
```

Program Output

```
Programming is great fun!
```

Example

```
1  // A simple C++ program
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Programming is ";
8      cout << "great fun!";
9      return 0;
10 }
```

Program Output

```
Programming is great fun!
```

Example

```
1  // An unruly printing program
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     cout << "The following items were top sellers";
8     cout << "during the month of June:";
9     cout << "Computer games";
10    cout << "Coffee";
11    cout << "Aspirin";
12    return 0;
13 }
```

Program Output

```
The following items were top sellersduring the month of June:Computer
gamesCoffeeAspirin
```

Example

```
1  // A well-adjusted printing program
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     cout << "The following items were top sellers" << endl;
8     cout << "during the month of June:" << endl;
9     cout << "Computer games" << endl;
10    cout << "Coffee" << endl;
11    cout << "Aspirin" << endl;
12    return 0;
13 }
```

Special Characters

Character	Name	Description
`//`	Double slash	Marks the beginning of a comment.
`#`	Pound sign	Marks the beginning of a preprocessor directive.
`< >`	Opening and closing brackets	Encloses a filename when used with the `#include` directive.
`( )`	Opening and closing parentheses	Used in naming a function, as in `int main()`
`{ }`	Opening and closing braces	Encloses a group of statements, such as the contents of a function.
`" "`	Opening and closing quotation marks	Encloses a string of characters, such as a message that is to be printed on the screen.
`;`	Semicolon	Marks the end of a complete programming statement.

Special Characters (Escape Characters)

There are some certain characters in C++ which can be used with the escape sequence (\) (escape characters or backslash (\) keys).

Control characters:

- ➢`\n` = Newline
- ➢`\b` = Backspace
- ➢`\t` = Horizontal tab
- ➢`\r` = Return

Punctuation characters:

- ➢`\"` = Double quote
- ➢`\'` = Single quote
- ➢`\\` = backslash

Special Characters (Escape Characters)

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

int main(){
    // Newline
    cout << "Hello" << '\n'
         << "Kurdistan!" << endl;

    // Backspace
    cout << "Hello\bKurdistan!" << endl;

    // Horizontal tab
    cout << "Name\tAge\tCity" << '\n';
    cout << "Alan\t18\tHawler" << '\n';
    cout << "Karzan\t20\tSlemani" << endl;

    // Return
    cout << "12345\rABCD" << endl;

    return 0;
}
```

Output

```
Hello
Kurdistan!
HellKurdistan!
Name    Age     City
Alan    18      Hawler
Karzan  20      Slemani
ABCD5
```

Special Characters (Escape Characters)

```cpp
#include <iostream>

using namespace std;

int main(){
    // Double quote
    cout << "This is a double quote: \"Hello\"" << endl;

    // Single quote
    cout
        << "This is a single quote: 'Hello'" << endl;

    // Backslash
    cout << "This is a backslash: \\\\" << endl;

    return 0;
}
```

Output

```
This is a double quote: "Hello"
This is a single quote: 'Hello'
This is a backslash: \\
```

Thank You