BASIC STRUCTURE OF C++ PROGRAM

INTRODUCTION

prepared by:

Gyro A. Madrona
Electronics Engineer

TOPIC OUTLINE

Types of Programming Language

IDE (Integrated Development Environment)

Basic Structure of a C++ Program

TYPES OF PROGRAMMING LANGUAGE

LOW-LEVEL

Machine Language

The natural language that the computer understands (**0's** and **1's**).

Sample program to add 5 and 3:

```
10111000

00000101

00000000

10111011

00000011

00000000

00010001

11011000
```

LOW-LEVEL

Assembly Language

A programming language that uses "abbreviations" or **mnemonics** in place of binary patterns.

Sample program to add 5 and 3:

```
MOV AX, 05

MOV BX, 03

ADD AX, BX
```

Assembler

A special program that **translates assembly** language mnemonics into machine language.

HIGH-LEVEL

High-Level Language

A programming language that uses **english-like** commands or instructions.

Sample program to add 5 and 3:

```
x = 5;

y = 3;

sum = x + y
```

Compiler

A special program that **translates high-level** language instructions into machine language.

POPULAR PROGRAMMING LANGUAGES

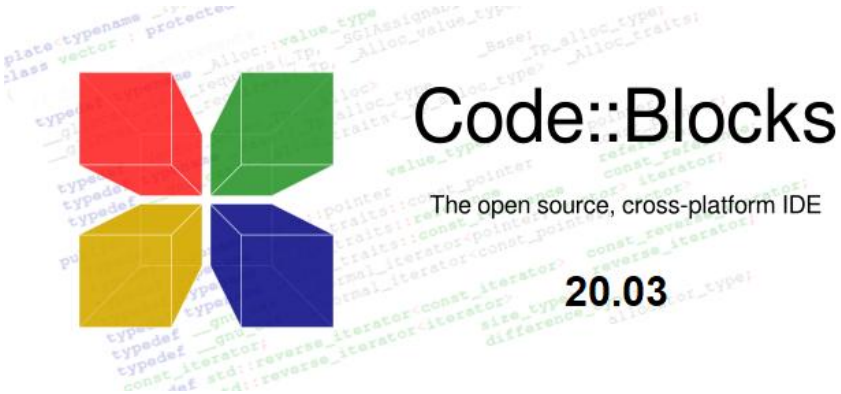
POPULAR PROGRAMMING LANGUAGES

INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

IDE

Integrated Development Environment (IDE)

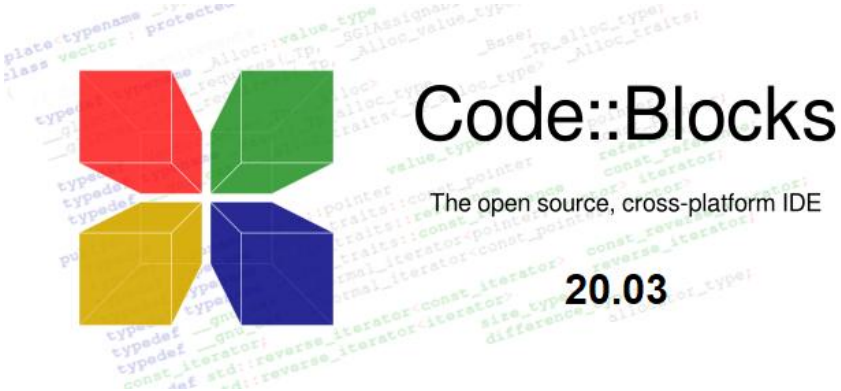
A software application that provides programmers with a comprehensive set of tools to **develop**, **write**, **test**, and **debug code**. It typically includes a code editor, compiler or interpreter, debugger, and build automation tools, all integrated into a single interface for software development process.



IDE

Integrated Development Environment (IDE)

A software application that provides programmers with a comprehensive set of tools to **develop**, **write**, **test**, and **debug code**. It typically includes a code editor, compiler or interpreter, debugger, and build automation tools, all integrated into a single interface for software development process.



BASIC STRUCTURE OF A C++ PROGRAM

MAIN FUNCTION

```
// My First C++ Program

#include <iostream>

using namespace std;

int main ( )

{

   cout << "Hello World!";

   return 0;

}
```

The main function **`main()`** is the entry point of the program where **execution begins**. Every C++ program must have a **`main()`** function.

FUNCTION'S BODY

```
// My First C++ Program

#include <iostream>

using namespace std;

int main ( )

{

    cout << "Hello World!";

    return 0;

}
```

{ }, everything between these braces is the function's body that defines what happens when **main** is called.

INSERTION OPERATOR

```
// My First C++ Program

#include <iostream>

using namespace std;

int main ( )

{

    cout << "Hello World!";

    return 0;

}
```

<<, indicates that what follows is inserted into **cout**.

STATEMENT

```
// My First C++ Program

#include <iostream>

using namespace std;

int main ( )

{

    cout << "Hello World!";

    return 0;

}
```

A **statement** is a single instruction that performs a specific action. Each statement typically ends with a semicolon `(;)`, which marks the end of the instruction.

One of the most common syntax errors in C++ is forgetting to end a statement with a semicolon.

C++ LIBRARY

```
// My First C++ Program

#include <iostream>

using namespace std;

int main ( )

{

    cout << "Hello World!";

    return 0;

}
```

괜찮아 (gwenchana)– If you're not familiar with Korean, you might not immediately understand what it means—unless you refer to a Korean-English dictionary.

The compiler does not automatically recognize standard library functions like **cout**, **cin**, or **endl** unless we specify their source. This is where the **namespace std** comes in.

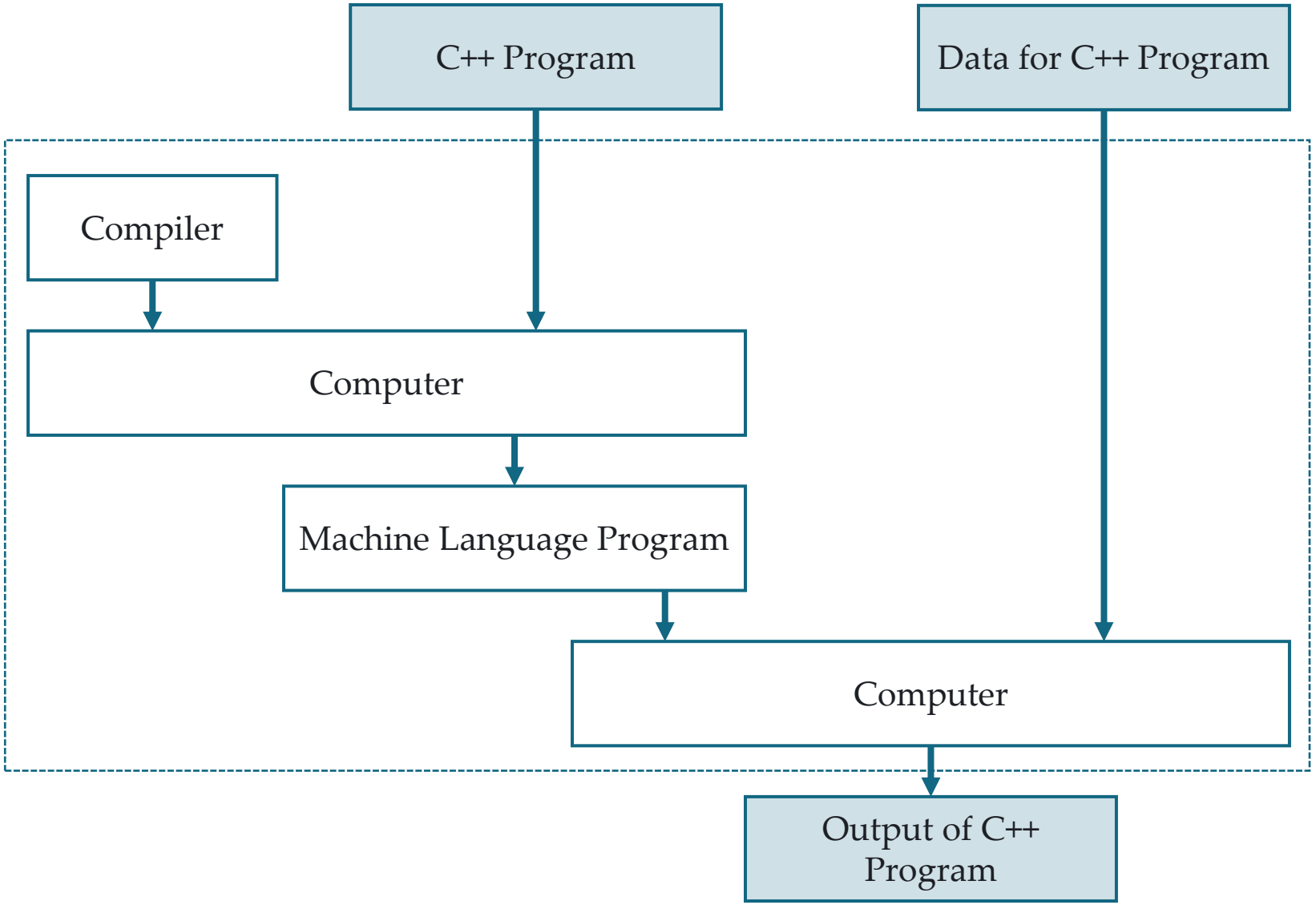
COMMENTS

```
// My First C++ Program

#include <iostream>

using namespace std;

int main ( )

{

    cout << "Hello World!";

    return 0;

}
```

Comments do not affect the operation of the program; however, they provide an important tool to **document** directly the source code what the program does and how it operates.

```
// line comment

/* block comment */
```

RUNNING A C++ PROGRAM



LABORATORY