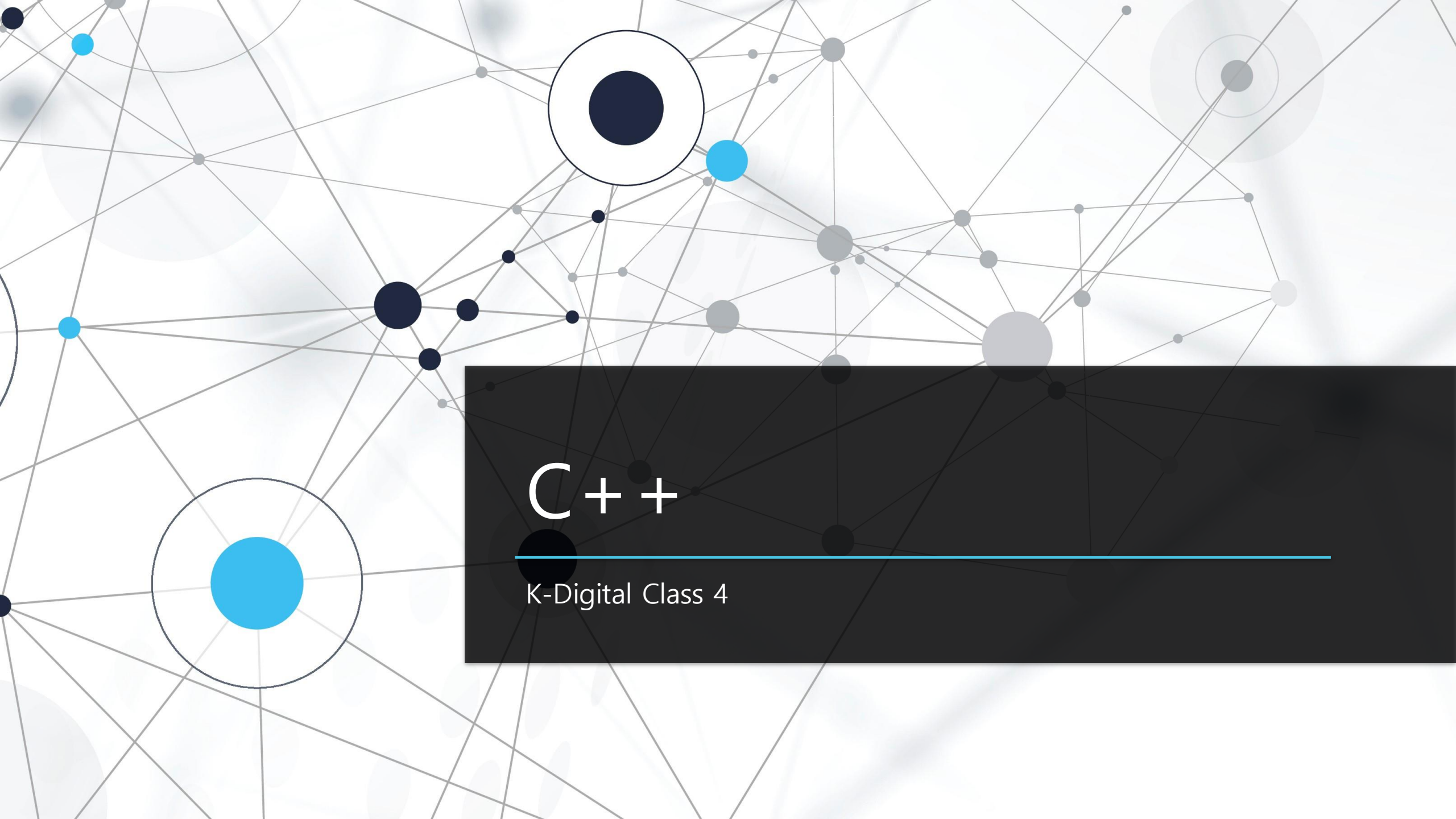


C++

K-Digital Class 4

C++ 개요

- C++는 Bjarne Stroustrup이 1979년 Bell Labs에서 개발한 High-Level Programming Language이다. C++는 Windows, Mac OS 및 다양한 UNIX/LINUX 버전과 같은 다양한 Platform에서 실행된다.
- Stroustrup이 작업할 기회가 있었던 Language 중 하나는 Simula라는 Language 였다. 이름에서 알 수 있듯이 주로 Simulation을 위해 설계된 Language 이다. 여기서 객체 지향(Object Oriented)의 개념을 C Language 에 추가 하여 1983 C++를 발표한다.
- ++ 연산자(Operator)를 추가 하여 C++라고 명명 하게 된다.
  - ++은 +1의 의미를 가진다.

C++의 특징

- C++은 절차 지향적이며 구조적 프로그래밍 언어이다.
- C++은 객체 지향 프로그래밍 언어이다.
- C++은 Compile(번역) 언어이다.
  - Compiler라는 매체를 통한 소스 파일을 기계어로 Compile(번역)하는 언어.

- 이처럼 C++은 세 가지 프로그래밍 방식을 모두 지원하는 언어이며, 따라서 다양한 방식으로 프로그램을 작성할 수 있다.
- 또한, 다양한 시스템에서의 프로그래밍을 지원하는 유용하고도 강력한 클래스 라이브러리들이 아주 많이 제공된다.
- 하지만 C++은 이렇게 언어로서 다양한 방식을 지원하며 강력하게 거듭났지만, 프로그래머의 측면에서 보면 이러한 다양한 기능을 모두 배워야 하는 부담으로 작용하기도 한다.

C++ 표준

- C++에 대한 표준은 ANSI(American National Standards Institute)와 ISO(International Organization for Standardization)가 표준화 작업을 진행한다.
- 1998년에 첫 C++ 국제 표준인 ISO/IEC 14882:1998이 제정되며, 이 표준을 C++98이라고 부르게 된다.
- 이후 C++98에서 단순한 기술적 개정만을 진행한 ISO/IEC 14882:2003(C++03)이 2003년에 공개되었다.
- 2011년에는 많은 언어적 특성이 추가된 ISO/IEC 14882:2011이 발표되며, 이 표준을 C++11이라고 부르게 된다.
- 이후 C++11의 사소한 버그 수정 및 약간의 기술적 개선을 진행한 ISO/IEC 14882:2014(C++14)가 2014년에 공개되었다.
- 2017년 파일 시스템, STL 병렬 알고리즘 처리 등이 추가되어 12월에 C++1z로 알려졌던 C++17 표준이 ISO/IEC 14882:2017이라는 정식 명칭으로 최종 승인되었다.

C++ PROGRAMMING

- 소스 파일(source file)의 작성
- 선행처리기(preprocessor)에 의한 선행처리
- 컴파일러(compiler)에 의한 컴파일
- 링커(linker)에 의한 링크
- 실행 파일(executable file)의 생성

C++ PROGRAMMING

소스 파일(source file)의 작성

- Programming에서 가장 먼저 해야 할 작업은 바로 프로그램을 작성하는 것이다.
- 다양한 에디터를 사용하여 C++ 문법에 맞게 논리적으로 작성된 프로그램을 원시 파일 또는 소스 파일이라고 한다.
- C++을 통해 작성된 소스 파일의 확장자는 대부분 .cpp 가 된다.

C++ PROGRAMMING

전처리기(preprocessor)에 의한 선행처리

- 전처리기(Preprocessor)는 실제 컴파일이 시작되기 전에 컴파일러에게 정보를 사전 처리하도록 지시하는 지시사항이다.
- Source file 중에서도 전 처리 문자(#)로 시작하는 전 처리 지시문의 처리 작업을 의미한다.
- 이러한 선행처리 작업은 전처리기(preprocessor)에 의해 처리된다.
- 전처리기는 코드를 생성하는 것이 아닌, 컴파일하기 전 컴파일러가 작업하기 좋도록 Source code를 재구성해주는 역할만을 한다.

C++ PROGRAMMING

컴파일러(compiler)에 의한 컴파일

- Computer는 0과 1로 이루어진 이진수로 작성된 기계어만을 이해할 수 있다.
- Source file은 개발자에 의해 C++ 언어로 작성되기 때문에 컴퓨터가 그것을 바로 이해할 수는 없다.
- 따라서 Source file 을 Computer가 알아볼 수 있는 기계어로 변환시켜야 하는데, 그 작업을 컴파일(compile)이라고 한다.
- Compile은 C/C++ compiler에 의해 수행되며, compile이 끝나 기계어로 변환된 파일을 Object file 이라고 한다.
- 이러한 Object file의 확장자는 .o 나 .obj 가 된다.

C++ PROGRAMMING

링커(linker)에 의한 링크

- Compiler에 의해 생성된 Object file은 운영체제(Operating System(OS))와의 Interface를 담당하는 시동 코드(Start-up code)를 가지고 있지 않다.
- 또한, 대부분의 C++ 프로그램에서 사용하는 표준 라이브러리 파일도 가지고 있지 않다.
- 하나 이상의 Object file과 Library file, Start-up code 등을 합쳐 하나의 파일로 만드는 작업을 링크(Link)라고 한다.
- Link는 링커(linker)에 의해 수행되며, Link가 끝나면 하나의 새로운 실행 파일이나 Library file이 생성된다.
- 이처럼 여러 개의 Source code file을 작성하여 최종적으로 링크를 통해 하나의 실행 파일로 만드는 것을 분할 컴파일이라고 한다.

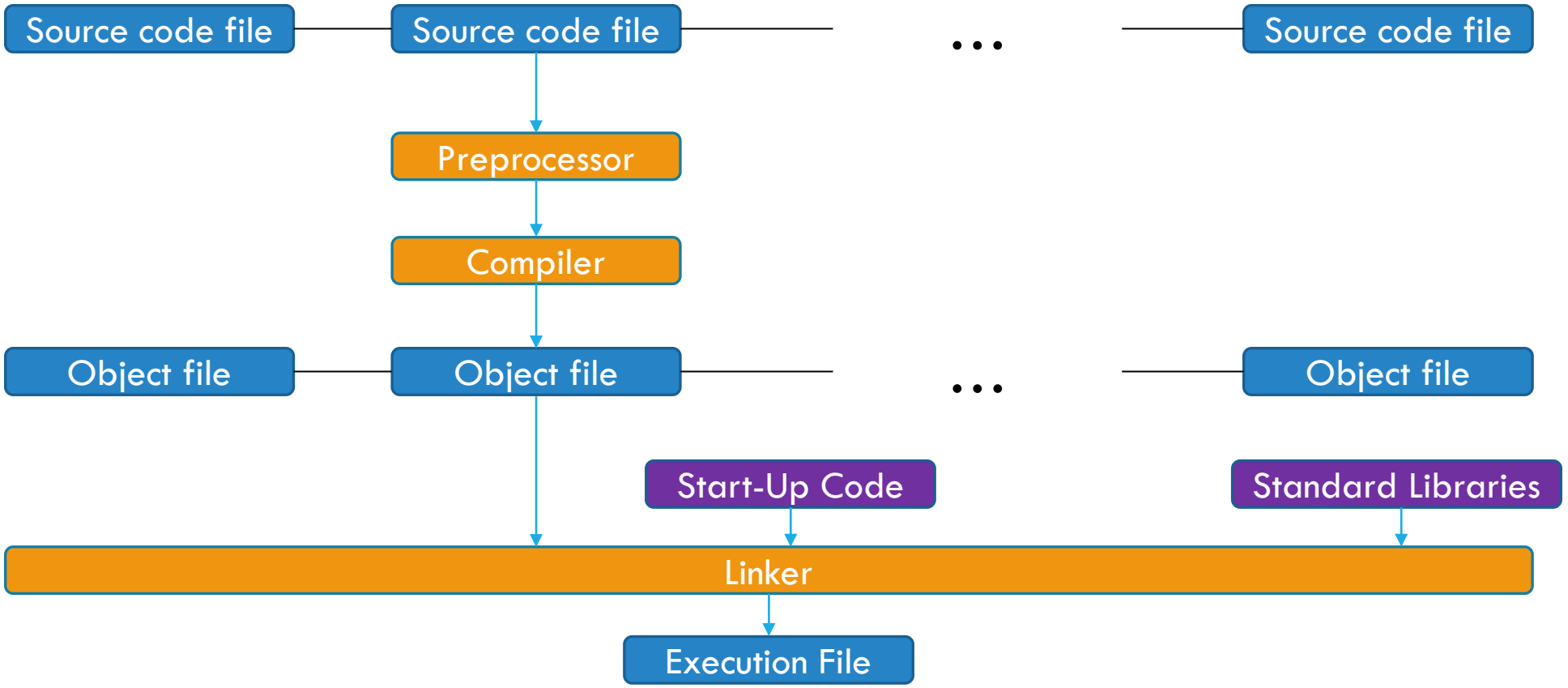
C++ 프로그램은 main함수를 가지고 있을 것이다. 그렇다면 이 main은 누가 호출하는 것이냐? 이것이 바로 스타트업 코드이다. 스타트업 코드의 기능은 간단히 얘기하면, CPU 레지스터 초기화와 정적 변수 초기화를 한 후 main을 호출하는 것이다.

C++ PROGRAMMING

실행 파일(executable file)의 생성

- Source code file은 전처리기(Preprocessor), Compiler 그리고 Linker에 의해 실행 파일로 변환된다.
- 최근 사용되는 개발 툴은 대부분 전처리기(Preprocessor), Compiler, Linker를 모두 내장하고 있으므로 Source code file에서 한 번에 실행 파일을 생성할 수 있다.
- Windows의 경우 이렇게 생성된 실행 파일의 확장자는 .exe 가 된다.
- Linux/Mac/Unix의 경우 실행 권한을 가진 확장자 없이 실행 파일이 생성된다.

C++ PROGRAMMING



C++ PROGRAMMING

Object-Oriented Programming

- C++는 객체 지향 개발의 네 가지를 포함하여 객체 지향 프로그래밍 (Object-Oriented Programming)을 완벽하게 지원한다.
  - Encapsulation(캡슐화)
  - Information hiding(정보의 은닉화)
  - Inheritance(상속)
  - Polymorphism(다형성)

C++ PROGRAMMING

Object-Oriented Programming

- Encapsulation : 객체의 속성(data fields)과 행위(methods)를 하나로 묶고, 실제 구현 내용 일부를 외부에 감추어 은닉한다.
- Information hiding : 은닉이라는 단어 때문에 캡슐화와 은닉화를 혼동하는 분이 많습니다. 은닉화는 캡슐화를 통해 얻어지는 "실제 구현 내용 일부를 외부에 감추는" 효과이다.
- Inheritance : Parent의 Method 혹은 Field를 Child class에게 상속해 주는 것이다. 즉 '기존 class에 method를 추가하거나 재정의하여 새로운 class를 정의한다.'
- Polymorphism : 하나의 객체에 여러 가지 타입을 대입할 수 있다는 것을 의미한다. 반대로, 단형성은 하나의 객체에 하나의 타입만 대응할 수 있다.

C++ PROGRAMMING

Basic syntax of a C++

HelloWorld.cpp > ...

```cpp
// HelloWorld.cpp

#include <iostream>

int main()
{
    std::cout << "Hello World!\n";

    return 0;
}

```

C++ PROGRAMMING

Structure of a C++ program

- Line1
  - 주석이며 프로그램의 동작에는 영향을 미치지 않는다. 코드나 프로그램에 관한 짧은 설명이나 관찰을 서술 하는데 사용한다.
- Line3
  - #으로 시작하는 행은 전처리기에서 읽고 해석하는 지시문이다. 컴파일이 시작되기 전에 해석되는 특수 라인이다. 이 경우 헤더 iostream으로 알려진 표준 C++ 코드를 포함하도록 전처리기에 지시한다.
- Line5
  - C/C++는 main 함수에서 실행이 시작 되기 때문에 반드시 있어야 하는 함수이다.
- Line6, 10
  - 6행의 여는 중괄호({)는 main 함수 정의의 시작을 나타내고 10행의 닫는 중괄호(})는 끝을 나타낸다.
- Line7
  - 이 줄은 C++ 문장(Statement)이다. 실제 동작을 지정하는 프로그램의 핵심이다. 모든 C/C++ 문장은 세미콜론(;)으로 끝난다.

C++ PROGRAMMING

Comments(주석)

- 프로그램이 수행하는 작업과 작동 방식을 소스 코드 내에서 직접 문서화하는 중요한 도구를 제공한다.
- C++는 코드 주석 처리의 두 가지 방법을 지원한다.
  - //
  - /* comment */
- Source code 설명을 HTML 문서로 저장 할 수 있는 doxygen type 주석을 권장 한다.

HelloWorld.cpp > ...

```cpp
// HelloWorld.cpp
/* This is Hello World C++ File*/
/* C++ comments can also
   * span multiple lines
*/
#include <iostream>

using namespace std;


/**
 * @brief Must need main function in C++ source code
 *
 * @return int
 */
int main()
{
    cout << "Hello World!\n";

    return 0;
}

```

C++ PROGRAMMING

Using namespace std

- cout은 표준 라이브러리의 일부이며 표준 C++ 라이브러리의 모든 요소는 namespace라고 하는 std namespace 내에서 선언되어 있다.
- std namespace에 있는 요소를 참조하기 위해 프로그램은 Library의 모든 요소 사용에 대해 자격을 부여하거나 (cout에 std::를 붙여서 수행한 것처럼) 해당 구성 요소의 가시성을 도입해야 한다. 이러한 구성 요소의 가시성을 도입하는 가장 일반적인 방법은 선언을 사용하는 것이다.

HelloWorld.cpp > ...

```cpp
// HelloWorld.cpp
/* This is Hello World C++ File*/
/* C++ comments can also
   * span multiple lines
*/
#include <iostream>

using namespace std;


/**
 * @brief Must need main function in C++ source code
 *
 * @return int
 */
int main()
{
    cout << "Hello World!\n";

    return 0;
}

```

C++ 변수(VARIABLE)

- 변수(variable)란 데이터(data)를 저장하기 위해 프로그램에 의해 이름을 할당 받은 메모리 공간을 의미한다.
- 즉, 변수란 데이터(data)를 저장할 수 있는 메모리 공간을 의미하며, 이렇게 저장된 값은 변경될 수 있다.
- C++에서 숫자 표현에 관련된 변수는 정수형 변수와 실수형 변수로 구분할 수 있다.
- 또 다시 정수형 변수는 char형, int형, long형, long long형 변수로, 실수형 변수는 float형, double형 변수로 구분된다.
- 관련된 데이터를 한 번에 묶어서 처리하는 사용자 정의 구조체(structure) 변수도 있다.

C++ 변수(VARIABLE)

변수(Variable)의 이름 생성 규칙

- C++에서는 변수의 이름을 비교적 자유롭게 지을 수 있다.
- 변수의 이름은 해당 변수에 저장될 데이터의 의미를 잘 나타내도록 짓는 것이 좋다.
- C++에서 변수의 이름을 생성할 때 반드시 지켜야 하는 규칙은 다음과 같다.
  - 변수의 이름은 영문자(대소문자), 숫자, Underscore(_)로만 구성된다.
  - 변수의 이름은 숫자로 시작될 수 없다.
  - 변수의 이름 사이에는 공백을 포함할 수 없다.
  - 변수의 이름으로 C++에서 미리 정의된 키워드(keyword)는 사용할 수 없다.
  - 변수 이름의 길이에는 제한이 없다.

C++에서는 변수의 이름에 대소문자를 구분하므로 이 점에 주의해야 한다.

C++ 변수(VARIABLE)

비트(bit)와 바이트(byte)

- Computer는 모든 data를 2진수(Binary)로 표현하고 처리한다.
- 비트(bit)란 Computer가 data를 처리하기 위해 사용하는 data의 최소 단위이다.
- 이러한 bit에는 2진수의 값(0과 1)을 단 하나만 저장할 수 있다.
- 바이트(byte)란 비트가 8개 모여서 구성되며, 한 문자(Character)를 표현할 수 있는 최소 단위다.

C++ 변수(VARIABLE)

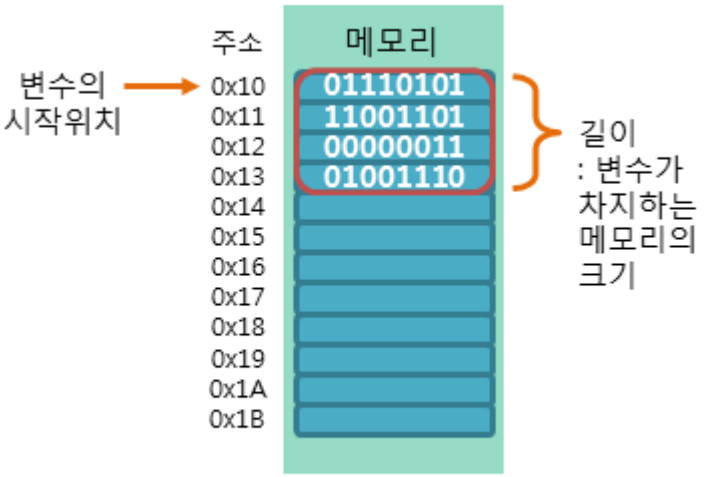
Variable(변수)과 Memory Address

- 변수는 기본적으로 메모리의 주소(address)를 기억하는 역할을 한다.
- 메모리 주소란 물리적인 메모리 공간을 서로 구분하기 위해 사용되는 일종의 식별자(identifier)다.
- 즉, 메모리 주소란 메모리 공간에서의 정확한 위치를 식별하기 위한 고유 주소를 의미한다.
- 변수를 참조할 때는 메모리의 주소를 참조하는 것이 아닌, 해당 주소에 저장된 데이터를 참조하게 된다.
- 따라서 변수는 데이터가 저장된 메모리의 주소 뿐만 아니라, 저장된 데이터의 길이와 형태에 관한 정보도 같이 기억해야 한다.

C++ 변수(VARIABLE)

Variable(변수)과 Memory Address

- 다음 그림은 메모리상에 변수가 어떤 식으로 저장되는지를 보여줍니다.
- 그림에서 하나의 메모리 공간에는 8개의 비트로 이루어진 1바이트의 데이터가 저장되어 있다.
- 해당 변수의 길이는 총 4개의 메모리 공간을 포함하므로, 해당 변수는 총 4바이트의 데이터를 저장하고 있는 것이다.



형태 : 메모리에 저장된 데이터를 해석하는 방법

C++ 변수(VARIABLE)

Declaring (Creating) Variables

- C++에는 다양한 유형의 변수가 있다. 예를 들면 다음과 같다.

- int - 123 또는 -123과 같이 소수 없이 정수(정수)를 저장한다.
- double - 19.99 또는 -19.99와 같은 소수를 사용하여 부동 소수점 숫자를 저장한다.
- char - 'a' 또는 'B'와 같은 단일 문자를 저장합니다. Char 값은 작은따옴표로 묶는다.
- string - "Hello World"와 같은 텍스트를 저장한다. 문자열 값은 큰따옴표(“”)로 묶는다.
- bool - true 또는 false의 두 가지 상태로 값을 저장한다.

C++ 변수(VARIABLE)

Declaring (Creating) Variables

- 변수(Variables)를 선언하려면 유형(Type)을 지정하거나 값을 할당한다.
- 여기서 type은 C++ type(예: int) 중 하나이고 variableName은 변수의 이름(예: x 또는 myName)이다. 등호(=)는 변수에 값을 할당하는 데 사용되는 대입 연산자이다.

- Syntax
- type variableName;
- type variableName = value;
- Ex)
  - int myNum = 15;

C++ 변수(VARIABLE)

Declare Multiple Variables

- 동일한 type의 변수를 둘 이상 선언하려면 쉼표(,)로 구분하여 선언한다.

- int x = 5, y = 6, z = 50;
- cout << x + y + z;

C++ 변수(VARIABLE)

Variable Scope in C++

- Scope는 프로그램의 한 영역이며, 대체로 변수를 선언할 수 있는 세 곳이 있다.

- 지역 변수(Local Variable)라고 하는 함수나 블록 내부에 정의.
- 형식 매개변수라고 하는 함수 매개 변수의 정의.
- 전역 변수(Global Variable)라고 하는 모든 함수 외부에서 정의.

C++ 변수(VARIABLE)

Local Variables

- 함수(function)나 블록(block) 내에서 선언된 변수는 지역 변수(local variable)이다. 해당 함수 또는 코드 블록(code block:{}) 내부에 있는 Statements(명령문)에서만 사용할 수 있다.
- 지역 변수는 자신의 외부에 있는 함수에서 사용할 수 없다.

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

int main () {
   // Local variable declaration:
   int a, b;
   int c;

   // actual initialization
   a = 10;
   b = 20;
   c = a + b;

   cout << c;

   return 0;
}
```

C++ 변수(VARIABLE)

Global Variables

- 전역 변수(global variables)는 일반적으로 프로그램의 맨 위에 있는 모든 함수 외부에서 정의된다. 전역 변수는 프로그램 수명(life-time) 내내 그 값을 유지(Keep)한다.
- 전역 변수는 모든 함수에서 access(접근)할 수 있다. 즉, 전역 변수는 선언 후 전체 프로그램에서 사용할 수 있다.

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

// Global variable declaration:
int g;

int main () {
   // Local variable declaration:
   int a, b;

   // actual initialization
   a = 10;
   b = 20;
   g = a + b;

   cout << g;

   return 0;
}
```

C++ 변수(VARIABLE)

Global Variables

- 프로그램은 지역(local) 및 전역(global) 변수에 대해 동일한 이름을 가질 수 있지만 함수 내부의 지역 변수 값이 우선한다.

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

// Global variable declaration:
int g = 20;

int main () {
   // Local variable declaration:
   int g = 10;

   cout << g;

   return 0;
}
```