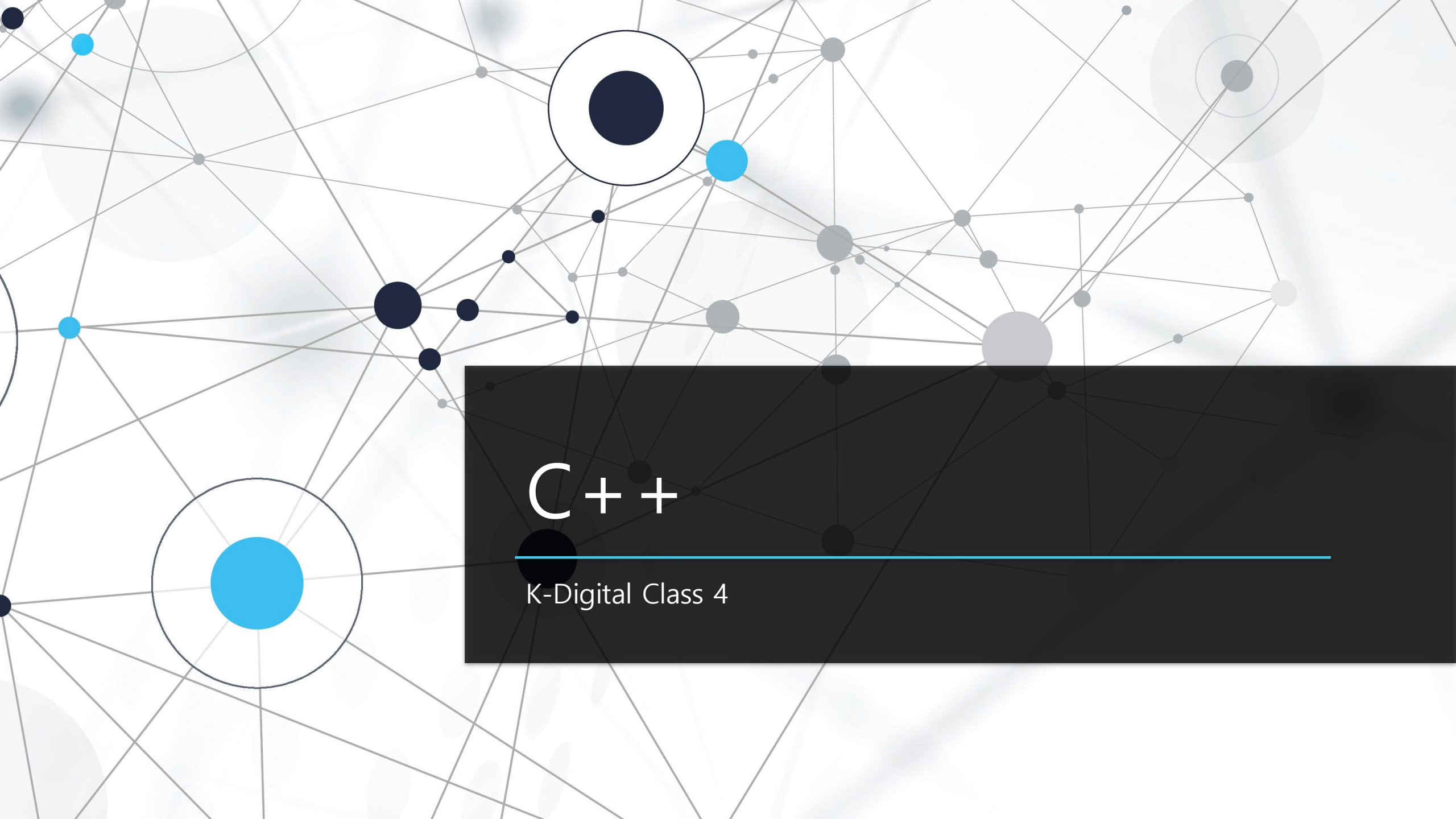

# C++

K-Digital Class 4

# C++ IOSTREAM

## C++ Standard Input/Output(표준 입출력) Class

- 사용자가 프로그램과 대화하기 위해서는 사용자와 프로그램 사이의 입출력을 담당하는 수단이 필요하다.
- C++의 모든 것은 Object(객체)로 표현되므로, 입출력을 담당하는 수단 또한 C언어의 함수와는 달리 모두 Object이다.
- C언어의 printf() 함수나 scanf() 함수처럼 C++에서도 iostream Header file(헤더 파일)에 표준 입출력 클래스를 정의하고 있다.
- C++에서는 cout Object로 출력 작업을, cin 객체로 입력 작업을 수행하고 있다.
- C++에서는 기존의 C언어 스타일처럼 printf() 함수나 scanf() 함수로도 입출력 작업을 수행할 수 있다.

# C++ IOSTREAM

## C++ Output (Print Text)

- cout 객체는 << 연산자와 함께 값을 출력하거나 텍스트를 출력하는 데 사용된다.
- std::cout << “output string”;
- 출력 연산자(<<)는 오른쪽에 위치한 출력할 데이터를 출력 스트림에 삽입한다.
- 이렇게 출력 스트림에 삽입된 데이터는 스트림을 통해 출력 장치로 전달되어 출력된다.

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

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

# C++ IOSTREAM

## New Lines

- 줄 바꿈(새 줄)을 삽입하려면 ₩n 문자를 사용할 수 있다.
- 또한 endl을 사용 하여 줄 바꿈(새 줄)을 삽입한다.

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

int main() {
  cout << "Hello World! \n";
  cout << "I am learning C++";
  return 0;
}
```

# C++ IOSTREAM

## C++ User Input

- cout이 값을 출력(인쇄)하는 데 사용된다는 면 이제 cin을 사용하여 사용자 입력을 받는다.
- cin은 extraction(추출) operator(연산자)(>>)를 사용하여 키보드에서 데이터를 읽는 미리 정의된 입력 스트림을 나타내는 객체이다.
- std::cin >> 저장 할 변수;

```
int x;
cout << "Type a number: "; // Type a number and press enter
cin >> x; // Get user input from the keyboard
cout << "Your number is: " << x; // Display the input value
```

# C++ STRINGS

- Strings are used for storing text.
- To use strings, you must include an additional header file in the source code, the <string> library

```
// Include the string library
#include <string>

// Create a string variable
string greeting = "Hello";
```

# C++ STRINGS

## String Concatenation

- + 연산자는 문자열 사이에 사용하여 함께 추가하여 새 문자열을 만들 수 있다.
- 이것을 Concatenation (연결)이라고 한다.

```
string firstName = "John ";
string lastName = "Doe";
string fullName = firstName + lastName;
cout << fullName;
```

# C++ STRINGS

## String Concatenation

- 출력 시 John과 Doe 사이에 공백을 만들기 위해 firstName 뒤에 공백을 추가했다.
- 따옴표(" " 또는 ' ')로 공백을 추가할 수도 있다.

```
string firstName = "John";
string lastName = "Doe";
string fullName = firstName + " " + lastName;
cout << fullName;
```

# C++ STRINGS

## String Concatenation - Append

- C++의 string은 실제로 문자열에 대해 특정 작업을 수행할 수 있는 함수를 포함하는 object이다.
- 예를 들어, 문자열을 append() 함수로 연결할 수도 있다.

```
string firstName = "John ";
string lastName = "Doe";
string fullName = firstName.append(lastName);
cout << fullName;
```

**+ 또는 append()를 사용할지 여부는 사용자에게 달려 있다.**
**이 둘의 주요 차이점은 append() 함수가 훨씬 더 빠르다는 것이다.**
**그러나 테스트 등의 경우에는 +를 사용하는 것이 더 쉬울 수 있다.**

# C++ STRINGS

## C++ Numbers and Strings

- C++는 덧셈과 문자열 연결 모두에 + 연산자를 사용한다.
- 숫자는 덧셈이 된다.
- 문자열은 문자열이 연결된다.

```
int x = 10;
int y = 20;
int z = x + y;    // z will be 30 (an integer)

string xx = "10";
string yy = "20";
string zz = xx + yy;   // zz will be 1020 (a string)
```

# C++ STRINGS

## C++ Numbers and Strings

```
string x = "10";
int y = 20;
string z = x + y;
```

# C++ STRINGS

## C++ String Length

- 문자열의 길이를 얻으려면 length() 함수를 사용한다.
- string의 길이를 얻기 위해 size() 함수를 사용하는 일부 C++ 프로그램을 볼 수 있다.
- 이것은 단지 length()의 alias(별칭)이다. length() 또는 size()를 사용하려는 경우 전적으로 프로그래머에게 달려 있다.

```
string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
cout << "The length of the txt string is: " << txt.length();
cout << "The length of the txt string is: " << txt.size();
```

# C++ STRINGS

## C++ Access Strings

- 대괄호 [] 안의 Index(색인)를 참조하여 문자열의 문자에 access할 수 있다.

```
string myString = "Hello";
cout << myString[0];
// Outputs H
```

# C++ STRINGS

## Omitting Namespace(Namespace 생략)

- Standard Namespace Library 없이 실행되는 일부 C++ 프로그램을 볼 수 있다.
- using namespace std 행은 생략하고 std 키워드로 대체할 수 있으며, 그 뒤에 string(및 cout) 객체에 대한 :: 연산자가 온다.
- Standard Namespace Library 를 포함할지 여부는 개발자에게 달려 있다.

```
#include <iostream>
#include <string>

int main() {
  std::string greeting = "Hello";
  std::cout << greeting;
  return 0;
}
```

# C++ CONTROL FLOW STATEMENTS

- C++ 프로그램이 원하는 결과를 얻기 위해서는 프로그램의 순차적인 흐름을 제어해야만 할 경우가 생긴다.
- 이때 사용하는 명령문을 제어문이라고 하며, 이러한 제어문에는 조건문, 반복문 등이 있다.
- 이러한 제어문에 속하는 명령문(statements)들은 중괄호({})로 둘러싸여 있으며, 이러한 중괄호 영역을 블록(block)이라고 한다.

# C++ CONTROL FLOW STATEMENTS

## 조건문(conditional statements)

- C++는 수학의 일반적인 논리 조건을 지원한다.
  - Less than: a < b
  - Less than or equal to: a <= b
  - Greater than: a > b
  - Greater than or equal to: a >= b
  - Equal to a == b
  - Not Equal to: a != b
- 이러한 조건을 사용하여 다른 결정에 대해 다른 작업을 수행할 수 있다.

- C++에는 다음과 같은 조건문이 있다.
  - 지정된 조건이 true인 경우 실행할 코드 블록을 지정하려면 'if'를 사용한다.
  - 동일한 조건이 false인 경우 'else'를 사용하여 실행할 코드 블록을 지정한다.
  - 첫 번째 조건이 false인 경우 테스트할 새 조건을 지정하려면 'else if'를 사용한다.
  - 실행할 많은 대체 코드 블록을 지정하려면 'switch'를 사용한.

# C++ CONTROL FLOW STATEMENTS

## The ‘if’ Statement

- if 문을 사용하여 조건이 true인 경우 실행할 C++ 코드 블록을 지정한다.
- if 는 소문자이다. 대문자(If 또는 IF)는 오류를 생성한다.

Syntax

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

# C++ CONTROL FLOW STATEMENTS

## The ‘else’ Statement

- else 문을 사용하여 조건이 false인 경우 실행할 코드 블록을 지정한다.

Syntax

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

# C++ CONTROL FLOW STATEMENTS

## The “else if” Statement

- 첫 번째 조건이 false인 경우 else if 문을 사용하여 새 조건을 지정한다.

Syntax

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

# C++ CONTROL FLOW STATEMENTS

## C++ Short Hand If Else - Ternary Operator

- 세 개의 피연산자로 구성 된 삼항 연산자로 알려진 short-handed if else도 있다.

Syntax

```
variable = (condition) ? expressionTrue : expressionFalse;
```

# C++ CONTROL FLOW STATEMENTS

## C++ Switch Statements

- switch 문은 if / else 문과 마찬가지로 주어진 조건 값의 결과에 따라 프로그램이 다른 명령을 수행하도록 하는 조건문이다.
- switch 문을 사용하여 실행할 많은 코드 블록 중 하나를 선택한다.
  - switch expression은 한 번 평가된다.
  - expression의 값은 각 case 값과 비교된다.
  - 일치하는 항목이 있으면 연결된 code block이 실행된다.

Syntax

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

# C++ CONTROL FLOW STATEMENTS

## C++ Switch Statements - The break Keyword

- C++가 break 키워드에 도달하면 switch 블록에서 나온다.
- 이렇게 하면 블록 내에서 더 많은 코드 및 case 테스트 실행이 중지된다.
- 일치하는 항목이 발견되고 작업이 완료되면 break 된다.더 많은 case 테스트가 필요하지 않다.
- Break문은 switch block의 나머지 모든 코드 실행을 " 무시 " 하기 때문에 많은 실행 시간을 절약할 수 있다.

Syntax

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

# C++ CONTROL FLOW STATEMENTS

## C++ Switch Statements - The default Keyword

- Default 는 일치하는 case가 없는 경우에 실행할 code를 지정한다.
- default 키워드는 switch의 마지막 문으로 사용해야 하며 break가 필요하지 않다.

Syntax

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```