C++ Programming

3rd Study: From C to C++ (3/4)

- Scoped enum
- Binary literal, separator
- std::string

C++ Korea 옥찬호 (utilForever@gmail.com)

Scoped enum

Strong-typed enumeration in C++11

Scoped enum

- C / C++에서 기존에 사용하던 열거형에 존재하는 문제들
  - A 열거체에서 사용한 T라는 이름을 B 열거체에서 사용할 수 없음
    - 전역(Global) 범위로 선언되기 때문
  - 서로 다른 열거체 변수끼리 더하거나 비교할 수 있음
    - 기본적으로 열거체의 타입은 int → 묵시적으로 변환되기 때문
- C++11에는 이런 문제들을 보완한 범위 지정 열거체를 제공
  - enum 대신 enum class를 사용
  - 명시적으로 타입을 지정하지 않으면 오류 (묵시적 변환 X)
  - 서로 다른 열거체 변수끼리 연산할 수 없음

Scoped enum: Example

C / C++98

```
int main()
{
    enum TrafficLight { Red, Yellow, Green };
    enum Job { Warrior, Ranger, Wizard, /* Green */ };
    enum Coffee : unsigned char { Latte = 10, Mocha = 25 };

    int jobNum = Warrior;
    int i = Green + Latte;

    if (Yellow == Ranger)
        std::cout << "Same!" << std::endl;
    else
        std::cout << "Different!" << std::endl;

    return 0;
}
```

Scoped enum: Example

C++11

```
int main()
{
    enum class TrafficLight { Red, Yellow, Green };
    enum class Job { Warrior, Ranger, Wizard, Green };
    enum class Coffee : unsigned char { Latte = 10, Mocha = 25 };

    Job jobNum = Job::Warrior;
    int i = static_cast<int>(TrafficLight::Green)
        + static_cast<int>(Coffee::Latte);

    // if (TrafficLight::Yellow == Job::Ranger)
    // …

    return 0;
}
```

Binary literal, separator

Convenient way to express binary and separator in C++14

Binary literal, separator

- C / C++에서는 10진수 외에 8진수, 16진수를 표현할 수 있음
  - 8진수 : 061 → 49, 16진수 : 0x3A → 58
  - 하지만 2진수를 표현하는 방법은 없었음
- C++14에서 2진수를 표현할 수 있는 방법 추가! : 0b~
- 또한, 기존 C / C++에서는 큰 숫자를 읽기 힘듬
  - 예를 들어, int INT_MAX = 2147483647; (21억 4748만 3647)
- C++14에는 구분자가 추가되어 큰 숫자를 읽기 쉬워짐
  - int INT_MAX = 21'4748'3647;

Binary literal, separator: Example

C++14

```
#include <iostream>

int main()
{
    int decimal = 52;
    int octal = 064;
    int hexadecimal = 0x34;
    int binary = 0b110100;  // C++14

    int maxInt_Cpp98 = 2147483647;
    int maxInt_Cpp14 = 21'4748'3647; // C++14


    return 0;
}
```

std::string

char[], char* in C → Way to store string literal in C++

std::string

- C에서는 문자열을 나타내기 위해 char[]나 char*를 사용
  - 하지만 비교, 연결, 길이 등 문자열 관련 함수들을 사용할 때 불편함
    - 비교 : strcmp, 연결 : strcat, 길이 : strlen
- C++에서는 std::string을 사용해 문자열을 나타냄
  - 먼저, <string> 헤더 파일 포함 → #include <string>
  - std::string str = “Hello, World”;
  - 비교, 연결, 길이 등의 문자열 동작들을 편리한 방법으로 제공
    - 비교 : ==, 연결 : +, 길이 : size() 또는 length()
    - <string> 헤더 파일에서 제공하는 다른 함수들을 알고 싶다면, http://en.cppreference.com/w/cpp/string/basic_string 참고

std::string: Example

C

```c
#include <stdio.h>

int main()
{
    char str1[30] = "Hello, World";
    char* str2 = "Hello, Word";

    if (strcmp(str1, str2) == 0) printf("Same!\n");
    else printf("Different!\n");

    strcat(str1, str2);
    printf("%s\n", str1);

    printf("%d, %d\n", strlen(str1), strlen(str2));
}
```

std::string: Example

C++

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

int main()
{
    std::string str1 = "Hello, World";
    std::string str2 = "Hello, Word";

    if (str1 == str2) std::cout << "Same!" << std::endl;
    else std::cout << "Different!" << std::endl;

    str1 = str1 + str2;
    std::cout << str1 << std::endl;

    std::cout << str1.size() << ' ' << str2.length() << std::endl;
}
```