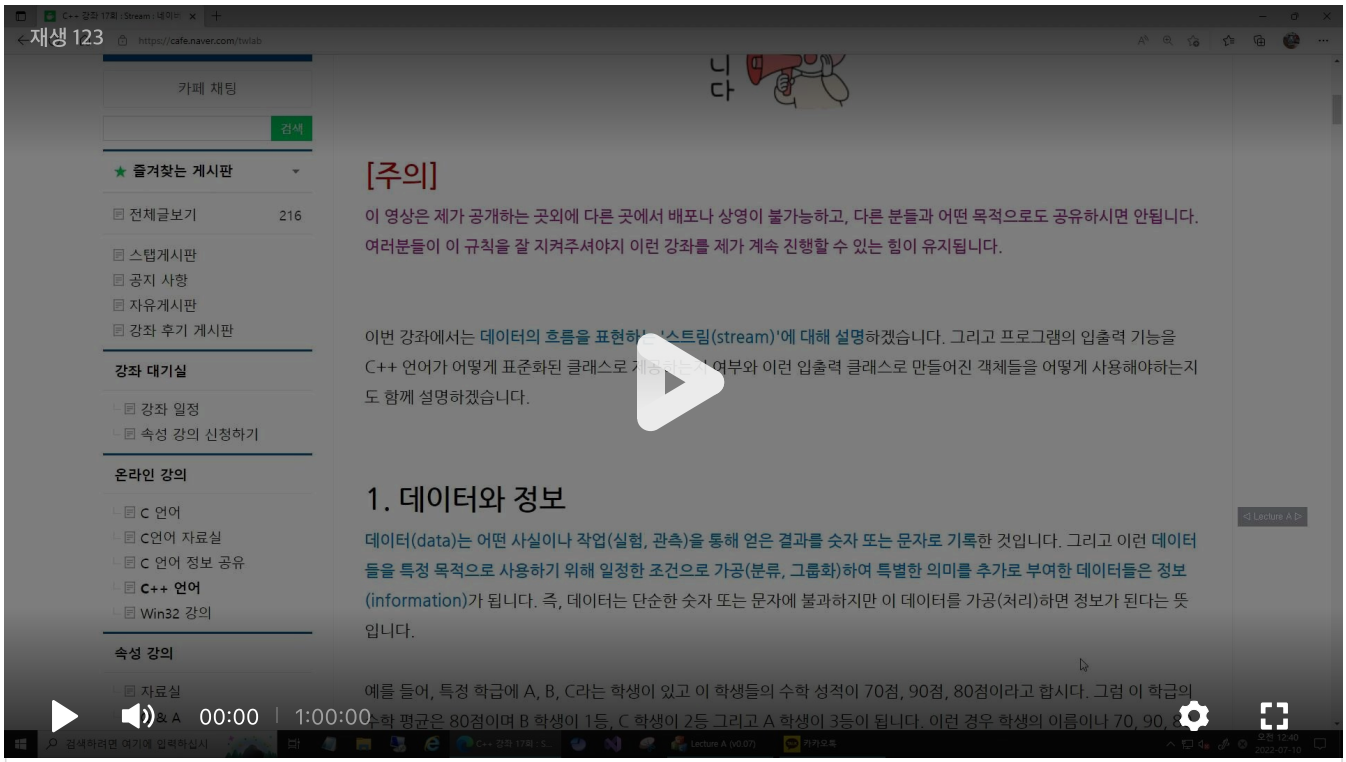
C++ 강좌 17회 : Stream

김성엽 카페매니저 + 구독 1:1 채팅
2022.07.06. 04:21 조회 304

[주의]

이 영상은 제가 공개하는 곳외에 다른 곳에서 배포나 상영이 불가능하고, 다른 분들과 어떤 목적으로도 공유하시면 안됩니다. 여러분들이 이 규칙을 잘 지켜주셔야지 이런 강좌를 제가 계속 진행할 수 있는 힘이 유지됩니다.

이번 강좌에서는 데이터의 흐름을 표현하는 '스트림(stream)'에 대해 설명하겠습니다. 그리고 프로그램의 입출력 기능을 C++ 언어가 어떻게 표준화된 클래스로 제공하는지 여부와 이런 입출력 클래스로 만들어진 객체들을 어떻게 사용해야하는지도 함께 설명하겠습니다.



C++ 강좌 17회 : Stream

네이버는 한 시간까지만 영상이 업로드 되는 관계로 끝 인사를 하지 못하고 영상이 끝납니다. 잘린 부분은 '여기까지 설명하고 이 강좌를 마칩니다.'라고 하는 말뿐이니 다른 내용이 없음을 알려드립니다.

1. 데이터와 정보

데이터(data)는 어떤 사실이나 작업(실험, 관측)은 통해 얻은 결과를 숫자 또는 문자로 기록한 것입니다. 그리고 이런 데이터들은 특정 목적으로 사용하기 위해 일정한 조건으로 가공(분류, 그룹화)하여 특별한 의미를 추가로 부여한 데이터들은 정보(information)가 됩니다. 즉, 데이터는 단순한 숫자 또는 문자에 불과하지만 이 데이터를 가공(처리)하면 정보가 된다는 뜻입니다.

예를 들어, 특정 학급에 A, B, C라는 학생이 있고 이 학생들의 수학 성적이 70점, 90점, 80점이라고 합시다. 그럼 이 학급의 수학 평균은 80점이며 B 학생이 1등, C 학생이 2등 그리고 A 학생이 3등이 됩니다. 이런 경우 학생의 이름이나 70, 90, 80과 같은 값은 '데이터'이고 이 데이터를 가공해서 얻은 수학 평균 점수나 등수는 '정보'가 된다는 뜻입니다.

2. 스트림

정보가 데이터를 가공하여 특별한 의미를 부여한 것이라면 스트림(stream)은 데이터의 흐름(방향)은 의미합니다. 즉, 특정 장치에 저장된 데이터가 다른 장치(또는 동일한 장치의 다른 영역으)로 이동하는 것을 스트림이라고 합니다. 예를 들어, 이동성이 없는 호수의 물처럼 데이터가 특정한 곳에 계속 유지될 수도 있지만, 강처럼 데이터가 다른 곳으로 이동할 수도 있는데 이렇게 데이터가 이동하는 행위를 표현한 것이 스트림입니다.

스트림에 대한 좀 더 구체적인 예를 들면 다음과 같습니다.

행위	데이터 이동 방향
키보드로 값을 입력	키보드 장치 ▶ 메모리 장치
값을 화면에 출력	메모리 장치 ▶ 출력 장치
파일에 값을 저장	메모리 장치 ▶ 디스크 장치
파일에서 값을 읽기	디스크 장치 ▶ 메모리 장치
인터넷으로 영화 보기	서버 컴퓨터의 영화 데이터 ▶ 개인 컴퓨터의 메모리 장치

결국 장치 간에 데이터를 입력 또는 출력하는 모든 행위가 스트림입니다.

3. C 언어와 입출력 함수

C 언어는 장치별 데이터 입출력 기능은 함수로 제공합니다. 그런데 장치에 따른 입출력 함수 이름이 모두 다르고 사용 방법도 달라 새로운 장치를 사용할 때마다 프로그래머는 해당 장치가 제공하는 함수들을 다시 공부해야 합니다.

예를 들어, C 언어는 키보드로 값을 입력받을 때 getc, gets, scanf와 같은 함수를 사용합니다. 그리고 파일에서 데이터를 입력받을 때는 read, fread, fscanf와 같은 함수를 사용합니다. 그리고 네트워크 장치로부터 데이터를 입력받을 때는 recv, recvfrom과 같은 함수를 사용합니다.

이렇게 장치마다 입출력을 위해 다른 이름의 함수를 사용하는 이유는 C 언어가 '네임스페이스', '클래스' 또는 '함수의 오버로딩(오버라이딩)' 같은 문법이 없어서 **유사한 작업은 하는 함수도 서로 다른 이름으로 만듣기 때문**입니다. 그래서 C 언어에서는 새로운 장치가 추가되면 해당 장치의 특성을 반영해 새로운 이름의 함수가 추가로 제공됩니다.

그런데 이렇게 추가되는 함수들이 단순히 이름만 다른 것이 아니라 **장치에 따라 제공되는 함수의 개수나 함수의 형식 그리고 함수간 호춛 순서까지도 다르기 때문에 입춛력 작업에 대한 표준 형식은 만듣기 어려웠습니다.** 그래서 C 언어를 사용하는 개발자들은 새로운 장치를 사용하게 될 때마다 해당 장치가 제공하는 함수의 종듀와 특성을 파악하기 위해 매번 다시 공부를 해야 하는 불편함이 있었습니다.

4. 입춛력 스트림

그런데 C++ 언어는 유사한 작업을 하는 함수들은 동일한 이름의 함수로 만들수 있습니다. 그래서 장치가 다르더라도 일정한 (표준적인) 형식의 코드를 사용해서 입출력 작업을 구성하는 것이 가능합니다. 그리고 **장치별로 입춛력 기능은 제공하는 클래스릁 만듣고 이 클래스듣은 다형성 구조로 만듣면, 새로운 장치릁 사용하더라도 개받자가 해당 장치에 대한 특별한 이해 없이도 표준적인 클래스릁 사용해서 원하는 입춛력 작업은 할 수** 있습니다.

그래서 **장치에 상관없이 동일한 형식으로 입춛력은 제공하기 위해 C++ 언어는 '스트림 클래스 계층 구조(Stream Class Hierarchy)'릁 제공**합니다. 하지만 스트림 클래스 계층 구조가 다형성 구조로 되어 있고 클래스 종듀도 많아서 초보 개발자들이 스트림과 관련된 전체 클래스를 이해하는 것은 어려운 일입니다. 따라서 이번 강좌에서는 스트림 클래스에서 가장 많이 사용되는 istream 클래스와 ostream 클래스를 가지고 설명하겠습니다.

먼저 이 두 클래스를 사용하려면 아래와 같이 **iostream 헤더 파일은 include** 해야 합니다. 그런데 이 헤더 파일은 .h 확장자를 생략한 것이 아니라 확장자가 없는 헤더 파일이기 때문에 iostream.h 라고 적으면 오듀가 발생합니다. 이처럼 C++ 언어에서 제공하는 표준 헤더 파일에는 .h 확장자가 없는 경우도 많기 때문에 헤더 파일을 사용할 때 주의해야 합니다.

```
#include <iostream>   // 표준 입출력 스트림 객체를 사용하기 위해
```

◆ istream - 표준 입력 스트림은 처리하는 클래스

C 언어와 표준 입력을 사용하는 코드를 비교하기 위해 C 언어로 정숫값 한 개와 실숫값 한 개를 입력받는 예제 코드를 작성해보면 다음과 같습니다. C 언어는 원하는 형태의 값을 입력받기 위해 개발자가 scanf_s 함수를 사용할 때 '%d', '%f'와 같은 형식 지정자를 사용해야 합니다. 즉, **표준 입력 작업은 위해 개받자가 입력되는 값의 형식에 따라 어떤 표현은 사용해야 하는지 먼저 공부**해야 한다는 뜻입니다. 그리고 scanf_s 함수가 입력받은 값을 main 함수에 선언된 변수에 저장하기 위해 주소 개념을 사용하기 때문에 **& 연산자릁 함께 사용**해야 한다는 점도 초보자들에게는 부담스러운 일입니다.

```c
#include <stdio.h>    // scanf_s 함수를 사용하기 위해

int main()
{
    int i_data = 0;        // 0으로 초기화
    float f_data = 0.0f;   // 0으로 초기화

    scanf_s("%d", &i_data); // 사용자에게 정숫값을 입력 받음
    scanf_s("%f", &f_data); // 사용자에게 실숫값을 입력 받음

    return 0;
}
```

반면 C++ 언어에서는 **표준 입력은 사용할 때 istream 클래스로 만든어진 cin 객체(console input)를 사용**하면 됩니다. cin 객체는 개발자가 선언하는 것이 아니라 이미 istream cin;처럼 선언되어 있기 때문에 iostream 헤더 파일을 include 했다면 그냥 사용하면 됩니다. 다만 **istream 클래스가 std 네임스페이스를 사용하기 때문에 cin 객체를 사용할 때는 std::cin 이라고 사용**해야 합니다. 아래의 코드는 정숫값 한 개와 실숫값 한 개를 입력받는 예제를 cin 객체를 사용해서 작업한 예제입니다.

```c
#include <iostream>  // 표준 입출력 스트림 객체를 사용하기 위해

int main()
{
    int i_data = 0;        // 0으로 초기화
    float f_data = 0.0f;   // 0으로 초기화

    std::cin >> i_data;    // 사용자에게 정숫값을 입력 받음
    std::cin >> f_data;    // 사용자에게 정숫값을 입력 받음

    return 0;
}
```

위 소스에서는 **cin 객체를 사용해서 정숫값을 입력받든, 실숫값을 입력받든 동일하게 '>>' 연산자만 사용해서 작업**이 가능합니다. 그리고 이때 C++ 언어는 데이터의 흐름을 함수 이름으로 표현하는 것보다 데이터가 이동하는 방향(장치▶변수)을 표현하는 '>>' 연산자를 재정의(연산자 오버로딩)해서 사용해서 **원하는 기능은 사용하기 위해 함수의 이름은 외워야 하는 분편함도 줄였습니다.** 그리고 이렇게 연산자 오버로딩을 사용하면 매개 변수로 전달되는 변수의 자료형을 구분할 수 있기 때문에 **C 언어처럼 '%d', '%f' 같은 형식 지정자를 추가로 사용할 필요도 없습니다.** 그리고 이 표현에서는 주소 개념도 사용되지 않기 때문에 **& 연사자도 추가로 사용되지 않습니다.**

만약, 연산자 오버로딩이 헷갈리신다면 아래에 링크한 글을 복습하시기 바랍니다.

C++ 강좌 09회 : 연산자 오버로딩
대한민국 모임의 시작, 네이버 카페
cafe.naver.com

결국 C++ 언어는 객체와 연산자 오버로딩 기술을 사용해서 **개발자가 표준 입력은 사용할 때 추가적인 공부없이 cin 객체와 '>>' 연산자만 사용해서 원하는 변수에 원하는 형식의 값은 입력받은 수 있게 만들어졌습니다.** 그리고 std 네임스페이스를 계속 사용하는 것이 불편하다면 아래와 같이 using namespace std; 코드를 추가해서 std:: 표현을 생략하는 것도 가능합니다.

```
#include <iostream>  // 표준 입출력 스트림 객체를 사용하기 위해

using namespace std;    // std::를 생략하기 위해서

int main()
{
    int i_data = 0;        // 0으로 초기화
    float f_data = 0.0f;   // 0으로 초기화

    cin >> i_data;    // 사용자에게 정숫값을 입력 받음
    cin >> f_data;    // 사용자에게 정숫값을 입력 받음

    return 0;
}
```

그리고 C 언어에서 정숫값과 실숫값을 연속으로 입력받을 때, scanf_s 함수를 두 번 연속으로 사용하지 않고 아래와 같이 한 번에 입력을 받듯이 C++ 언어도 **cin 객체를 사용할 때 '>>' 연속으로 사용하면 한 번에 표현하는 것이 가능**합니다.

```
scanf_s("%d %f", &i_data, &f_data);    // C 언어 표현
cin >> i_data >> f_data;                  // C++ 언어 표현
```

◆ ostream - 표준 출력 스트림을 처리하는 클래스

C 언어와 표준 출력을 사용하는 코드를 비교하기 위해 C 언어로 정숫값 한 개와 실숫값 한 개를 출력하는 예제 코드를 작성해 보면 다음과 같습니다. 표준 출력에서도 C 언어는 원하는 형태의 값을 출력하기 위해 개발자가 printf 함수를 사용할 때 '%d', '%f'와 같은 형식 지정자를 사용해야 합니다.

```
#include <stdio.h>  // printf 함수를 사용하기 위해!

int main()
{
    int i_data = 5;
    float f_data = 1.2f;

    printf("i_data = %d, f_data = %f\n", i_data, f_data); // 정숫값과 실숫값을 출력

    return 0;
}
```

위 예제는 아래와 같이 출력됩니다.

```
i_data = 5, f_data = 1.200000

이 창을 닫으려면 아무 키나 누르세요...
```

반면 C++ 언어는 표준 출력은 사용할 때 ostream 클래스로 만들어진 cout 객체(console output)를 사용하면 됩니다. cout 객체도 개발자가 선언하는 것이 아니라 이미 ostream cout;처럼 선언되어 있기 때문에 그냥 사용하면 됩니다. 위에서 소개한 정숫값과 실숫값을 출력하는 C 언어 예제를 cout 객체를 사용하도록 재구성하면 다음과 같습니다. 여기서 cout 객체에 사용된 'endl'은 줄바꿈 작업은 할 때 사용하는 것이며 이 값 대신에 '₩n'을 사용해도 동일한 결과가 나옵니다. cout 객체도 데이터 출력은 의미하는 함수 이름 대신 '<<' 연산자를 재정의(연산자 오버로딩)해서 사용하고 있으며 연속으로 나열하는 것도 가능합니다.

```
#include <iostream>  // 표준 입출력 스트림 객체를 사용하기 위해

using namespace std;    // std::를 생략하기 위해서

int main()
{
    int i_data = 5;
    float f_data = 1.2f;

    cout << i_data << ", "  << f_data << endl;  // 정숫값과 실숫값을 출력

    return 0;
}
```

위 예제는 아래와 같이 출력됩니다. 위 코드에서 i_data 변수와 f_data 변수의 값만 출력하게 하면 '51.2'처럼 두 값이 붙어서 출력됩니다. 그래서 두 값을 구분하기 위해 ", " 문자열을 출력하는 코드를 추가했습니다.

```
5, 1.2

이 창을 닫으려면 아무 키나 누르세요...
```

그래서 스트림 클래스와 cin, cout 객체를 배우게 되면 간단한 실습 예제를 만들때 C 언어에서 사용하는 scanf_s, printf 같은 함수는 더 이상 사용하지 않게 됩니다.

5. 표준 입출력 스트림 클래스는 정말 편리한가?

cin, cout 객체를 사용하는 코드가 더 편리한지 확인하려면 동일한 작업을 scanf_s, printf 함수를 사용하는 코드와 비교해 보면 됩니다. 예를 들어, 정숫값 한 개를 입력받아 출력하는 예제를 C 언어 형식의 예제와 C++ 언어 형식의 예제를 만들어 보면 다음과 같습니다. 먼저 C 언어로 작성한 예제입니다.

```c
#include <stdio.h>   // scanf_s printf 함수를 사용하기 위해

int main()
{
    int data = 0;       // 0으로 초기화

    scanf_s("%d", &data); // 사용자에게 정숫값을 입력 받음
    printf("%d\n", data); // 입력받은 정숫값을 출력 함

    return 0;
}
```

위 예제는 아래와 같이 출력됩니다. 5는 사용자가 입력한 값입니다.

```
5
5

이 창을 닫으려면 아무 키나 누르세요...
```

위 예제를 C++ 언어의 cin, cout 객체를 사용해서 재구성한 예제입니다.

```c
#include <iostream>  // 표준 입출력 스트림 객체를 사용하기 위해
using namespace std;    // std::를 생략하기 위해서

int main()
{
    int data = 0;       // 0으로 초기화

    cin >> data;          // 사용자에게 정숫값을 입력 받음
    cout << data << endl; // 입력받은 정숫값을 출력 함

    return 0;
}
```

두 예제 코드를 보면 **C++ 형식의 예제가 '형식 지정자'도 사용하지 않고 작업에 해당하는 함수 이름 대신 '<<' 을 사용하기 때문에 좀더 단순**한건 사실입니다. 그래서 C++ 언어를 공부하거나 공부한 사람들은 C++ 언어에서 scanf_s 함수나 printf 함수를 사용하면 큰 문제가 있는 것처럼 이야기 하는 사람들도 있습니다.

하지만 **기본적인 기능만 사용하는 경우에는 C++ 언어로 작성한 코드가 더 단순**해 보이겠지만 **예외처리나 출력 형식은 변경하는 코드가 추가되면 과연 더 편리한 것이 맞는지 의문**이 생길 것입니다. 예를 들어, 위 예제는 정숫값을 입력받는 작업인데 개발자가 정숫값이 아닌 문자나 문자열을 입력하면 정상적으로 값이 출력되지 않습니다. 위 예제 코드를 실행해서 아래와 같이 'aaa'라고 입력하면 0이라고 출력됩니다.

```
aaa
0

이 창을 닫으려면 아무 키나 누르세요...
```

그래서 **사용자가 정숫값이 아닌 다른 형식의 값을 입력하면 잘못 입력했음을 알리고 다시 입력하게 코드를 수정**해보겠습니다. 먼저 C 언어는 아래와 같이 코드를 구성하면 됩니다. 아래의 코드에서는 scanf_s 함수에 한 개의 형식 지정자('%d')만 사용되었기 때문에 **정상적으로 값이 입력되면 1이 반환**됩니다. 하지만 **원하는 형식의 값이 입력되지 않으면 해당 형식 지정자의 사용 횟수가 제외된 상태로 값이 반환되기 때문에 0이 반환**됩니다. 그래서 scanf_s 함수의 반환값을 체크해서 사용자가 값을 잘못 입력했으면 그 사실을 화면에 출력하고 정상적인 값을 입력할 때까지 해당 작업을 반복하게 구성했습니다.

```c
#include <stdio.h>   // scanf_s printf 함수를 사용하기 위해

int main()
{
    int data = 0;       // 0으로 초기화

    printf("정숫값을 입력하세요: ");
    while (!scanf_s("%d", &data)) { // 사용자에게 정숫값을 입력 받음
        rewind(stdin);  // 표준 입력 버퍼에 입력된 값을 무시함
        printf("잘못된 값을 입력했습니다. 다시 입력하세요: ");
    }
    printf("입력된 정숫값: %d\n", data); // 입력은 정숫값을 출력 함

    return 0;
}
```

위 예제는 아래와 같이 출력됩니다. 'aaa'와 't'를 입력하면 오류처리하고 정상적으로 7을 입력하면 종료됩니다.

```
정숫값을 입력하세요: aaa
잘못된 값을 입력했습니다. 다시 입력하세요: t
잘못된 값을 입력했습니다. 다시 입력하세요: 7
입력된 정숫값: 7

이 창을 닫으려면 아무 키나 누르세요...
```

이제 위 기능을 C++ 언어의 cin, cout 객체를 사용해서 수정해 보면 다음과 같습니다. **cin 객체는 '<<' 연산자 뒤에 사용된 자료형과 일치하는 형식의 값이 입력되었는지 cin 객체의 fail 멤버 함수를 사용해서 확인이 가능**합니다. fail 함수가 1을 반환하면 원하지 않는 형식의 값이 입력되었다는 뜻이기 때문에 이 값이 0이 반환될 때까지 반복하면서 사용자에게 값을 입력받도록 코드를 구성한 것입니다. 그리고 **값이 잘못 입력되었으면 오류 상태를 기억하는 내부 변수를 초기화하고 표준 입력에 입력되어 있는 값도 무효화** 시켜야 합니다. 그래서 cin 객체의 clear, ignore 멤버 함수를 사용했습니다.

```cpp
#include <iostream>      // 표준 입출력 스트림 객체를 사용하기 위해
using namespace std;     // std::를 생략하기 위해서

int main()
{
    int data = 0;

    cout << "정숫값을 입력하세요: ";
    cin >> data;     // 정수값을 입력 받는다.

    // 사용자가 원하는 형식으로 값을 입력했는지 체크한다.
    while (cin.fail()) {
        cin.clear();              // 내부적으로 설정된 오류 상태 값을 모두 초기화한다.
        cin.ignore(100, '\n');    // 100개의 입력 값을 무효화 시킨다.
                                  // 하지만 중간에 '\n' 문자를 만나면 중단한다.
        cout << "잘못된 값을 입력했습니다. 다시 입력하세요: ";
        cin >> data;    // 정수값을 다시 입력 받는다.
    }

    cout << "입력된 정숫값: " << data << endl;  // 입력 받은 값을 출력한다.
    return 0;
}
```

이제 C 언어로 작성한 코드와 C++ 언어로 작성한 코드를 비교해보면 C++ 언어로 작성한 스트림 기반 표준 입출력 코드가 더 간단하고 편한지 의문이 생길 것입니다. 왜냐하면 cin 객체에서 제공하는 fail, clear, ignore 함수의 존재와 사용 순서 그리고 사용 형식은 알지 못한다면 구현이 불가능합니다. 즉, cin 객체를 사용하는 장점 중에 장치의 특성은 이해하지 않고도 사용할 수 있다는 것은 이제 의미가 없습니다.

그리고 cout 객체도 값을 그대로 출력할 때는 아래와 같이 printf 함수를 사용하는 것보다 cout 객체를 사용하는 것이 더 편해 보일 것입니다.

```cpp
int data = 7;
printf("data: %d\n", data);  // C 언어 형식
cout << "data: " << data << endl;  // C++ 언어 형식
```

그런데 위 코드에서 값을 출력하는 형식을 7이 아닌 [7] 처럼 data 변수의 값은 다섯 칸의 공간은 확보하고 그 내부에 값은 출력하게 코드를 수정해 보면 다음과 같습니다. '[ ]' 기호를 사용한 것은 중간에 공백이 있음을 보여주기 위한 것일뿐 다른 의미는 없습니다. C 언어는 다섯 칸의 공간을 확보하고 출력하려면 %d를 %5d로 수정하면 됩니다. 하지만 C++ 언어의 cout 객체는 width 멤버 함수를 호출해야 하기 때문에 개발자가 width 함수의 존재와 매개 변수의 의미를 추가로 알아야 합니다.

```
int data = 7;
printf("data: [%5d]\n", data);  // C 언어 형식

// C++ 언어 형식
cout.width(5);  // 값을 출력할 때 공간을 다섯 칸  확보하도록 지정
cout << "data: " << data << endl;
```

그리고 이번엔 다섯 칸의 공간을 확보하는데 숫자가 부족한 부분에 공백 문자가 아닌 '0'문자가 출력되게 하려면 아래와 같이 코드를 수정하면 됩니다. 예를 들어, [7]이 아닌 [00007]으로 출력되게 하려는 것입니다. 이번에도 C 언어는 형식지정자를 %5d에서 %05d로만 수정하면 작업이 끝납니다. 하지만 C++ 언어의 cout 객체는 부족한 부분은 공백 문자가 아닌 '0' 문자로 변경하기 위해 cout 객체의 fill 멤버 함수른 호춛하여 '0' 문자로 변경할 것은 지정해야 합니다.

```
int data = 7;
printf("data: [%05d]\n", data);  // C 언어 형식

// C++ 언어 형식
cout.fill('0');   //  빈 공간을 '0'으로 채운다.
cout.width(5);  // 값을 출력할 때 공간을 다섯 칸  확보하도록 지정
cout << "data: " << data << endl;
```

마지막으로 좀더 극단적인 예시를 들어보겠습니다. 아래의 코드는 성적처리 프로그램을 만들 때 학생의 이름과 해당 학생의 국어, 영어, 수학 점수 그리고 이 점수듣의 총점, 평균은 춛력하는 코드입니다. 이 코드는 C 언어에서 printf 함수를 사용한 예시입니다.

```
printf(" %-8s%5d%6d%6d%7d%6.0f\n", name, kor, eng, math, total, average);
```

그런데 위 코드를 C++ 언어의 cout 객체를 사용해서 재구성하면 다음과 같습니다. 개발자에 따라 호불호가 있을지 모르겠지만 기본 춛력 형식이 아닌 조금만 다른 형태가 적용되면 생각보다 코드가 많이 복잡해지고 앋아야 할 것이 많아진다는 것을 느끼게 될 것입니다. 개인적인 의견이지만 스트림이라는 개념은 정말 진보적이고 좋은 개념이 맞습니다. 하지만 스트림도 상황에 따라 좋은 것이지 무조건 좋다고 보기는 어렵기 때문에 개발자가 잘 판단해서 스트림을 사용여부를 결정하면 됩니다.

```
cout.flags(ios::left);    // 왼쪽 정렬을 사용한다.
cout.width(8);            // 8칸을 확보!
cout << name;
cout.flags(ios::right);   // 오른쪽 정렬을 사용한다.
cout.width(5);            // 5칸을 확보!
cout << kor;
cout.width(6);            // 6칸을 확보!
cout << eng;
cout.width(6);            // 6칸을 확보!
cout << math;
cout.width(7);            // 7칸을 확보!
cout << total;
cout.width(6);            // 6칸을 확보!
cout << (int)m_average;
```

위 코드에 사용된 함수 중에 flags 함수나 진법 변환과 관련된 내용은 아래의 링크를 참고하세요.

표준 출력 스트림의 양식 변경하기
: C++ 언어 관련 전체 목차 http://blog.naver.com/tipsware/2210285599003이 강좌에서는 표준 출력 스…
blog.naver.com

6. C++ 언어에서 C 언어 표준 함수를 사용하면 문제가 되는가?

C++ 언어는 C 언어에서 확장된 프로그래밍 언어입니다. 즉, C 언어의 특성을 가지고 있는 프로그래밍 언어입니다. 따라서 많은 문법이 C 언어 표준을 따르고 있기 때문에 C 언어 표준 함수를 사용해도 아무런 문제가 발생하지 않습니다. 그런데도 C++ 언어를 사용하는 개발자들은 C++ 언어로 작성된 코드에서 C 언어 표준 함수를 사용하면 마치 문제가 있는 것처럼 지적하는 경우가 많습니다.

하지만 스트림 클래스는 개발자가 더 편하게 프로그래밍을 할 수 있게 만들어진 것일뿐 더 효율적이지도 않고 더 강력한 표현도 아닙니다. 그냥 더 편하게 사용할 수 있는 표현일 뿐인데 이 마저도 전문적으로 사용하게 되면 오히려 더 사용이 불편해지는 경향이 있습니다. 그래서 이런 점을 더 부각하기 위해 C++ 언어 강좌임에도 불구하고 대부분 예제에서 입출력 기능을 스트림이 아닌 printf 함수와 scanf_s 함수로 설명한 것입니다.

프로그래밍 언어는 프로그램을 개발하는데 사용되는 표현법일 뿐입니다. 철학도 종교도 아니기 때문에 개발자가 원하는 시점에 더 효율적으로 사용할 수 있는 표현을 사용하면 됩니다.

김성엽 M 작성자

ㅎㅎㅎ 맞아요!

2022.09.13. 21:10 답글쓰기

카일 3

강의 잘 들었습니다. 다시 들으니 전에는 안 들렸던 부분이 들립니다^^

2022.11.26. 16:23 답글쓰기

김성엽 M 작성자

그래서 C++ 강좌도 시간 날때마다 반복해서 보는것이 좋습니다 ㅎ

2022.11.26. 16:28 답글쓰기

카일 3

김성엽 네~ ^^

2022.11.26. 16:29 답글쓰기

조민희 3

C가 얼마나 잘 만들어졌는지 다시금 느끼게 되네요 ㅎㅎ

2022.12.18. 23:14 답글쓰기

김성엽 M 작성자

그래서 나중에 하다보면 C 언어로 돌아가는 경우가 많습니다. 그런데 그 C 언어로 만든 코드가 C++를 많이 닮아있다는 것을 느끼게 될 것입니다. 결국 C++가 가지고 있는 철학만 잘 활용하면 됩니다.

2022.12.19. 00:39 답글쓰기

임레이 3

```
#include <iostream>
using namespace std;
int main() {

char c_num = 65;
int i_num = 65;

printf("%d\n", c_num); // 65
cout << c_num << endl; // A

printf("%c\n", i_num); // A
cout << i_num << endl; // 65

return 0;
}
```

printf는 형식 지정자를 통해서 출력 방식을 정할 수 있었는데, ostream 클래스는 출력하는 변수의 자료형에 맞게 출력 되는 건가요?
char형은 1바이트 내의 부호 있는 숫자를 저장하는 자료형이라고 알고 있는데, 그냥 c++에서는 char형을 문자 형식으로 출력 하는 게 기본 값이라서 그런가요?

이해가 되지 않는 부분은 c_num은 65라는 10진수 숫자를 저장하고 있는데, cout << c_num 을 하면 65가 아닌 A가 출력 되는 점입니다.

char c_num = 7; cout << c_num; 에선 비프음이 들린다는 것 또한 문자 형식으로 출력하고 있다는 거겠죠?

2023.03.30. 02:35 답글쓰기

김성엽 M 작성자

네 << 연산자에 대한 연산자 오버로딩이 되어 있어서 그런것입니다. 연산자 오버로딩은 해당 변수의 자료형을 가지고 판단하기 때문에 c_num이 char 자료형으로 선언되어 있어서 그렇게 되는 것입니다. 만약 c_num 변수의 값을 int 로 출력하고 싶다면 (short)c_num 또는 (int)c_num 처럼 형변환을 앞에 해주면 정수값으로 출력됩니다.

2023.03.30. 12:24 답글쓰기

dh221009

댓글을 남겨보세요

등록