

C++ Programming

2nd Study: From C to C++ (2/4)

- Range-based for
- malloc, free → new, delete
- NULL → nullptr
- static_cast

C++ Korea 옥찬호 (utilForever@gmail.com)



Range-based for

Python-like “for” syntax

Range-based for

- 반복문 중 가장 많이 사용하는 for 문은
지정된 횟수만큼 반복하는 작업에도 사용하지만
배열이나 각종 컨테이너에 있는 각 요소들에
무언가를 수행하기 위해 순회하는 목적으로도 많이 사용함
- 하지만, 조건문에 배열이나 컨테이너에
몇 개의 요소가 들어있는지 명시해야된다는 단점이 존재!
- 요소들의 개수에 상관없이 반복문을 수행할 수 없을까?
→ 범위 기반 for문의 등장!

Range-based for: Example

C / C++98

```
#include <iostream>

int main()
{
    int arr[] = { 1, 2, 3, 4, 5 };

    for (int i = 0; i < 5; ++i)
        std::cout << arr[i] << std::endl;

    return 0;
}
```

C++11

```
#include <iostream>

int main()
{
    int arr[] = { 1, 2, 3, 4, 5 };

    for (auto& i : arr)
        std::cout << i << std::endl;

    return 0;
}
```

Range-based for: Example

Python

```
def mean(seq):  
    n = 0.0  
  
    for x in seq:  
        n += x  
  
    return n / len(seq)
```

C++11

```
auto mean(const Sequence& seq)  
{  
    auto n = 0.0;  
  
    for (auto& x : seq)  
        n += x;  
  
    return n / seq.size();  
}
```

The background is a solid blue color. It is decorated with several squares of various sizes and positions. Some squares are solid blue, while others are white outlines. They are scattered across the slide, with some overlapping the text.

malloc, free → new, delete

Dynamic allocation syntax in C++

malloc, free → new, delete

- C에서는 동적 메모리 할당을 위해 malloc, calloc 등을 사용
또한 동적 메모리 반환을 위해 free를 사용
- C++에서는 malloc, calloc 등을 대신해 new를 사용하고
free를 대신해 delete를 사용 → 쉬운 문법을 제공
- 주의해야 할 점, new는 delete로, new[]는 delete[]로!
 - 단일 객체에 대한 동적 메모리 할당은 new, 반환은 delete!
 - 배열 객체에 대한 동적 메모리 할당은 new[], 반환은 delete[]!

malloc, free → new, delete: Example

C

```
int main()
{
    int i;
    int** arr = (int**)malloc(sizeof(int*) * 5);
    for (i = 0; i < 5; ++i)
        arr[i] = (int*)malloc(sizeof(int) * 5);

    // ...

    for (i = 0; i < 5; ++i)
        free(arr[i]);
    free(arr);
    arr = NULL;

    return 0;
}
```


malloc, free → new, delete: Example

C++

```
int main()
{
    int** arr = new int*[5];
    for (int i = 0; i < 5; ++i)
        arr[i] = new int[5];

    // ...

    for (int i = 0; i < 5; ++i)
        delete[] arr[i];
    delete[] arr;
    arr = NULL;

    return 0;
}
```

malloc, free → new, delete: Example


C++

```
#include <iostream>

int main()
{
    int* p1 = new int;
    delete p1;
    p1 = NULL;

    int* p2 = new int[10];
    delete[] p2;
    p2 = NULL;

    return 0;
}
```

The background is a solid blue color. It is decorated with several squares of various sizes and positions. Some squares are solid blue, while others are white outlines. They are scattered across the slide, with a higher concentration in the top-left and bottom-right areas.

NULL → nullptr

Real null pointer in C++11

NULL → nullptr

- C에서 널 포인터를 나타내기 위해 NULL이나 상수 0을 사용
- 하지만 NULL은 진짜 널 포인터를 가리킬까?
 - #define NULL 0, 따라서 NULL = 0
- NULL이나 상수 0을 사용해 함수에 인자로 넘기는 경우, int 타입으로 추론되는 문제 발생 → nullptr 키워드의 등장!
- 포인터 변수와 마찬가지로, 크기는 4바이트
- 주의할 점, nullptr은 타입이 아니다!
 - nullptr은 널 포인터 리터럴(Literal), 실제 타입은 std::nullptr_t!

NULL → nullptr: Example

C / C++98

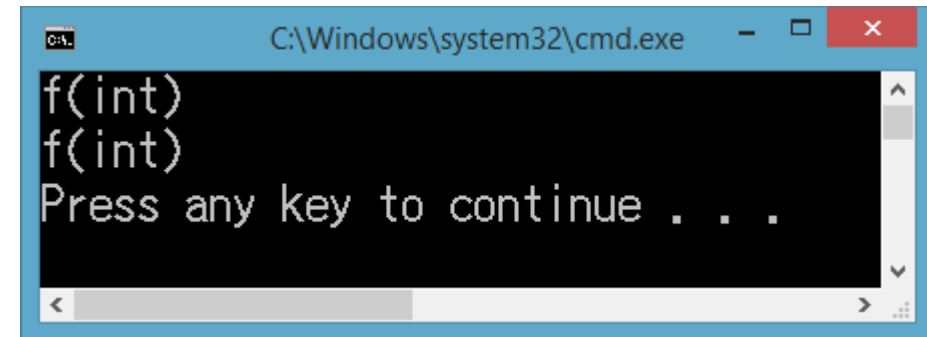
```
#include <iostream>

void f(int a) { std::cout
<< "f(int)" << std::endl; }

void f(int* p) { std::cout
<< "f(int*)" << std::endl; }

int main()
{
    f(0);
    f(NULL);

    return 0;
}
```



NULL → nullptr: Example

C++11

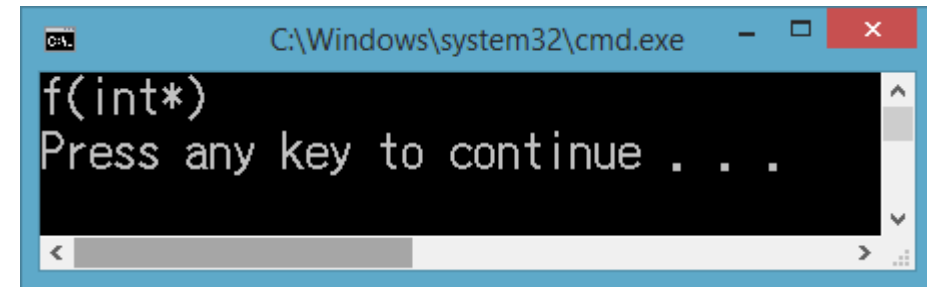
```
#include <iostream>

void f(int a) { std::cout
<< "f(int)" << std::endl; }

void f(int* p) { std::cout
<< "f(int*)" << std::endl; }

int main()
{
    f(nullptr);

    return 0;
}
```



NULL → nullptr: Example

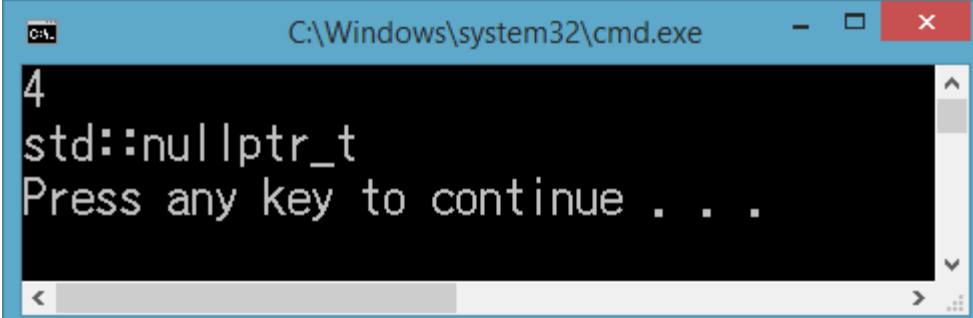
C++11

```
#include <iostream>

int main()
{
    std::cout << sizeof(nullptr)
               << std::endl;

    std::cout << typeid(nullptr)
               .name() << std::endl;

    return 0;
}
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The output of the program is displayed as follows: the number "4" on the first line, "std::nullptr_t" on the second line, and "Press any key to continue . . ." on the third line. The window includes standard Windows window controls (minimize, maximize, close) in the top right corner.

NULL → nullptr: Example

C++98

```
#include <iostream>

int main()
{
    int* p1 = new int;
    delete p1;
    p1 = NULL;

    int* p2 = new int[10];
    delete[] p2;
    p2 = NULL;

    return 0;
}
```

C++11

```
#include <iostream>

int main()
{
    int* p1 = new int;
    delete p1;
    p1 = nullptr;

    int* p2 = new int[10];
    delete[] p2;
    p2 = nullptr;

    return 0;
}
```


static_cast

Explicit type conversion in C++

static_cast

- C에서는 ()를 통해 명시적(Explicit) 캐스팅을 할 수 있음
- C의 명시적 캐스팅은 개발자의 실수를 그대로 용납하기 때문에 차후에 오류가 생길 가능성이 있음
- C++에서는 좀 더 명시적인 캐스팅 방법들을 통해 개발자가 캐스팅의 목적을 명확하게 명시함으로써 개발자의 의도를 컴파일러에게 전달 → 오류를 방지함
 - static_cast, dynamic_cast, const_cast, reinterpret_cast를 사용
 - 캐스트-이름<타입>(표현식);

static_cast

- static_cast : const를 제외한 모든 명확한 타입 변환에 사용
 - http://en.cppreference.com/w/cpp/language/static_cast
- dynamic_cast : 기본 타입에 대한 포인터나 참조자를 파생 타입에 대한 포인터나 참조자로 안전하게 변환
 - http://en.cppreference.com/w/cpp/language/dynamic_cast
- const_cast : const 객체를 const가 아닌 타입으로 변환
 - http://en.cppreference.com/w/cpp/language/const_cast
- reinterpret_cast : 비트 구성 형식을 저수준에서 재해석
 - http://en.cppreference.com/w/cpp/language/reinterpret_cast

static_cast: Example

C

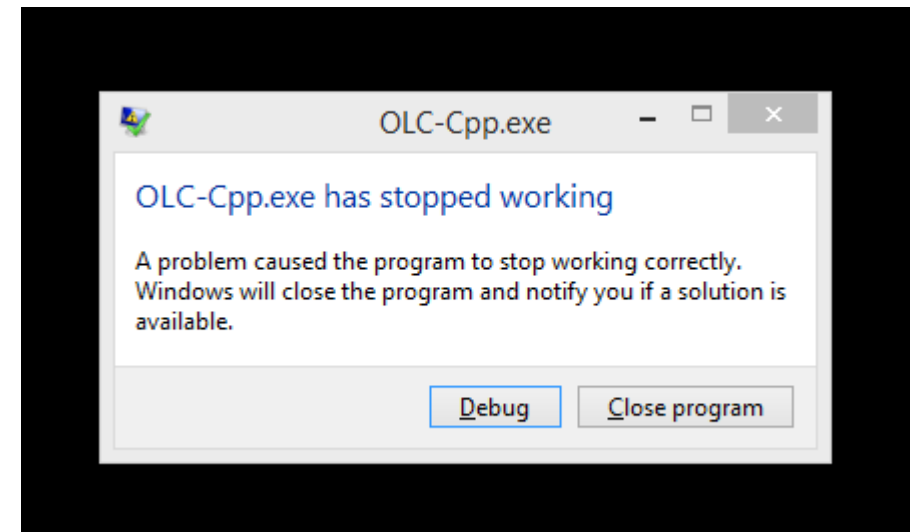
```
#include <iostream>

int main()
{
    char* str = "Hello, World";

    int* pi = (int*)str;
    char* pc = (char*)pi;

    std::cout << pc << std::endl;

    return 0;
}
```



static_cast: Example

C++





```
#include <iostream>

int main()
{
    char* str = "Hello, World";

    int* pi1 = static_cast<int*>(str);
    char* pc1 = static_cast<char*>(*pi1);

    std::cout << pc1 << std::endl;

    return 0;
}
```

	Code	Description ▾
		IntelliSense: invalid type conversion
		IntelliSense: invalid type conversion
	C2440	'static_cast': cannot convert from 'int' to 'char *'
	C2440	'static_cast': cannot convert from 'char *' to 'int *'