Day-25

Non-allocating Placement Allocation, Non-throwing Allocation

Non-Allocating Placement(비할당 배치)

- 비할당 배치 (ptr)type을 사용하면 개별 객체의 메모리 위치(이전 할당 위치)를 명시적으로 지정할 수 있음

```
// Stack Memory
char buffer[8];
int* x = new (buffer) int;
short* y = new (x+1) short[2];
// x,y 를 할당 해제할 필요가 없음
```

```
// Heap Memory
unsigned* buffer 2 = new unsigned[2];
double* z = new (buffer2) double;
delete[] buffer2; // ok
// delete[] z; // ok, 하지만 좋지 않은 행동
```

Non-Allocating Placement and Objects

- 분명하지 않은 객체가 아닌 객체의 배치 할당은 런타임이 객체가 범위를 벗어난 경우를 감지할 수 없으므로 객체 소멸자를 명시적으로 호출해야 함

```
struct A{
    ~() { cout << "destructor"; }
};

char buffer[10];
```

```
auto x = new (buffer) A();
// delete x; // 런타임 에러, 'x'가 유효한 힙 메모리 포인터가 아님
x->~A(); // print "destructor"
```

Non-Throwing Allocation

- new 연산자는 std::nothrow 객체를 전달하여 Non-Throwing Allocation을 허용함
- 메모리 할당이 실패하면 std::bad alloc 예외를 던지는 대신 NULL 포인터를 반환

```
int* arr = new (std::nothrow) int[very_large_size];
```

참고 : new는 할당된 크기가 0인 경우에도 NULL 포인터를 반환할 수 있음

- std::nothrow는 할당된 객체 자체가 예외를 던질 수 없다는 것을 의미하지 않음

```
struct A{
    A() { throw std::runtime_error{};}
};
A* arr = new (std::nothrow) A; // throw std::runtime_error
```