생성자
(Chapter 7 Constructors and other tools)

숭실대학교
김강희 교수
(khkim@ssu.ac.kr)

순서

- 이론:
  - 생성자 관련 사항들
  - 인자 전달
  - 인라인 함수
  - 관련 C++11 문법들

생성자에 대해 추가로 알아야 할 것들

- 디폴트 생성자
  - 호출 방법
    - Matrix A; // yes!
    - Matrix A(); // no!
  - 누가 작성하나?
    - 다른 생성자들이 없으면 컴파일러에 의해 자동 생성됨
- 생성자의 명시적 호출
  - Matrix A(3,3); A = Matrix(5,5);
    - 익명의 객체가 생성되어 객체 A 에 지정됨
- 생성자 안에서 멤버 변수 초기화의 다른 방식 -> 초기화 섹션
  - MyMatrix::MyMatrix(int cy, int cx) : Matrix(cy, cx) { … }
  - MyMatrix::MyMatrix(int cy, int cx) : dy(cy), dx(cx) { … }

인자(파라미터)의 전달

- Call-by-value : 인자의 복사 작업이 있음
  - 단순한 데이터 타입들(char, int, double, ...)에 대해서는 크게 문제되지 않음
  - 복잡한 데이터 타입들(class)에 대해서는 크게 문제될 수 있음
  - 예1: Matrix(Matrix obj)
  - 예2: Matrix(Matrix *obj)

- Call-by-reference : 인자의 복사 작업이 없음
  - 예: Matrix(Matrix &obj)

인라인 함수들(inline functions)

- class 선언 내부에 함수 body 를 포함하는 경우에는 함수 선언 앞에 inline 키워드가 필요 없음
- class 선언 외부에 있는 함수 body 를 inline 화하기 위해서는 함수 선언 앞에 inline 키워드가 필요함
- inline 함수는 호출되는 지점에서, 함수 호출 코드가 존재하지 않고, 함수 body 코드가 존재함
  - 작은 크기의 함수에 대해서만 inline 하는 것이 좋음

관련 C++11 문법들

- C++11 는 member initialization 라는 기능을 제공함

```
class Matrix {
 private:
      int dy = 1; // default value
      int dx = 1; // default value
 public:
      Matrix() { ... } ;
}
```

- C++11 는 한 생성자가 다른 생성자를 호출하는 것을 허용함

```
class Matrix {
  Matrix(int cy, int cx) : dy(cy), dx(cx) { ... }
  Matrix() : Matrix(1, 1) { ... }
}
```

Matrix.h

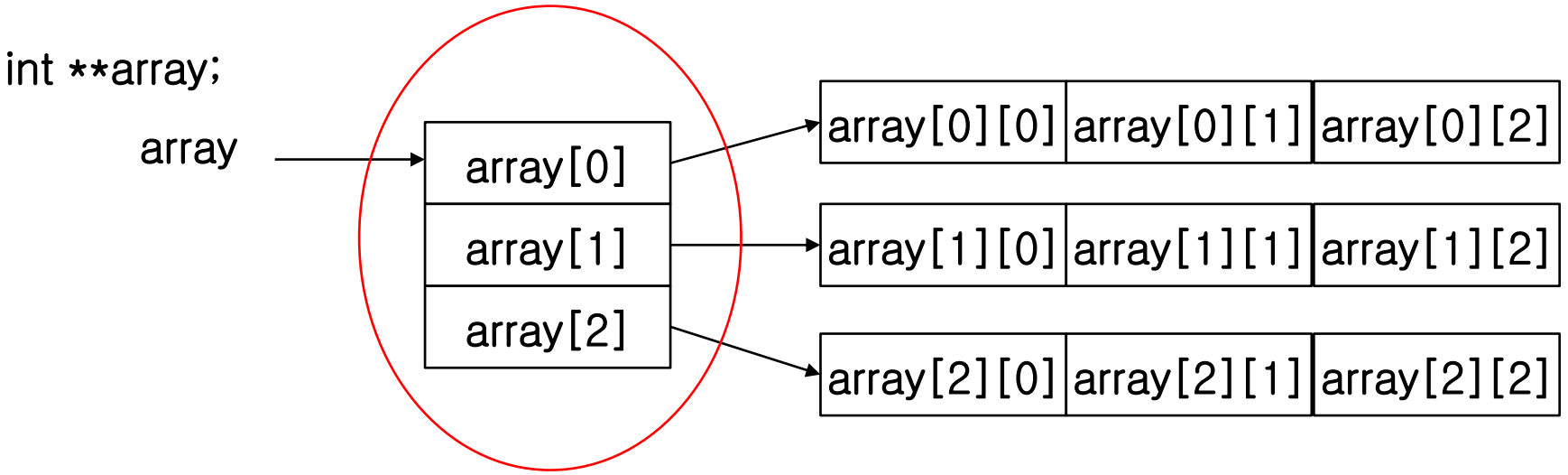
```
1  #pragma once
2  #include <iostream>
3  #include <cstdlib>
4
5  using namespace std;
6
7  class Matrix {
8  private:
9    static int nAlloc;
10   static int nFree;
11   int dy;
12   int dx;
13   int **array;
14   void alloc(int cy, int cx);
15 public:
16   static int get_nAlloc();
17   static int get_nFree();
18   int get_dy();
19   int get_dx();
20   int** get_array();
21   Matrix();
22   Matrix(int cy, int cx);
23   Matrix(const Matrix *obj);
24   Matrix(const Matrix &obj);
25   Matrix(int *arr, int col, int row);
26   ~Matrix();
```

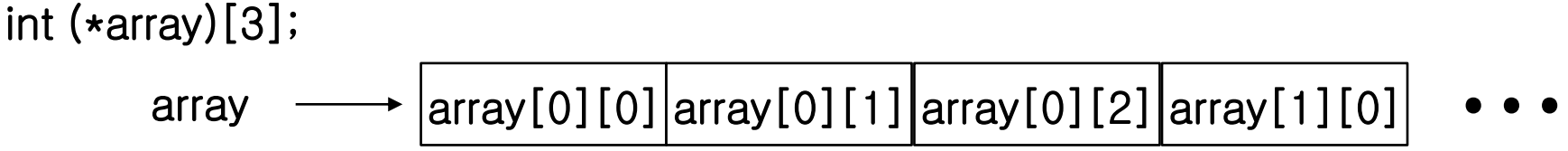
Matrix 생성자들

```
1   #include "Matrix.h"
2
3   int Matrix::nAlloc = 0;
4   int Matrix::nFree = 0;
5
6   int Matrix::get_nAlloc() { return nAlloc; }
7
8   int Matrix::get_nFree() { return nFree; }
9
10  int Matrix::get_dy() { return dy; }
11
12  int Matrix::get_dx() { return dx; }
13
14  int **Matrix::get_array() { return array; }
15
16  void Matrix::alloc(int cy, int cx) {
17    if ((cy < 0) || (cx < 0)) return;
18    dy = cy;
19    dx = cx;
20    array = new int*[dy];
21    for (int y = 0; y < dy; y++)
22      array[y] = new int[dx];
23    for (int y = 0; y < dy; y++)
24      for (int x = 0; x < dx; x++)
25        array[y][x] = 0;
26
27    nAlloc++;
28  }
```

```
30  Matrix::Matrix() { alloc(0, 0); }
31
32  Matrix::~Matrix() {
33    for (int y = 0; y < dy; y++)
34      delete array[y];
35    delete array;
36
37    nFree++;
38  }
39
40  Matrix::Matrix(int cy, int cx) {
41    alloc(cy, cx);
42    for (int y = 0; y < dy; y++)
43      for (int x = 0; x < dx; x++)
44        array[y][x] = 0;
45  }
46
47  Matrix::Matrix(const Matrix *obj) {
48    alloc(obj->dy, obj->dx);
49    for (int y = 0; y < dy; y++)
50      for (int x = 0; x < dx; x++)
51        array[y][x] = obj->array[y][x];
52  }
53
54  Matrix::Matrix(const Matrix &obj) {
55    alloc(obj.dy, obj.dx);
56    for (int y = 0; y < dy; y++)
57      for (int x = 0; x < dx; x++)
58        array[y][x] = obj.array[y][x];
59  }
60
61  Matrix::Matrix(int *arr, int col, int row) {
62    alloc(col, row);
63    for (int y = 0; y < dy; y++)
64      for (int x = 0; x < dx; x++)
65        array[y][x] = arr[y * dx + x];
66  }
```

참고: 포인터들의 배열 vs 2차원 배열의 포인터





주차별 강의 내용

1. ~~교과목 개요~~
2. ~~C/C++ 기초~~
3. ~~구조체와 클래스~~
4. ~~생성자와 벡터 → 과제#1~~
5. 연산자 오버로딩과 프렌즈
6. 포인터와 동적 배열
7. 상속 → 과제#2
8. 중간고사(4/20일 pm7:30)
9. 다형성과 가상 함수, 템플릿
10. 연결된 자료구조들(5/5일 휴강 → 보강 5/11일 50분, 5/18일 50분) → 과제#3
11. 예외 처리
12. 표준 템플릿 라이브러리(STL)
13. 분할 컴파일과 네임스페이스 → 과제#4
14. 스트림과 파일 입출력
15. 기말고사 (6/8일 pm7:30)