객체지향 프로그래밍: C++ 기본사항(1)

2020. 9. 3

순천향대학교 컴퓨터 공학과

내용

- 첫번째 C++ 프로그램 분석
- 기본 타입
- 스트링 클래스
- 수식과 연산자
- 제어구조
  - If
  - Switch
  - While
  - Do-while
  - For
  - 범위 기반 for
- 배열

첫 번째 프로그램의 분석

hello.cpp

```
01  // 첫 번째 예제 프로그램
02  #include <iostream>
03  using namespace std;
04
05  int main()
06  {
07      cout << "Hello World!"<< endl;
08      return 0;
09  }
```

주석

헤더파일

이름 공간 설정

함수 선언

화면에 문자열 출력

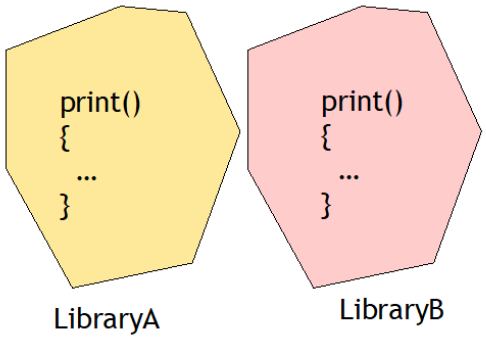
#include <iostream>

- <iostream> 헤더 파일
  - C++ 표준 입출력을 위한 클래스, 객체 선언
  - cout, cin, >>, <<, endl 등을 포함
  - 카보드 입력이나 화면 출력시 반드시 필요

- C++ 표준에서는 헤더파일에 .h의 확장자를 붙이지 않는다.

using namespace std;

- 변수 이름이나 함수 이름과 같은 수많은 이름(식별자)들은 이름 공간(name space)이라고 하는 영역으로 분리되어 저장
  - 프로그램에서 사용되는 이름들간의 충돌 방지
  - 개발자 자신만의 이름 공간을 생성할 수 있게 함
  - 이름 충돌 사례
    - 프로젝트를 여러 명이 나누어 개발하는 경우
    - 다른 사람이 작성한 소스코드나 목적 파일을 사용하는 경우



using namespace std; (2)

- std는 C++ 표준에서 정의한 **표준 이름 공간**
  - <iostream> 헤더 파일에 선언된 모든 이름은 std 이름 공간에 속함
  - C++ 표준 라이브러리에 선언된 이름 포함 (C 라이브러리, C++ 입출력 라이브러리, C++ STL 라이브러리 포함)
- std 이름 공간에 선언된 이름을 접근하기 위해 std:: 접두어 사용
  - std::cout, std::cin, std::endl
  - '::'은 범위 지정 연산자
- using 지시어를 사용하면 이름 접근시 std:: 생략 가능

using namespace std; (2)

- Java의 ‘import package’와 유사

```
java.time.LocalDate today = java.time.LocalDate.now();

import java.time.*
LocalDate today = LocalDate.now();
```

cout 객체

- **스크린** 출력 장치에 연결된 **표준 C++** 출력 스트림 객체
- <iostream> 헤더 파일에 선언
- std 이름 공간에 선언: **std::cout**으로 사용

hello.cpp

```
01  // 첫 번째 예제 프로그램
02  #include <iostream>
03  using namespace std;
04
05  int main()
06  {
07      cout << "Hello World!"<< endl;
08      return 0;
09  }
```

주석

헤더파일

이름 공간 설정

함수 선언

endl() 함수가 호출되어, **cout** 스트림 버퍼에 '**\n**' 삽입 후에 버퍼 내용을 장치에 출력하게 한다

연산자 '<<'

- 스트림 삽입 연산자(stream insertion operator)
  - C++ 기본 산술 시프트 연산자(<<)가 스트림 삽입 연산자로 중복 정의됨
  - ostream 클래스에 구현됨
  - 오른쪽 피연산자를 왼쪽 스트림 객체에 삽입
  - cout 객체에 연결된 화면에 출력

- 여러 개의 << 연산자로 여러 값 출력

```
cout << "Hello\n" << "첫 번째 맛보기입니다.";
```

데이터 입력 받기

```
#include <iostream>
using namespace std;

int main() {
  int width, height;

  cout << "너비를 입력하세요>>";
  cin >> width; // 키보드로부터 너비를 읽어 width 변수에 저장

  cout << "높이를 입력하세요>>";
  cin >> height; // 키보드로부터 높이를 읽어 height 변수에 저장

  int area = width*height;  // 사각형의 면적 계산
  cout << "면적은 " << area << "₩n"; // 면적을 출력하고 다음 줄로 이동
}
```

```
너비를 입력하세요>>3
높이를 입력하세요>>5
면적은 15
```

cin, >> 연산자를 이용한 키 입력

- cin: 표준 입력 장치인 키보드를 연결하는 C++ 입력 스트림 객체
- >> 연산자: 스트림 추출 연산자(stream extraction operator)
  - C++ 산술 시프트 연산자(>>)가 <iostream> 헤더 파일에 스트림 추출 연산자로 중복 정의됨
  - 입력 스트림에서 값을 읽어 변수에 저장
- 연속된 >> 연산자를 사용하여 여러 값 입력 가능

```
cout << "너비와 높이를 입력하세요>>";
cin >> width >> height;
cout << width << '₩n' << height << '₩n';
```

cin, >> 연산자를 이용한 키 입력(2)

- 다음의 모든 기본 타입에 대해서 >> 연산자로 데이터 입력 가능
  - bool, char, short, int, long, float, double

```
int i;
cin >> i;          // 정수를 읽어서 i에 저장

double f;
cin >> f;          // 실수를 읽어서 f에 저장
```

C++ 기본 데이터 타입

- 정수형: short, int, long, long long
- 실수형: float, double, long double
- 문자형: char
- 부울형: bool

bool 타입

- bool형의 변수는 참(true) 또는 거짓(false)만을 가질 수 있다.

```
#include <iostream>
using namespace std;


int main()
{
        bool b;
        b = true;
        return 0;
}
```

string 클래스

- C++ 표준 클래스
- #include <string> 필요
- 문자열의 크기에 따른 제약 없음
- 문자열 복사, 비교, 수정 등을 위한 다양한 함수와 연산자 제공
- C의 스트링보다 다루기 쉬움

예제: C++ string 클래스

```
#include <iostream>
#include <string>   // string 사용시 포함되어야 함
using namespace std;

int main()
{
        string s1 = "Good";
        string s2 = "Morning";
        string s3 = s1 + " " + s2 + "!";

        cout << s3 << endl;

        return 0;
}
```

예제: C++ string 클래스

- 문자열 비교

```
string s1 = "Good";
string s2 = "Bad";
bool b = (s1 == s2);
```

- 문자열과 숫자 연결

```
string s1 = "사과";
string s2;

s2 = s1 + " " + to_string(10) + "개"; // 수치 사용시 오류
cout << s2 << endl;
```

예제: C++ string 클래스

- 문자열 입력

```
string name;

cout << “Enter your name: ”;
cin >>name;
cout <<name <<“are welcomed”<<endl;
```

문자열: C 스타일


```
#include <iostream>
using namespace std;

int main() {
  char name[11]; // 영문은 ?까지 저장할 수 있다.

   cout << "이름을 입력하세요>>";
   cin >> name; // 키보드로부터 문자열을 읽는다.

   cout << "이름은 " << name << "입니다\n"; // 이름을 출력한다.
}
```

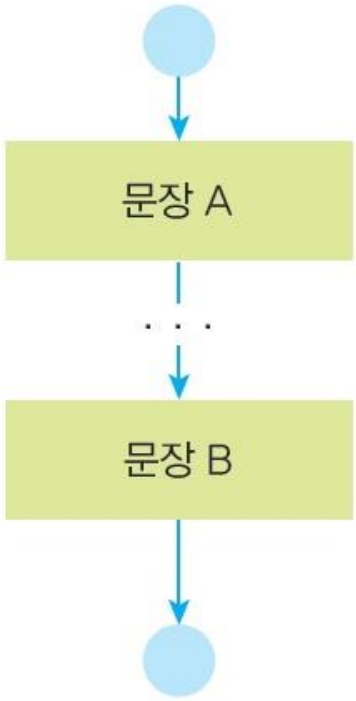
수식과 연산자

- 산술 연산자

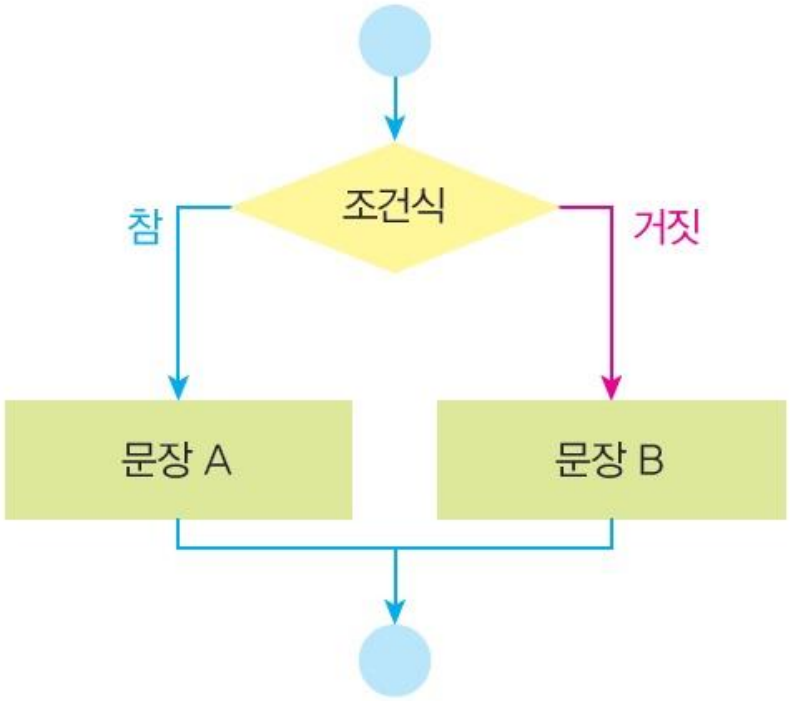
연산자	의미
x + y	x와 y를 더한다
x − y	x에서 y를 뺀다
x * y	x와 y를 곱한다
x / y	x를 y로 나눈다
x % y	x를 y로 나눈 나머지

- 증가/감소 연산자
  - ++, --

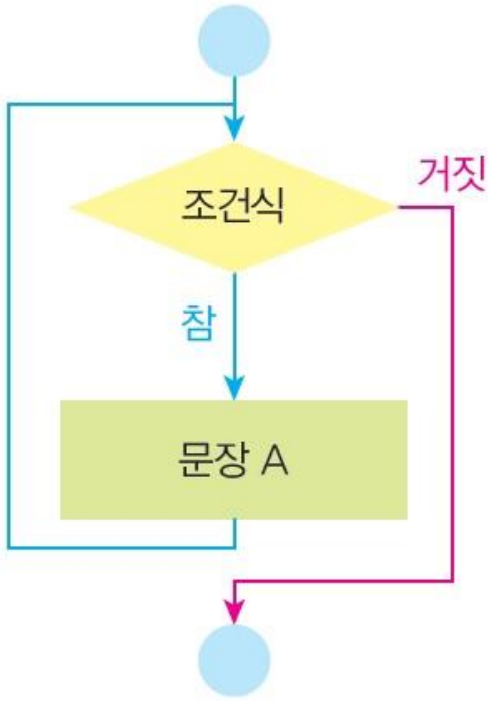
제어구조



순차 구조



선택 구조



반복 구조

관계 연산자

표 2.1 관계 연산자

연산자	의미
x == y	x와 y가 같은가?
x != y	x와 y가 다른가?
x > y	x가 y보다 큰가?
x < y	x가 y보다 작은가?
x >= y	x가 y보다 크거나 같은가?
x <= y	x가 y보다 작거나 같은가?

예제

```
#include <iostream>
using namespace std;

int main() {
        bool b;
        b = (1 == 2);

        cout << std::boolalpha;
        cout << b << endl;

        return 0;
}
```

부울 값을 true/false로 출력되게,
다른 경우에는 1/0이 출력

논리 연산자

표 2.2 논리 연산자

연산자	의미
x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
x || y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참

if-else문

문법 2.1 if-else 문

```
if ( 조건식 ) {
    문장1
}
else {
    문장2
}
```

예제

- 하나의 예로 사용자로부터 받은 두 개의 정수 중에서 더 큰 수를 찾는 프로그램을 작성하여 보자.

예제: 두 수중에서 큰 수 구하기

분석 → 알고리듬 작성 → 코딩

중첩 if-else문

문법 2.2 중첩 if-else 문

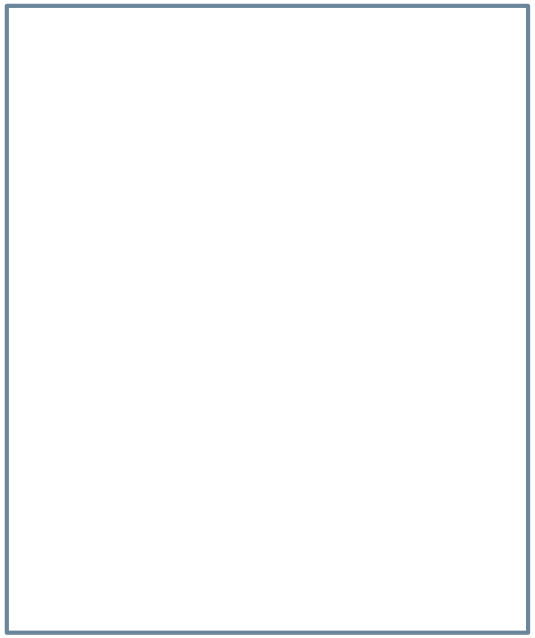
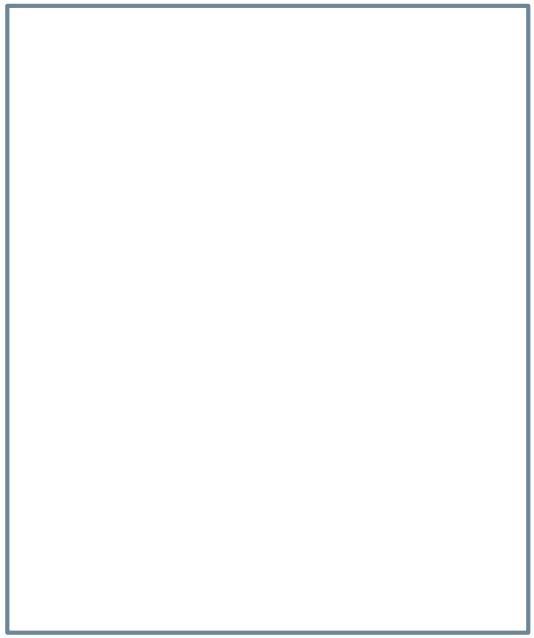
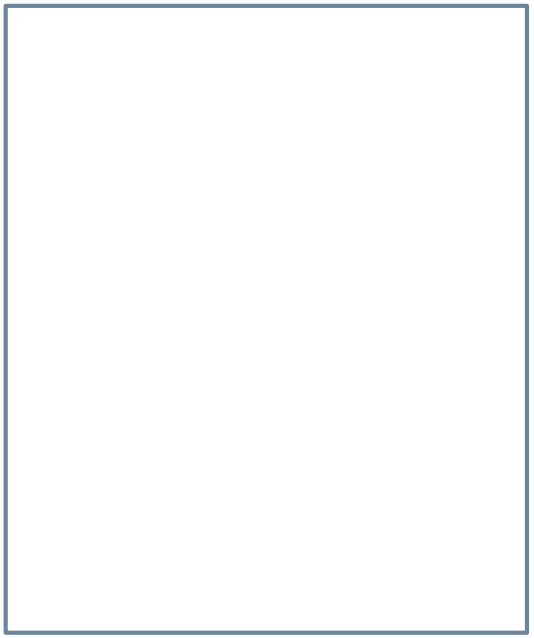
```
if ( 조건식 )  {
    문장1
}
else if {
    문장2
}
else {
    문장3
}
```

예제

- 사용자로부터 나이를 받아서 어린이(12세 이하), 청소년(19세 이하), 성인을 구분하는 프로그램을 작성하여 보자.

예제: 나이별 구분

분석 → 알고리듬 작성 → 코딩

Lab: 3개 정수 중에서 큰 수 찾기

- 사용자로부터 3개의 정수를 입력받고, 이중에서 가장 큰 수를 찾는 프로그램을 작성해보자.

Lab: 3개 정수 중에서 큰 수 찾기

분석 → 알고리듬 작성 → 코딩

switch문

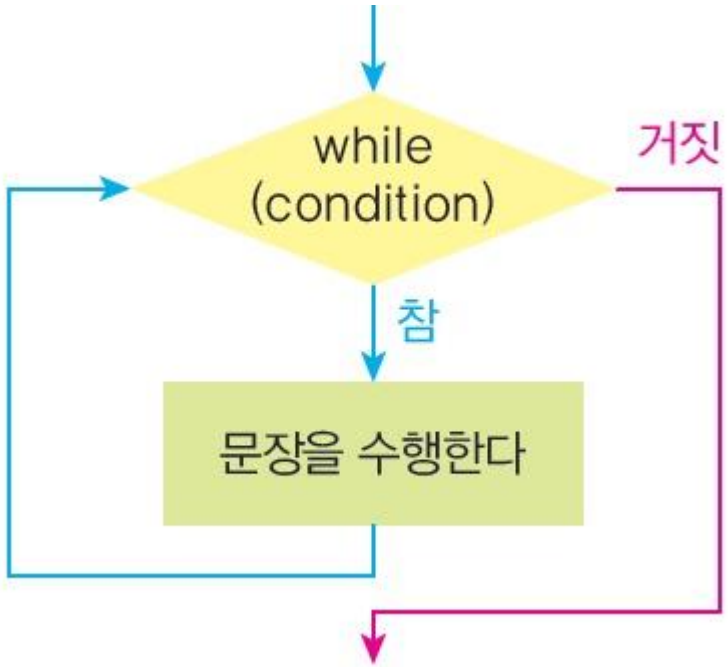
```
int main() {
        int number;

        cout << "숫자를 입력하시오:";
        cin >> number;
        switch (number) {
        case 0:
                cout << "zero\n";
                break;
        case 1:
                cout << "one\n";
                break;
        case 2:
                cout << "two\n";
                break;
        default:
                cout << "many\n";
                break;
        }
        return 0;
}
```

while 루프

문법 2.3 while 루프

```
while (조건식) {
    문장
}
```



do-while루프

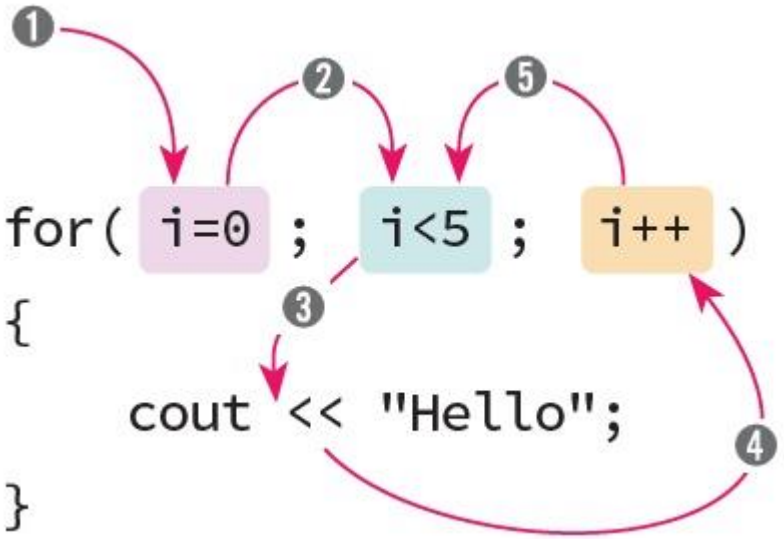
문법 2.4 do-while 문

```
do {
    문장
} while( 조건식 );
```

for루프

문법 2.5 for 루프

```
for( 초기식 ; 조건식 ; 증감식 ) {
    문장
}
```



break 문장 예제

```
#include <iostream>
using namespace std;

int main()
{
        for (int i = 1; i < 10; i++)
        {
                cout << i << " ";
                if (i == 4)
                        break;
        }
        return 0;
}
```

continue 문장 예제

```
#include <iostream>
using namespace std;

int main()
{
	int i = 0;

	do {
		i++;
		cout << "continue 문장 전에 있는 문장" << endl;
		continue;
		cout << "continue 문장 후에 있는 문장" << endl;
	} while (i < 3);

	return 0;
}
```

Lab: 숫자 맞추기 게임

- 프로그램은 1부터 100사이의 정수를 저장하고 있고 사용자는 질문을 통하여 그 정수를 알아맞히려고 노력한다.
  - 1~100사이의 난수 발생: rand() % 100 + 1

```
int main()
{
        srand(time(NULL));

        int answer = rand() %100 +1;    // 정답
        int guess;
        int tries = 0;

        do {
                cout << "정답을 추측하여 보시오: ";
                cin >> guess;
                tries++;

                if (guess >answer)
                        cout << "제시한 정수가 높습니다.\n";
                if (guess <answer)
                        cout << "제시한 정수가 낮습니다.\n";
        } while (guess != answer);

        cout << "축하합니다. 시도 횟수=" << tries << endl;

        return 0;
}
```

배열

- **배열(array)**은 같은 종류의 데이터들이 순차적으로 메모리에 저장되는 자료 구조이다.

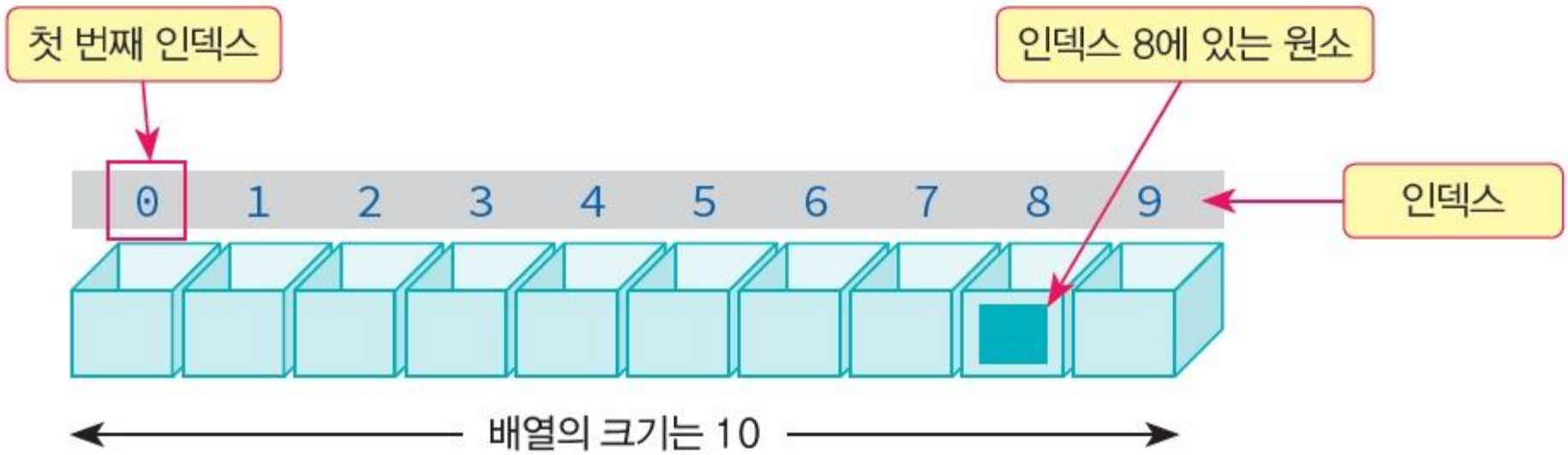


그림 2.1 배열의 개념

배열 선언

문법 2.6 배열 선언

배열의 초기화

```
int sales[5] = { 100, 200, 300, 400, 500 };
int sales[5] = { 100, 200, 300 };
int sales[] = { 100, 200, 300, 400, 500, 600, 700 };
```

	0	1	2	3	4
sales	100	200	300	400	500

보편적 초기화

- 보편적 초기화(universal initialization)는 C++11에서 지원
- 모든 초기화에 중괄호 { ... }을 사용한다. ('='없이 초기화)

```
int scores[] { 10, 20, 30 }; // 다음과 같음: int scores[] = { 10, 20, 30 };
int a { 0 };                 // int a=0;과 동일하다.
string s { "hello" };        // string s="hello";
vector<string> list { "alpha", "beta", "gamma" }; // 벡터 생성시 초기화
```

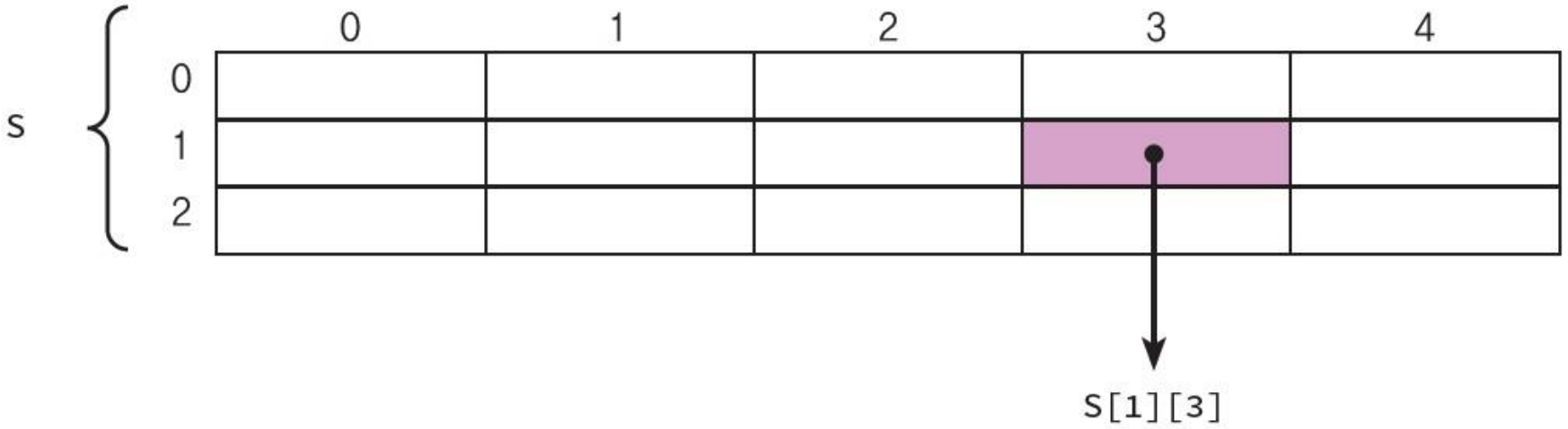
Lab: 배열에서 최대값 찾기

- 크기가 100인 배열을 1부터 100 사이의 난수(=rand()%100+1)로 채우고 배열 요소 중에서 최대값을 찾아보자

2차원 배열의 선언

- C/C++에서, 다차원 배열은 배열의 배열(arrays of arrays)로 표현

```
int s[3][5];
```



2차원 배열의 초기화

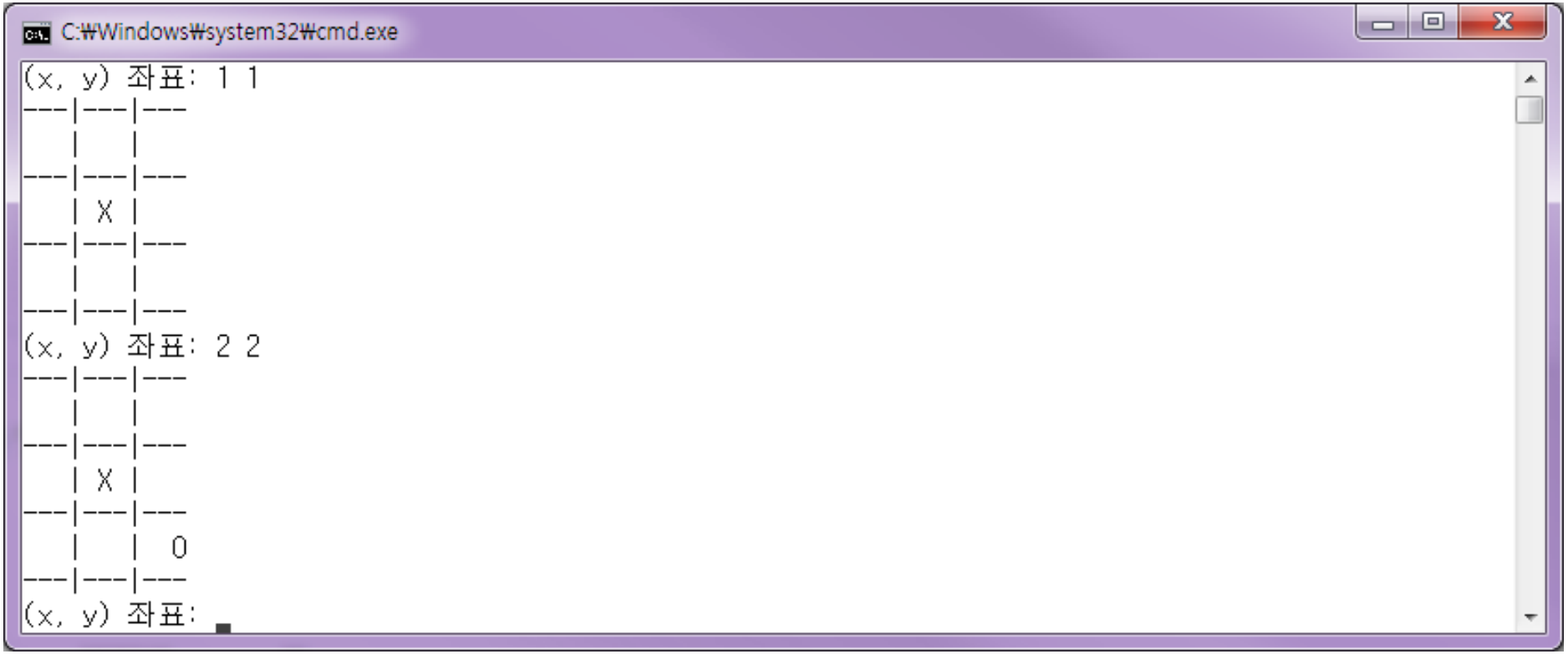
```
int s[3][5] = {
        { 1,  2,  3,  4,  5 },        // 첫 번째 행의 요소들의 초기값
        { 2,  4,  6,  8,  10 },       // 두 번째 행의 요소들의 초기값
        { 3,  6,  9, 12,  15 }        // 세 번째 행의 요소들의 초기값
};
```

s

	0	1	2	3	4
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

Lab: Tic-Tac-Toe 게임

- Tic-Tac-Toe 게임은 2명의 경기자가 오른쪽과 같은 보드를 이용하여서 번갈아가며 O와 X를 놓는 게임이다.



```
int main()
{       char board[3][3];
        int x, y, k, i;

        for (x = 0; x < 3; x++) // 보드를 초기화한다.
                for (y = 0; y < 3; y++) board[x][y] = ' ';

        for (k = 0; k < 9; k++) { // 사용자로부터 위치를 받아 보드에 표시
                cout << "(x, y) 좌표: ";
                cin >> x >> y;
                board[x][y] = (k % 2 == 0) ? 'X' : 'O'; // 순번따라 게임자 결정

                for (I = 0; I < 3; i++) { // 보드를 화면에 그린다.
                   cout <<  " ---|---|--- " << 두이;
                   cout << board[i][0] <<  "   |  " << board[i][1] <<  "   |   "
                        << board[i][2] << 두이;
                }
                cout <<  " ---|---|--- " << 두이;
        }
        return 0;
}
```

* 확장 사항: 승부를 판단하고, 그 결과를 출력, 승부가 결정되면 게임 종료/반복

범위-기반 for루프 (range-based for loop)

- 배열에 포함된 모든 값에 대하여 반복할 때 유용
- Java의 for-each loop과 동일함

문법 2.7 범위-기반 for 루프

```
for( 변수 : 범위 ) {
    문장
}
```

- 범위는 배열, 컨테이너 등으로 표현
- 범위에 포함된 모든 요소에 대해서 반복
- 반복할 때마다 변수에 해당 요소의 값이 할당

예제

```
#include <iostream>
using namespace std;

int main()
{
        int list[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

        for (int i : list) {
                cout << i << " ";
        }
        cout << endl;
}
```