Array and String

프로그래밍 입문(2)

Topics

- **배열 (Array)**
  - **고정 길이 배열 vs. 동적 할당 배열**
- 문자열 (String)
  - char[] 배열
  - string 클래스

Array

- 배열(array)는 **같은 데이터형 여러 개의 값을 연속적으로** 저장하는 데이터 구조입니다.

- 지난 100년간의 지구 기온 변화를 월별로 추적하여 분석하는 프로그램.
  - 12 x 100 = 1200개의 기온을 저장해야 함.
  - `double jan1920 = 14.8, feb1920 = 14.7, mar1920 = 14.9, ..., sep2020 = 15.7;`
  - 2000년의 평균 기온이 알고 싶다면?
    - `avg = (jan2000 + feb2000 + ... + dec2000)/12;`
  - 그렇다면 2010년의 평균 기온이 궁금할 때는...? 2019년은?

Array

- 의미적으로 같은 데이터를 갖는 여러 변수를 하나로 통합하여 표현할 수 있다는 것은 프로그래밍을 간결하게 하기 위해 매우 중요.

- 배열은 특정 데이터형의 자료들을 한 데 묶어 같은 이름을 붙이고, 이들에게 각각 번호를 부여하는 방식.

- e.g.) 학급에서 학생들에게 고유한 번호를 순차적으로 부여, 각각의 학생은 2-1반 1번 학생, 2번 학생 등으로 가리킬 수 있음.

배열의 선언

- 이미 배운 기본 데이터형(primitive types)의 배열에 대해서 배우게 됨.
- 기본 데이터형을 바탕으로 배열을 선언 → 배열은 복합 데이터형.
- <배열의 데이터형> <배열 이름>[<배열 크기>];
  - `int arr[3]; double db_arr[5];`

배열의 크기

- 배열의 크기로는 다음과 같은 것을 쓸 수 있습니다.
  - 정수: `1, 5, 10, 256,` etc.
  - 변수: `n, len, size,` etc.
  - 표현식(expression): `len*2, n*size`, etc.
- 다만 고정된 값이 아닌 변수/표현식 등을 사용할 때는 주의가 필요함.

배열의 초기화

- 변수와 마찬가지로 배열도 초기화가 가능합니다.
- 배열을 초기화하지 않는 경우, 할당된 메모리에 어떤 값이 들어있을지는 예측이 불가능합니다 (변수와 동일).
- 초기화는 { } 안에 값을 넣는 것으로 가능합니다.
  - `int array[3] = { 1, 2, 3 };`
- 값을 지정하지 않는 경우에는 모두 0으로 초기화됩니다.
  - `int array[3] = { };`

배열의 초기화

- 초기화 시에는 배열의 데이터형과 호환되는 값을 넣어 주어야 합니다.
  - `int a[3] = { 1, 10, 100 };` → **OK**,
    `int a[3] = { 1, 'c', 100 }` → **Warning!**
- 값의 수 또한 배열의 크기보다 커서는 안 됩니다.
  - `int a[2] = { 1, 5 };` → **OK**,
    `int a[2] = { 1, 5, 3 };` → **Error!**
- 작은 것은 가능합니다. 나머지는 자동으로 0으로 채워집니다.
  - `int a[3] = {1, 3};` → `a = { 1, 3, 0 }`
- 배열의 크기가 실행시간에 결정되는 경우, 초기화가 불가능합니다.
  - `int n; cin >> n; //user input.`
    `int a[n] = {};` → **Error!**

배열의 원소에 접근

- 배열의 원소(element)에는 인덱스(index)를 사용하여 접근합니다.
- 배열의 크기가 n일 때, 인덱스는 0~n-1의 값을 갖습니다.
- e.g.) `int a[3] = { 3, 2, 1};`
  `a[0] = 3;  a[1] = 2;  a[2] = 1;`
- 각각의 인덱스로 지정된 원소들은 하나의 변수처럼 동작.
  - `a[1] = 5;` → `a = { 3, 5, 1 }`
- 배열의 n번째 원소는 n-1의 인덱스를 갖는다는 점을 명심해야 합니다.
- 인덱스로는 변수나 복잡한 표현식 또한 사용이 가능합니다.
  - `a[i], a[i*(n-1)], a[n/2];`

배열의 원소에 접근

- 배열의 크기보다 큰 인덱스에 접근하는 경우.
  - `int a[10] = {};`
    - `cout << a[100]; //`엉뚱한 값이 튀어나옴.
- 컴파일 단계에서 아무런 에러를 표시하지 않기 때문에, 예측 불가능한 문제를 일으킬 수 있어 주의해야 합니다.
- 특히 메모리의 값을 바꾸는 것도 제약에 없기 때문에 위험합니다.
  - `a[100] = 123; //`문제없이 값이 대입됨.

범위를 벗어난 인덱스의 문제

- 하나의 배열에서 벗어난 범위의 인덱스가 다른 배열에 할당된 메모리를 참조하게 되는 경우.

- int a[3] = {}; int b[5] = {};

- 두 배열의 인덱스 차이를 계산할 수 있음.

  - int diff = b - a;

  - a[diff]는 이제 b[0]를 가리키게 됨 (a[diff+1] → b[1], etc.).

- a[diff] = 10; → b[0]의 값도 같이 바꾸어 버림.

Fixed Length Array

- 고정 길이 배열(Fixed length array) 또는 고정 배열은 컴파일시 배열의 크기가 결정되어 고정됨.

- `cin >> n; int a[n];`과 같이 n의 값을 입력으로 받는 형태의 사용방식은 원래 C++의 표준으로는 금지된 부분.

- 이런 가변 길이 배열(Variable Length Array)은 g++ 등의 컴파일러가 자동 변환하는 형태로 지원해 주기 때문에 실제 실행이 가능.

동적 할당 배열

- 동적으로 메모리를 할당하여 배열의 크기를 실행시간에 결정하기 위해서는 `new` 키워드를 사용합니다.
  - `int* a = new int[len];`
- 동적으로 할당된 배열은 필요가 없어지면 반드시 `delete[]`를 사용하여 메모리를 반환해야 합니다.

동적 할당이 필요한 이유?

- 왜 실행 시간에 배열 크기를 결정하는 것이 중요한가?
- e.g.) 나라별 사람들의 나이를 입력받아 저장하는 프로그램.
- `int ages[n];`
- `n`은 나라별 사람들의 수.
- 모든 나라에 대해 동작하도록 프로그램을 만들려면?

동적 할당이 필요한 이유?

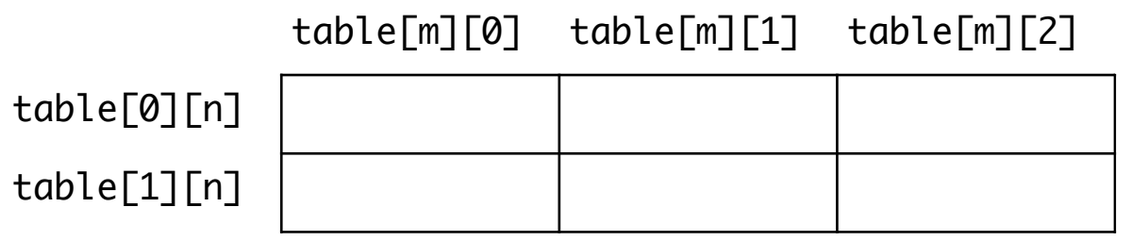
- 인구가 많은 나라에 맞춰 배열 크기 결정 (중국, 인도…?).
- 요구되는 배열 크기 중 가장 큰 것에 맞추어야 모든 입력에 대해 동작하는 프로그램을 만들 수 있음.
- 인구가 적은 나라에서는 수억개의 공간이 그대로 낭비되게 됨.
- 각 나라별 인구수를 입력받고 그에 맞춰 동적으로 배열을 생성하는 것이 메모리 낭비를 막고 효율적인 프로그램을 만들 수 있습니다.

고정 배열 vs. 동적 할당 배열

- 고정 배열은 정보가 스택(Stack)에 저장됩니다.
- 동적 할당 배열은 실행시간에 메모리 할당을 요청하여 힙(Heap)에 데이터를 저장합니다.
- 고정 배열은 속도가 빠르지만, 저장할 수 있는 정보의 크기에 한계가 있습니다 (Stack Overflow).
- 동적 할당 배열은 훨씬 더 큰 데이터를 다룰 수 있게 해주지만, 속도가 느립니다.
- 상황에 알맞게 둘을 사용하는 것이 필요합니다.

다차원 배열

- 다차원 배열(Multi-dimensional Array)은 원소가 한 줄로 늘어서는 것이 아닌 그보다 많은 차원을 갖도록 선언된 것을 말합니다.
  - int table[2][3];



    - 3의 크기를 갖는 int형 배열이 2개 늘어서 있는 것.

- 2차원 배열은 우리가 일반적으로 보는 엑셀의 테이블을 생각하면 됩니다.

- 3차원 배열은 루빅스 큐브와 같은 형태를 생각하면 됩니다. 

다차원 배열의 초기화

- 다차원 배열도 일차원 배열과 유사한 형태로 초기화가 가능합니다.
- `int a[3][2] = { {0, 1}, {1, 2}, {2, 3} };`
- `int a[5][2] = {}; //`모두 0으로 초기화
- `int a[3][2] = {{1, }, {2, }, {3, }}`
  - 일부는 값을 입력하고, 나머지는 0으로 초기화

다차원 배열 원소에 접근

- `int array[3][2][4];`
- 왼쪽부터 순차적으로 인덱스를 사용하여 접근.
- `array[0]~array[2]`: 각각 `int[2][4]`인 배열을 원소로 갖는 배열들.
- `array[0][0]~array[2][1]`: 각각 `int[4]`인 배열을 원소로 갖는 배열들.

Summary

- 배열
- 고정 배열 vs. 동적 할당 배열
- 다차원 배열