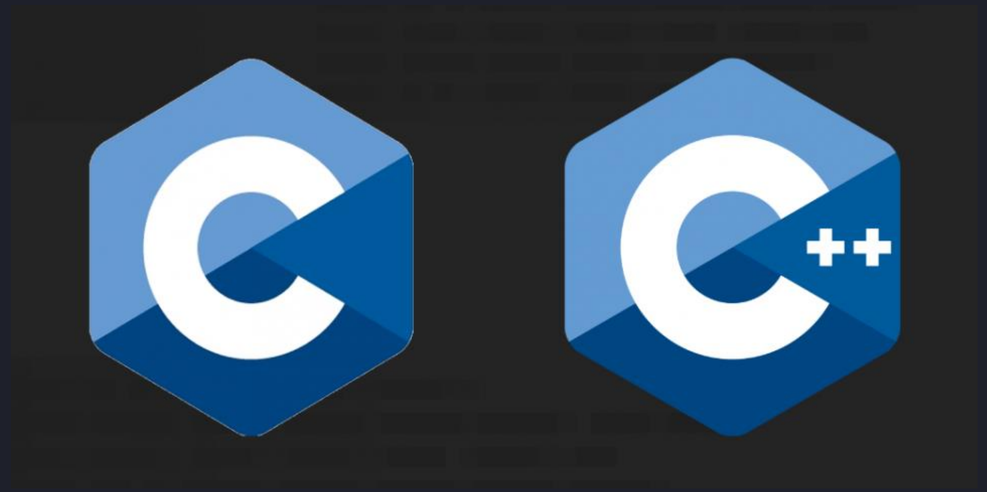


SPLIT 튜터링 5회차

:vector와 string

기계항공공학부 17학번 김기성

Contents

array의 단점

- 프로그래머 입장에서 신경써줘야 하는 것들이 많습니다.
  - size를 프로그래머가 직접 트래킹해야 합니다.
    - array의 전체 길이를 구하는 함수 std::size가 있기는 합니다.
    - 하지만 여유롭게 길이를 잡아놓고, 실제 들어간 원소의 개수만 세고 싶을 때에는 직접 세어야 합니다.
  - size를 바꿀 수 없습니다.
    - size를 바꾸고 싶은 경우, 더 큰 array를 새로 만들어 거기에 모든 원소를 붙여넣기 해야합니다.
  - 잘못된 인덱스 접근에 예외 처리를 직접 해주어야 합니다.
    - 하지 않을 경우 어떤 일이 생길까요?
  - 각종 편의 기능들이 없습니다.
    - ex) integer array에서 3이 처음으로 등장하는 인덱스를 반환하는 함수를 작성하세요.
  - 기타 등등 여러 불편한 사항들이 있습니다...

std::vector

- Array의 크기를 동적으로 바꿀 수 있게 한 배열 기반 컨테이너입니다.
- #include <vector>를 통해 vector 헤더파일을 포함시켜줍니다.
- 자료형은 “std::vector<데이터 타입>”으로 선언합니다.
  - array와 같이 중괄호로 감싼 리스트로 초기화하는 리스트 초기화(list initialization)가 가능합니다.
- 컨테이너의 내장함수를 사용하기 위해서는 .(dot operator)를 사용하면 됩니다.
  - 내장함수는 저번에 알려드린 사이트 (https://cplusplus.com/reference/vector/)에서 확인할 수 있습니다.

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

int main() {
    vector<int> v1 = { 1, 2, 3, 4, 5 };
    cout << "v1의 크기: " << v1.size() << endl;
    cout << "v1[1] = " << v1[1] << endl;
    cout << "v1의 첫번째 원소: " << v1.front() << endl;
    cout << "v1의 마지막 원소: " << v1.back() << endl;
    return 0;
}
```

std::vector

- vector는 동적 배열이기 때문에 원하는만큼 원소를 추가할 수 있습니다.

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

int main() {
    vector<int> v; // empty vector
    for (int i = 1; i <= 5 ; ++i) {
        v.push_back(2 * i);
    }
    for (int i = 0; i < v.size(); ++i) {
        cout << "v[" << i << "] = " << v[i] << endl;
    }
    return 0;
}
```

```
v[0] = 2
v[1] = 4
v[2] = 6
v[3] = 8
v[4] = 10
```

- 이외에도, 지우기, 특정 위치 추가하기/지우기, 바꾸기 등 다양한 내장함수를 지원합니다.

Modifiers:

assign	Assign vector content (public member function)
push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace	Construct and insert element (public member function)
emplace_back	Construct and insert element at the end (public member function)

생성자(constructor) 사용하기

- 모든 자료형(int, char, vector 등)의 뒤에 소괄호를 붙여 함수처럼 호출하면, 객체를 생성하여 돌려줍니다.
- 생성자 사용법
  - 빈 괄호로 호출하는 것을 기본 생성자(default constructor)라고 하며, 기본값으로 지정된 객체를 생성해줍니다.(int->0, bool->false 등)
  - 각 자료형이 지정한 매개변수를 입력해주면 해당하는 객체를 반환해줍니다.
- 변수를 선언할 때 변수 이름 뒤에 괄호를 붙이면, 해당 자료형의 생성자를 호출해 그 변수에 대입해줍니다.
- int, char 등 기본 자료형을 사용할 때는 굳이 쓰지 않지만, vector, string 등의 자료형에는 유용한 생성자들이 정의되어 있습니다.

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

int main() {
    int a = int(); // a = 0
    int b = int(1); // b = 1
    int c(2); // c = 2
    cout << a << " " << b << " " << c << " ";
    return 0;
}
```

std::vector의 생성자 사용하기

- https://cplusplus.com/reference/vector/vector/vector/
- C++ 레퍼런스 사이트에는 생성자 사용법을 친절하게 알려줍니다.
- 생성자를 이용해 벡터를 선언하는 다양한 방법

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

int main() {
    // v1 = {0, 0, 0, 0, 0}
    vector<int> v1 = vector<int>(5);
    // v2 = {1, 1, 1, 1, 1}
    vector<int> v2(5, 1);
    // v3 = {1, 2, 3, 4, 5}
    vector<int> v3({ 1, 2, 3, 4, 5 });
    return 0;
}
```

std::string

- Array기반 문자열의 단점을 보완한 문자열 컨테이너입니다.
- #include<string>을 통해 string 파일을 포함시켜줍니다.
- 자료형은 “std::string”으로 선언합니다.
  - array와 같이 중괄호로 감싼 리스트로 초기화하는 리스트 초기화(list initialization)가 가능합니다.
  - 큰따옴표를 사용한 문자열 초기화도 가능합니다.
- 내장함수는 역시 (https://cplusplus.com/reference/string/string/?kw=string)에서 확인할 수 있습니다.
  - 참고: 문자열의 길이를 반환하는 함수는 size()가 아니라 length()입니다!
- 대괄호를 통해 string의 각 문자에 접근할 수 있습니다. 인덱싱으로 접근한 문자는 char형태로 반환됩니다.

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

int main() {
    string s = "Hello, World!";
    cout << s.length();
    return 0;
}
```

std::string을 이용해 문자열 다루기

- string에는 vector보다 더 다양한 내장함수들이 있습니다.
- 예를 들어, 더하기 연산(+)은 두 string을 이은 string을 반환해줍니다.
  - +=연산도 지원합니다.
  - string += string뿐만 아니라, string += char 형태의 연산도 지원합니다!
- substr(int pos, int len)은 pos로부터 길이 len의 부분 문자열을 반환합니다.
  - pos랑 len을 따로 지정해주지 않으면 어떤 식으로 동작할까요? 아래 레퍼런스를 보고 추측해봅시다.
  - https://cplusplus.com/reference/string/string/substr/

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

int main() {
    string name = "Kim giseong";
    cout << name.substr(4, 7);
    return 0;
}
```

```
giseong
C:\Users\ASUS\source\
이 창을 닫으려면 아무
```

std::string을 이용해 문자열 다루기

- replace(int pos, int len, string str) 함수는 pos부터 길이 len의 부분 문자열을 str으로 대체합니다.
  - https://cplusplus.com/reference/string/string/replace/
- erase(int pos, int len)은 pos부터 길이 len의 부분 문자열을 지웁니다.
  - https://cplusplus.com/reference/string/string/erase/
- insert(int pos, string str)은 pos에 str을 삽입합니다.
  - https://cplusplus.com/reference/string/string/insert/

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

int main() {
    string s = "Kim is a genious";
    cout << s << endl;
    s.insert(7, "not ");
    cout << s << endl;
    s.erase(7, 4);
    cout << s << endl;
    s.replace(0, 3, "Lee");
    cout << s << endl;
    return 0;
}
```

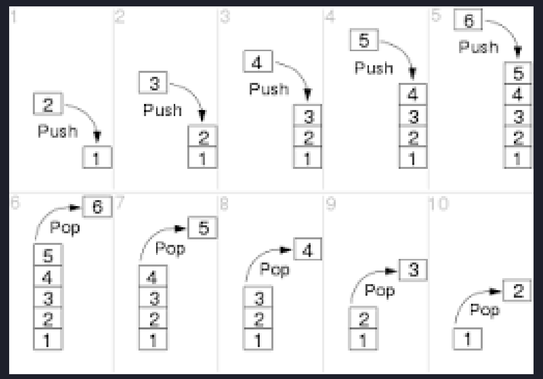
```
Kim is a genious
Kim is not a genious
Kim is a genious
Lee is a genious
```

여담: char 자료형에 대하여

- 컴퓨터는 사실 문자를 모릅니다.
  - char도 내부적으로 숫자로 저장됩니다!
  - 출력했을 때 알파벳으로 보이는 이유는, char 자료형은 알파벳으로 출력하게끔 출력함수가 구현되어있기 때문입니다.
- 따라서 char 자료형도 결국 int같은 숫자입니다!
  - 각 알파벳과 정수가 1대1 대응 관계입니다.
  - 정수와 알파벳 사이의 대응관계는 ascii 코드표에 정리되어 있습니다.
  - https://namu.wiki/w/%EC%95%84%EC%8A%A4%ED%82%A4%20%EC%BD%94%EB%93%9C
- 당연히 덧셈과 뺄셈같은 연산이 가능합니다.
  - 주의할 점은, 문자를 담기 위한 자료형이기 때문에 좀 작습니다.
  - char b = ‘a’+1을 수행한 후 b를 출력하면 어떻게 될까요?

연습문제

- 연습문제1)
  - vector를 이용하여 스택을 구현해 유저의 쿼리에 응답해봅시다.
    - insert x : 정수 x를 stack에 삽입합니다.
    - top : stack의 제일 꼭대기에 있는 원소를 출력합니다.
    - pop : stack의 제일 꼭대기에 있는 원소를 제거합니다.
    - size : stack의 사이즈를 출력합니다.
    - quit : 프로그램을 종료합니다.




```
insert 5
insert 2
insert 10
top
10
pop
top
2
size
2
quit
```

- 연습문제2)
  - 숫자를 표기한 string을 입력받아 정수로 변환하여 반환하는 int string_to_integer(string str)을 작성하세요.
    - 예를 들어, “41256”을 입력받으면 41256을 반환해야 합니다.
    - 참고) string 헤더파일에는 같은 역할을 하는 stoi라는 함수가 구현되어 있습니다. 다음에는 그냥 갖다 쓰세요!

연습문제


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

int main() {
    vector<int> stack;
    while (true) {
        string query;
        cin >> query;
        if (query == “insert") {

        }
        else if (query == "top") {

        }
        else if (query == "pop") {

        }
        else if (query == "size") {

        }
        else {

        }
    }
    return 0;
}
```