ROVIN 2010 Robot Vision & Intelligence Lab.

고급프로그래밍

고급문법 3

Professor Jeong, Mun-Ho

Robot Vision & Intelligence Laboratory
Kwangwoon University
(02-940-5625, mhjeong@kw.ac.kr)

Schedule

주차	주제		과제	퀴즈
1	과목소개	교과목 소개 (1), C++ 시작 (2)		
2	C++	C++ 프로그래밍의 기본 (3), **클래스와 객체 (4)**	1	1
3		객체생성과 사용 (5)	2	2
4		함수와 참조 (6, 3/26), 복사 생성자와 함수중복(7)	3	3
5		static, friend, 연산자중복 (8, 4/2), 연산자중복 상속(9)	4	4
6		상속 (10, 4/9), 가상함수와 추상클래스 (11)		5
7		템플릿과 STL (12, 4/16), 입출력(13)	5	
8		**중간고사**		
9		파일 입출력(14), 예외처리 및 C 사용(15)		6
10		람다식(16, 5/7) , 멀티스레딩(17, 5/9)	6	7
11		멀티스레딩(18, 5/14), 고급문법(19, 5/16)		8
12		고급문법 2(20, 5/21), 고급문법 3(21, 5/23)		
13	병렬프로그래밍	병렬프로그래밍		
14		병렬프로그래밍		
15		**기말고사**		

map 컨테이너

- 특징
  - ('키', '값')의 쌍을 원소로 저장하는 제네릭 컨테이너
    - 동일한 '키'를 가진 원소가 중복 저장되면 오류 발생
  - '키'로 '값' 검색
  - 많은 응용에서 필요함
  - #include <map> 필요
- 맵 컨테이너 생성 예
  - 영한 사전을 저장하기 위한 맵 컨테이너 생성 및 활용
    - 영어 단어와 한글 단어를 쌍으로 저장하고, 영어 단어로 검색

```
// 맵 생성
Map<string, string> dic;                    // 키는 영어 단어, 값은 한글 단어

// 원소 저장
dic.insert(make_pair("love", "사랑"));      // ("love", "사랑") 저장
dic["love"] = "사랑";                       // ("love", "사랑") 저장

// 원소 검색
string kor = dic["love"];                   // kor은 "사랑"
string kor = dic.at("love");                // kor은 "사랑"
```

map 클래스의 주요 멤버와 연산자

멤버와 연산자 함수	설명
insert(pair<> &element)	맵에 '키'와 '값'으로 구성된 pair 객체 element 삽입
at(key_type& key)	맵에서 '키' 값에 해당하는 '값' 리턴
begin()	맵의 첫 번째 원소에 대한 참조 리턴
end()	맵의 끝(마지막 원소 다음)을 가리키는 참조 리턴
empty()	맵이 비어 있으면 true 리턴
find(key_type& key)	맵에서 '키' 값에 해당하는 원소를 가리키는 iterator 리턴
erase(iterator it)	맵에서 it가 가리키는 원소 삭제
size()	맵에 들어 있는 원소의 개수 리턴
operator[key_type& key]()	맵에서 '키' 값에 해당하는 원소를 찾아 '값' 리턴
operator=()	맵 치환(복사)

예제 - map

- map 컨테이너를 이용하여 (영어, 한글) 단어를 쌍으로 저장하고, 영어로 한글을 검색하는 사전을 작성하라.

```
저장된 단어 개수 3
찾고 싶은 단어>> apple
사과
찾고 싶은 단어>> lov
없음
찾고 싶은 단어>> love
사랑
찾고 싶은 단어>> exit
종료합니다...
```

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

int main() {
    map<string, string> dic; // 맵 컨테이너 생성
    // 단어 3개를 map에 저장
    dic.insert(make_pair("love", "사랑")); // ("love", "사랑") 저장
    dic.insert(make_pair("apple", "사과")); // ("apple", "사과") 저장
    dic["cherry"] = "체리"; // ("cherry", "체리") 저장

    cout << "저장된 단어 개수 " << dic.size() << endl;

    string eng;
    while (true) {
        cout << "찾고 싶은 단어>> ";
        getline(cin, eng); // 사용자로부터 키 입력
        if (eng == "exit")
            break;  // "exit"이 입력되면 종료
        if(dic.find(eng) == dic.end()) // eng '키'를 끝까지 찾았는데 없음
            cout << "없음" << endl;
        else
            cout << dic[eng] << endl; // dic에서 eng의 값을 찾아 출력
    }
    cout << "종료합니다..." << endl;
}
```

map with iterator

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

int main() {
   map<string, string> dic; // 맵 컨테이너 생성
   // 단어 3개를 map에 저장
   dic.insert(make_pair("love", "사랑")); // ("love", "사랑") 저장
   dic.insert(make_pair("apple", "사과")); // ("apple", "사과") 저장
   dic["cherry"] = "체리"; // ("cherry", "체리") 저장

   map<string, string>::iterator itr;
   for(itr = dic.begin(); itr != dic.end(); itr++)
      cout << "1:" << itr->second << endl;

   for(itr = dic.begin(); itr != dic.end(); )
   {
      cout << "1:" << itr->second << endl;
      itr = itr+1;  //Compile Error
   }

}
```

오늘의 학습내용

- 고급문법
  - array
  - pair & tuple

std::array

std::array

- <array>
- vector와 유사하나, 선언시에 크기가 고정됨

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

int main()
{
    array<int, 3> arr = {1, 2, 3}; //<자료형,크기>
    cout << "Array size = " << arr.size() << endl;

    for(auto item : arr)
        cout << item << " ";
    cout << endl;
}
```

std::array

- 대입연산자 (=)

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

int main()
{
    array<int, 3> a = {1, 2, 3};
    array<int, 3> b;

    b = a;

    for(auto item : b)
        cout << item << " ";
    cout << endl;
}
```

std::array

- 포인터 변수 아님 → 객체

```
int main()
{
    array<int, 3> a = {1, 2, 3};

    int k = a[0];
    int m = *(a + 1);      ○ Invalid operands to binary expression
    int n = a.at(0);
}
```

std::array

- 포인터 변수 아님 → 객체

```
std::array <int, 5> my_arr = {1, 2, 3, 4, 5};
int array [5] = {1, 2, 3, 4, 5};
array = {2, 3, 4, 5, 6}; // 오류
my_arr = {2, 3, 4, 5, 6};
```

std::array

- iterator

```cpp
#include <array>
#include <iostream>

int main() {
  array<int, 6> data = {1, 2, 4, 5, 5, 6};

  cout << "forward iterator : ";
  for (auto itr = data.cbegin(); itr != data.cend(); ++itr) {
    std::cout << *itr << " ";
  }
  cout << endl;

  cout << "backward iterator : ";
  for (auto itr = data.crbegin(); itr != data.crend(); ++itr) {
    cout << *itr << " ";
  }
  cout << endl;
}
```

std::array

- sort

```
#include <algorithm>
#include <array>
#include <iostream>

int main() {
  array<int, 6> arr = {7, 1, 5, 6, 3, 4};
  sort(arr.begin(), arr.end());

  for (int item : arr)
      cout << item << " ";
}
```

std::array

- front, back, fill

```
:

int main(void)
{
    array<string, 3> arr = {"front", "mid", "back" };

    //front, back
    cout << " arr.front() : " << arr.front() << endl;
    cout << " arr.back()  : " << arr.back() << endl;
    cout << endl;
    //fill
    arr.fill("filled");
    cout << " arr fill() : " << arr[0] << ", " << arr[1] << ", " << arr[2] << endl;
    cout << endl;
    //swap
    array<string, 3> arr2 = { "swaped1", "swaped2", "swaped3" };
    arr.swap(arr2);
    cout << " arr swap() : " << arr[0] << ", " << arr[1] << ", " << arr[2] << endl;
}
```

std::array

- C array → array container

```
#include <iostream>
#include <array>
#include <algorithm>
using namespace std;
int main()
{
   int arr[] = { 1, 2, 3, 4, 5};
   const int n = sizeof(arr));
    array<int, n>  A;

    copy(begin(arr), end(arr), A.begin());
// copy_n(begin(arr), n, begin(A));
// move(begin(arr), end(arr), A.begin());
   for (auto item: A)
      cout << item << " ';
}
```

std::array

- std::out_of_range exception

```
#include <iostream>
#include <array>

using namespace std;

int main()
{
    array<int, 3>  arr{1,2,3};

    try{
        int value = arr.at(10);
    }
    catch (const out_of_range& e){
        cout << “out_of_range error:” << e.what() << endl;
    }
}
```

std:: array 2차원

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

int main() {
    size_t  nRow  = 4;
    size_t  nCol    = 5;
    array<array<int, nCol>, nRow> arValue;

    // 값 할당
    for (size_t  i = 0; i < nRow; ++i) {
        for (size_t  j = 0; j < nCol; ++j) {
            arValue[i][j] = i * j;
        }
    }
    // 값 출력
    for (const auto& row : arValue) {
        for (const auto& item : row) {
            cout << item << " ";
        }
        cout << endl;
    }
}
```

실습

- 다음의 실행 결과는 ?

```
:
int main()
{
        array<int,5>  arr = { 1,2,3,4,5 };
        array<int, 5> arr2 = { 5,4,3,2,1 };

        arr = arr2;

        cout << "arr: ";
        for (int item : arr)
            cout << item << ' ';
        cout << endl;

        cout << "arr2: ";
        for (int item : arr2)
            cout << item << ' ‘;
}
```

pair & tuple

pair

- #include <utility>
- 두 개의 서로 다른 형식의 객체를 한 단위로 묶는 템플릿 클래스
- 멤버 변수 first와 second는 public
- 비교 연산은 first를 먼저 비교하고, 같으면 second 비교

```
int main()
{
   pair<string, int> p1("Hong", 1000);
   cout << "(" << p1.first << ", " << p1.second << ")" << endl;

   pair<string, int> p2("Hong", 900);
   cout << "(" << p2.first << ", " << p2.second << ")" << endl;

   if(p1 == p2)          cout << "p1 = p2" << endl;
   else if(p1 != p2)     cout << "p1 != p2" << endl;

   if(p1 > p2)           cout << "p1 > p2" << endl;
   else if(p1 < p2)      cout << "p1 < p2" << endl;
}
```

pair

- make_pair : 생성자 호출없이 객체 초기화
- 중괄호 초기화 가능

```
int main()
{
    pair<int,double> v1, v2;
    v1 = pair<int,double>(10, 10.1);
    v2 = make_pair(10, 10.1);  // v1과 v2는 동일

    pair<int,pair<double,double>> p1, p2;
    p1 = make_pair(10, make_pair(10.1, 10.2));
    p2 = { 10, {10.1, 10.2}};  //p1과 p2는 동일

    cout << p1.first << " "
         << p1.second.first << " " << p1.second.second << endl;
}
```

tuple

- <tuple>
- 서로 다른 형식의 객체를 임의의 개수로 한 단위로 묶는 템플릿 클래스

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

tuple<string,char,double> student(int id)
{
    if (id == 0) return make_tuple("Tom",'A',3.8);
    if (id == 1) return make_tuple("Jain",'B',3.3);
    if (id == 2) return make_tuple("John",'C',2.5);

    throw invalid_argument("undefined id");
}
```

tuple

- <tuple>
- 서로 다른 형식의 객체를 임의의 개수로 한 단위로 묶는 템플릿 클래스

```
int main()
{
    auto student0 = student(0);
    cout << "ID: 0, "
         << "Name: " << get<0>(student0) << endl
         << "Grade: "<< get<1>(student0) << endl
         << "GPA: "  << get<2>(student0) << endl<<endl;

    auto[name, grade, gpa] = student(1);
    cout << "ID: 2, "
            << "Name: " << name  << endl
            << "Grade: "<< grade << endl
            << "GPA: "  << gpa   << endl<<endl;
}
```