3주차

The Coding Test Academic
C o n f e r e n c e

코딩테스트 학술회

INDEX

1. C++ 추가 문법 – pair, tuple

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

int main()
{
    int a,c;
    char b;

    pair<int, char> pi;
    pi = { 1, 'a'};
    cout << pi.second << "\n"; //a

    tie(a, b) = pi;
    cout << "first: " << a << " second: "<< b <<"\n"; //first: 1 second: a


    tuple <int, char, int> tl;
    tl = make_tuple(1,'a',3);
    a = get<0>(tl);
    cout<<a<<"\n";
    tie(a,b,c) = tl;
    cout << "first: "<<a<<" second: "<<b<<" third: "<<c;
    return 0;
}
```

	Pair	Tuple
사용	두가지 값 담고 싶을 때	세가지 이상의 값 담고 싶을 때
선언 및 할당	pair<int, int> pi; pi = {1, 2}; //or make_pair(1, 2);	tuple<int, int, int> tl; tl = {1, 2, 3}; //or make_tuple(1, 2, 3);
원소 접근	pi.first, pi.second int a, b; tie(a, b) = pi;	int a,b,c; a=get<0>tl;

1. C++ 추가 문법 – map

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

map<string, int> m;


int main()
{
    m.insert({ "Emma", 100 });
    m.insert({ "Kevin", 200 });
    m.insert({ "Amy", 300});

    if (m.find("Emma") != m.end())
    {
        cout << "find" << endl;
    }
    else {
        cout << "not find" << endl;
    }

    //인덱스기반 순회
    for (auto iter = m.begin() ; iter !=  m.end(); iter++)
    {
        cout << iter->first << " " << iter->second << endl;
    }
    cout << endl;

    //범위기반 순회
    for (auto iter : m) {
        cout << iter.first << " " << iter.second << endl;
    }

    return 0;

}
```

A map

각 노드가 key와 value 쌍으로 이뤄진 트리
중복 허용 x. 정렬 보장

```
//값 넣기
map.insert({key, value});
map.emplace(key, value);

//값 변경 및 추가
map[key] = value;

//값 삭제
map.erase(key)
map.erase(m.begin(), m.end());
map.erase(m.begin()+3);
map.clear();

//크기
map.size()

//검색
map.find(key)
```

1. C++ 추가 문법 – map

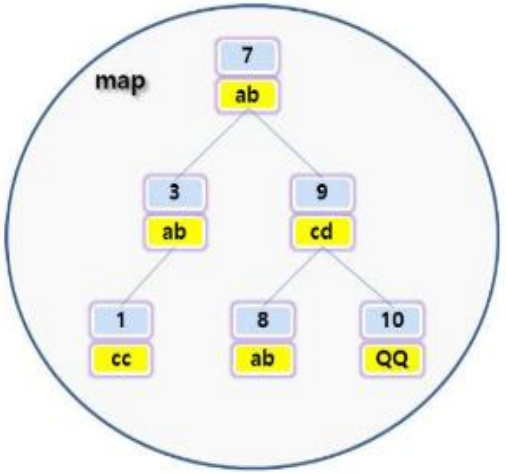
```cpp
#include <iostream>
#include <map>
using namespace std;

map<string, int> m;


int main()
{
    m.insert({ "Emma", 100 });
    m.insert({ "Kevin", 200 });
    m.insert({ "Amy", 300});

    if (m.find("Emma") != m.end())
    {
        cout << "find" << endl;
    }
    else {
        cout << "not find" << endl;
    }

    //인덱스기반 순회
    for (auto iter = m.begin() ; iter !=  m.end(); iter++)
    {
        cout << iter->first << " " << iter->second << endl;
    }
    cout << endl;

    //범위기반 순회
    for (auto iter : m) {
        cout << iter.first << " " << iter.second << endl;
    }

    return 0;

}
```

A map

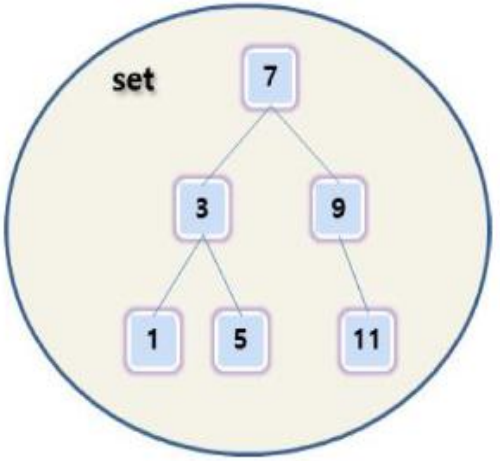
각 노드가 key와 value 쌍으로 이뤄진 트리
중복 허용 x. 정렬 보장

```
//값 넣기
map.insert({key, value});
map.emplace(key, value);

//값 변경 및 추가
map[key] = value;

//값 삭제
map.erase(key)
map.erase(m.begin(), m.end());
map.erase(m.begin()+3);
map.clear();

//크기
map.size()

//검색
map.find(key)
```

1. C++ 추가 문법 – set, multiset



STL map

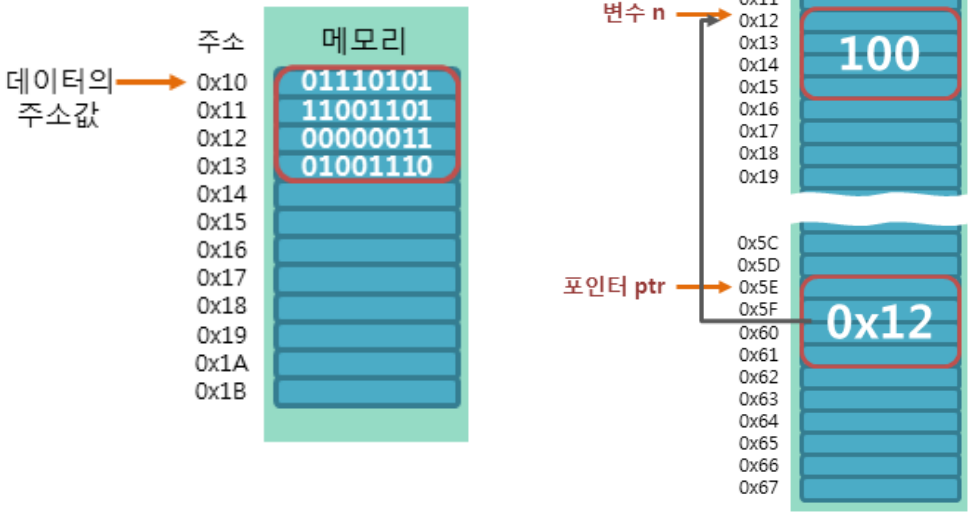


set

A set

특정 순서에 따라 고유한 요소를 저장하는 컨테이너
중복 요소x

1. C++ 추가 문법 - 포인터



A 주소값

데이터가 저장된 메모리의 시작 주소, 1바이트 크기의 메모리 공간으로 나눠 표현

B 포인터

메모리의 주소값을 저장하는 변수

```
int n = 100;
int *ptr = &n; //포인터 선언

cout << ptr << "\n";
cout << *ptr << "\n"; //참조 연산자 사용
```

```
0x7ffeb19bdbac
100
```

Call by reference, value?

2. 순열

$$ {}_nP_r = n(n-1)(n-2)\cdots(n-r+1) $$

$$ = \frac{n!}{(n-r)!} $$

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;


void printP(vector<int> &v){
    for(int i=0; i<v.size(); i++){
        cout << v[i] << " ";
    }
    cout <<"\n";
}

int main(){
    int a[3] = {1,2,3};
    vector<int> v;
    //오름차순
    for(int i =0; i< 3; i++)v.push_back(a[i]);
    do{
        printP(v);
    }while(next_permutation(v.begin(), v.end()));

    v.clear();

    //내림차순

    for(int i = 2; i >= 0; i--)v.push_back(a[i]);
    do{
        printP(v);
    }while(prev_permutation(v.begin(), v.end()));
    return 0;
}
```

A 순열

nPr : 서로 다른 n개, 중복을 허락하지 않고 r개를 일렬로 나열하는 수

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int a[3] = {1, 2, 3};
vector<int> v;

void printP(vector<int> &v){
    for(int i=0; i<v.size(); i++){
        cout << v[i] << " ";
    }
    cout <<"\n";
}

void makePermutation(int n, int r, int depth){
    if(r == depth){
        printP(v);
        return;
    }
    for(int i = depth; i < n; i++){
        swap(v[i], v[depth]);
        makePermutation(n, r, depth + 1);
        swap(v[i], v[depth]);
    }
    return;
}

int main(){
    for(int i =0; i< 3; i++)v.push_back(a[i]);
    makePermutation(3, 3, 0);
    return 0;
}
```

2. 조합

$${}_nC_r = \text{서로 다른 n 개 중에서 겹치지 않도록 r 개를 선택하는 방법의 수}$$

$${}_nC_r = \frac{\text{뽑아서 줄세우기}}{\text{줄세우기}}$$

$${}_nC_r = \frac{{}_nP_r}{r!} = \frac{\frac{n!}{(n-r)!}}{r!} = \frac{n!}{r!\,(n-r)!}$$

n! : 전체 / r! : 뽑힌것 / (n-r)! : 안뽑힌것

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;


void printP(vector<int> &v){
    for(int i=0; i<v.size(); i++){
        cout << v[i] << " ";
    }
    cout <<"\n";
}

int main(){
    int a[3] = {1,2,3};
    vector<int> v;
    //오름차순
    for(int i =0; i< 3; i++)v.push_back(a[i]);
    do{
        printP(v);
    }while(next_permutation(v.begin(), v.end()));

    v.clear();

    //내림차순

    for(int i = 2; i >= 0; i--)v.push_back(a[i]);
    do{
        printP(v);
    }while(prev_permutation(v.begin(), v.end()));
    return 0;
}
```

A 순열

순서 없이 n개 뽑는 수

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int a[3] = {1, 2, 3};
vector<int> v;

void printP(vector<int> &v){
    for(int i=0; i<v.size(); i++){
        cout << v[i] << " ";
    }
    cout <<"\n";
}

void makePermutation(int n, int r, int depth){
    if(r == depth){
        printP(v);
        return;
    }
    for(int i = depth; i < n; i++){
        swap(v[i], v[depth]);
        makePermutation(n, r, depth + 1);
        swap(v[i], v[depth]);
    }
    return;
}

int main(){
    for(int i =0; i< 3; i++)v.push_back(a[i]);
    makePermutation(3, 3, 0);
    return 0;
}
```