Basics: From C to C++

Computer Programming for Engineers (DSAF003-42)

Fall, 2021

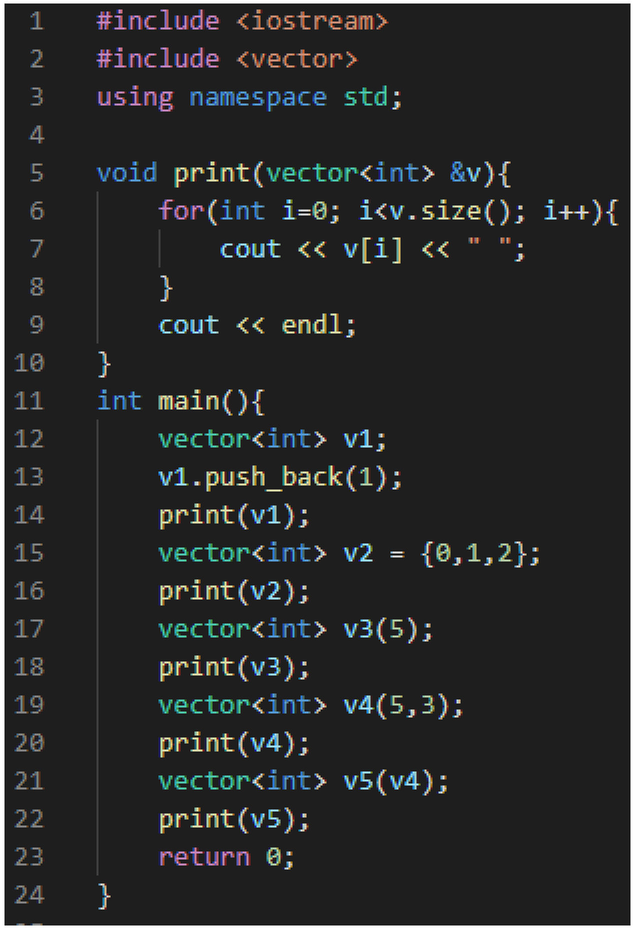
Practice 11 : STL

Instructor:

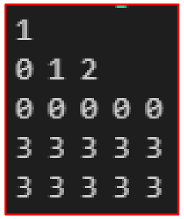
Youngjoong Ko (nlp.skku.edu)

Declaration vector

- **vector is a sequence container that encapsulates dynamic size arrays.**



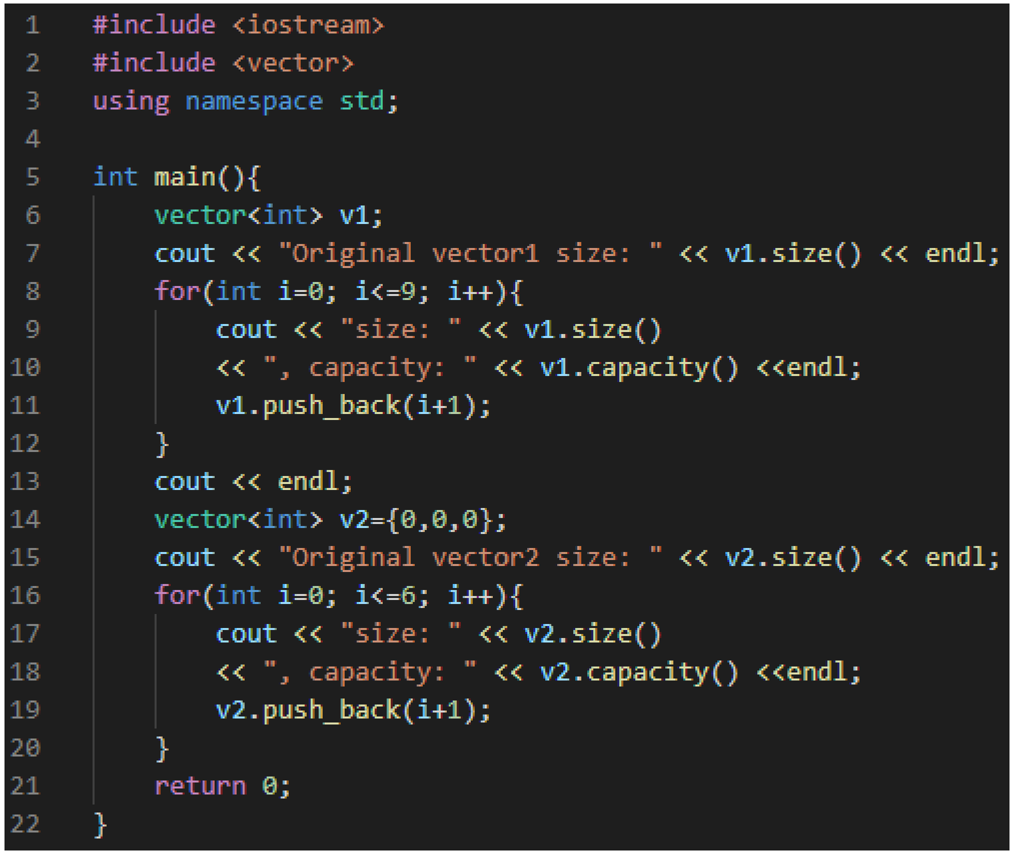
```cpp
#include <iostream>
#include <vector>
using namespace std;

void print(vector<int> &v){
    for(int i=0; i<v.size(); i++){
        cout << v[i] << " ";
    }
    cout << endl;
}
int main(){
    vector<int> v1;
    v1.push_back(1);
    print(v1);
    vector<int> v2 = {0,1,2};
    print(v2);
    vector<int> v3(5);
    print(v3);
    vector<int> v4(5,3);
    print(v4);
    vector<int> v5(v4);
    print(v5);
    return 0;
}
```



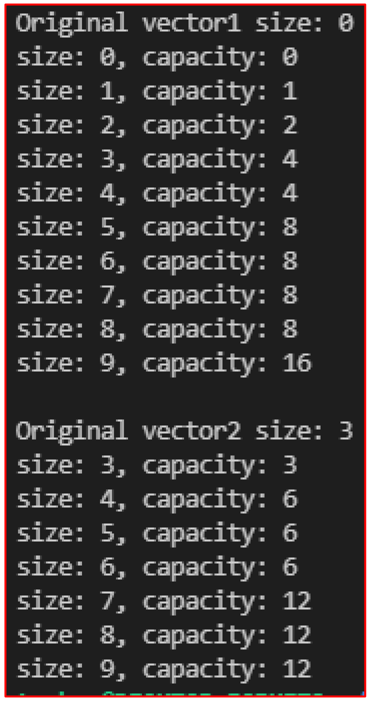
```
1
0 1 2
0 0 0 0 0
3 3 3 3 3
3 3 3 3 3
```

Size & Capacity

- when capacity is not enough, the capacity is **doubled**.



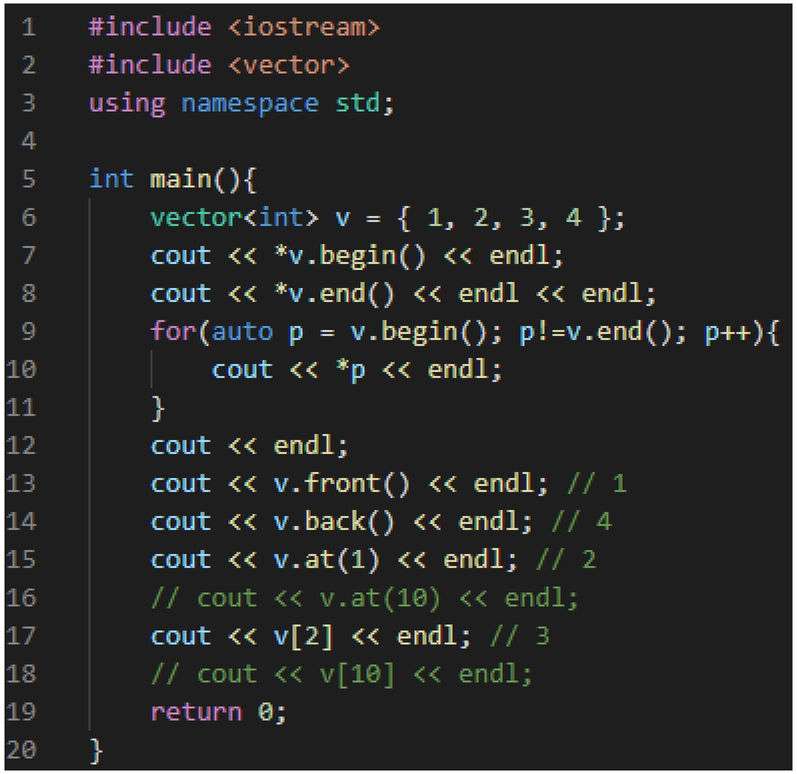
```cpp
#include <iostream>
#include <vector>
using namespace std;

int main(){
    vector<int> v1;
    cout << "Original vector1 size: " << v1.size() << endl;
    for(int i=0; i<=9; i++){
        cout << "size: " << v1.size()
        << ", capacity: " << v1.capacity() <<endl;
        v1.push_back(i+1);
    }
    cout << endl;
    vector<int> v2={0,0,0};
    cout << "Original vector2 size: " << v2.size() << endl;
    for(int i=0; i<=6; i++){
        cout << "size: " << v2.size()
        << ", capacity: " << v2.capacity() <<endl;
        v2.push_back(i+1);
    }
    return 0;
}
```



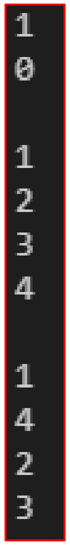
```
Original vector1 size: 0
size: 0, capacity: 0
size: 1, capacity: 1
size: 2, capacity: 2
size: 3, capacity: 4
size: 4, capacity: 4
size: 5, capacity: 8
size: 6, capacity: 8
size: 7, capacity: 8
size: 8, capacity: 8
size: 9, capacity: 16

Original vector2 size: 3
size: 3, capacity: 3
size: 4, capacity: 6
size: 5, capacity: 6
size: 6, capacity: 6
size: 7, capacity: 12
size: 8, capacity: 12
size: 9, capacity: 12
```

Slicing vector

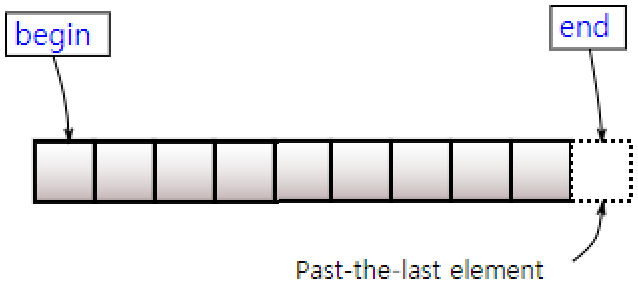
- at():Returns a reference to the element at position n in the vector.
- This is in contrast with member operator[], that does not check against bounds.



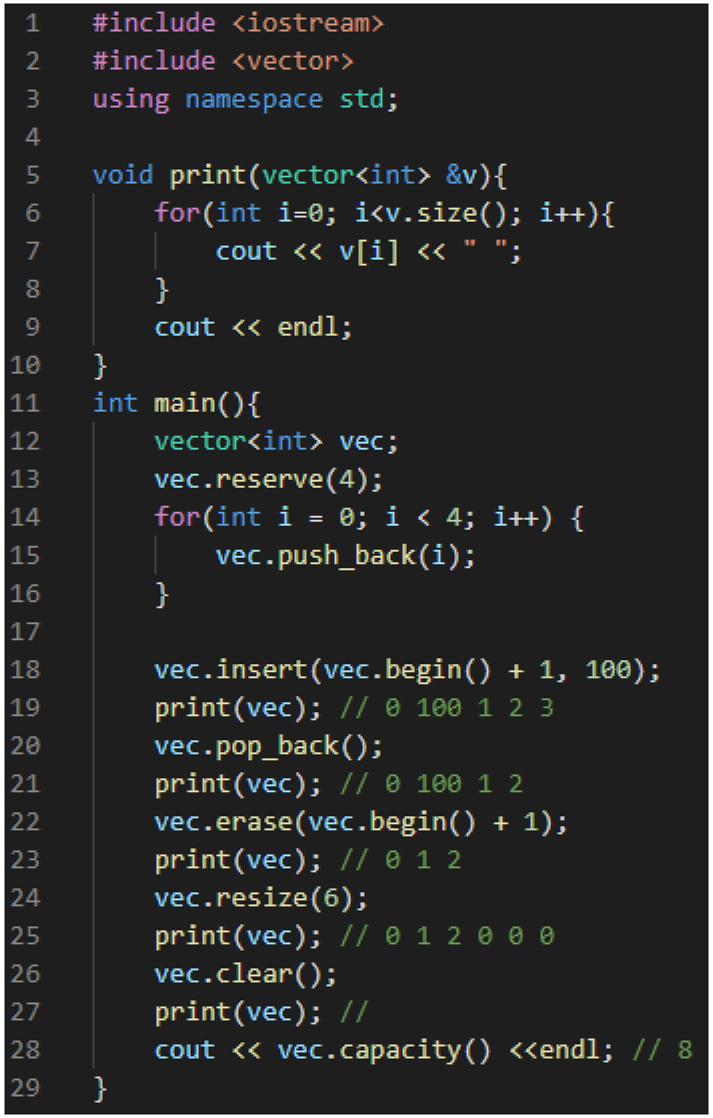
```cpp
#include <iostream>
#include <vector>
using namespace std;

int main(){
    vector<int> v = { 1, 2, 3, 4 };
    cout << *v.begin() << endl;
    cout << *v.end() << endl << endl;
    for(auto p = v.begin(); p!=v.end(); p++){
        cout << *p << endl;
    }
    cout << endl;
    cout << v.front() << endl; // 1
    cout << v.back() << endl; // 4
    cout << v.at(1) << endl; // 2
    // cout << v.at(10) << endl;
    cout << v[2] << endl; // 3
    // cout << v[10] << endl;
    return 0;
}
```



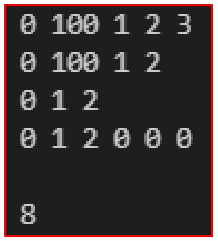
```
1
0

1
2
3
4

1
4
2
3
```



Vector member function

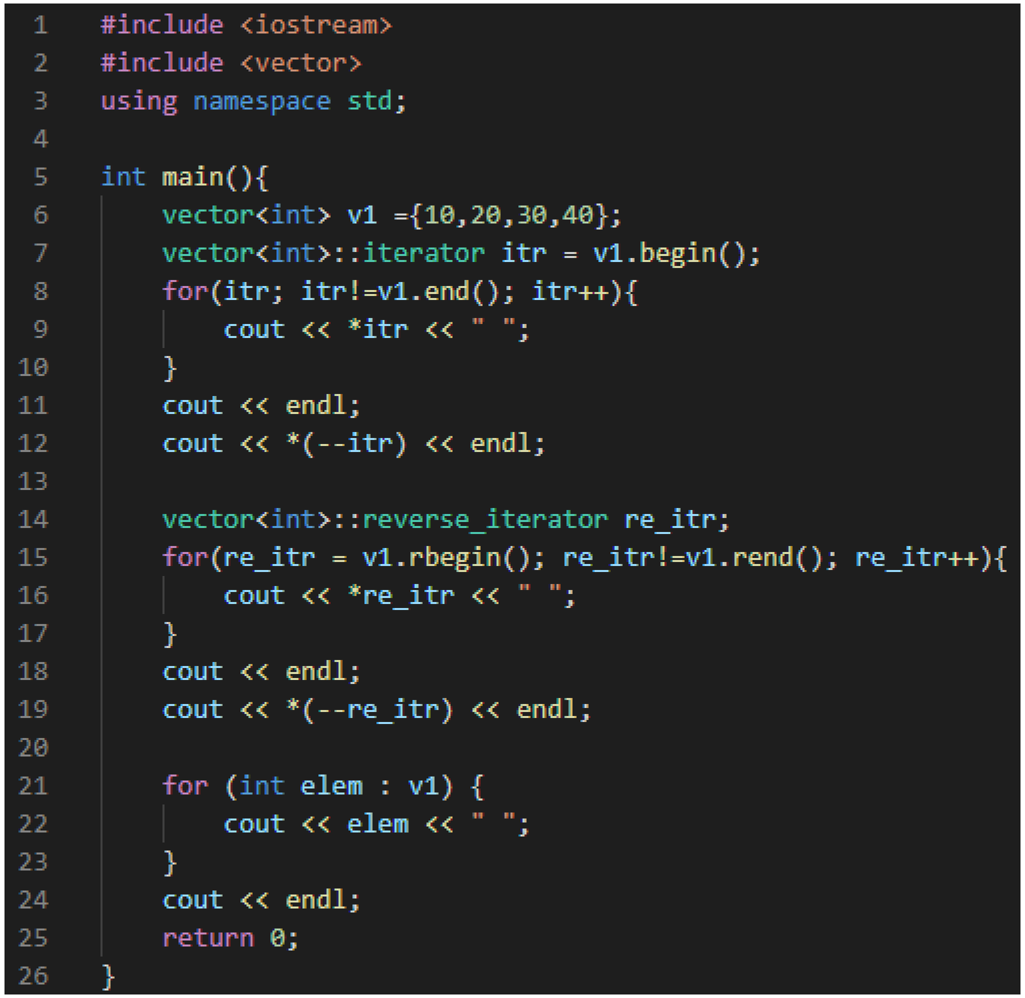


```cpp
#include <iostream>
#include <vector>
using namespace std;

void print(vector<int> &v){
    for(int i=0; i<v.size(); i++){
        cout << v[i] << " ";
    }
    cout << endl;
}
int main(){
    vector<int> vec;
    vec.reserve(4);
    for(int i = 0; i < 4; i++) {
        vec.push_back(i);
    }

    vec.insert(vec.begin() + 1, 100);
    print(vec); // 0 100 1 2 3
    vec.pop_back();
    print(vec); // 0 100 1 2
    vec.erase(vec.begin() + 1);
    print(vec); // 0 1 2
    vec.resize(6);
    print(vec); // 0 1 2 0 0 0
    vec.clear();
    print(vec); //
    cout << vec.capacity() <<endl; // 8
}
```

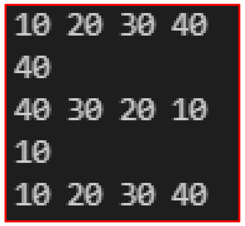


```
0 100 1 2 3
0 100 1 2
0 1 2
0 1 2 0 0 0

8
```

Vector Iterator



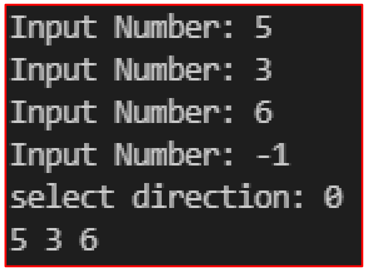
```cpp
#include <iostream>
#include <vector>
using namespace std;

int main(){
    vector<int> v1 ={10,20,30,40};
    vector<int>::iterator itr = v1.begin();
    for(itr; itr!=v1.end(); itr++){
        cout << *itr << " ";
    }
    cout << endl;
    cout << *(--itr) << endl;

    vector<int>::reverse_iterator re_itr;
    for(re_itr = v1.rbegin(); re_itr!=v1.rend(); re_itr++){
        cout << *re_itr << " ";
    }
    cout << endl;
    cout << *(--re_itr) << endl;

    for (int elem : v1) {
        cout << elem << " ";
    }
    cout << endl;
    return 0;
}
```



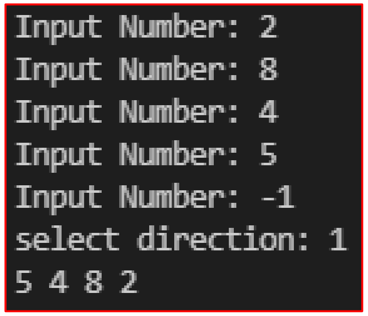
```
10 20 30 40
40
40 30 20 10
10
10 20 30 40
```

Exercise 1

- **Define main function**
  - Get integer values until -1 entered
  - User can select direction in which to output
  - After selection, The system outputs all values of the vector in the direction.



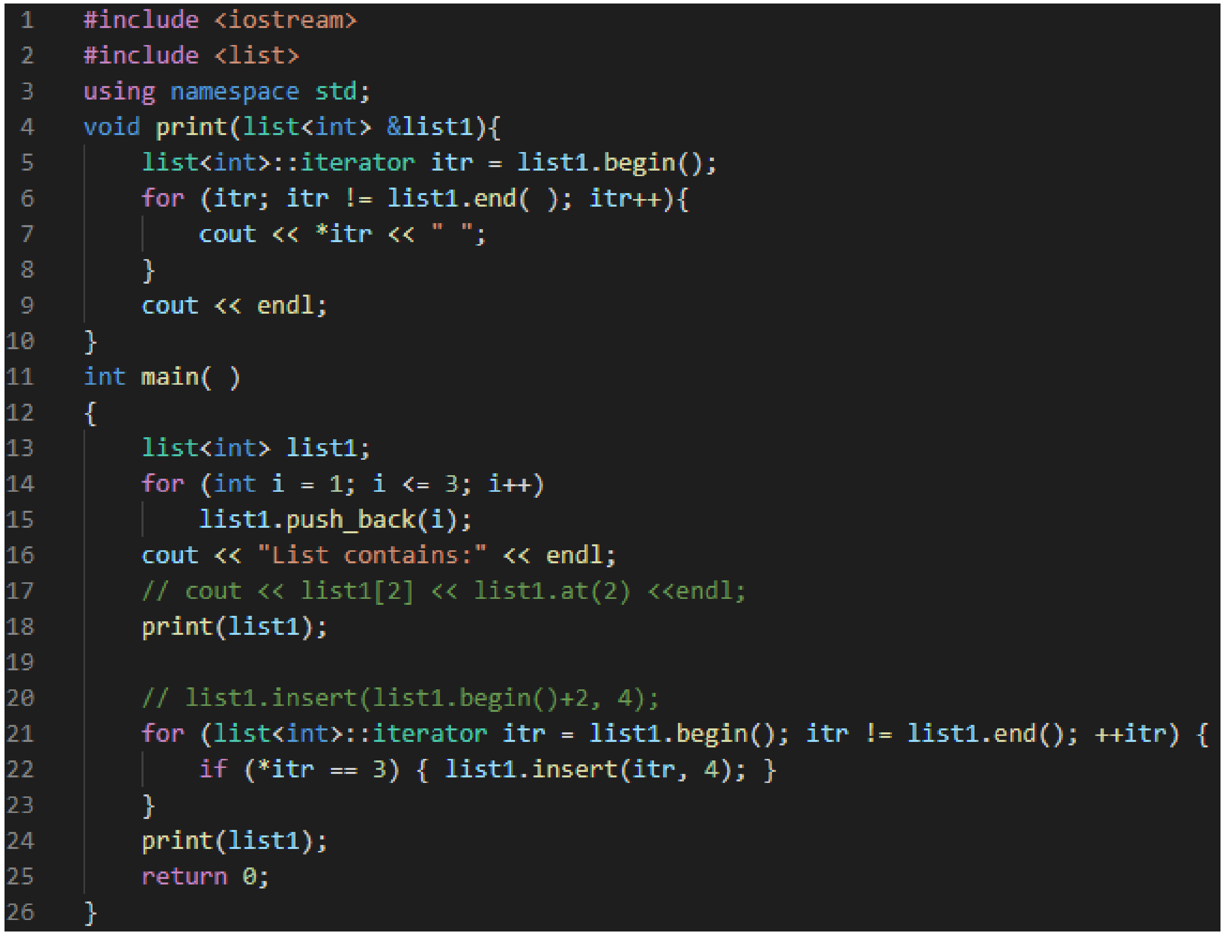
```
Input Number: 5
Input Number: 3
Input Number: 6
Input Number: -1
select direction: 0
5 3 6
```



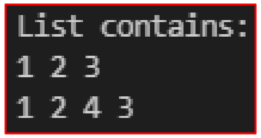
```
Input Number: 2
Input Number: 8
Input Number: 4
Input Number: 5
Input Number: -1
select direction: 1
5 4 8 2
```

List example

- using double linked list structure
- cannot using [ ] (random access) operator



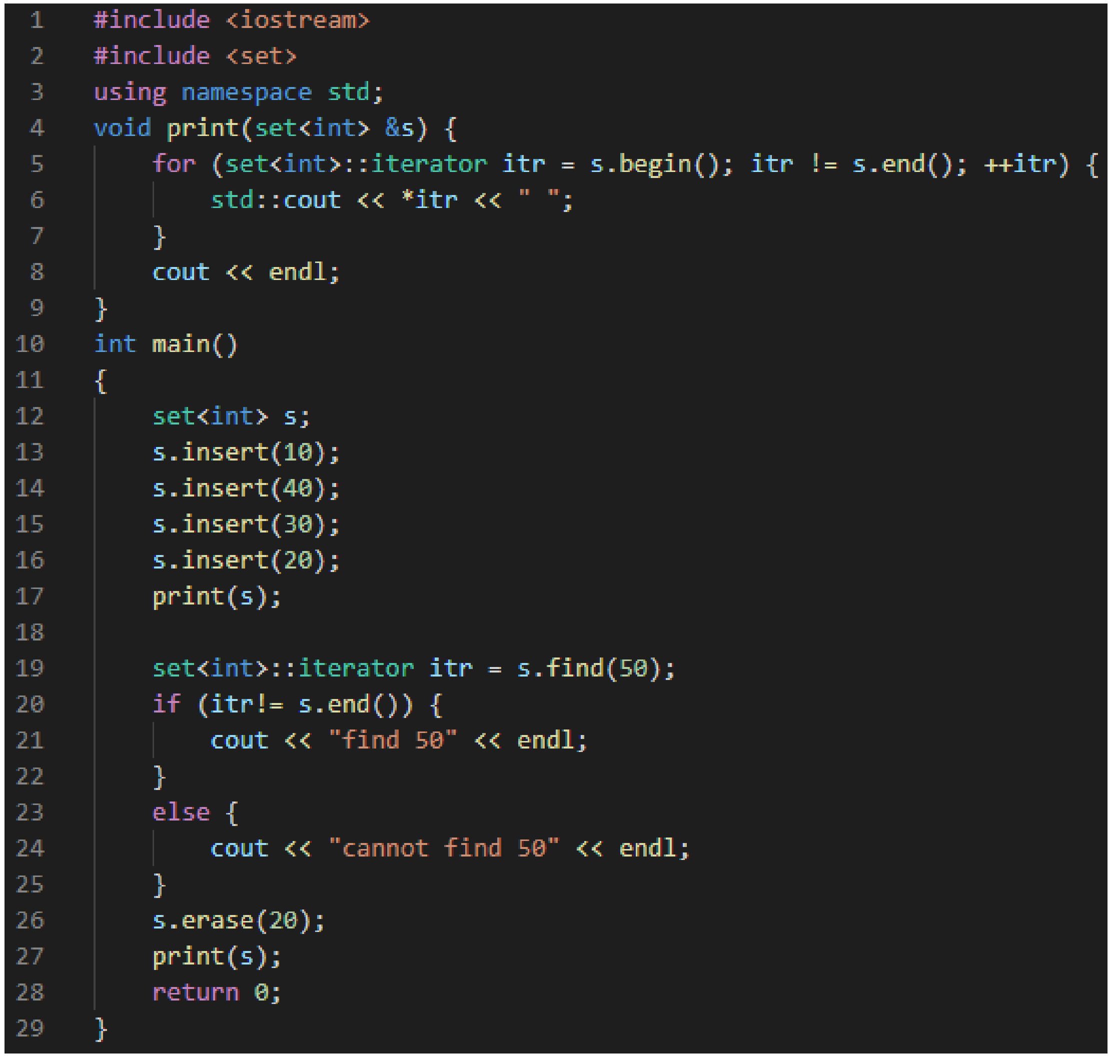
```cpp
#include <iostream>
#include <list>
using namespace std;
void print(list<int> &list1){
    list<int>::iterator itr = list1.begin();
    for (itr; itr != list1.end( ); itr++){
        cout << *itr << " ";
    }
    cout << endl;
}
int main( )
{
    list<int> list1;
    for (int i = 1; i <= 3; i++)
        list1.push_back(i);
    cout << "List contains:" << endl;
    // cout << list1[2] << list1.at(2) <<endl;
    print(list1);

    // list1.insert(list1.begin()+2, 4);
    for (list<int>::iterator itr = list1.begin(); itr != list1.end(); ++itr) {
        if (*itr == 3) { list1.insert(itr, 4); }
    }
    print(list1);
    return 0;
}
```



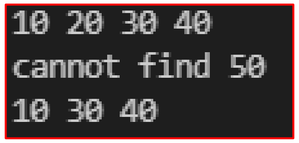
```
List contains:
1 2 3
1 2 4 3
```

Set example

- store unique keys using binary tree



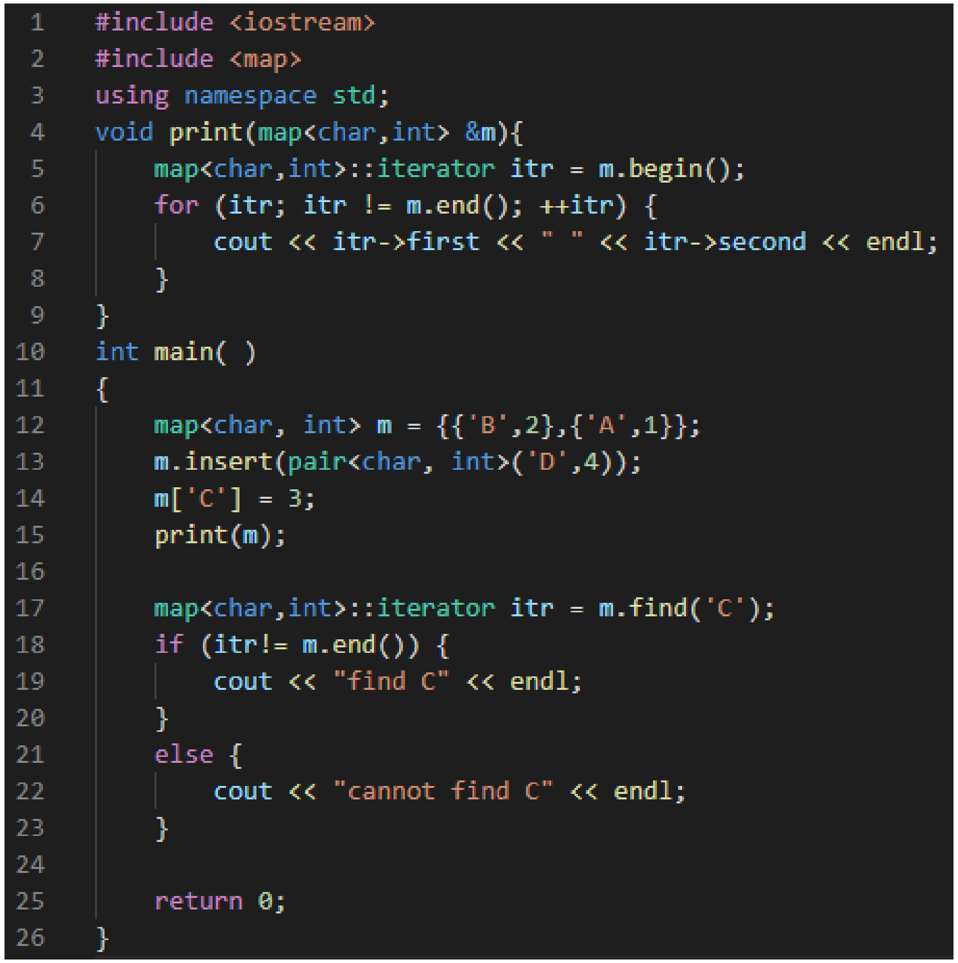
```cpp
#include <iostream>
#include <set>
using namespace std;
void print(set<int> &s) {
    for (set<int>::iterator itr = s.begin(); itr != s.end(); ++itr) {
        std::cout << *itr << " ";
    }
    cout << endl;
}
int main()
{
    set<int> s;
    s.insert(10);
    s.insert(40);
    s.insert(30);
    s.insert(20);
    print(s);

    set<int>::iterator itr = s.find(50);
    if (itr!= s.end()) {
        cout << "find 50" << endl;
    }
    else {
        cout << "cannot find 50" << endl;
    }
    s.erase(20);
    print(s);
    return 0;
}
```



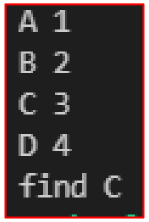
```
10 20 30 40
cannot find 50
10 30 40
```

Map example

- store key-value data using binary tree as set
- key and value stored in std::pair (first = key, second=value)



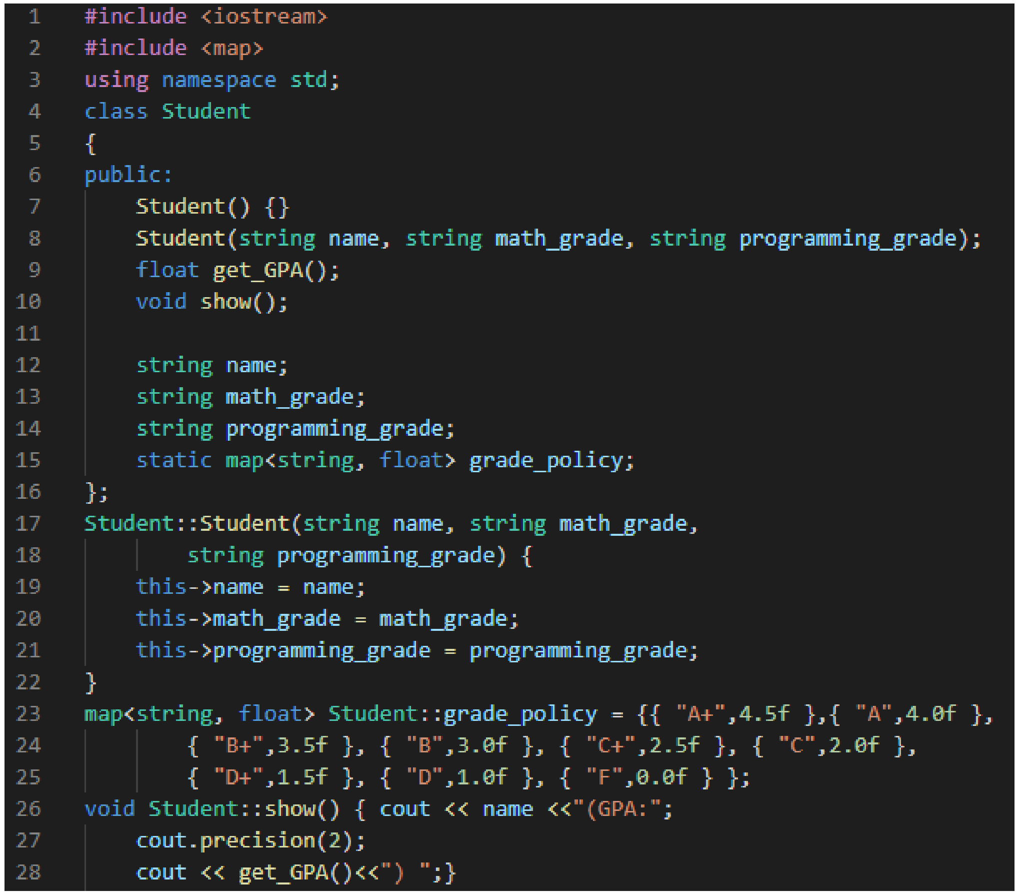
```cpp
#include <iostream>
#include <map>
using namespace std;
void print(map<char,int> &m){
    map<char,int>::iterator itr = m.begin();
    for (itr; itr != m.end(); ++itr) {
        cout << itr->first << " " << itr->second << endl;
    }
}
int main( )
{
    map<char, int> m = {{'B',2},{'A',1}};
    m.insert(pair<char, int>('D',4));
    m['C'] = 3;
    print(m);

    map<char,int>::iterator itr = m.find('C');
    if (itr!= m.end()) {
        cout << "find C" << endl;
    }
    else {
        cout << "cannot find C" << endl;
    }

    return 0;
}
```



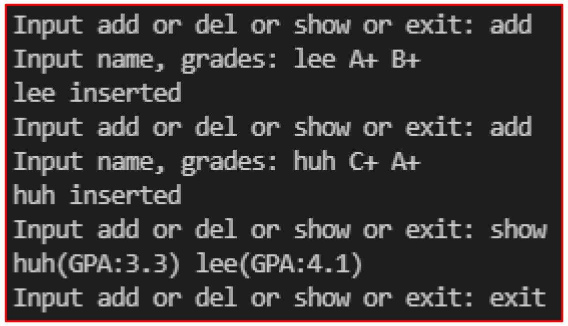
```
A 1
B 2
C 3
D 4
find C
```

Exercise 2

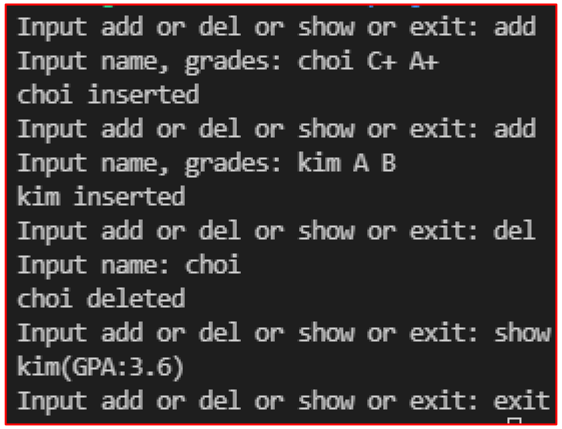
- **Define main function and get_GPA() of Student class**
  - Grade scores follows skku's policy. (credits: math=3, programming=2)
  - User can add&delete student info and look all studen.
  - Input order= name math programming



```cpp
#include <iostream>
#include <map>
using namespace std;
class Student
{
public:
    Student() {}
    Student(string name, string math_grade, string programming_grade);
    float get_GPA();
    void show();

    string name;
    string math_grade;
    string programming_grade;
    static map<string, float> grade_policy;
};
Student::Student(string name, string math_grade,
        string programming_grade) {
    this->name = name;
    this->math_grade = math_grade;
    this->programming_grade = programming_grade;
}
map<string, float> Student::grade_policy = {{ "A+",4.5f },{ "A",4.0f },
        { "B+",3.5f }, { "B",3.0f }, { "C+",2.5f }, { "C",2.0f },
        { "D+",1.5f }, { "D",1.0f }, { "F",0.0f } };
void Student::show() { cout << name <<"(GPA:";
    cout.precision(2);
    cout << get_GPA()<<") ";}
```



```
Input add or del or show or exit: add
Input name, grades: lee A+ B+
lee inserted
Input add or del or show or exit: add
Input name, grades: huh C+ A+
huh inserted
Input add or del or show or exit: show
huh(GPA:3.3) lee(GPA:4.1)
Input add or del or show or exit: exit
```



```
Input add or del or show or exit: add
Input name, grades: choi C+ A+
choi inserted
Input add or del or show or exit: add
Input name, grades: kim A B
kim inserted
Input add or del or show or exit: del
Input name: choi
choi deleted
Input add or del or show or exit: show
kim(GPA:3.6)
Input add or del or show or exit: exit
```