|  |
| --- |
| **2019년 12월 10일 실습보고서** |
| **7조 조윤직 송재원 양석준 진영** |
| **실습자료1 : [STL 알고리즘]** |
| **소스코드** |
| //-student클래스  #pragma once  #include<iostream>  using namespace std;  class Student  {  private:    public:  string name;  int score;  Student() :Student("", 0) {  }  Student(string name, int score) :name(name), score(score) {  }  Student(string name) :Student(name, 0) {  }  friend ostream& operator<<(ostream& out, Student& copy);  friend ostream& operator<<(ostream& out, const Student\* copy);    bool operator==(string check){  return this->name == check;  }  bool operator==(const Student& check) {  return (this->name == check.name && this->score == check.score);  }  bool operator<(const Student& check) {  return this->name < check.name;  }  bool operator>(const Student& check) const {  return this->name > check.name;  }  string getname() const {  return name;  }  bool operator()(const Student& check)const {  return (this->name == check.name && this->score == check.score);  }  };  \main  #include <iostream>  using namespace std;  #include "Student.h"  #include <algorithm>  #include <vector>  #include <functional>//for placeholder  template<typename T>class stdGreater {  public:  bool operator()(const T& t1, const T& t2) {  return t1.name < t2.name;  }  };  ostream& operator<<(ostream& out, Student& copy) {  out << "이름 : " << copy.name << ", 성적 : " << copy.score;  return out;  }  //studen\* 출력 함수  ostream& operator<<(ostream& out, const Student\* copy) {  out << "이름 : " << copy->name << ", 성적 : " << copy->score;  return out;  }  template<typename T1>  void print(const typename vector<T1> vec, const string& ss,const string& tab="\t") {  cout << ss << endl;  for (T1 i : vec) {  cout << i << tab;  }  cout << endl;  }  int random() {  return rand() % 100;  }  bool checkkim(const Student& std) {  if (std.getname().find("김") != string::npos)  return true;  else  return false;  }  class checkName {//비교함수 객체에 넣어 객체만들기  private:  string name;  public:  checkName(const string& name):name(name){}  bool operator()(const Student& std)const {  return this->name == std.getname();  }  };  bool checkName2(const Student& std1, const Student& std2) {  return std1.getname() == std2.getname();  }  int main()  {  vector<int> vec(10);  // 값을 생성해주는 함수  generate(vec.begin(), vec.end(), random);  sort(vec.begin(), vec.end());  print(vec, "출력");  sort(vec.begin(), vec.end(), greater<int>());  print(vec, "출력");  vector<Student> vecstu;  vecstu.push\_back(Student("이길동", 20));  vecstu.push\_back(Student("김길동", 80));  vecstu.push\_back(Student("김아동", 80));  vecstu.push\_back(Student("김호동", 80));  vecstu.push\_back(Student("호김동", 70));  vecstu.push\_back(Student("조길동", 40));  vecstu.push\_back(Student("박길동", 20));  print(vecstu, "정렬전 학생", "\n");  sort(vecstu.begin(), vecstu.end());  print(vecstu, "정렬후 학생", "\n");  //위치 찾기 find ,string -객체의 operator==필요  cout << "김호동 찾기\n";  vector<Student>::iterator it = find(vecstu.begin(), vecstu.end(), "김호동");  if (it != vecstu.end()) {  cout << distance(vecstu.begin(), it) << "번 째에 존재\n";  }  cout << endl;  //위치 찾기 find ,studend -객체의 operator==필요  cout << "김호동,80점 찾기\n";  it = find(vecstu.begin(), vecstu.end(), Student("김호동",80));  if (it != vecstu.end()) {  cout << distance(vecstu.begin(), it) << "번 째에 존재\n";  }  cout << endl;  //김씨 찾기  it = vecstu.begin();  cout << "이름에 김 들어가는 사람\n";  while (true)  {  it = find\_if(it, vecstu.end(), checkkim);  if (it == vecstu.end())  break;  cout << distance(vecstu.begin(), it) << "번 째에 이름 김 들어감\n";  it++;  }  //조건을 함수 operator()를 가진 객체로 생성해서 사용 -따로 클래스 만들기  checkName std("박길동");  it = vecstu.begin();  cout << "\n박길동 찾기\n";  while (true)  {  it = find\_if(it, vecstu.end(), std);  if (it == vecstu.end())  break;  cout << distance(vecstu.begin(), it) << "번 째에 박길동\n";  it++;  }  //조건 함수 객체로 생성해서 사용 -따로클래스 만들지 않고  // 그냥 객체 클래스 자체에 operator() 을 추가하여  Student std2("조길동", 40);  it = vecstu.begin();  cout << "\n조길동,40 찾기\n";  while (true)  {  it = find\_if(it, vecstu.end(), std2);  if (it == vecstu.end())  break;  cout << distance(vecstu.begin(), it) << "번 째에 조길동-40점\n";  it++;  }  //외부 정의 함수로 조건 주기  using namespace std::placeholders;  it = vecstu.begin();  cout << "\n김아동 찾기\n";  while (true)  { // 이부분에서 매개가 두개가 들어가는 함수에 매개 하나를 주기위해  it = find\_if(it, vecstu.end(), bind(checkName2, Student("김아동", 40), \_1));  if (it == vecstu.end())  break;  cout << distance(vecstu.begin(), it) << "번 째에 김아동\n";  it++;  }  //람다 함수-이용  it = vecstu.begin();  cout << "\n람다 이용-> 호김동 찾기\n";  while (true)  { // 이부분에서 매개가 두개가 들어가는 함수에 매개 하나를 주기위해  //[]->(리턴타입 생략가능){함수 정의 ;}  it = find\_if(it, vecstu.end(), [](const Student& std)->bool {return std.getname() == "호김동"; });  if (it == vecstu.end())  break;  cout << distance(vecstu.begin(), it) << "번 째에 호김동\n";  it++;  }  } |
| **실행결과** |
|  |