|  |
| --- |
| **2019년 12월 03일 실습보고서** |
| **7조 조윤직 송재원 양석준 진영** |
| **실습자료1 : [템플릿상속,STL벡터]** |
| **소스코드** |
| \main  #include <iostream>  #include <vector>  #include "Student.h"  #include "TArrayDataBak.h"  #include <deque>  #include <list>  #include <map>  #include <stack>  #include <algorithm>  #include <queue>  using namespace std;  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 T3>  ostream& operator<<(ostream& out, const TArrayData<T3>& copy) {  copy.showData();  return out;  }  template<typename T, typename T2>  bool Search(T\* arr, int sizearr, T2 find, int& index) {  for (int i = 0; i < sizearr; i++) {  if (arr[i] == find) {  index = i;  return true;  }  }  return false;  cout << "값 찾지 못함\n";  }  template<typename T4>  void printArr(const vector<T4>& v) {  //vector를 const로 받아와서 -- const\_iterator로 안하면 오류 발생  typename vector<T4>::const\_iterator itt;  for (itt = v.begin(); itt != v.end(); itt++) {  cout << \*itt << endl;  }  }  //배열 동적할당 다 삭제해주는 함수  template<typename T1>  void deleteVectorele(vector<T1>& v) {  //v가 비워질 때까지 실행  while (!v.empty())  {  //동적할당 제거, vector 원소-팝  delete v.back();  v.pop\_back();  }  }  bool KorGreater(const Student& t1, const Student& t2) {  return t1.score > t2.score;  }  int main()  { cout<<”7조 조윤직,양석준,박진영,송재원<<endl;  //실습1 -vector 값 찾아서 지우기  vector<int > intarr;  vector<int > intarr2;  intarr.push\_back(10);  intarr.push\_back(20);  intarr.push\_back(30);  intarr.push\_back(40);  intarr.push\_back(50);  intarr.push\_back(60);  printArr(intarr);  vector<int>::iterator it;  it = intarr.begin();  for (it; it != intarr.end(); it++) {  if (\*it == 50) {  cout << "제거\n";  intarr.erase(it);  break;  }  }  printArr(intarr);    //실습 2 swap /\*  intarr2.push\_back(40);  intarr2.push\_back(90);  intarr2.push\_back(100);  intarr2.push\_back(120);  intarr2.push\_back(150);  cout << "intarr2출력\n";  printArr(intarr2);  cout << "intarr 주소 : " << &intarr << endl;  cout << "intarr2 주소 : " << &intarr2 << endl;  cout << "=====swap=====\n";  intarr.swap(intarr2);  cout << "intarr 주소 : " << &intarr << endl;  cout << "intarr2 주소 : " << &intarr2 << endl;  cout << "intarr출력\n";  printArr(intarr);  cout << "intarr2출력\n";  printArr(intarr2);  //\*/  //벡터와 동일하지만 앞에서부터도 들어 갈수 있는 deque/\*  deque<int> de;  de.push\_front(20);  de.push\_front(30);  de.push\_front(40);  de.push\_front(50);  de.push\_back(20);  de.push\_back(20);  deque<int>::iterator it2;  it2 = de.begin();  for (; it2 < de.end(); it2++) {  cout << \*it2 << "\t";  }  cout << "\n";  for (int i = 0; i < de.size(); i++) {  cout << de[i] << "\t";  }  //\*/  //List /\*  list <int> arr3;  arr3.push\_back(80);  arr3.push\_back(40);  arr3.push\_back(10);  arr3.push\_back(70);  arr3.push\_back(30);  //sort실행  for (int i:arr3) {  cout << i << "\t";  }  cout << "\n";  cout << "sort(default) 실행" << endl; arr3.sort();  for (int i : arr3) {  cout << i << "\t";  }  cout << "\n";  greater<int> g;  //greater함수나 디폴트매개로 주는 sort는 각각 less(<),greater(>)가  //정의 되어있는 클래스(,타입)만 가능- 정의시에는 const를 꼭!  cout << "sort(greater) 실행" << endl; arr3.sort(g);  for (int i : arr3) {  cout << i << "\t";  }  //\*/  //사용자 정의 클래스 sort /\*  list<Student> stdarr;  stdarr.push\_back(Student("greanjoa1", 20));  stdarr.push\_back( Student("greanjoa2", 40));  stdarr.push\_back( Student("greanjoa3", 50));  stdarr.push\_back( Student("greanjoa4", 70));  stdarr.push\_back( Student("greanjoa5", 10));  for (Student s : stdarr) {  cout << s << "\n";  }  stdarr.sort();  cout << "점수 정렬\n";  for (Student s : stdarr) {//클래스의 오퍼레이터 오름차  cout << s << "\n";  }  cout << "점수 정렬\n";  stdarr.sort(greater<Student>());//클래스 의 오퍼레이터 내림차  for (Student s : stdarr) {  cout << s << "\n";  }  cout << "이름 정렬\n";  stdarr.sort(stdGreater<Student>());//외부클래스함수  for (Student s : stdarr) {  cout << s << "\n";  }  cout << "점수 정렬\n";  stdarr.sort(KorGreater);//외부-정의함수  for (Student s : stdarr) {  cout << s << "\n";  }  //\*/  //map  cout << "키로 접근 하는 map!\n";  map<string, Student> stdd;  stdd["홍길동"] = Student("홍길동", 20);  stdd["김길동"] = Student("김길동", 20);  stdd["진길동"] = Student("진길동", 40);  stdd["강길동"] = Student("강길동", 30);  stdd["이길동"] = Student("이길동", 50);  cout << stdd["강길동"] << endl;  // Stack  cout << "lastinfirstout-stack\n";  stack<int> sta;  sta.push(2);  sta.push(5);  sta.push(2);  sta.push(1);  while (!sta.empty()) {  cout << sta.top() << "\t";  sta.pop();  }  //priority\_queue- /\*  cout << "자동 정렬 priority\_queue\n";  std::priority\_queue<int> intpq;  //이렇게 정렬함수를 시작부터 정의해줄수도 있음  std::priority\_queue<int, vector<int>, greater<int> > pq;  intpq.push(4);  intpq.push(1);  intpq.push(2);  intpq.push(4);  intpq.push(2);  intpq.push(8);  cout << "intpq : ";  while (!intpq.empty()) {  cout << intpq.top() << "\t";  intpq.pop();  }  cout << "\n--pq : ";  pq.push(4);  pq.push(1);  pq.push(2);  pq.push(4);  pq.push(2);  pq.push(8);  while (!pq.empty()) {  cout << pq.top() << "\t";  pq.pop();  }  //\*/  } |
| **실행결과** |
|  |
|  |