|  |
| --- |
| **2019년 11월 26일 실습보고서** |
| **7조 조윤직 송재원 양석준 진영** |
| **실습자료1 : [stack,que]** |
| **소스코드** |
| // ArrayData.h  #pragma once  #include <iostream>  using namespace std;  template<typename T>  class TArrayData  {  private:  T\* data;  int capacity;  int used;  public:  TArrayData(const int& capacity);  TArrayData(const TArrayData& arr);  TArrayData(TArrayData&& arr);  TArrayData();  ~TArrayData();  void addElement(T num);  bool full() const;  int getCapacity() const;  int getUsed() const {  return used;  }  void emptyArray();  void showData() const;  void operator=(const TArrayData& copy);  void operator=(TArrayData&& copy);  TArrayData getObject() {  return \*this;  }    };  template<typename T>  inline TArrayData<T>::TArrayData(const int& capacity)  {  used = 0;  this->capacity = capacity;  data = new T[capacity];  }  template<typename T>  inline TArrayData<T>::TArrayData(const TArrayData& arr):capacity(arr.capacity),used(arr.used)  {  data = new T[capacity];  for (int i = 0; i < used; i++) {  data[i] = arr.data[i];  }  }  template<typename T>  inline TArrayData<T>::TArrayData(TArrayData&& arr):used(arr.used),capacity(arr.capacity)  {  data = arr.data;  arr.data = NULL;  }  template<typename T>  inline TArrayData<T>::TArrayData() :TArrayData(10)  {  }  template<typename T>  inline TArrayData<T>::~TArrayData()  {  if (data != NULL) {  delete[] data;  }  }  template<typename T>  inline void TArrayData<T>::addElement(T num)  {  if (!full())  data[used++] = num;  else  cout << "data 꽉 참\n";  }  template<typename T>  inline bool TArrayData<T>::full() const  {  return used >= capacity;  }  template<typename T>  inline int TArrayData<T>::getCapacity() const  {  return capacity;  }  template<typename T>  inline void TArrayData<T>::emptyArray()  {  used = 0;  }  template<typename T>  inline void TArrayData<T>::showData() const  {  cout << "used :" << used << "배열 출력\n";  for (int i = 0; i < used; i++) {  cout << i<<" : "<<data[i] << endl;  }  }  template<typename T>  inline void TArrayData<T>::operator=(const TArrayData& copy)  {  if (data != NULL) {  delete[] data;  }  data = new T[copy.capacity];  capacity = copy.capacity;  used = copy.used;  for (int i = 0; i < used; i++) {  data[i] = copy.data[i];  }  }  template<typename T>  inline void TArrayData<T>::operator=(TArrayData&& copy)  {  if (data != NULL) {  delete[] data;  }    capacity = copy.capacity;  used = copy.used;  data = copy.data;  copy.data = NULL;  }  //student.h  #pragma once  #include<iostream>  using namespace std;  class Student  {  private:  string name;  int score;  public:  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);    bool operator==(string check){  return this->name == check;  }  bool operator==(Student& check) {  return (this->name == check.name && this->score == check.score);  }  };  //main.cpp  #include <iostream>  #include "Student.h"  #include "ArrayData.h"  using namespace std;  ostream& operator<<(ostream& out,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";  }  int main() {  cout << "7조 박진영 양석준 송재원 조윤직\n";  /\*  int arr[5] = { 5,4,3,2,1 };  int index;    if (Search(arr, 5, 3,index)) {  cout << arr[index] << endl;  }  string id[3] = { "greenjoa1","greenjoa2","greenjoa3" };  if (Search(id, 3, "greenjoa2", index)) {  cout << id[index]<<" "<<index << endl;  }  Student std[3] = { Student("greenjoa1",70),Student("greenjoa2",50),Student("greenjoa3",80) };  if (Search(std, 3, "greenjoa3", index)) {  cout << std[index] << " " << index << endl;  }    if (Search(std, 3, Student("greenjoa3",80), index)) {  cout << std[index] << " " << index << endl;  }  \*/  TArrayData<Student> std;  std.addElement(Student("greejoa1", 10));  std.addElement(Student("greejoa2", 20));  std.addElement(Student("greejoa4", 40));  std.addElement(Student("greejoa5", 50));  cout << std << endl;  cout << "=======복사 생성자 =============\n";  TArrayData<Student> stdcopy = std;  cout << stdcopy<<endl;  cout << "=======이동 생성자 =============\n";  TArrayData<Student> std2 = move(stdcopy);  cout << std2 << endl;  cout << "=======복사 연산자 =============\n";  TArrayData<Student> stdcopy2;  stdcopy2 = std2;  cout << stdcopy2 << endl;  cout << "=======이동 연산자 =============\n";  std2 = move(std);  cout << std2;      } |
| **실행결과** |
|  |