**Homework 7**

|  |
| --- |
| 그림입니다. 원본 그림의 이름: YU_UI_RGB-10.png 원본 그림의 크기: 가로 2256pixel, 세로 3047pixel 프로그램 이름 : Adobe ImageReady |

|  |  |
| --- | --- |
| 과목명 | 객체지향프로그래밍과자료구조 |
| 교수님 | 김영탁 교수님 |
| 이 름 | 김주환 |
| 학 번 | 21812158 |
| 일 자 | 2021.10.20.수 |

|  |
| --- |
| /\* main.cpp - T\_Array<Student> \*/  /\* Description  \* 템플릿 구현  \* STL의 기능을 직접 구현  \* 오름/내림차순 선택/퀵정렬  \* Programmed by J. H. Kim  \* Last updated : 2021-10-15 \*/  #include <iostream>  #include <fstream>  #include <string>  #include <random>  #include "T\_Array.h"  #include "Student.h"  using namespace std;  #define ELEMENTS\_PER\_LINE 10  #define SAMPLE\_LINES 5  #define NUM\_ELEMENTS 500  #define MIN\_NUM\_ELEMENTS 10  #define NUM\_STUDENTS 10  Student students[10] = {  Student(21811000, string("Kim, G-M"), Date(1990, 10, 5), Time(3, 0, 30), 3.57),  Student(21611075, string("Yoon, S-M"), Date(1990, 4, 5), Time(7, 30, 0), 4.37),  Student(21411015, string("Hwang, S-S"), Date(1989, 1, 10), Time(2, 0, 50), 2.72),  Student(21611054, string("Lee, K-M"), Date(1991, 5, 15), Time(5, 30, 0), 3.35),  Student(21311340, string("Hong, G-M"), Date(1990, 2, 5), Time(1, 10, 0), 3.89),  Student(21811001, string("Kim, M-J"), Date(1992, 11, 7), Time(4, 1, 25), 3.50),  Student(21611076, string("Yoon, S-H"), Date(1993, 5, 1), Time(8, 35, 20), 4.40),  Student(21411016, string("Hwang, S-H"), Date(1988, 2, 13), Time(6, 1, 55), 2.08),  Student(21611055, string("Lee, K-S"), Date(1996, 6, 14), Time(3, 26, 32), 3.59),  Student(21311341, string("Hong, M-E"), Date(1997, 3, 6), Time(0, 10, 10), 3.97),  };  void main() {  ofstream fout;  T\_Array<Student> studentArray(NUM\_STUDENTS, "Array of Students");  Student\* pStudent;  fout.open("output.txt");  if (fout.fail()) {  cout << "Fail to open output.txt file for results !!" << endl;  exit;  }  for (int i = 0; i < NUM\_STUDENTS; i++) {  pStudent = &students[i];  studentArray.insert(i, \*pStudent);  }  fout << "Elements in studentArray after initialization :" << endl;  studentArray.fprint(fout, 1);  // 오름차순 선택정렬  fout << "Elements in studentArray after selection sorting by increasing order of arrival time :" << endl;  studentArray.selection\_sort(INCREASING);  studentArray.fprint(fout, 1);  // 내림차순 퀵정렬  fout << "Elements in studentArray after quick sorting by decreasing order of arrival time :" << endl;  studentArray.quick\_sort(DECREASING);  studentArray.fprint(fout, 1);  fout.close();  } |
| /\* Time.h \*/  #ifndef TIME\_H  #define TIME\_H  #include <iostream>  using namespace std;  class Time {  friend ostream& operator<<(ostream&, const Time&);  public:  Time(); // default constructor  Time(int h, int m, int s); // 생성자  Time& operator+(int s); // 연산자 오버로딩  int elasedSec(); // 지난 초 계산  Time getTime(); // Time 정보 반환    // 비교연산자  bool operator<(Time&);  bool operator<=(Time&);  bool operator>(Time&);  bool operator>=(Time&);  bool operator==(Time&);  private:  bool isValidTime(int, int, int); // 유효한 Time 정보인지 판단  int hour;  int min;  int sec;  };  #endif |
| /\* Time.cpp \*/  #include <iomanip>  #include "Time.h"  ostream& operator<<(ostream& fout, const Time& t) {  fout << "(" << setw(2) << t.hour << ":" << setw(2) << t.min << ":" << setw(2) << t.sec << ")";  return fout;  }  Time::Time()  : hour(0), min(0), sec(0) {  // cout << "constructor for Time << endl;  }  Time::Time(int h, int m, int s)  : hour(h), min(m), sec(s) {  // cout << "constructor for Time << endl;  }  Time& Time::operator+(int s) {  Time tTm(0, 0, 0);  int et = this->elasedSec();  et += s;  tTm.sec = et % 60;  et /= 60;  tTm.min = et % 60;  et /= 60;  tTm.hour = et % 60;  return tTm;  }  int Time::elasedSec() {  return 3600 \* hour + 60 \* min + sec;  }  Time Time::getTime() {  return Time(hour, min, sec);  }  bool Time::operator<(Time& t) {  if (this->elasedSec() < t.elasedSec())  return true;  else  return false;  }  bool Time::operator<=(Time& t) {  if (this->elasedSec() <= t.elasedSec())  return true;  else  return false;  }  bool Time::operator>(Time& t) {  if (this->elasedSec() > t.elasedSec())  return true;  else  return false;  }  bool Time::operator>=(Time& t) {  if (this->elasedSec() >= t.elasedSec())  return true;  else  return false;  }  bool Time::operator==(Time& t) {  if (this->elasedSec() == t.elasedSec())  return true;  else  return false;  }  bool Time::isValidTime(int h, int m, int s) {  if (0 <= h && h <= 24)  if (0 <= m && m <= 60)  if (0 <= s && s <= 60)  return true;  return false;  } |
| /\* Date.h \*/  #ifndef DATE\_H  #define DATE\_H  #include <iostream>  using namespace std;  #define WEEKDAY\_AD01Jan01 MON // the weekday of AD Jan 1.  #define DAYS\_PER\_WEEK 7  class Date {  friend ostream& operator<<(ostream&, const Date&); // 출력  public:  // 생성자 & 소멸자  Date(); // default constructor  Date(int y, int m, int d); // constructor  ~Date(); // destructor  int getWeekDay(); // 요일 반환  int getElapsedDaysFromAD010101() const; // get elapsed days from AD 1. 1. 1.  int getElapsedDaysFromAD010101(Date) const;  // 비교연산자  bool operator<(const Date&) const;  bool operator<=(const Date&) const;  bool operator>(const Date&) const;  bool operator>=(const Date&) const;  bool operator==(const Date&) const;  private:  bool isValidDate(int y, int m, int d); // 유효한 Date 정보인지 판단  int year;  int month;  int day;  };  bool isLeapYear(int y); // check whether the given year y is leap year  int getYearDay(int year, int month, int day); // 해당 연도의 몇번째 일인지 반환  #endif |
| /\* Date.cpp\*/  #include <iomanip>  #include "Date.h"  ostream& operator<<(ostream& fout, const Date& d) {  fout << "(" << setw(2) << d.year << "." << setw(2) << d.month << "." << setw(2) << d.day << ")";  return fout;  }  Date::Date()  : year(0), month(0), day(0) {  // cout << "constructor for Date" << endl;  }  Date::Date(int y, int m, int d)  : year(y), month(m), day(d) {  // cout << "constructor for Date" << endl;  }  Date::~Date() {  // cout << "destructor for Date" << endl;  }  int Date::getWeekDay() {  int weekDay\_AD010101 = 1;  int weekDay;  int elapsedDays = 0;  elapsedDays = getElapsedDaysFromAD010101();  weekDay = (elapsedDays + weekDay\_AD010101 - 1) % 7; // 최초는 월욜이면 추가 안하면 대지않나?  // cout << ", Elapsed days from AD Jan. 1, 1 (" << elapsedDays << ")";  return weekDay;  }  int Date::getElapsedDaysFromAD010101() const {  int yearDay;  int elpsDay = 0;  for (int y = 1; y < this->year; y++) {  if (isLeapYear(y)) elpsDay += 366;  else elpsDay += 365;  }  yearDay = getYearDay(this->year, this->month, this->day);  elpsDay += yearDay;  return elpsDay;  }  int Date::getElapsedDaysFromAD010101(Date d) const {  return d.getElapsedDaysFromAD010101();  }  bool Date::operator<(const Date& d) const {  int tl, tr;  tl = getElapsedDaysFromAD010101();  tr = d.getElapsedDaysFromAD010101();  if (tl < tr) return true;  else return false;  }  bool Date::operator<=(const Date& d) const {  int tl, tr;  tl = getElapsedDaysFromAD010101();  tr = d.getElapsedDaysFromAD010101();  if (tl <= tr) return true;  else return false;  }  bool Date::operator>(const Date& d) const {  int tl, tr;  tl = getElapsedDaysFromAD010101();  tr = d.getElapsedDaysFromAD010101();  if (tl > tr) return true;  else return false;  }  bool Date::operator>=(const Date& d) const {  int tl, tr;  tl = getElapsedDaysFromAD010101();  tr = d.getElapsedDaysFromAD010101();  if (tl >= tr) return true;  else return false;  }  bool Date::operator==(const Date& d) const {  int tl, tr;  tl = getElapsedDaysFromAD010101();  tr = d.getElapsedDaysFromAD010101();  if (tl == tr) return true;  else return false;  }  bool Date::isValidDate(int y, int m, int d) {  int days\_month[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };  if (isLeapYear(y)) days\_month[2] = 29;  if ((m >= 1) && (m <= 12) && (d >= 1) && (d <= days\_month[m])) return true; // 범위 내의 정보인지 확인  else {  cout << "Illegal date! (" << m << ", " << d << ") ==> Program aborted." << endl;  return false;  }  }  bool isLeapYear(int y) {  if ((y % 4 == 0 && y % 100 != 0) || y % 400 == 0)  return true;  else  return false;  }  int getYearDay(int year, int month, int day) {  int days\_month[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };  int yearDay = 0;  if (isLeapYear(year))  days\_month[2] = 29;  for (int i = 1; i < month; i++)  yearDay += days\_month[i];  return yearDay + day;  } |
| /\* Person.h \*/  #ifndef PERSON\_H  #define PERSON\_H  #include <string>  #include "Date.h"  #include "Time.h"  class Person {  friend ostream& operator<< (ostream& fout, const Person& p) { // 출력  fout << " Person [name: " << p.name << "]";  return fout;  }  public:  Person() { name = "noname"; }  Person(string n) { name = n; }  protected:  Date dateOfBirth;  string name;  };  #endif |
| /\* Student.h \*/  #ifndef STUDENT\_H  #define STUDENT\_H  #include "Person.h"  class Student : public Person {  friend ostream& operator<< (ostream&, const Student&);  public:  Student(); // default constructor  Student(int id, string n, Date d, Time avt, double gpa);  bool operator<(Student& st); // compare by arrival time  bool operator<=(Student& st); // compare by arrival time  bool operator>(Student& st); // compare by arrival time  bool operator>=(Student& st); // compare by arrival time  private:  int st\_id;  double gpa;  Time arrivalTime;  };  #endif |
| /\* Student.cpp \*/  #include <iomanip>  #include "Student.h"  ostream& operator<< (ostream& fout, const Student& st) {  fout << "Student [ st\_id: " << setw(5) << st.st\_id;  fout << ", name: " << setw(10) << std::left << st.name;  fout.precision(2);  fout.setf(ios::fixed);  fout.setf(ios::showpoint);  fout << ", date\_of\_birth: " << std::right << st.dateOfBirth;  fout << ", arrival: " << std::right << st.arrivalTime;  fout << ", gpa: " << setw(5) << st.gpa << "]";  return fout;  }  Student::Student()  : Person(), st\_id(0), gpa(0) {  arrivalTime = Time(0, 0, 0);  }  Student::Student(int id, string n, Date d, Time avt, double gpa)  : Person(n), st\_id(id), gpa(gpa) {  dateOfBirth = d;  arrivalTime = avt;  }  bool Student::operator<(Student& st) {  if (this->arrivalTime < st.arrivalTime)  return true;  else  return false;  }  bool Student::operator<=(Student& st) {  if (this->arrivalTime <= st.arrivalTime)  return true;  else  return false;  }  bool Student::operator>(Student& st) {  if (this->arrivalTime > st.arrivalTime)  return true;  else  return false;  }  bool Student::operator>=(Student& st) {  if (this->arrivalTime >= st.arrivalTime)  return true;  else  return false;  } |
| /\* Template class T\_Array.h \*/  #ifndef T\_Array\_H  #define T\_Array\_H  #include <iostream>  #include <iomanip>  #include "Time.h"  using namespace std;  enum SortingOrder { INCREASING, DECREASING };  template<typename T>  class T\_Array {  public:  T\_Array(int n, string nm); // constructor  ~T\_Array(); // destructor  string getName() { return name; }  void reserve(int new\_capacity);  void insert(int i, T element);  void remove(int i);  T& at(int i);  void set(int i, T& element);  void shuffle();  void selection\_sort(SortingOrder sortOrder = INCREASING);  void quick\_sort(SortingOrder sortOrder = INCREASING);  void fprint(ofstream& fout, int elements\_per\_line);  bool isValidIndex(int i);  T& operator[](int index) { return t\_array[index]; }  private:  T\* t\_array;  int num\_elements;  int capacity;  string name;  };  template<typename T>  T\_Array<T>::T\_Array(int new\_capacity, string nm) { // constructor  t\_array = (T\*) new T[new\_capacity];  if (t\_array == NULL) {  cout << "Error in creation of dynamic array of size (" << new\_capacity << ") !!" << endl;  exit;  }  capacity = new\_capacity;  num\_elements = 0;  name = nm;  }  template<typename T>  T\_Array<T>::~T\_Array() { // destructor  if (t\_array != NULL)  delete[] t\_array;  }  template<typename T>  bool T\_Array<T>::isValidIndex(int index) {  if ((index < 0) || (index > num\_elements))  return false;  else  return true;  }  template<typename T>  void T\_Array<T>::reserve(int new\_capacity) {  if (capacity >= new\_capacity)  return; // already big enough  T\* t\_newGA = (T\*) new T[new\_capacity];  if (t\_newGA == NULL)  {  cout << "Error in creation of dynamic array of size (" << new\_capacity << ") !!" << endl;  exit;  }  cout << this->getName() << " expands capacity to " << setw(3)  << new\_capacity << endl;  for (int i = 0; i < num\_elements; i++)  t\_newGA[i] = t\_array[i];  delete[] t\_array;  t\_array = t\_newGA;  capacity = new\_capacity;  }  template<typename T>  void T\_Array<T>::insert(int i, T new\_element) {  if (num\_elements >= capacity) // full ?  {  int new\_capa;  new\_capa = ((2 \* capacity) > 1) ? 2 \* capacity : 1;  reserve(new\_capa);  }  if (isValidIndex(i))  {  for (int j = num\_elements - 1; j >= i; j--)  t\_array[j + 1] = t\_array[j]; //shift up elements in one position  t\_array[i] = new\_element;  num\_elements++;  }  }  template<typename T>  void T\_Array<T>::remove(int i) {  if (isValidIndex(i))  {  for (int j = i + 1; j < num\_elements; j++)  t\_array[j - 1] = t\_array[j]; //shift down elements in one position  num\_elements--;  }  if (num\_elements < (capacity / 2))  {  int new\_capacity = capacity / 2;  T\* t\_newGA = (T\*) new T[new\_capacity];  if (t\_newGA == NULL)  {  return; // new memory allocation failed.  // Just return without modification.  }  cout << this->getName()  << " reduces capacity to " << setw(3)  << new\_capacity << endl;  for (int i = 0; i < num\_elements; i++)  t\_newGA[i] = t\_array[i];  delete[] t\_array;  t\_array = t\_newGA;  capacity = new\_capacity;  }  }  template<typename T>  T& T\_Array<T>::at(int i) {  if (isValidIndex(i))  return t\_array[i];  }  template<typename T>  void T\_Array<T>::set(int i, T& element) {  if (isValidIndex(i))  t\_array[i] = element;  }  template<typename T>  void T\_Array<T>::shuffle() {  srand(time(0));  int index1, index2;  int rand\_1, rand\_2;  T temp;  for (int i = 0; i < num\_elements; i++) {  rand\_1 = rand();  rand\_2 = rand();  index1 = ((rand\_1 << 15) | rand\_2) % num\_elements;  rand\_1 = rand();  rand\_2 = rand();  index2 = ((rand\_1 << 15) | rand\_2) % num\_elements;  temp = t\_array[index1];  t\_array[index1] = t\_array[index2];  t\_array[index2] = temp;  }  }  template<typename T>  void T\_Array<T>::selection\_sort(SortingOrder sortOrder) {  int index\_min, index\_max; // index of the element with minimum value  T minValue; // minimum value  T maxValue;  for (int i = 0; i < num\_elements - 1; i++)  {  if (sortOrder == INCREASING) { // sorting in increasing (non\_decreasing) order  index\_min = i;  minValue = t\_array[i];  for (int j = i + 1; j < num\_elements; j++)  {  if (t\_array[j] < minValue) // T must provide operator<() overloading !!  {  index\_min = j;  minValue = t\_array[j];  }  }  if (index\_min != i) // if a smaller element is found, then swap  {  /\* minValue is t\_array[min] \*/  t\_array[index\_min] = t\_array[i];  t\_array[i] = minValue;  }  }  else { // sorting in decreasing (non\_increasing) order  index\_max = i;  maxValue = t\_array[i];  for (int j = i + 1; j < num\_elements; j++)  {  if (t\_array[j] > maxValue) // T must provide operator>() overloading !!  {  index\_max = j;  maxValue = t\_array[j];  }  }  if (index\_max != i) // if a smaller element is found, then swap  {  /\* maxValue is t\_array[max] \*/  t\_array[index\_max] = t\_array[i];  t\_array[i] = maxValue;  }  }  } // end for  }  template<typename T>  int \_partition(T\* array, int size, int left, int right, int pivotIndex,  SortingOrder sortOrder=INCREASING) {  T pivotValue, temp; // pivot value  int newPI; // new pivot index  /\* place the pivot element at right-position \*/  pivotValue = array[pivotIndex];  array[pivotIndex] = array[right];  array[right] = pivotValue; // Move pivot to array[right]  newPI = left; // newPI is the index that points the position  // where pivot element will be finally re-located  for (int i = left; i <= (right - 1); i++) {  if (sortOrder == INCREASING) // sorting in increasing order  {  if (array[i] <= pivotValue) // T must provide operator<=() overloading !!  {  temp = array[i];  array[i] = array[newPI];  array[newPI] = temp;  newPI = newPI + 1;  // note: all elements in left of index newPI are equal or smaller than pivot\_value  }  }  else // sorting in decreasing (non\_increasing) order  {  if (array[i] > pivotValue) // T must provide operator>() overloading !!  {  temp = array[i];  array[i] = array[newPI];  array[newPI] = temp;  newPI = newPI + 1;  // note: all elements in left of index newPI are greater than pivot\_value  }  }  } // end for  // swap array[newPI] and array[right]; Move pivot element to its final place  temp = array[newPI];  array[newPI] = array[right];  array[right] = temp;  return newPI;  }  template<typename T>  void \_quick\_sort(T\* array, int size, int left, int right,  SortingOrder sortOrder=INCREASING) {  int pI, newPI; // pivot index  if (left >= right)  {  return;  }  else  {//select a pI (pivotIndex) in the range left ≤ pI ≤ right  pI = (left + right) / 2;  }  newPI = \_partition(array, size, left, right, pI, sortOrder);  if (left < (newPI - 1)) {  \_quick\_sort(array, size, left, newPI - 1, sortOrder);  // recursively sort elements on the left of pivotNewIndex  }  if ((newPI + 1) < right) {  \_quick\_sort(array, size, newPI + 1, right, sortOrder);  // recursively sort elements on the right of pivotNewIndex  }  }  template<typename T>  void T\_Array<T>::quick\_sort(SortingOrder sortOrder) {  int pI, newPI; // pivot index  \_quick\_sort(this->t\_array, num\_elements, 0, num\_elements - 1, sortOrder);  }  template<typename T>  void T\_Array<T>::fprint(ofstream& fout, int elements\_per\_line)  {  int count = 0;  while (count < num\_elements)  {  for (int i = 0; i < elements\_per\_line; i++)  {  fout << t\_array[count] << " ";  count++;  if (count % elements\_per\_line == 0)  fout << endl;  }  }  fout << endl;  }  #endif |
|  |