객체지향프로그래밍과 자료구조 (실습)

Lab 7. (보충설명) STL Vector, Generic Template Array, Generic Algorithm

정보통신공학과

교수 김 영 탁

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Outline

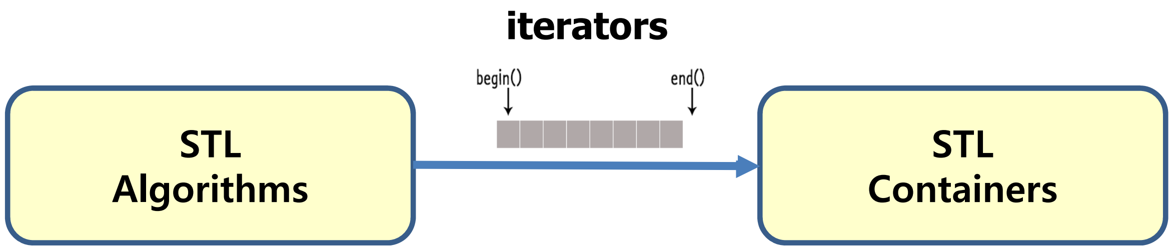
- **Standard Template Library**
- **Template class for Generic Array with Generic Algorithms**
  - Creation of Template Dynamic Array
- **Applications of Generic Array**
  - insert()
  - remove()
  - printSample()
  - selection_sort()
  - quick_sort()
  - merge_sort()
- **Oral Tests**

Standard Template Library (STL)

Standard Template Library (STL)

◆ Standard Template Library (STL)

- Algorithm: sort(), binary_search(), random_shuffle()
- Container
  - Sequence container: vector, list, slist, deque
  - Container Adaptor: queue, stack
  - Associative container: set, map
- Iterator



sort()
random_shuffle()

sequence container(vector, list, deque)
container adaptor(queue, stack)
associative container(set, map)

STL Algorithm

◆ **<algorithm>** **(참조 http://www.cplusplus.com/reference/algorithm/)**

Category	Library Operations (example)
Non-modifying sequence operations	find(), count(), search(), equal()
Modifying sequence operations	copy(), swap(), fill(), generate(), reverse(), shuffle()
Partitions	partition()
Sorting	sort(), partial_sort(), is_sorted()
Binary search	binary_search()
Merge	merge(), set_union()
Heap	make_heap(), sort_heap()
Min / Max	min(), max()
Other	lexicographical_compare(), next_permutation(), prev_permutation()

STL Sequences (ordered collections)

Sequences (arrays/linked lists): ordered collections	
vector	배열과 같은 동적 배열로서 객체를 삽입하거나 제거할 때 자동으로 자신의 크기를 조정하는 능력을 갖는다. vector의 end에 요소를 삽입하거나, 마지막 요소를 제거하는 동작은 단지 일정한 시간을 필요로 하는데, 크기 조정이 일어나지 않기 때문이다. 그러나 처음 또는 중간에 새로운 요소를 삽입하거나 기존 요소를 삭제하는 것은 그 삽입/삭제 대상 요소 이후의 모든 요소들에 대한 재배치 동작을 실행해야하기 때문에 전체 요소 개수에 비례하는 시간이 걸린다.
list	이중 연결 리스트 (doubly linked list)이며, 각 요소들은 근접하지 않는 메모리에 저장될 수 있다. 전체 원소 개수에 선형으로 비례하는 접근 시간과 느린 검색 성능을 갖지만, 한 번 위치가 찾아지면 짧은 시간 내에 삽입과 제거가 가능하다.
slist	단일 연결 리스트 (singly linked list) 이며, 각 요소들이 근접하지 않는 메모리에 저장될 수 있다. 전체 원소 개수에 선형으로 비례하는 접근 시간과 느린 검색 성능을 갖지만, 한 번 위치가 찾아지면 짧은 시간 내에 삽입과 제거가 가능하다. slist는 이중 연결 리스트보다 삽입과 제거에서 약간 효율적이며 적은 메모리를 사용한다. 하지만 단지 전방향 (forward)으로만 진행할 수 있다.
deque	double-ended queue 시작과 끝에서 삽입과 삭제 기능을 상수 시간에 수행

STL Container Adaptors

Container Adaptor	
queue	FIFO (first in first out) Queue 인터페이스를 push/pop/front/back 연산들로 제공한다. front(), back(), push_back(), 그리고 pop_front() 연산을 지원하는 어느 시퀀스 (예를 들면 list 그리고 deque) 들도 큐의 구현에 사용될 수 있다
priority queue	우선순위 큐 인터페이스를 push/pop/top 연산들로 제공한다 (높은 우선순위 요소가 위에 존재한다). 연산 front(), push_back(), 그리고 pop_back() 를 지원하는 어느 임의 접근 시퀀스도 priority_queue를 인스턴스화하는데 사용될 수 있다(예를 들면 vector , deque). 이것은 힙을 사용하여 구현된다. 어느 요소가 높은 우선순위를 갖는 지와 먼저 pop되어야 하는지를 결정하기 위한 비교 (comparison) 기능을 지원해야 한다
stack	LIFO (last in first out) stack 인터페이스를 push/pop/top 연산들을 통해 지원한다. 마지막에 삽입된 요소가 top에 존재한다. back(), push_back(), 그리고 pop_back()을 지원하는 어떤 시퀀스 자료구조 (예를 들면 vector, list, 그리고 deque)도 stack으로 사용할 수 있다.

STL Associative Containers (unordered collections)

Associative containers: unordered collections	
Set (집합)	집합 연산 union, intersection, difference, symmetric difference 그리고 inclusion 테스트를 제공한다. 데이터의 형은 반드시 비교 연산자 < 또는 명시된 비교연산자 함수를 통해서 구현되어야 한다; 이러한 비교 연산 또는 비교연산자 함수는 반드시 strict weak ordering을 보장해야 하며, 그렇지 않으면 행위가 정의되지 않는다. 일반적으로 자가 균형 (self-balancing) 이진 탐색 트리를 사용해서 구현된다. set에서 요소를 삽입하고 제거하는 것은 set에서 가리키는 iterator를 무효화하지 않는다.
multiset	set과 같지만 중복 요소들을 허용하며, 수학적으로 중복집합(Multiset)에 해당한다.
Map (맵)	연관 배열이며, 한 데이터 아이템(키)에서 다른 것(값)으로 매핑 (mapping)을 허용한다. 키의 타입은 반드시 비교 연산 < 또는 사용자가 정의한 비교연산자를 통해 구현되어야 한다; 이러한 비교 연산 또는 비교연산자 함수는 반드시 strict weak ordering을 보장해야 하며, 아니면 행위가 정의되지 않는다. 일반적으로 자가 균형 (self-balancing) 이진 탐색 트리를 사용해서 구현된다.
multimap	map과 같지만 중복 키들을 허용한다.

STL Vector

- **Similar to array:**
  - Has base type
  - Stores collection of base type values
  - Recall: arrays are fixed size
  - Vectors: "arrays that can grow and shrink‟ during program execution
  - vectors are formed from a template class in the Standard Template Library (STL)
- **Collections and Vectors**
  - a collection S of n elements stored in a certain linear order
    - list, sequence
    - index (or rank) of range [0, n-1]
  - vector: a sequence that supports access to its elements by their indices
- **Declared differently:**
  - Syntax: vector<Base_Type>
    - Indicates template class
    - Any type can be "plugged in" to Base_Type
    - Produces "new" class for vectors with that type
  - Example declaration:

```
vector<int> v;
typename vector<int>::iterator p;   p = v.begin();
```

STL Vector 주요 멤버 함수

STL vector 주요 멤버함수	설명

begin()
end()
	첫 번째 원소를 가리키는 반복자 (iterator)를 반환
	마지막 원소 다음 (after end) 위치 를 가리키는 반복자(iterator)를 반환

at(n)
operator[n]
front()
back()
	n 번째 원소를 참조할 때 사용, 범위 점검을 함
	n 번째 원소를 참조할 때 사용, 범위 점검을 하지 않음
	첫 번째 원소의 참조를 반환
	마지막 원소의 참조를 반환

empty()
size()
resize()
capacity()
reserve()
	원소가 아무것도 없으면 true, 하나라도 존재하면 false
	원소의 개수를 반환
	vector의 크기를 변경하고, default 값이나 임의의 값으로 초기화
	vector에 할당된 메모리의 크기를 반환
	지정한 크기 만큼의 메모리를 미리 할당

clear()
erase()
insert()
push_front()
pop_front()
push_back()
pop_back()
swap()
sort()
random_shuffle()
	vector의 모든 원소를 제거
	지정된 위치의 원소나 지정된 범위의 원소를 삭제
	지정된 위치에 지정된 값을 삽입
	제공되지 않음
	제공되지 않음
	vector의 끝에 원소를 추가
	vector의 마지막 원소를 제거
	v1.swap(v2)일 때 v1과 v2를 swap시킴
	제공되지 않음, <algorithm> 라이브러리에서 제공
	제공되지 않음, <algorithm> 라이브러리에서 제공

class Time

```
/* Time.h */
#ifndef TIME_H
#define TIME_H
#include <iostream>

using namespace std;

class Time
{
    friend ostream& operator<<(ostream&, const Time&);
public:
    Time() { hour = 0; min = 0; sec = 0; } // default constructor
    Time(int h, int m, int s);
    int elasedSec() const;
    Time getTime() const { return Time(hour, min, sec); }
    bool operator<(const Time&) const;
    bool operator<=(const Time&) const;
    bool operator>(const Time&) const;
    bool operator>=(const Time&) const;
    bool operator==(const Time&) const;
private:
    bool isValidTime(int, int, int);
    int hour;
    int min;
    int sec;
};
#endif
```

class Date

```
/* 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&);
public:
    Date();  // default constructor
    Date(int y, int m, int d); // constructor
    ~Date(); // destructor
    int getWeekDay();
    int getElapsedDaysFromAD010101() const; //  get elapsed days from AD 1. 1. 1.
    int getElapsedDaysFromAD010101(Date) const;
    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);
    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);
#endif
```

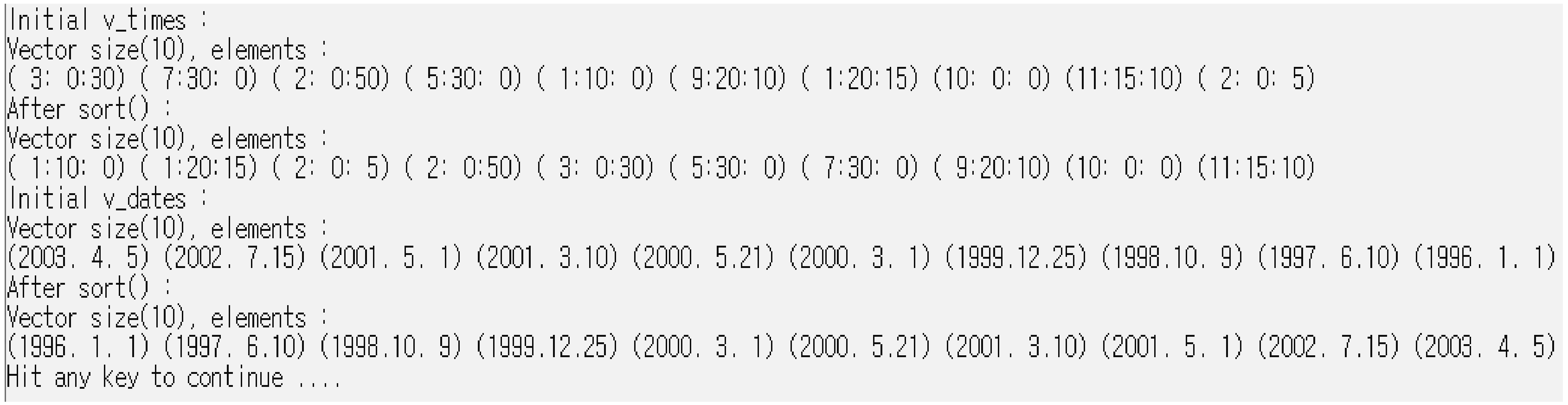
VectorHandler.h

```
/* VectorHandler*/
#ifndef VECTOR_HANDLER_H
#define VECTOR_HANDLER_H
#include <vector>
#include <algorithm>

template<typename T>
void printVector(vector<T>& v)
{
    string typeName = typeid(T).name();
    cout << "Vector size(" << v.size() << "), elements : ₩n";
    typename vector<T>::iterator p;
    for (p = v.begin(); p != v.end(); p++)
    {
        cout << *p << " ";
        if ((typeName == "class Date") || (typeName == "class Time"))
            continue;
        else
            cout << endl;
    }
    cout << endl;
}
#endif
```

Using vector for class Time and class Date

```
/** main - T_Array<Student, K>  (1) */

#include <iostream>
#include <conio.h> # for _getch()
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include "VectorHandler.h"
using namespace std;

#define NUM_TIMES 10
#define NUM_DATES 10
#define NUM_STUDENTS 10

void main()
{
    Time times[NUM_TIMES] =
    {
        Time(3, 0, 30), Time(7, 30, 0), Time(2, 0, 50), Time(5, 30, 0), Time(1, 10, 0),
        Time(9, 20, 10), Time(1, 20, 15), Time(10, 0, 0), Time(11, 15, 10), Time(2, 0, 5)
    };
/** main - T_Array<Student, K>  (2) */

    vector<Time> v_times(times, times+ NUM_TIMES);
    cout << "Initial v_times : " << endl;
    printVector(v_times);
    sort(v_times.begin(), v_times.end());
    cout << "After sort() : " << endl;
    printVector(v_times);

    Date dates[10] =
    {
        Date(2003, 4, 5), Date(2002, 7, 15), Date(2001, 5, 1), Date(2001, 3, 10), Date(2000, 5, 21),
        Date(2000, 3, 1), Date(1999, 12, 25), Date(1998, 10, 9), Date(1997, 6, 10), Date(1996, 1, 1)
    };
    vector<Date> v_dates(dates, dates + NUM_DATES);
    cout << "Initial v_dates :" << endl;
    printVector(v_dates);
    sort(v_dates.begin(), v_dates.end());
    cout << "After sort() : " << endl;
    printVector(v_dates);

    cout << "Hit any key to continue .... ";
    _getch();
    cout << endl;
}
```

실행 결과



```
Initial v_times :
Vector size(10), elements :
( 3: 0:30) ( 7:30: 0) ( 2: 0:50) ( 5:30: 0) ( 1:10: 0) ( 9:20:10) ( 1:20:15) (10: 0: 0) (11:15:10) ( 2: 0: 5)
After sort() :
Vector size(10), elements :
( 1:10: 0) ( 1:20:15) ( 2: 0: 5) ( 2: 0:50) ( 3: 0:30) ( 5:30: 0) ( 7:30: 0) ( 9:20:10) (10: 0: 0) (11:15:10)
Initial v_dates :
Vector size(10), elements :
(2003. 4. 5) (2002. 7.15) (2001. 5. 1) (2001. 3.10) (2000. 5.21) (2000. 3. 1) (1999.12.25) (1998.10. 9) (1997. 6.10) (1996. 1. 1)
After sort() :
Vector size(10), elements :
(1996. 1. 1) (1997. 6.10) (1998.10. 9) (1999.12.25) (2000. 3. 1) (2000. 5.21) (2001. 3.10) (2001. 5. 1) (2002. 7.15) (2003. 4. 5)
Hit any key to continue ....
```

class Person

```
/* Person.h */

#ifndef PERSON_H
#define PERSON_H
#include <iostream>
#include <string>
#include "Date.h"
#include "Time.h"

class Person
{
    friend ostream& operator<< (ostream& fout, const Person& p)
    {
        fout << " Person [name: " << p.name << "]";
        return fout;
    }
public:
    Person() { name = "nobody"; }
    Person(string n) { name = n; }
    void setName(string n) { name = n; }
    string getName() { return name; }
    void setDoB(Date dob) { dateOfBirth = dob; }
    const Date getDoB() const { return dateOfBirth; }
    void setArrivalTime(Time t) { arrivalTime = t; }
    const Time getArrivalTime() const { return arrivalTime; }
protected:
    string name;
    Date dateOfBirth; // date of birth
    Time arrivalTime;
};
#endif
```

class Student

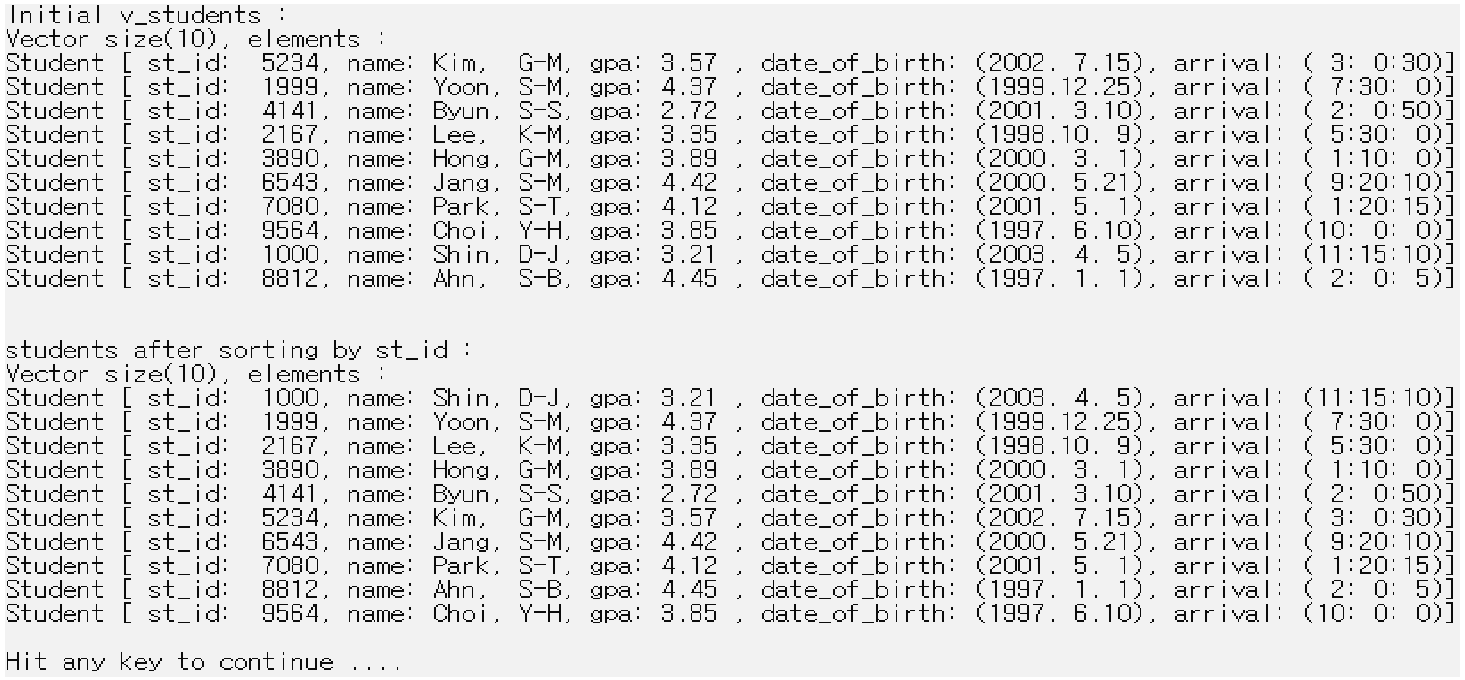
```
/* Student.h */

#ifndef STUDENT_H
#define STUDENT_H

#include "Person.h"
#include "Date.h"
#include "Time.h"
class Student : public Person
{
    friend ostream & operator<< (ostream &, Student &);
public:
    Student();  // default constructor
    Student(int s_id, string n, Date dob, Time avt, double gpa);
    //void getKey(string keyName, void* pKey);
    bool operator<(const Student&) const;
    bool operator<=(const Student&) const;
    bool operator>(const Student&) const;
    bool operator>=(const Student&) const;
    bool operator==(const Student&) const;
private:
    int st_id;
    double gpa;
};

#endif
```

```
/* Student.cpp (1) */
#include "Student.h"
#include <iomanip>

ostream &operator<<(ostream &fout, Student &st)
{
    fout << "Student [ st_id: " << setw(5) << st.st_id;
    fout << ", name: " << setw(7) << std::left << st.name;
    fout << ", gpa: ";
    fout.precision(2);
    fout.setf(ios::fixed);
    fout.setf(ios::showpoint);
    fout << setw(5) << st.gpa;
    fout << ", date_of_birth: " << st.dateOfBirth;
    fout << ", arrival: " << std::right << st.arrivalTime << "]";

    return fout;
}

Student::Student()  // default constructor
{
    name = "nobody";
    arrivalTime = Time(0, 0, 0);
    st_id = 0;
    gpa = 0.0;
}
Student::Student(int id, string n, Date dob, Time avt, double g)
{
    name = n;
    dateOfBirth = dob;
    arrivalTime = avt;
    st_id = id;
    gpa = g;
}
```

```
/* Student.cpp (2) */

bool Student::operator<(const Student& other) const
{
     if (this->st_id < other.st_id)
          return true;
     else
          return false;
}
bool Student::operator<=(const Student& other) const
{
     if (this->st_id <= other.st_id)
          return true;
     else
          return false;
}
bool Student::operator>(const Student& other) const
{
     // . . . .
}
bool Student::operator>=(const Student& other) const
{
    // . . . .
}
bool Student::operator==(const Student& other) const
{
     // . . . .
}
```

Using vector for class Student

```
/** main - T_Array<Student, K>  (1) */
#include <iostream>
#include <conio.h> # for _getch()
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include "VectorHandler.h"
using namespace std;

#define NUM_TIMES 10
#define NUM_DATES 10
#define NUM_STUDENTS 10

void main()
{

    /* Step 2 - Testing class Person, Student */
    Student students[NUM_STUDENTS] =
    {
        Student(5234, string("Kim,  G-M"), Date(2002, 7, 15), Time(3, 0, 30), 3.57),
        Student(1999, string("Yoon, S-M"), Date(1999, 12, 25), Time(7, 30, 0),  4.37),
        Student(4141, string("Byun, S-S"), Date(2001, 3, 10), Time(2, 0, 50), 2.72),
        Student(2167, string("Lee,  K-M"), Date(1998, 10, 9), Time(5, 30, 0), 3.35),
        Student(3890, string("Hong, G-M"), Date(2000, 3, 1), Time(1, 10, 0),  3.89),
        Student(6543, string("Jang, S-M"), Date(2000, 5, 21), Time(9, 20, 10),  4.42),
        Student(7080, string("Park, S-T"), Date(2001, 5, 1), Time(1, 20, 15),  4.12),
        Student(9564, string("Choi, Y-H"), Date(1997, 6, 10), Time(10, 0, 0),  3.85),
        Student(1000, string("Shin, D-J"), Date(2003, 4, 5), Time(11, 15, 10),  3.21),
        Student(8812, string("Ahn,  S-B"), Date(1997, 1, 1), Time(2, 0, 5), 4.45),
    };
/** main - T_Array<Student, K>  (2) */


    vector<Student> v_students(students, students + NUM_DATES);
    cout << "Initial v_students :" << endl;
    printVector(v_students);
    sort(v_students.begin(), v_students.end());
    cout << "₩nstudents after sorting by st_id :" << endl;
    printVector(v_students);
    cout << "Hit any key to continue .... ";
    _getch();
    cout << endl;
}
```



```
Initial v_students :
Vector size(10), elements :
Student [ st_id:  5234, name: Kim,  G-M, gpa: 3.57 , date_of_birth: (2002. 7.15), arrival: ( 3: 0:30)]
Student [ st_id:  1999, name: Yoon, S-M, gpa: 4.37 , date_of_birth: (1999.12.25), arrival: ( 7:30: 0)]
Student [ st_id:  4141, name: Byun, S-S, gpa: 2.72 , date_of_birth: (2001. 3.10), arrival: ( 2: 0:50)]
Student [ st_id:  2167, name: Lee,  K-M, gpa: 3.35 , date_of_birth: (1998.10. 9), arrival: ( 5:30: 0)]
Student [ st_id:  3890, name: Hong, G-M, gpa: 3.89 , date_of_birth: (2000. 3. 1), arrival: ( 1:10: 0)]
Student [ st_id:  6543, name: Jang, S-M, gpa: 4.42 , date_of_birth: (2000. 5.21), arrival: ( 9:20:10)]
Student [ st_id:  7080, name: Park, S-T, gpa: 4.12 , date_of_birth: (2001. 5. 1), arrival: ( 1:20:15)]
Student [ st_id:  9564, name: Choi, Y-H, gpa: 3.85 , date_of_birth: (1997. 6.10), arrival: (10: 0: 0)]
Student [ st_id:  1000, name: Shin, D-J, gpa: 3.21 , date_of_birth: (2003. 4. 5), arrival: (11:15:10)]
Student [ st_id:  8812, name: Ahn,  S-B, gpa: 4.45 , date_of_birth: (1997. 1. 1), arrival: ( 2: 0: 5)]


students after sorting by st_id :
Vector size(10), elements :
Student [ st_id:  1000, name: Shin, D-J, gpa: 3.21 , date_of_birth: (2003. 4. 5), arrival: (11:15:10)]
Student [ st_id:  1999, name: Yoon, S-M, gpa: 4.37 , date_of_birth: (1999.12.25), arrival: ( 7:30: 0)]
Student [ st_id:  2167, name: Lee,  K-M, gpa: 3.35 , date_of_birth: (1998.10. 9), arrival: ( 5:30: 0)]
Student [ st_id:  3890, name: Hong, G-M, gpa: 3.89 , date_of_birth: (2000. 3. 1), arrival: ( 1:10: 0)]
Student [ st_id:  4141, name: Byun, S-S, gpa: 2.72 , date_of_birth: (2001. 3.10), arrival: ( 2: 0:50)]
Student [ st_id:  5234, name: Kim,  G-M, gpa: 3.57 , date_of_birth: (2002. 7.15), arrival: ( 3: 0:30)]
Student [ st_id:  6543, name: Jang, S-M, gpa: 4.42 , date_of_birth: (2000. 5.21), arrival: ( 9:20:10)]
Student [ st_id:  7080, name: Park, S-T, gpa: 4.12 , date_of_birth: (2001. 5. 1), arrival: ( 1:20:15)]
Student [ st_id:  8812, name: Ahn,  S-B, gpa: 4.45 , date_of_birth: (1997. 1. 1), arrival: ( 2: 0: 5)]
Student [ st_id:  9564, name: Choi, Y-H, gpa: 3.85 , date_of_birth: (1997. 6.10), arrival: (10: 0: 0)]

Hit any key to continue ....
```

Template Array (T_Array)

Template Class for Generic Array with Generic Algorithms

```
/* Template class T_Array.h (1) */
#ifndef T_Array_H
#define T_Array_H
#include <iostream>
#include <iomanip>
#include "Date.h"
#include "Time.h"
using namespace std;
enum SortingOrder { INCREASING, DECREASING };

template<typename T, typename K>
class T_Array
{
public:
     T_Array(int n, string nm); // constructor
     ~T_Array(); // destructor
     int size() { return num_elements; }
     bool empty() { return num_elements == 0; }
     string getName() { return name; }
     void insert(int i, T element);
     int sequential_search(string keyName, K search_key);
       // search and return the index; -1 if not found
     int binary_search(string keyName, K search_key);
     void selection_sort(string keyName, SortingOrder sortOrder);
     void merge_sort(string keyName, SortingOrder sortOrder);
     void print(int elements_per_line);
     bool isValidIndex(int i);
     T& operator[](int index) { return t_array[index]; }
/* Template class T_Array.h (2) */

private:
    void _mergeSort(T* arr, T* tmp_array, int left, int right, string keyName,
           SortingOrder sortOrder);
    T *t_array;
    int num_elements;
    int capacity;
    string name;
};

template<typename T, typename K>
T_Array<T, K>::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, typename K>
T_Array<T, K>::~T_Array() // destructor
{
    if (t_array != NULL)
        delete[] t_array;
}
```

```
/* Template class T_Array.h (3) */

template<typename T, typename K>
bool T_Array<T, K>::isValidIndex(int index)
{
    if ((index < 0) || (index >= num_elements))
        return false;
    else
        return true;
}

template<typename T, typename K>
void T_Array<T, K>::insert(int i, T new_element)
{
    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 class T_Array.h (4) */

template<typename T, typename K>
int T_Array<T, K>::sequential_search(string keyName, K search_key)
{
    int index;
    string keytype;
    K key;

    for (int index = 0; index < num_elements; index++)
    {
        t_array[index].getKey(keyName, &key);
        if (key == search_key)
            return index;
    }
    return -1;
}
```

```
/* Template class T_Array.h (5) */

template<typename T, typename K>
int T_Array<T, K>::binary_search(string keyName, K search_key)
{
    int low, mid, high;
    int loop = 1;

    low = 0;
    high = num_elements - 1;
    K mid_key;

    while (low <= high)
    {
        cout << setw(2) << loop << "-th loop: current search range [" << setw(3)
               << low << ", " << setw(3) << high << "]" << endl;
        mid = (low + high) / 2;
        t_array[mid].getKey(keyName, &mid_key);
        if (mid_key== search_key)
            return mid;
        else if (mid_key> search_key)
            high = mid - 1;
        else
            low = mid + 1;
        loop++;
    }
}
```

```
/* Template class T_Array.h (6) */

template<typename T, typename K>
void T_Array<T, K>::selection_sort(string keyName, SortingOrder sortOrder)
{
    int index_min, index_max; // index of the element with minimum value
    T tempElement;

    K minKey, maxKey, key;
    for (int i = 0; i < num_elements - 1; i++)
    {
        if (sortOrder == INCREASING)
        { // sorting in increasing (non_decreasing) order
            index_min = i;
            t_array[i].getKey(keyName, &key);
            minKey = (K)key;
            for (int j = i + 1; j < num_elements; j++)
            {
                t_array[j].getKey(keyName, &key);
                if ((K)key < minKey)
                {
                    index_min = j;
                    minKey = (K)key;
                }
            }
            if (index_min != i) // if a smaller element is found, then swap
            {
                tempElement = t_array[index_min];
                t_array[index_min] = t_array[i];
                t_array[i] = tempElement;
            }
        }
/* Template class T_Array.h (7) */

        else
        { // sorting in decreasing (non_increasing) order
            index_max = i;
            t_array[i].getKey(keyName, &key);
            maxKey = (K)key;
            for (int j = i + 1; j < num_elements; j++)
            {
                t_array[j].getKey(keyName, &key);
                if ((K)key > maxKey)
                {
                    index_max = j;
                    maxKey = (K)key;
                }
            }
            if (index_max != i) // if a smaller element is found, then swap
            {
                tempElement = t_array[index_max];
                t_array[index_max] = t_array[i];
                t_array[i] = tempElement;
            }
        }
    }  // end for
}
```

```
/* Template class T_Array.h (8) */
template<typename T, typename K>
void T_Array<T, K>::_mergeSort(T* arr, T* tmp_array, int left, int right, string keyName,
        SortingOrder sortOrder)
{
    T entry;
    K key_i, key_j;
    //printf("... invoke MergeSort(left=%d, right=%d):₩n", left, right);

    if (left >= right)
        return;
    int i, j, k, mid = (left + right) / 2;
    _mergeSort(arr, tmp_array, left, mid, keyName, sortOrder);
    _mergeSort(arr, tmp_array, mid + 1, right, keyName, sortOrder);

            . . . . . // 이부분은 직접 구현할 것
}

template<typename T, typename K>
void T_Array<T, K>::merge_sort(string keyName, SortingOrder sortOrder)
{
    T* tmp_array = new T[num_elements];
    if (tmp_array == NULL)
    {
        cout << "Error in creation of tmp_array (size = %d) in mergeSort() !!!" << endl;
        exit;
    }
    _mergeSort(t_array, tmp_array, 0, num_elements - 1, keyName, sortOrder);
    delete[] tmp_array;
}
```

```
/* Template class T_Array.h (9) */

template<typename T, typename K>
void T_Array<T, K>::print(int elements_per_line)
{
    int count = 0;
    while (count < num_elements)
    {
        for (int i = 0; i < elements_per_line; i++)
        {
            cout << t_array[count] << "   ";
            count++;
            if (count % elements_per_line == 0)
                cout << endl;
        }
    }
    cout << endl;
}
#endif
```

Template Class의 Generic 알고리즘사용을 위한 class Student의 기능 추가

```
/* Student.h */

#ifndef STUDENT_H
#define STUDENT_H

#include "Person.h"
#include "Date.h"
#include "Time.h"
class Student : public Person
{
    friend ostream & operator<< (ostream &, Student &);
public:
    Student();  // default constructor
    Student(int s_id, string n, Date dob, Time avt, double gpa);
    void getKey(string keyName, void* pKey);
    bool operator<(const Student&) const;
    bool operator<=(const Student&) const;
    bool operator>(const Student&) const;
    bool operator>=(const Student&) const;
    bool operator==(const Student&) const;
private:
    int st_id;
    double gpa;
};

#endi
```

class Student

```
void Student::getKey(string keyName, void* pKey)
{
    if (keyName == "ST_ID")
        *(int *)pKey = this->st_id;
    else if (keyName == "ST_NAME")
        *(string *)pKey = this->name;
    else if (keyName == "GPA")
        *(double *)pKey = this->gpa;
    else if (keyName == "ARRIVAL_TIME")
        *(Time *)pKey = this->arrivalTime;
    else if (keyName == "BIRTH_DATE")
        *(Date *)pKey = this->dateOfBirth;
    else
        pKey = NULL;
}
```

Template Array의
Generic Algorithm 응용 예제

main()

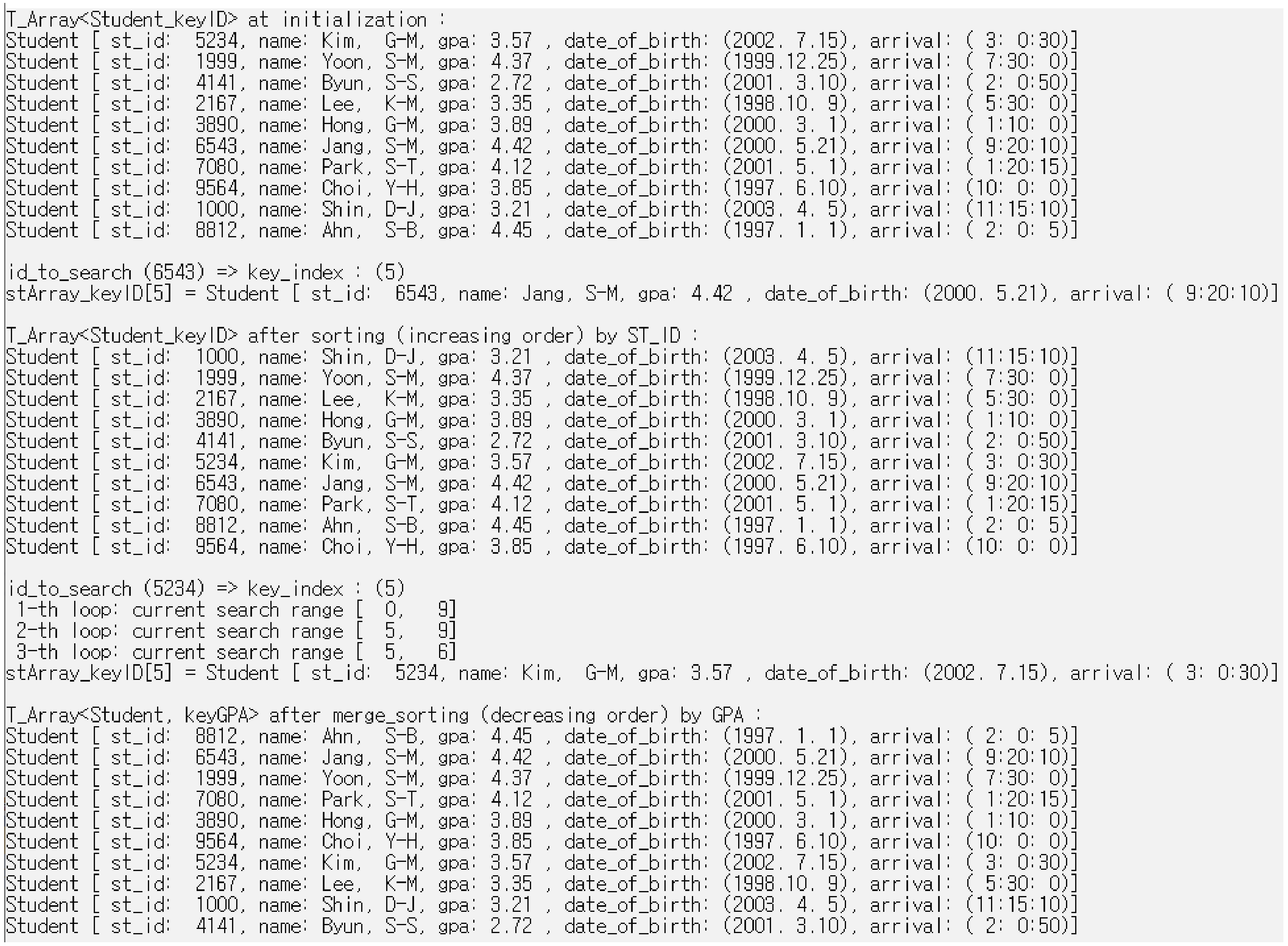
```
/** main - T_Array<Student, K>  (1) */

#include <iostream>
#include <conio.h> # for _getch()
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include "T_Array.h"
#include "Student.h"
#include "VectorHandler.h"
using namespace std;

#define STEP_1
#define STEP_2
#define STEP_3
#define STEP_4
#define NUM_TIMES 10
#define NUM_DATES 10
#define NUM_STUDENTS 10

void main()
{
```

```
Student students[NUM_STUDENTS] =
{
    Student(5234, string("Kim,  G-M"), Date(2002, 7, 15), Time(3, 0, 30), 3.57),
    Student(1999, string("Yoon, S-M"), Date(1999, 12, 25), Time(7, 30, 0),  4.37),
    Student(4141, string("Byun, S-S"), Date(2001, 3, 10), Time(2, 0, 50), 2.72),
    Student(2167, string("Lee,  K-M"), Date(1998, 10, 9), Time(5, 30, 0), 3.35),
    Student(3890, string("Hong, G-M"), Date(2000, 3, 1), Time(1, 10, 0),  3.89),
    Student(6543, string("Jang, S-M"), Date(2000, 5, 21), Time(9, 20, 10),  4.42),
    Student(7080, string("Park, S-T"), Date(2001, 5, 1), Time(1, 20, 15),  4.12),
    Student(9564, string("Choi, Y-H"), Date(1997, 6, 10), Time(10, 0, 0),  3.85),
    Student(1000, string("Shin, D-J"), Date(2003, 4, 5), Time(11, 15, 10),  3.21),
    Student(8812, string("Ahn,  S-B"), Date(1997, 1, 1), Time(2, 0, 5), 4.45),
};
```

```
/** main - T_Array<Student, K>  (3) */

#ifdef STEP_3
    /* Step 3 - Testing template array class T_Array<T, K> */
    Student* pSt;
    T_Array<Student, int> stArray_keyID(NUM_STUDENTS, "T_Array<Student, keyST_ID>");
    for (int i = 0; i < NUM_STUDENTS; i++)
    {
        stArray_keyID.insert(i, students[i]);
    }
    cout << "T_Array<Student_keyID> at initialization : " << endl;
    stArray_keyID.print(1);

    stArray_keyID.selection_sort(string("ST_ID"), INCREASING);
    cout << "₩nT_Array<Student_keyID> after sorting (increasing order) by ST_ID : " << endl;
    stArray_keyID.print(1);

    T_Array<Student, double> stArray_keyGPA(NUM_STUDENTS, "T_Array<Student, keyGPA>");
    for (int i = 0; i < NUM_STUDENTS; i++)
    {
        stArray_keyGPA.insert(i, students[i]);
    }
    stArray_keyGPA.merge_sort(string("GPA"), DECREASING);
    cout << "₩nT_Array<Student, keyGPA> after merge_sorting (decreasing order) by GPA : " << endl;
    stArray_keyGPA.print(1);
    cout << "Hit any key to continue .... ";
    _getch();
    cout << endl;
#endif
```

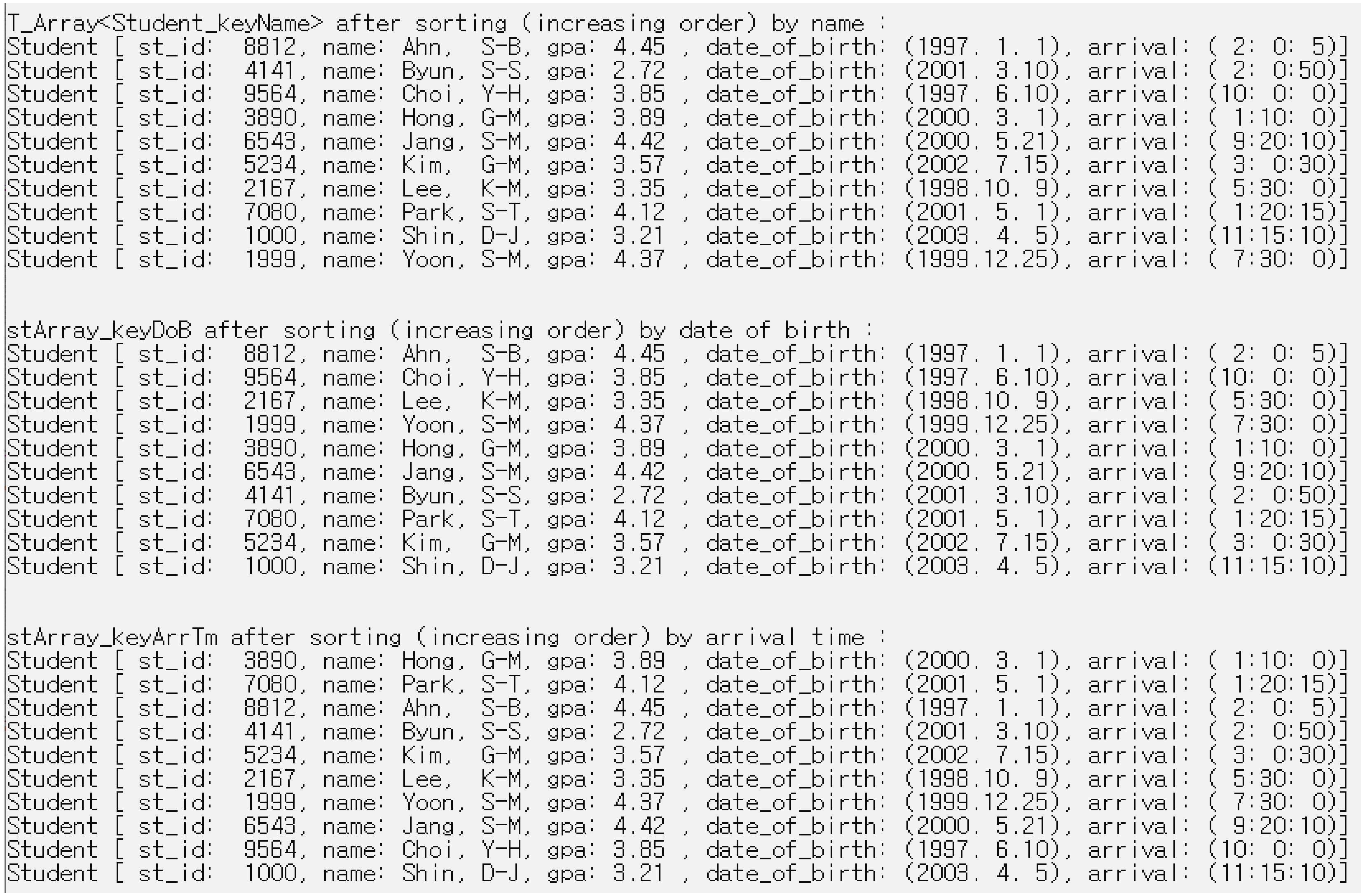
```
/** main - T_Array<Student, K>  (3) */

#ifdef STEP_4
    T_Array<Student, string> stArray_keyName(NUM_STUDENTS, "T_Array<Student,
      keyName>");
    for (int i = 0; i < NUM_STUDENTS; i++)
    {
        stArray_keyName.insert(i, students[i]);
    }

    stArray_keyName.selection_sort(string("ST_NAME"), INCREASING);
    cout << "₩nT_Array<Student_keyName> after sorting (increasing order) by name : " << endl;
    stArray_keyName.print(1);

    T_Array<Student, Date> stArray_keyDoB(NUM_STUDENTS,
      "Array of Students, date of birth as key");
    for (int i = 0; i < NUM_STUDENTS; i++)
    {
        stArray_keyDoB.insert(i, students[i]);
    }
    stArray_keyDoB.selection_sort(string("BIRTH_DATE"), INCREASING);
    cout << "₩nstArray_keyDoB after sorting (increasing order) by date of birth : " << endl;
    stArray_keyDoB.print(1);

    T_Array<Student, Time> stArray_keyTime(NUM_STUDENTS,
      "Array of Students, arrival time as key");
    for (int i = 0; i < NUM_STUDENTS; i++)
    {
        stArray_keyTime.insert(i, students[i]);
    }
    stArray_keyTime.selection_sort(string("ARRIVAL_TIME"), INCREASING);
    cout << "₩nstArray_keyArrTm after sorting (increasing order) by arrival time : " << endl;
    stArray_keyTime.print(1);

#endif
}
```

실행 결과 (1)



```
T_Array<Student_keyID> at initialization :
Student [ st_id:  5234, name: Kim,  G-M, gpa: 3.57 , date_of_birth: (2002. 7.15), arrival: ( 3: 0:30)]
Student [ st_id:  1999, name: Yoon, S-M, gpa: 4.37 , date_of_birth: (1999.12.25), arrival: ( 7:30: 0)]
Student [ st_id:  4141, name: Byun, S-S, gpa: 2.72 , date_of_birth: (2001. 3.10), arrival: ( 2: 0:50)]
Student [ st_id:  2167, name: Lee,  K-M, gpa: 3.35 , date_of_birth: (1998.10. 9), arrival: ( 5:30: 0)]
Student [ st_id:  3890, name: Hong, G-M, gpa: 3.89 , date_of_birth: (2000. 3. 1), arrival: ( 1:10: 0)]
Student [ st_id:  6543, name: Jang, S-M, gpa: 4.42 , date_of_birth: (2000. 5.21), arrival: ( 9:20:10)]
Student [ st_id:  7080, name: Park, S-T, gpa: 4.12 , date_of_birth: (2001. 5. 1), arrival: ( 1:20:15)]
Student [ st_id:  9564, name: Choi, Y-H, gpa: 3.85 , date_of_birth: (1997. 6.10), arrival: (10: 0: 0)]
Student [ st_id:  1000, name: Shin, D-J, gpa: 3.21 , date_of_birth: (2003. 4. 5), arrival: (11:15:10)]
Student [ st_id:  8812, name: Ahn,  S-B, gpa: 4.45 , date_of_birth: (1997. 1. 1), arrival: ( 2: 0: 5)]

id_to_search (6543) => key_index : (5)
stArray_keyID[5] = Student [ st_id:  6543, name: Jang, S-M, gpa: 4.42 , date_of_birth: (2000. 5.21), arrival: ( 9:20:10)]

T_Array<Student_keyID> after sorting (increasing order) by ST_ID :
Student [ st_id:  1000, name: Shin, D-J, gpa: 3.21 , date_of_birth: (2003. 4. 5), arrival: (11:15:10)]
Student [ st_id:  1999, name: Yoon, S-M, gpa: 4.37 , date_of_birth: (1999.12.25), arrival: ( 7:30: 0)]
Student [ st_id:  2167, name: Lee,  K-M, gpa: 3.35 , date_of_birth: (1998.10. 9), arrival: ( 5:30: 0)]
Student [ st_id:  3890, name: Hong, G-M, gpa: 3.89 , date_of_birth: (2000. 3. 1), arrival: ( 1:10: 0)]
Student [ st_id:  4141, name: Byun, S-S, gpa: 2.72 , date_of_birth: (2001. 3.10), arrival: ( 2: 0:50)]
Student [ st_id:  5234, name: Kim,  G-M, gpa: 3.57 , date_of_birth: (2002. 7.15), arrival: ( 3: 0:30)]
Student [ st_id:  6543, name: Jang, S-M, gpa: 4.42 , date_of_birth: (2000. 5.21), arrival: ( 9:20:10)]
Student [ st_id:  7080, name: Park, S-T, gpa: 4.12 , date_of_birth: (2001. 5. 1), arrival: ( 1:20:15)]
Student [ st_id:  8812, name: Ahn,  S-B, gpa: 4.45 , date_of_birth: (1997. 1. 1), arrival: ( 2: 0: 5)]
Student [ st_id:  9564, name: Choi, Y-H, gpa: 3.85 , date_of_birth: (1997. 6.10), arrival: (10: 0: 0)]

id_to_search (5234) => key_index : (5)
 1-th loop: current search range [  0,   9]
 2-th loop: current search range [  5,   9]
 3-th loop: current search range [  5,   6]
stArray_keyID[5] = Student [ st_id:  5234, name: Kim,  G-M, gpa: 3.57 , date_of_birth: (2002. 7.15), arrival: ( 3: 0:30)]

T_Array<Student, keyGPA> after merge_sorting (decreasing order) by GPA :
Student [ st_id:  8812, name: Ahn,  S-B, gpa: 4.45 , date_of_birth: (1997. 1. 1), arrival: ( 2: 0: 5)]
Student [ st_id:  6543, name: Jang, S-M, gpa: 4.42 , date_of_birth: (2000. 5.21), arrival: ( 9:20:10)]
Student [ st_id:  1999, name: Yoon, S-M, gpa: 4.37 , date_of_birth: (1999.12.25), arrival: ( 7:30: 0)]
Student [ st_id:  7080, name: Park, S-T, gpa: 4.12 , date_of_birth: (2001. 5. 1), arrival: ( 1:20:15)]
Student [ st_id:  3890, name: Hong, G-M, gpa: 3.89 , date_of_birth: (2000. 3. 1), arrival: ( 1:10: 0)]
Student [ st_id:  9564, name: Choi, Y-H, gpa: 3.85 , date_of_birth: (1997. 6.10), arrival: (10: 0: 0)]
Student [ st_id:  5234, name: Kim,  G-M, gpa: 3.57 , date_of_birth: (2002. 7.15), arrival: ( 3: 0:30)]
Student [ st_id:  2167, name: Lee,  K-M, gpa: 3.35 , date_of_birth: (1998.10. 9), arrival: ( 5:30: 0)]
Student [ st_id:  1000, name: Shin, D-J, gpa: 3.21 , date_of_birth: (2003. 4. 5), arrival: (11:15:10)]
Student [ st_id:  4141, name: Byun, S-S, gpa: 2.72 , date_of_birth: (2001. 3.10), arrival: ( 2: 0:50)]
```

실행 결과 (2)



```
T_Array<Student_keyName> after sorting (increasing order) by name :
Student [ st_id:  8812, name: Ahn,  S-B, gpa: 4.45 , date_of_birth: (1997. 1. 1), arrival: ( 2: 0: 5)]
Student [ st_id:  4141, name: Byun, S-S, gpa: 2.72 , date_of_birth: (2001. 3.10), arrival: ( 2: 0:50)]
Student [ st_id:  9564, name: Choi, Y-H, gpa: 3.85 , date_of_birth: (1997. 6.10), arrival: (10: 0: 0)]
Student [ st_id:  3890, name: Hong, G-M, gpa: 3.89 , date_of_birth: (2000. 3. 1), arrival: ( 1:10: 0)]
Student [ st_id:  6543, name: Jang, S-M, gpa: 4.42 , date_of_birth: (2000. 5.21), arrival: ( 9:20:10)]
Student [ st_id:  5234, name: Kim,  G-M, gpa: 3.57 , date_of_birth: (2002. 7.15), arrival: ( 3: 0:30)]
Student [ st_id:  2167, name: Lee,  K-M, gpa: 3.35 , date_of_birth: (1998.10. 9), arrival: ( 5:30: 0)]
Student [ st_id:  7080, name: Park, S-T, gpa: 4.12 , date_of_birth: (2001. 5. 1), arrival: ( 1:20:15)]
Student [ st_id:  1000, name: Shin, D-J, gpa: 3.21 , date_of_birth: (2003. 4. 5), arrival: (11:15:10)]
Student [ st_id:  1999, name: Yoon, S-M, gpa: 4.37 , date_of_birth: (1999.12.25), arrival: ( 7:30: 0)]


stArray_keyDoB after sorting (increasing order) by date of birth :
Student [ st_id:  8812, name: Ahn,  S-B, gpa: 4.45 , date_of_birth: (1997. 1. 1), arrival: ( 2: 0: 5)]
Student [ st_id:  9564, name: Choi, Y-H, gpa: 3.85 , date_of_birth: (1997. 6.10), arrival: (10: 0: 0)]
Student [ st_id:  2167, name: Lee,  K-M, gpa: 3.35 , date_of_birth: (1998.10. 9), arrival: ( 5:30: 0)]
Student [ st_id:  1999, name: Yoon, S-M, gpa: 4.37 , date_of_birth: (1999.12.25), arrival: ( 7:30: 0)]
Student [ st_id:  3890, name: Hong, G-M, gpa: 3.89 , date_of_birth: (2000. 3. 1), arrival: ( 1:10: 0)]
Student [ st_id:  6543, name: Jang, S-M, gpa: 4.42 , date_of_birth: (2000. 5.21), arrival: ( 9:20:10)]
Student [ st_id:  4141, name: Byun, S-S, gpa: 2.72 , date_of_birth: (2001. 3.10), arrival: ( 2: 0:50)]
Student [ st_id:  7080, name: Park, S-T, gpa: 4.12 , date_of_birth: (2001. 5. 1), arrival: ( 1:20:15)]
Student [ st_id:  5234, name: Kim,  G-M, gpa: 3.57 , date_of_birth: (2002. 7.15), arrival: ( 3: 0:30)]
Student [ st_id:  1000, name: Shin, D-J, gpa: 3.21 , date_of_birth: (2003. 4. 5), arrival: (11:15:10)]


stArray_keyArrTm after sorting (increasing order) by arrival time :
Student [ st_id:  3890, name: Hong, G-M, gpa: 3.89 , date_of_birth: (2000. 3. 1), arrival: ( 1:10: 0)]
Student [ st_id:  7080, name: Park, S-T, gpa: 4.12 , date_of_birth: (2001. 5. 1), arrival: ( 1:20:15)]
Student [ st_id:  8812, name: Ahn,  S-B, gpa: 4.45 , date_of_birth: (1997. 1. 1), arrival: ( 2: 0: 5)]
Student [ st_id:  4141, name: Byun, S-S, gpa: 2.72 , date_of_birth: (2001. 3.10), arrival: ( 2: 0:50)]
Student [ st_id:  5234, name: Kim,  G-M, gpa: 3.57 , date_of_birth: (2002. 7.15), arrival: ( 3: 0:30)]
Student [ st_id:  2167, name: Lee,  K-M, gpa: 3.35 , date_of_birth: (1998.10. 9), arrival: ( 5:30: 0)]
Student [ st_id:  1999, name: Yoon, S-M, gpa: 4.37 , date_of_birth: (1999.12.25), arrival: ( 7:30: 0)]
Student [ st_id:  6543, name: Jang, S-M, gpa: 4.42 , date_of_birth: (2000. 5.21), arrival: ( 9:20:10)]
Student [ st_id:  9564, name: Choi, Y-H, gpa: 3.85 , date_of_birth: (1997. 6.10), arrival: (10: 0: 0)]
Student [ st_id:  1000, name: Shin, D-J, gpa: 3.21 , date_of_birth: (2003. 4. 5), arrival: (11:15:10)]
```

Check Points for Generic Algorithms on Generic Array (1)

- 일반화된 템플릿 배열에서 일반화된 알고리즘을 사용하기 위하여 필요한 사항
  - 하나의 데이터 레코드의 다양한 항목을 키 값으로 사용할 수 있도록 T_Array<T, K>를 구성할 수 있어야 함
  - 알고리즘에서 사용되는 기준 keyName을 인수로 지정하여, 지정된 키를 기준으로 비교하도록 구현되어 있어야 함
  - 이렇게 동일한 구조의 알고리즘을 다양한 자료형에 적용할 수 있도록 구현하는 것이 일반화된 알고리즘 (Generic Algorithm)이라 함

Check Points for Generic Algorithms on Generic Array (2)

- **Entry<T, K>에서 T 자료형으로 데이터 레코드의 포인터 형을 사용**
  - 동일한 데이터 레코드를 다양한 키 값을 기준으로 처리할 때, T_Array 배열에 해당 데이터 레코드를 내용을 모두 복사하지 않고, 데이터 레코드가 저장되어 있는 곳을 가리키는 포인터를 가지도록 함
  - 데이터 레코드가 큰 규모일 때 메모리 사용을 효율적으로 할 수 있게 하며, 처리시간도 빨라 짐 (데이터 레코드의 전체 복사가 필요 없음)
  - 하나의 키 값을 기준으로 한 처리 결과를 기반으로 공유하고 있는 데이터 레코드를 직접 update 할 수 있으며, update 된 내용을 다른 알고리즘에서도 직접 사용할 수 있음

Lab 7. Oral Test

(1) 클래스 템플릿이 무엇이며, 클래스 템플릿을 사용하는 장점과 단점에 대하여 예를 들어 설명하라.

(2) STL (Standard Template Library)는 무엇이며, 어떤 기능 (자료구조 또는 알고리즘)이 제공되는가에 대하여 표를 만들어 설명하라. (Keypoints: container, algorithm, iterator)

(3) STL vector, STL list, STL deque이 제공하는 주요 멤버함수 들에 대하여 설명하라.

(4) 알고리즘 추상화 (algorithm abstraction)는 무엇이며, 알고리즘 추상화를 어떻게 구현하는가를 대하여 퀵 정렬을 예로 들어 설명하라. 이 일반화된 알고리즘이 다수의 속성을 가지는 클래스 배열에 사용될 수 있게 하고, 알고리즘에 사용되는 기준 키의 이름 (keyName)이 알고리즘 호출에 전달되는 구조를 사용할 것.