

Data Structures

Lab # 13



- 1. Selection sort
- 2. Bubble sort
- 3. Insertion sort
- 4. Heap sort 1
- 5. Heap sort 2

■ 실습에서 사용할 파일

- ❖ E-campus에서 파일 다운로드
- ❖ Student.h , Student.cpp : 학생의 정보를 저장하고 출력하기 위한 객체
- ❖ Swap.cpp : 배열 내에서 학생의 위치를 변경하기 위한 함수가 정의됨

■ 그 외 제공한 샘플 코드는 참고용으로만 사용

- ❖ BubbleSort.h, QuickSort.h, ...

1. Exercise

■ 문 제

- ❖ 학생들의 정보를 담고있는 배열을 **Selection Sort** 기법을 이용하여 정렬하시오.
 - 정렬의 기준 : 이름, 오름차순 (문자열 비교함수의 결과를 이용하여 정렬함)
- ❖ Selection Sort 함수를 SelectionSort.h 파일에 구현할 것

■ 참고 사항

- ❖ 샘플 코드의 SelectionSort.cpp 파일을 참고하여 구현할 것

■ 예 제

id	Name	gpa	정렬 →	id	Name	gpa
2003200111	이웅재	3.0		2004200121	권오준	3.2
2004200121	권오준	3.2		2005200132	김진일	2.7
2005200132	김진일	2.7		2003200111	이웅재	3.0

■ Selection Sort 예제 및 참고 코드

	8
	5
	2
	6
	9
	3
	1
	4
	0
	7

```
int MinIndex(Student values[], int startIndex, int endIndex)
// Post: Returns the index of the smallest value in
//       values[startIndex]..values[endIndex].
{
    int indexOfMin = startIndex;
    for (int index = startIndex + 1; index <= endIndex; index++)
    {
        //Student의 이름을 기준으로 현재 탐색하는학생과 최소값을 갖는 학생을 비교한다.
        if (/*-----*/)
            indexOfMin = index;
    }
    return indexOfMin;
}

void SelectionSort(Student values[], int numValues)
// Post: The elements in the array values are sorted by key.
{
    int endIndex = numValues-1;
    for (int current = 0; current < endIndex; current++)
        Swap(values[current],
              values[MinIndex(values, current, endIndex)]);
}
```

■ 테스트 드라이버

```
#include <iostream>
#include "Student.h"
#include "SelectionSort.h"
using namespace std;

int main()
{
    Student stu[100];
    stu[0].InitValue(2003200111, "이웅재", 3.0);
    stu[1].InitValue(2004200121, "권오준", 3.2);
    stu[2].InitValue(2005200132, "김진일", 2.7);

    SelectionSort(stu, 3);
    Print(cout, stu, 3);
    return 0;
}
```

2. Exercise

■ 문 제

- ❖ 학생들의 정보를 담고있는 배열을 **Bubble Sort** 기법을 이용하여 정렬하시오.
 - 정렬의 기준 : 이름, 오름차순 (문자열 비교함수의 결과를 이용하여 정렬함)
- ❖ BubbleSort함수를 BubbleSort.cpp 파일에 구현할 것

■ 참고 사항

- ❖ BubbleSort.cpp파일을 참고하여 구현할 것

■ Bubble Sort 예제 및 참고 코드

- ❖ 그림은 Bubble down으로 정렬
- ❖ 코드는 Bubble up으로 구현되어 있음

6 5 3 1 8 7 2 4

```
#include "BubbleSort.h"

void BubbleUp(Student values[], int startIndex, int endIndex)
// Post: Adjacent pairs that are out of order have been
//       switched between values[startIndex]..values[endIndex]
//       beginning at values[endIndex].
{
    for (int index = endIndex; index > startIndex; index--)
        //현재 학생 값과 이전 학생의 값을 이름을 기준으로 비교함
        if (/*-----*/)
            Swap(values[index], values[index-1]);
}

void BubbleSort(Student values[], int numValues)
// Post: The elements in the array values are sorted by key.
{
    int current = 0;

    while (current < numValues - 1)
    {
        BubbleUp(values, current, numValues-1);
        current++;
    }
}
```


■ 테스트 드라이버

```
#include <iostream>
#include "Student.h"
#include "BubbleSort.h"
using namespace std;

int main( )
{
    Student stu[100];
    stu[0].InitValue(2003200111, "이동재", 3.0);
    stu[1].InitValue(2004200121, "권오준", 3.2);
    stu[2].InitValue(2005200132, "김진일", 2.7);

    BubbleSort(stu, 3);
    Print(cout, stu, 3);
    return 0;
}
```

3. Exercise

■ 문 제

- ❖ 학생들의 정보를 담고있는 배열을 **Insertion Sort** 기법을 이용하여 정렬하시오.
 - 정렬의 기준 : 이름, 오름차순 (문자열 비교함수의 결과를 이용하여 정렬함)
- ❖ InsertionSort 함수를 InsertionSort.cpp 파일에 구현할 것

■ 참고 사항

- ❖ InsertionSort.cpp 파일을 이용하여 구현할 것
- ❖ 홈페이지의 Lecture Notes의 Lab12 내의 Lab12-3.zip 파일 참조

■ Insertion Sort 예제 및 참고 코드

6 5 3 1 8 7 2 4

```
#include "InsertionSort.h"

void InsertItem(Student values[], int startIndex, int
endIndex)
// Post: values[0]..values[endIndex] are now sorted.
{
    bool finished = false;
    int current = endIndex;
    bool moreToSearch = (current != startIndex);

    while (moreToSearch && !finished)
    {
        //이름을 기준으로 현재 학생과 이전학생을 비교한다.
        if (/*-----*/)
        {
            Swap(values[current], values[current-1]);
            current--;
            moreToSearch = (current != startIndex);
        }
        else
            finished = true;
    }
}

void InsertionSort(Student values[], int numValues)
// Post: The elements in the array values are sorted by key.
{
    for (int count = 0; count < numValues; count++)
        InsertItem(values, 0, count);
}
```

■ 테스트 드라이버

```
#include <iostream>
#include "Student.h"
#include "InsertionSort.h"
using namespace std;

int main()
{
    Student stu[100];
    stu[0].InitValue(2003200111, "이웅재", 3.0);
    stu[1].InitValue(2004200121, "권오준", 3.2);
    stu[2].InitValue(2005200132, "김진일", 2.7);

    InsertionSort(stu, 3);
    Print(cout, stu, 3);
    return 0;
}
```

4. Exercise

■ 문 제

- ❖ 학생들의 정보를 담고있는 배열을 **Heap Sort** 기법을 이용하여 정렬하시오.
 - 정렬의 기준 : 이름, 오름차순 (문자열 비교함수의 결과를 이용하여 정렬함)
- ❖ HeapSort 함수를 HeapSort.cpp 파일에 구현할 것
- ❖ Student 클래스에 \leq , $<$ 등 필요한 비교 연산자를 overloading하여 구현하시오. (ReheapDown()과 ReheapUp() 함수를 수정하지 마시오)

■ 참고 사항

- ❖ Chapter 10의 강의노트에 있는 함수들을 이용하여 구현할 것

■ Heap Sort 참고 코드

```
template <class ItemType >
void HeapSort ( ItemType values [ ] , int numValues )
// Post: Sorts array values[ 0 . . numValues-1 ] into
// ascending order by key
{
    int index ;

    // Convert array values[0..numValues-1] into a heap
    for (index = numValues/2 - 1; index >= 0; index--)
        ReheapDown ( values , index , numValues - 1 ) ;

    // Sort the array.
    for (index = numValues - 1; index >= 1; index--)
    {
        Swap (values [0] , values[index]);
        ReheapDown (values , 0 , index - 1);
    }
}
```

5. Exercise

■ 문 제

- ❖ Heap sort를 이용한 정렬 과정에서 배열 값의 변화를 출력하시오.
- ❖ 입력이 아래와 같이 주어졌다고 가정한다.

25	17	36	2	3	100	1	19	7
----	----	----	---	---	-----	---	----	---

- ❖ 배열 값이 변화하는 과정을 살펴보고, Heap Sort를 이해하시오.

■ 참고 코드

```
template <class ItemType >
void HeapSort ( ItemType values [ ] , int numValues )
// Post: Sorts array values[ 0 . . numValues-1 ] into
// ascending order by key
{
    int index ;

    // 배열 입력을 출력하시오
    // Convert array values[0..numValues-1] into a heap
    for (index = numValues/2 - 1; index >= 0; index--)
        ReheapDown ( values , index , numValues - 1 ) ;

    // Heap으로 변경된 배열을 출력하시오
    // Sort the array.
    for (index = numValues - 1; index >= 1; index--)
    {
        Swap (values [0] , values[index]);
        ReheapDown (values , 0 , index - 1);
        // 변경된 배열을 출력하시오
    }
}
```