

# Lab 5

**Data Structures**  
**C++ for C Coders**

한동대학교 김영섭 교수  
idebtor@gmail.com

bubble sort  
insertion sort  
quicksort  
selection sort  
#if

## Lab 05:

---

1. Working with sort algorithms:
  - Bubble sort, Insertion sort, Selection sort, Quicksort
2. Handling multiple source files, include files, and external functions.
3. Creating header file (#include file)
4. Using a macro "#if", "#else" and "#endif"
5. Files provided
  - bubble.cpp, insertion.cpp, selection.cpp, quicksort.cpp
  - sort.cpp
  - sortx.exe, sortx

## Lab 05: Step 1 – Learn sort algorithms

---

In this lab, we would like to write a program that works like **sortx.exe**.

1. Run **sortx.exe** provided in this lab and experience how it works.
2. Each file shown below contains one specific sort algorithm.
  - bubble.cpp, insertion.cpp, quicksort.cpp, selection.cpp
3. Build and run each file separately.
  - Study the algorithms by your own and understand how each algorithm works.
4. Build and run the source file with **-DDEBUG** may help you trace the code.
  - For example, the following commands show you compiling without or with -DDEBUG.  
\$ g++ bubble.cpp -o sort  
\$ g++ bubble.cpp -o sort -DDEBUG

## Lab 05: Step 2 – Create `print_list.cpp`

---

- Now we would like to build one executable `sort.exe` that uses all those sort functions and **`print_list()`** function defined in separate files.
- Take a look at **`sort.cpp`**
  - This is a skeleton file provided.
  - Your code should go in this file.
- Create **`print_list.cpp`**
  - Create this external file that contains **`print_list()`** function.
  - Copy **`print_list()`** from one of sort files into its own file named `print_list.cpp`.
  - Add a prototype for **`print_list()`** in **`sort.cpp`**.

## Lab 05: Step 3 – Create sort.h

- Create **nowic/include/sort.h** file that contains all four sort function prototypes.
- The head file such as **sort.h** must begin and end with a macro to prevent it from duplicated inclusion. For example,

```
#ifndef SORT_H
#define SORT_H

// declare the function prototypes for sort.h
void bubbleSort(int *list, int n);
...
...

// This is the end of the header guard
#endif
```

Good explanation can be found here: <https://boycoding.tistory.com/144>

## Lab 05: Step 4 – Edit sort.cpp

---

- In `sort.cpp`, we want to invoke sort functions as defined in four sort files. Then, we need to **exclude** `main()` and `print_list()` in those files. Otherwise, it causes errors because of a multiply defined functions.
- In each sort file, set the macro `#if 0`. This will actually exclude all codes between `#if 0` and `#endif` from compiling.
- Add the following line to in `sort.cpp`.
  - `#include "sort.h"`

## Lab 05: Step 5 – Build the executable, sort.exe

---

- Let us suppose we place source files and include folders as shown below:
  - `~/nowic/include/sort.h`
  - `~/nowic/src/bubble.cpp`, `insertion.cpp`, `quicksort.cpp`, `selection.cpp`, `print_list.cpp`
- The following command line would **not work** since it cannot find `sort.h`.

```
$ g++ sort.cpp bubble.cpp insertion.cpp quicksort.cpp selection.cpp  
print_list.cpp -o sort
```

- Use **`-I`** option such that the compiler looks for header files in **`dir`** folder.  
**`../`** goes up by one level, **`../..`** by two levels in the tree folder structure.

```
$ g++ sort.cpp bubble.cpp insertion.cpp quicksort.cpp selection.cpp  
print_list.cpp -I../include -o sort
```



## Lab 05: Sample run without -DDEBUG

```
$ g++ sort.cpp bubble.cpp insertion.cpp quicksort.cpp selection.cpp  
print_list.cpp -I../../include -o sort
```

may be different depending on your folder structure

```
$ ./sort
```

b for bubble, i for insertion, q for quick, s for selection

Enter an algorithm to sort(x to exit): q

UNSORTED(10):

3 4 1 7 0 9 6 5 2 8

SORTED(10):

0 1 2 3 4 5 6 7 8 9

b for bubble, i for insertion, q for quick, s for selection

Enter an algorithm to sort(x to exit): x



## Lab 05: Sample run with -DDEBUG

```
$ g++ sort.cpp bubble.cpp selection.cpp insertion.cpp quicksort.cpp print_list.cpp -I../..//include -o  
sort -DDEBUG
```

```
$ ./sort
```

```
b for bubble, i for insertion, q for quick, s for selection
```

```
Enter an algorithm to sort(x to exit): b
```

```
UNSORTED(10):
```

```
3 4 1 7 0 9 6 5 2 8
```

```
BUBBLE SORTING...
```

```
3 1 4 0 7 6 5 2 8 9
```

```
1 3 0 4 6 5 2 7 8 9
```

```
1 0 3 4 5 2 6 7 8 9
```

```
0 1 3 4 2 5 6 7 8 9
```

```
0 1 3 2 4 5 6 7 8 9
```

```
0 1 2 3 4 5 6 7 8 9
```

```
0 1 2 3 4 5 6 7 8 9
```

```
0 1 2 3 4 5 6 7 8 9
```

```
0 1 2 3 4 5 6 7 8 9
```

```
SORTED(10):
```

```
0 1 2 3 4 5 6 7 8 9
```

```
b for bubble, i for insertion, q for quick, s for selection
```

```
Enter an algorithm to sort(x to exit):
```

may be different depending  
on your folder structure

## Lab 05:

---

- Files to submit:
  - `sort.h`, `print_list.cpp`, `sort.cpp`
- Due:
  - 11:55 pm, 다음 강의 시간 전날밤 **11:55 PM**까지
- Grade:
  - 1.0

# Lab 5

**Data Structures**  
**C++ for C Coders**

한동대학교 김영섭 교수  
idebtor@gmail.com

insertion sort  
selection sort  
bubble sort  
quick sort  
#if