본 PSet 은 저의 강의 경험과 학생들의 의견 및 Stanford CS106 과 Harvard CS50 같은 강의에서 수집된 자료를 토대로 작성되었습니다. 본 PSet 에 문제가 있거나, 질문 혹은 의견이 있다면, 언제든지 알려 주시면 감사하겠습니다. 강의 개선에 많은 도움이 되겠습니다.
idebtor@gmail.com

# PSet - Sorting

## 목차



# Getting Started

사용자가 아래와 같이 일부 정렬 알고리즘을 상호적으로 테스트할 수 있는 프로그램을 작성하려고 합니다.

```
PS C:\GitHub\nowicx\psets\pset3sorting> ./sortingx
        0       1       2       3       4       5       6       7       8       9
        ...
       40      41      42      43      44      45      46      47      48      49
        MENU[ sort=Bubble order=Ascending N=50 show_n=20 per_line=10 ]
        B - Bubblesort          n - set N samples and initialize
        I - Insertionsort       r - randomize(shuffle) samples
        M - Mergesort           m - max samples to show: show_n
        Q - Quicksort           l - max samples to show: per_line
        S - Selectionsort       o - order[Ascending/Descending]
                                s - sort()
        Command(q to quit):
```

설명을 단계별로 따르다 보면 이 PSet 의 모든 항목을 구현하게 될 것입니다. 프로그램을 구현하면서 다음 주제들을 다룰 것입니다.

- 정렬 알고리즘: 버블, 삽입, 병합, 퀵, 선택 정렬
- 알고리즘 셔플/랜덤화
- 정적 라이브러리, 헤더 파일 사용
- 함수 포인터를 일급 객체로 사용

- **rand(), srand(), %, new, delete, nothrow, assert()** 사용
- C++ 문자열 스트림 **<sstream>** 사용
- STL 의 **map** 컨테이너, **pair, make_pair** 사용

먼저 **github 에서 nowic/psets/pset3, nowic/include, nowic/lib** 을 가져오면 다음 파일들을 확인할 수 있습니다.

- **pset3.pdf** - 본 파일
- **random.pdf** - 난수 생성에 대한 읽기 자료
- **sorting.cpp** - 다양한 정렬 알고리즘을 상호적으로 실행하는 뼈대 코드
- **sortingx.exe, sortingx** - 자신의 실행 결과와 비교할 수 있는 실행 답안
- **include/nowic.h** - I/O 함수: **GetInt(), GetChar(), GetString()** 등등
- **include/sort.h** - 정렬 함수의 프로토타입이 정의된 파일
- **include/rand.h** - 난수 함수 프로토타입이 정의된 파일
- **lib/libnowic.a, libsort.a, librand.a** - Windows 용 사용자 정의 정적 라이브러리
  **lib/libnowic_mac.a, libsort_mac.a, librand_mac.a** - MacOS 용

## 정적 라이브러리 사용하기

정적 라이브러리와 빌드 프로세스에 대해 공부한 내용을 여기서 직접 사용해봅시다. nowic/include 폴더의 nowic.h, sort.h 및 rand.h 를 모두 살펴보세요. 예를 들어, 컴파일하는 동안 다음 에러 메시지가 나타날 경우 어떤 부분을 수정해야 하는지 알 수 있습니다.

```
$ g++ sorting.cpp -I../../include -L../../lib  -lsort -o sorting              # error
```

```
PS C:\GitHub\nowicx\psets\pset3sorting> g++ sorting.cpp -I../../include -L../../lib  -lsort -o sorting
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/10.2.0/../../../../x86_64-w64-mingw32/bin/ld.exe: C:\User
s\user\AppData\Local\Temp\ccgTXJzz.o:sorting.cpp:(.text+0x485): undefined reference to `GetChar(std::__cxx11:
:basic_string<char, std::char_traits<char>, std::allocator<char> >)'
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/10.2.0/../../../../x86_64-w64-mingw32/bin/ld.exe: C:\User
s\user\AppData\Local\Temp\ccgTXJzz.o:sorting.cpp:(.text+0x594): undefined reference to `GetInt(std::__cxx11::
basic_string<char, std::char_traits<char>, std::allocator<char> >)'
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/10.2.0/../../../../x86_64-w64-mingw32/bin/ld.exe: C:\User
s\user\AppData\Local\Temp\ccgTXJzz.o:sorting.cpp:(.text+0x649): undefined reference to `GetInt(std::__cxx11::
basic_string<char, std::char_traits<char>, std::allocator<char> >)'
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/10.2.0/../../../../x86_64-w64-mingw32/bin/ld.exe: C:\User
s\user\AppData\Local\Temp\ccgTXJzz.o:sorting.cpp:(.text+0x6c3): undefined reference to `GetInt(std::__cxx11::
basic_string<char, std::char_traits<char>, std::allocator<char> >)'
collect2.exe: error: ld returned 1 exit status
PS C:\GitHub\nowicx\psets\pset3sorting>
```

아래와 같은 에러 메시지는 linking( ld )가 실행 파일을 빌드하기 위한 GetChar()을 찾을 수 없다는 의미입니다.

"...:undefined reference to 'GetChar(std:...) collect2.exe: ld returned 1 exit status

'GetChar()'과 같은 I/O 함수는 libnowic.a 에 정의되어 있습니다. 해당 라이브러리가 존재하는지 확인하거나 명령줄이 정확하게 입력되었는지 확인하세요. 이 예시에서는 명령줄이 다음과 같아야 합니다:

```
$g++ sorting.cpp -I../../include -L../../lib  -lsort -lnowic -o sorting
```

## map 과 함수 포인터 사용하기

이 pset 에서 map 과 함수 포인터를 광범위하게 사용할 것입니다. Map 과 함수 포인터가 익숙하지 않은 분들은 먼저 이 주제를 복습하고 오길 권장합니다.

- map - STL(Standard Template Library)의 연관 컨테이너
- pair, make_pair() - STL 의 pair 객체 구성 및 함수
- 함수 포인터, 일급 객체인 함수 포인터 배열

# 메뉴 옵션 및 상태 구현하기

먼저 임의의 개수의 샘플을 정렬해서 결과를 출력하고자 합니다.

- 뼈대 코드 **sorting.cpp** 를 따릅니다. **본 파일의 설명과 더불어 뼈대 파일 sorting.cpp 의 설명을 따르셔도 됩니다.** 뼈대 코드를 작성하고 실행할 수 있지만, 아래에서 볼 수 있듯이 기능이 매우 제한적입니다.

**실행 예시: sorting.cpp**

```
PS C:\GitHub\nowicx\psets\pset3sorting> g++ sorting.cpp -I../../include -L../../lib -lnowic -lsort -o sorting
PS C:\GitHub\nowicx\psets\pset3sorting> ./sorting
your code here
your code here
your code here
your code here
your code here
your code here
        0       1       2       3       4       5       6       7       8       9
        ...
       40      41      42      43      44      45      46      47      48      49
        MENU[ sort=Your code here order=Your code here N=50 show_n=20 per_line=10 ]
        B - Bubblesort          n - set N samples and initialize
        I - Insertionsort       r - randomize(shuffle) samples
        M - Mergesort           m - max samples to show: show_n
        Q - Quicksort           l - max samples to show: per_line
        S - Selectionsort       o - order[Ascending/Descending]
                                s - sort()
        Command(q to quit):
```

- 가이드라인으로 제공된 **sorting.exe** 처럼 작동하도록 **sorting.cpp** 를 완성하세요.

**실행 예시: sortingx.exe**

```
PS C:\GitHub\nowicx\psets\pset3sorting> ./sortingx
        0       1       2       3       4       5       6       7       8       9
        ...
       40      41      42      43      44      45      46      47      48      49
        MENU[ sort=Bubble order=Ascending N=50 show_n=20 per_line=10 ]
        B - Bubblesort          n - set N samples and initialize
        I - Insertionsort       r - randomize(shuffle) samples
        M - Mergesort           m - max samples to show: show_n
        Q - Quicksort           l - max samples to show: per_line
        S - Selectionsort       o - order[Ascending/Descending]
                                s - sort()
        Command(q to quit):
```

사용자가 옵션을 상호적으로 설정할 수 있도록 정렬 옵션을 보여줍니다. MENU 행은 정렬의 현재 상태를 출력합니다.

- sort: 선택된 알고리즘

- N: 정렬할 샘플의 개수
- order: 현재 정렬 순서 (오름차순 또는 내림차순)
- show_n: 작업 후 출력할 최대 샘플 수
- per_line: 한 줄당 출력할 최대 샘플 수

## Step 1: B/I/M/Q/S 옵션 - 정렬 알고리즘 고르기

버블, 삽입, 병합, 퀵, 선택 정렬과 같은 정렬 알고리즘을 제공합니다.

- **sort.h** 에 정의된 버블 정렬을 사용하여 sort_fp 라는 정렬 함수의 함수 포인터를 정의하세요.

```
// Define q sort function pointer variable 'sort_fp' variable and initialize it
// with the bubblesort function, 'bubblesort'.

cout << "your code here\n";                                    //set currunt sort_fp
```

- 사용자가 새로운 알고리즘을 선택하면 아래와 같이 재설정합니다.

```
    case 'M':
        cout << "your code here\n";
        break;
```

## Step 2: n 옵션 - 샘플 수 N 설정 및 초기화하기

Libnowic.a 에 있는 GetInt()를 사용하여 사용자로부터 정수를 입력받습니다. 사용자의 입력값이 1 보다 작으면 에러 메시지를 출력한 후 메뉴로 이동합니다. 사용자의 입력이 올바른 경우 다음을 수행합니다.

1. 샘플의 개수 N 을 사용자가 입력한 새로운 값으로 설정합니다.
2. 새 목록을 할당하기 전에 이전 목록의 할당을 해제합니다.
3. 새 데이터 샘플에 메모리를 할당합니다.
4. 0 부터 N - 1 까지의 숫자로 목록을 채웁니다.

**노트:** C++을 배우고 있으므로 **malloc()**과 **free()**가 아닌 **new** 와 **delete** 를 사용하세요. 프로그램을 종료하기 전, 세션 중에 동적으로 할당한 메모리를 **할당 해제**하세요.

## Step 3: r 옵션

목록이 새로 생성되거나 정렬되면 새로운 숫자로 채워집니다. 따라서 일반적으로 정렬 전에 목록의 요소들을 섞어야 합니다.

뼈대 코드에 나온 것처럼 두 가지 방법으로 코딩하세요.

```
    case 'r':
        // your code here
        #if 1
        randomize_bruteforce(list, N);
        #else
        randomize(list, N);
        #endif
        break;
```

**방법 1: randomize_bruteforce()**
모든 샘플은 목록의 첫 번째 요소부터 시작해서 0 부터 N-1 사이의 '실제' (가짜가 아닌) 난수로 생성된 인덱스에 의해 무작위로 선택된 요소와 맞바꿉니다. 0 부터 n - 1 까지의 고유 난수 n 개를 생성하여 배열에 저장해야 합니다. 이를 해결할 수 있는 간단한 방법은 brute force 알고리즘입니다.
예를 들어, 정수 배열 **a[]**이 있고 **n = 100** 이라고 가정합시다.

```
void randomize_bruteforce(int list[], int n) {
   Set a[0] = 0, a[1] = 1, … , a[99] = 99.
   For loop from i = 0 to 99:
       Get a random number r.
       Swap two values: a[i] and a[r]
```

필요하다면 **random.pdf** 에 설명된 **rand()**와 **srand()**를 참조하세요.

**방법 2: randomize()**
유명한 셔플 알고리즘이 여기에 설명되어 있습니다. 피셔-예이츠 셔플 알고리즘입니다.
**nowic/include/rand.h** 에 정의되어 있으며 **librand.a** 에서 사용할 수 있습니다.
**randomize_bruteforce()**를 구현한 후, **randomize()** 함수로도 테스트해보세요.

## Step 4: m 옵션 & l 옵션

**목록이 너무 긴 경우**, 목록의 앞부분과 뒷부분의 일부만 출력하고자 합니다. **show_n** 은 출력할 샘플의 총개수를 지정합니다.

- **show_n** 이 샘플의 총개수인 **N** 보다 작으면, 목록의 중간 부분에 위치한 요소는 출력되지 않을 수 있습니다. 이 경우를 제외하면 모든 요소가 출력됩니다.
- **per_line** 한 줄당 출력할 샘플의 개수를 결정합니다.
- 처음 시작 시 **show_n** 과 **per_line** 은 각각 20 과 10 으로 설정되어 있습니다.

이 옵션들은 **sort.h** 에 정의된 **printlist()**를 통해 이미 구현되어 있습니다. 입력값을 받고 필요에 따라 show_n 과 per_line 을 설정하면 됩니다.

```
void printlist(int *list, int N, int show_n, int per_line);
```

## Step 5: o 옵션

이 옵션은 정렬 순서 함수를 전환합니다. 현재 comparator 함수 포인터 **comp_fp** 가 오름차순이라면, 내림차순으로 설정합니다. 반대의 경우에도 동일하게 작동합니다.

```
      case 'o': // use comp_fp, ::less, more and a ternary operator
          cout << "o: your code here\n";       // one-line code, use
          break;
```

## Step 6: s 옵션

이 옵션은 이전에 이미 설정된 알고리즘을 실행합니다. 함수 포인터 **sort_fp** 를 사용해서 정렬을 실행합니다. 이 코드는 한 줄이면 충분합니다.

```
      case 's':
          begin = clock();
          cout << "s: your code here\n";   // one-line code, use sort_fp, comp_fp
```

```
            bubblesort(list, N);            // remove this line
            show_timeit(begin);
            break;
```

### Step 7: MENU 상태

"sort"와 "order"에 대한 MENU 상태를 출력하기 위해 comp_fp 와 sort_fp 를 key 로 사용하여 각각 comp_map[]과 sort_map[]에서 매핑된 문자열을 가져옵니다.

```
do {
    ...
    stringstream ss;
    ss << "\tMENU[ sort=" << "Your code here" << " order=" << "Your code here";
    ss << " N=" << N << " show_n=" << show_n << " per_line=" << per_line << " ]";
   ...
```

예를 들어, comp_map 에는 comparator 함수 포인터가 key 로, 함수 설명은 value 로 설정되어 있어 다음 코드는 "Descending"을 출력합니다.

```
cout << comp_map[more];
```

예를 들어, sort_map 에는 정렬 함수 포인터가 key 로, 함수 설명은 value 로 설정되어 있어 다음 코드는 "Selection"을 출력합니다.

```
cout << comp_map[selectionsort];
```

## 과제 제출

- 소스 파일 상단에 아래와 같이 아너 코드 문장을 적고 서명하세요.
  **On my honor, I pledge that I have neither received nor provided improper assistance in the completion of this assignment.**
  서명: _____ 분반: _____ 학번: _____
- 제출하기 전에 코드가 제대로 컴파일이 되고 실행되는지 확인하세요. 제출 직전에 급하게 코드를 수정한 후 코드가 제대로 컴파일이 될 거라고 짐작하지 않는 게 좋습니다. "거의" 작동하는 코드도 틀린 것입니다.
- 과제가 컴파일 및 실행된다면, 마감 기한 전까지 과제의 일부만 완성했더라도 제출하기 바랍니다. 컴파일 및 실행되지 않는다면 제출하지 마세요. 마감 시간 이후 24 시간 이내 제출하면, 만점에서 25% 감점하고 채점합니다. 그 이상 늦은 것은 채점하지 않으며, 0 점 처리합니다.
- 제출 후, **마감 기한 전까지** 수정 및 재제출이 가능합니다. 파일 하나만 수정하더라도 해당 파일과 관련된 파일들을 모두 재제출해야 합니다. 재제출 횟수는 제한 없습니다. 마감 기한 전에 **가장 마지막으로** 제출된 파일을 채점할 것입니다.

### 제출 파일 목록, 마감 기한, 배점

**제출 파일 목록: 다음 파일들을 piazza 폴더에 제출하세요.**

- sorting.cpp

**마감 기한:** 11:55 pm