객체지향프로그래밍과 자료구조 (실습)

실습 1. 구조체 배열의 정렬과 탐색

정보통신공학과

교수 김 영 탁

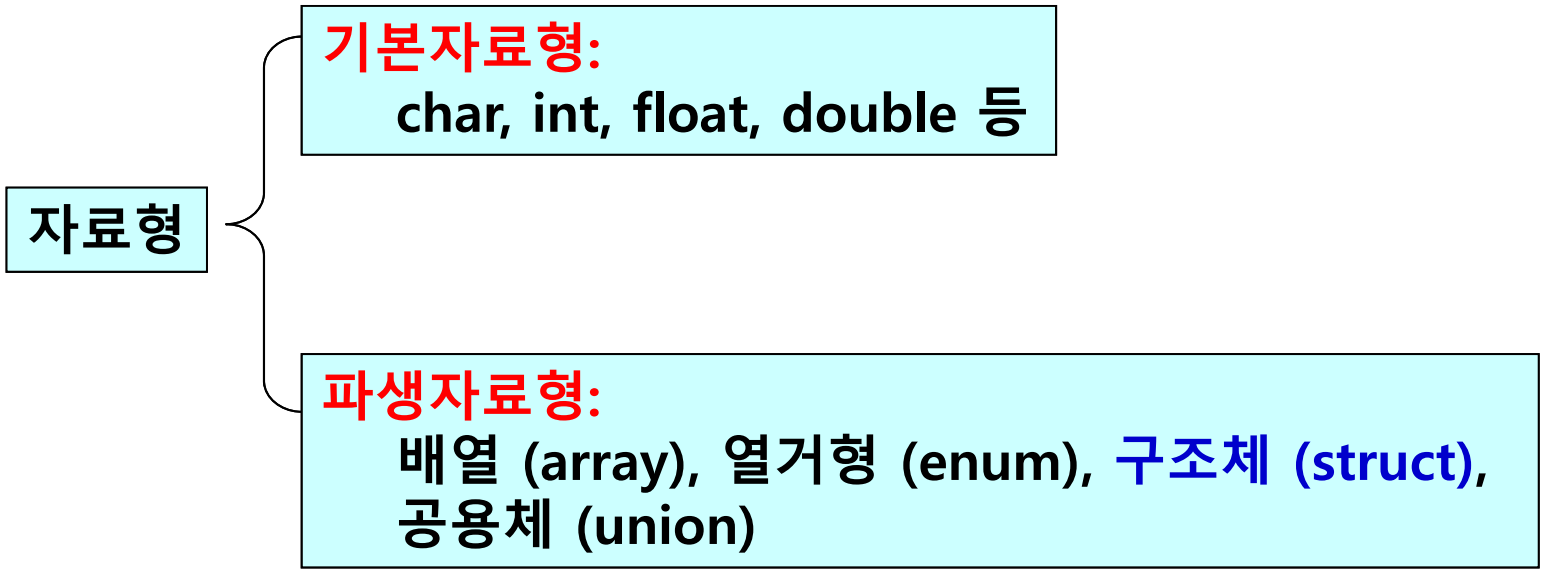
(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Outline

- 구조체
- 구조체 배열
- 구조체 배열의 정렬
- 구조체 배열의 탐색

자료형의 분류

◆ Data Types

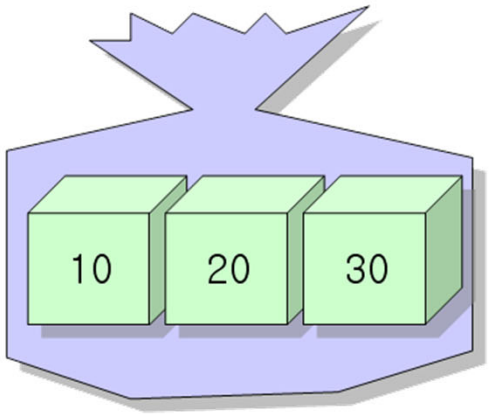


구조체와 배열

◆ 구조체 vs 배열

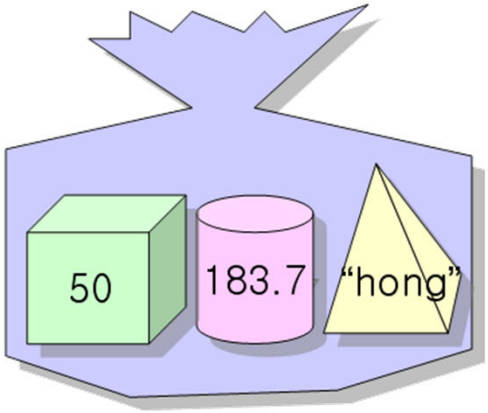
```
int data[50];
```

```
struct  student
{
    int st_id;
    char  name[50];
    double  avg_score;
    . . . .
}
```



배열

같은 자료 유형의 집합



구조체

다른 자료 유형의 집합

구조체 선언의 예

```
// x값과 y값으로 이루어지는 화면의 좌표
struct point {
    int x;        // x 좌표
    int y;        // y 좌표
};
```

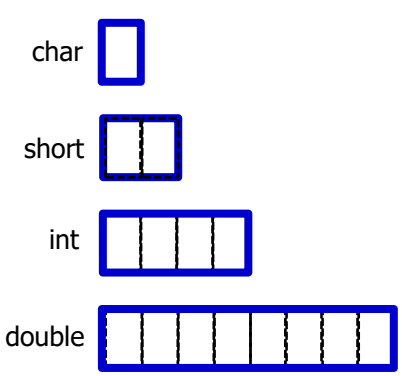
```
// 복소수
struct complex {
    double real;    // 실수부
    double imag;    // 허수부
};
```

```
// 날짜
struct date {
    int month;
    int day;
    int year;
};
```

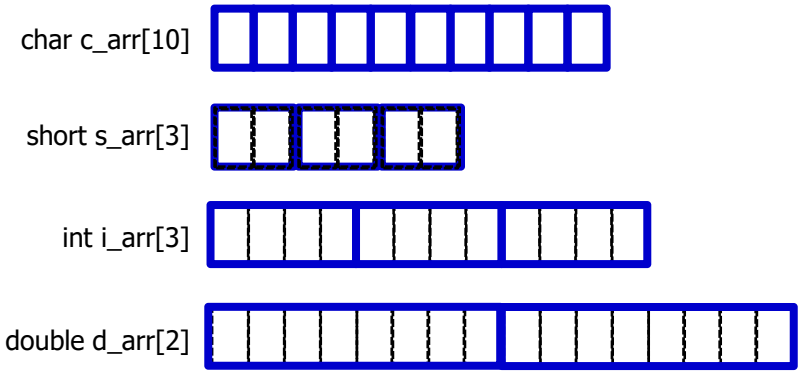
```
// 사각형
struct rect {
    int x;
    int y;
    int width;
    int length;
};
```

```
// 직원
struct employee {
    char name[20];    // 이름
    int age;    // 나이
    int gender;    // 성별
    int salary;    // 월급
};
```

배열과 구조체의 비교

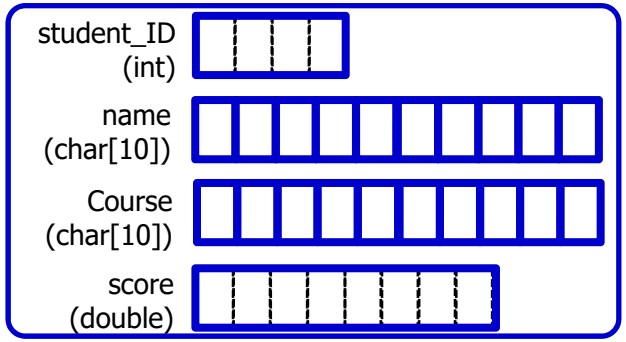


(a) 기본 자료형



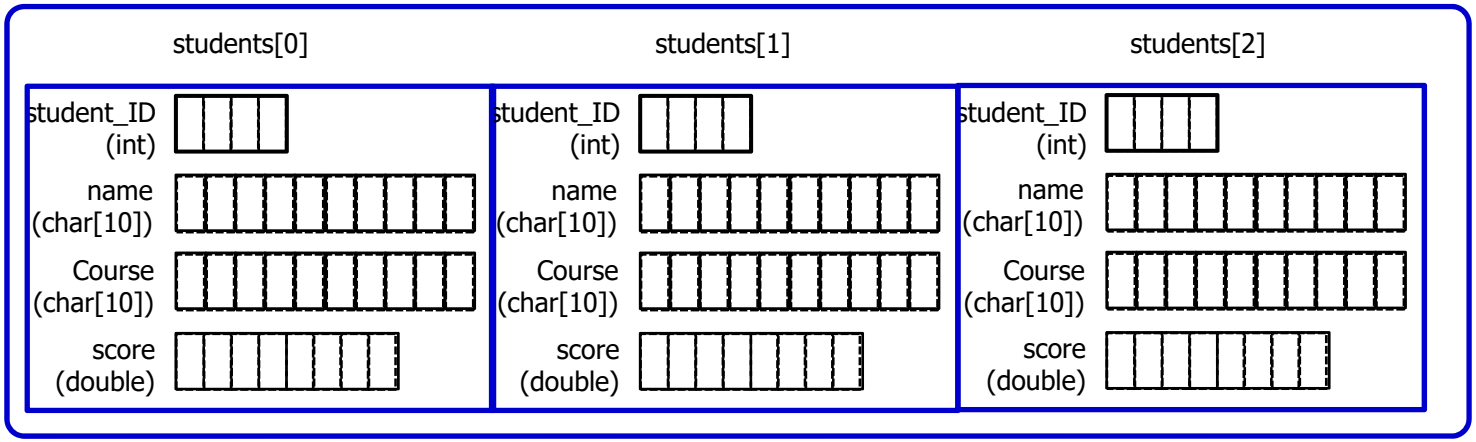
(b) 기본자료형의 배열

```
struct student
{
    int student_ID;
    char name[10];
    char course[10];
    double score;
};
```



(c) 구조체

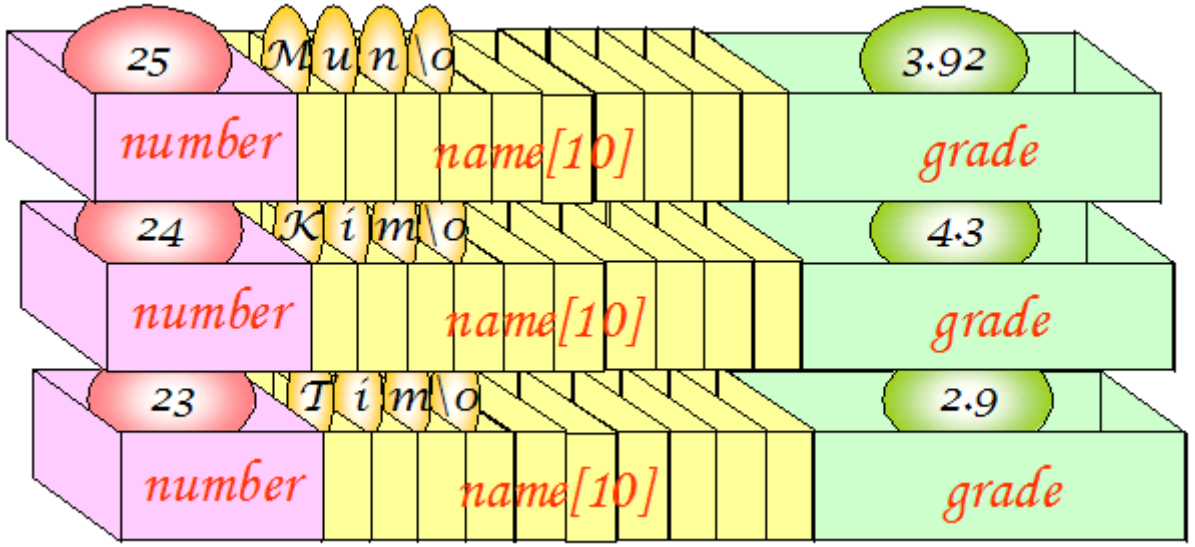
struct student
students[3];



(d) 구조체 배열

구조체 배열 (Array of Struct)

◆ 구조체를 여러 개 모은 것



구조체 배열 (Array of Struct)

◆ 구조체 배열의 선언

```
typedef struct S_Student {
    int number;
    char name[20];
    double grade;
} Student;

int main(void)
{
    Student list[100];    // 구조체의 배열 선언


    list[2].number = 27;
    strcpy(list[2].name, "홍길동");
    list[2].grade = 178.0;
}
```

구조체 배열의 초기화

◆ 구조체 배열의 초기화

```
typedef struct S_Student {
    int number;
    char name[20];
    double grade;
} Student;

Student students[3] = {
    { 1, "Park", 172.8 },
    { 2, "Kim", 179.2 },
    { 3, "Lee", 180.3 }
};
```

Student

```
/* Student.h */

#include <iostream>
using namespace std;

typedef struct s_Student
{
    int st_id;
    char name[16];
    char dept[16];
    int grade;
    double gpa;
} Student;

void initStudents(Student students[], int st_ids[], int num_students);
void fprintStudent(ostream& fout, Student *pSt);
void fprintStudentIDs(Student students[], int num_students);
void fprintBigArrayOfStudent_IDs(ostream& fout, Student students[], int num_students, int
num_first_last);
void sortStudentsByID(Student students[], int num_students);
void sortStudentsByGPA_ID(Student students[], int num_students);
void fprintBigArrayOfStudent_GPA_IDs(ostream& fout, Student students[], int num_students,
   int num_first_last);
Student *searchStudentByID(Student students[], int num_students, int st_ID);
```

genBigRandArray()

```
/* BigRandGen.cpp */
#include <stdio.h>
#include <stdlib.h>

void genBigRandArray(int randArray[], int num_rands)
{
        char *flag;
        int count = 0;
        unsigned int u_int32 = 0;
        unsigned int bigRand;

        flag = (char *)calloc(sizeof(char), num_rands);
        while (count < num_rands)
        {
              u_int32 = ((long)rand() << 15) | rand();
              bigRand = u_int32 % num_rands;
              if (flag[bigRand] == 1) {
                 continue;
              }

              flag[bigRand] = 1;
              randArray[count++] = bigRand;

              }
        }
}
```

```
void initStudents(Student students[], int st_ids[], int num_students)
{
	int name_len;
	int j, grade;
	double br;

	srand(time(0));
	for (int i = 0; i<num_students; i++)
	{
		students[i].st_id = st_ids[i];

		name_len = rand() % 11 + 5;
		students[i].name[0] = rand() % 26 + 'A';
		for (j = 1; j<name_len; j++)
		students[i].name[j] = rand() % 26 + 'a';
		students[i].name[j] = '₩0';

		name_len = rand() % 4 + 2;
		for (j = 0; j<name_len; j++)
		students[i].dept[j] = rand() % 26 + 'A';
		students[i].dept[j] = '₩0';

		students[i].grade = rand() % 8 + 1;

		students[i].gpa = (rand() % 10000)/100.0;
	}
}
```

fprintStudent()

```
void fprintStudent(ostream& fout, Student *pSt)
{
      int count = 0;
      fout.setf(ios::fixed);
      fout.setf(ios::showpoint);
      fout.precision(2);

      fout << "Student(ID: " << setw(4) << pSt->st_id;
      fout << ", Name: " << setw(16) << pSt->name;
      fout << ", Dept : " << setw(6) << pSt->dept;
      fout << ", Grade : " << pSt->gpa;
}
```

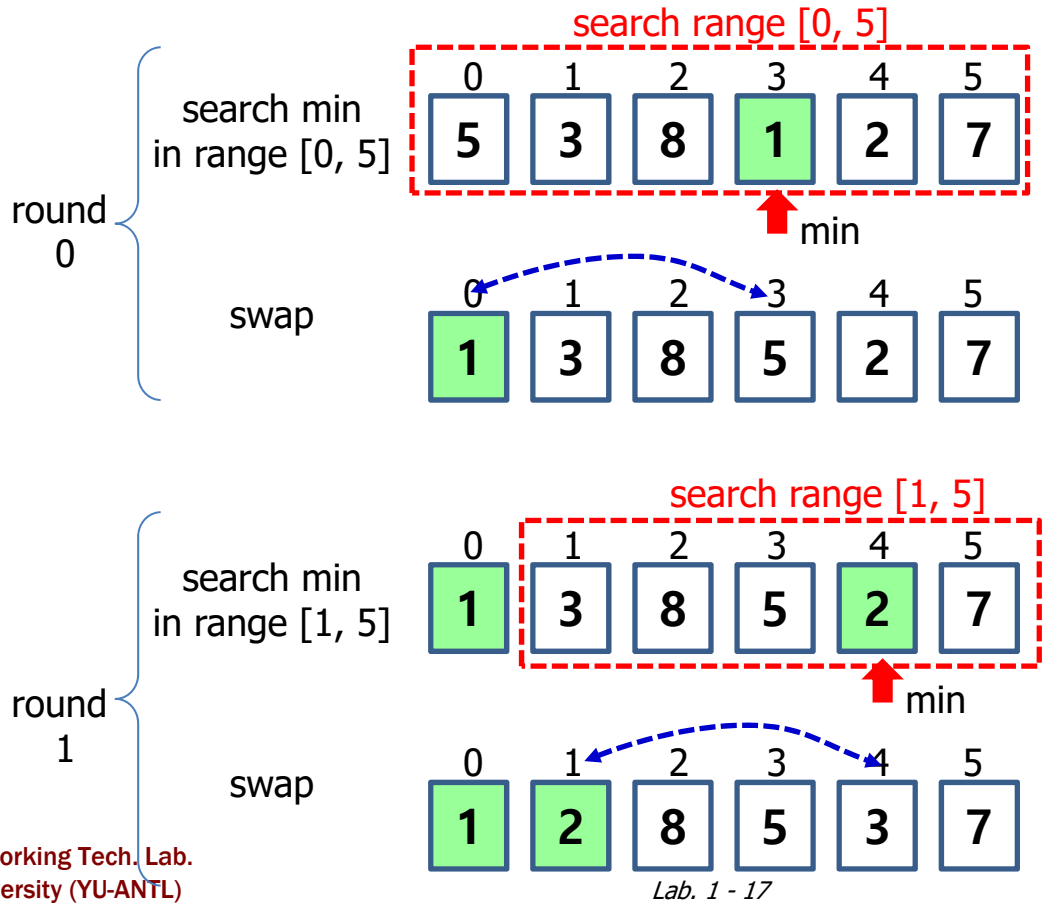
fprintBigArrayOfStudent_IDs()

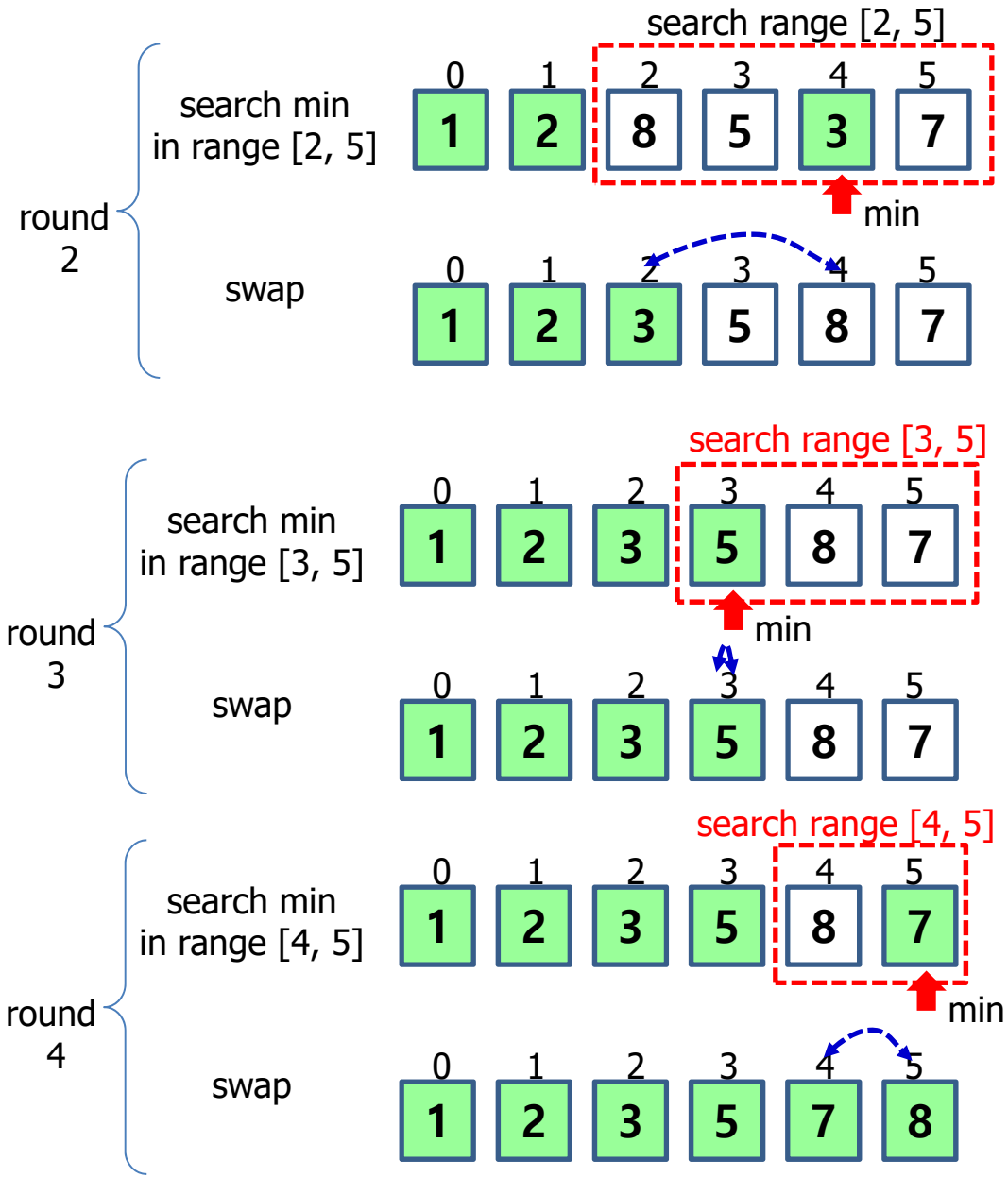
```
void fprintBigArrayOfStudent_IDs(ostream& fout, Student students[],
    int num_students,  int num_first_last)
{
    Student *pSt;
    int count = 0;
    if (num_students <= num_first_last * 2)
    {
        for (int i = 0; i < num_students; i++)
        {
            pSt = &students[i];

            fout << setw(6) << pSt->st_id;
            count++;
            if ((count % 20) == 0)
            fout << endl;
        }
        return;
    }
```

```
    for (int i = 0; i < num_first_last; i++)
    {
        pSt = &students[i];

        fout << setw(6) << pSt->st_id;
        count++;
        if ((count % 20) == 0)
        fout << endl;
    }

    fout << " . . . . . " << endl;
    for (int i = num_students - num_first_last; i < num_students; i++)
    {
          pSt = &students[i];

          fout << setw(6) << pSt->st_id;
          count++;
          if ((count % 20) == 0)
          fout << endl;
    }
}
```

선택정렬(selection sort)

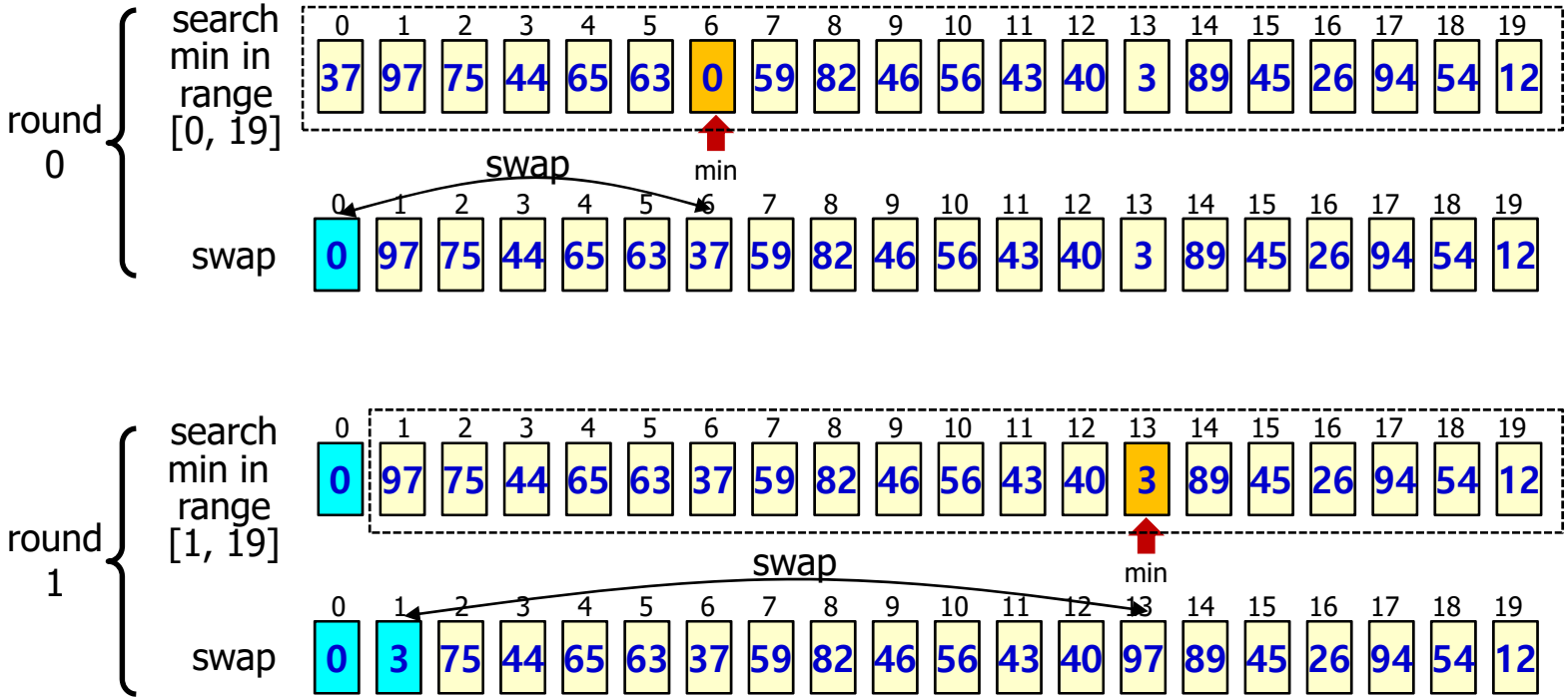
- 선택정렬(selection sort): 정렬이 안된 숫자들 중에서 최소값을 선택하여 배열의 첫 번째 요소와 교환
- 몇 개의 단계만 살펴보자.

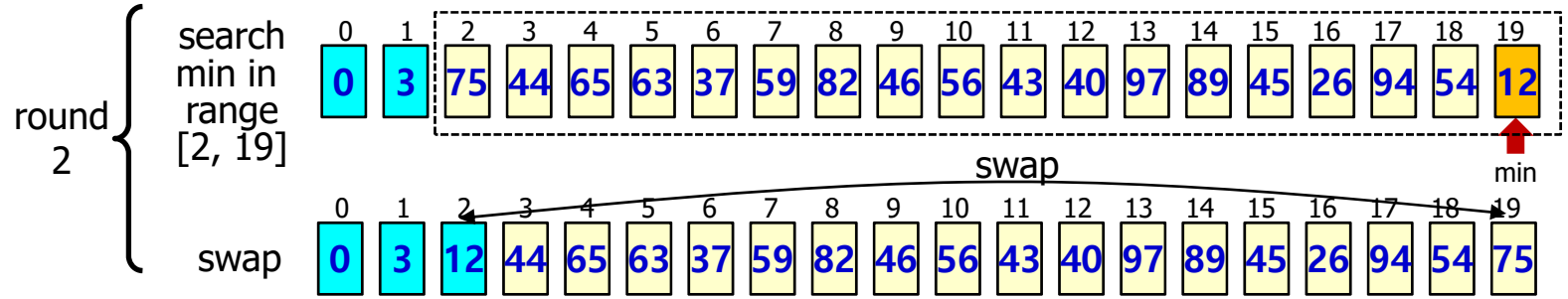



search range [2, 5]
round 2
search min in range [2, 5]
0 1 2 3 4 5
1 2 8 5 3 7
min
swap
0 1 2 3 4 5
1 2 3 5 8 7
search range [3, 5]
round 3
search min in range [3, 5]
0 1 2 3 4 5
1 2 3 5 8 7
min
swap
0 1 2 3 4 5
1 2 3 5 8 7
search range [4, 5]
round 4
search min in range [4, 5]
0 1 2 3 4 5
1 2 3 5 8 7
min
swap
0 1 2 3 4 5
1 2 3 5 7 8

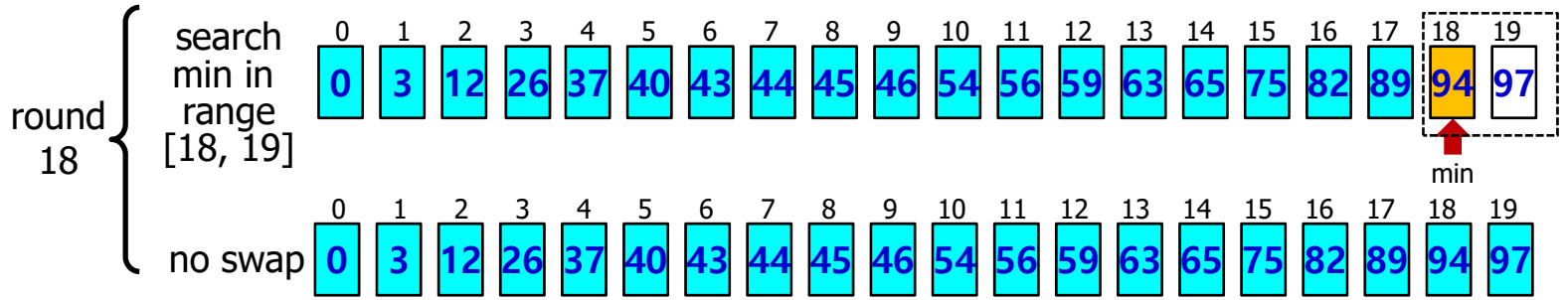
선택정렬(selection sorting)

◆ 선택정렬(selection sort): 정렬이 안된 숫자들 중에서 최소값을 선택하여 배열의 첫 번째 요소와 교환




round 2
search min in range [2, 19]
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
0 3 75 44 65 63 37 59 82 46 56 43 40 97 89 45 26 94 54 12
min
swap
swap
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
0 3 12 44 65 63 37 59 82 46 56 43 40 97 89 45 26 94 54 75

round 3 ~ 17 • • • •


round 18
search min in range [18, 19]
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
0 3 12 26 37 40 43 44 45 46 54 56 59 63 65 75 82 89 94 97
min
no swap
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
0 3 12 26 37 40 43 44 45 46 54 56 59 63 65 75 82 89 94 97

```
void sortStudentsByID(Student students[], int num_students)
{
	int i, j, mx;
	char minName[16] = { 0 };
	Student tmp;
	int min_st, min_ID;

	for (i = 0; i<num_students - 1; i++)
	{
		min_st = i;
		min_ID = students[min_st].st_id;
		for (j = i + 1; j<num_students; j++)
		{
			if (students[j].st_id < min_ID)
			{
			min_st = j;
			min_ID = students[j].st_id;
			}
		}

		if (min_st != i)
		{
			tmp = students[i];
			students[i] = students[min_st];
			students[min_st] = tmp;
		}
	}
}
```

순차탐색 (Sequential Search)

◆ **순차 탐색은 배열의 원소를 순서대로 하나씩 꺼내서 탐색키와 비교하여 원하는 값을 찾아가는 방법**

- 배열에 포함된 원소가 정렬되어 있지 않을 수 있음
- 가장 빠른 탐색 결과: 맨 처음 원소가 찾고자 하는 원소일 때
- 가장 늦은 탐색 결과: 맨 뒤 원소가 찾고자 하는 원소 일 때
- 만약 찾고자 하는 원소가 배열에 포함되어 있지 않을 때 : -1 반환
- N개의 원소가 포함된 배열에서의 탐색에 걸리는 시간의 평균: N/2

순차탐색

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
37	97	75	44	65	63	0	59	82	46	56	43	40	3	89	45	26	94	54	12

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
37	97	75	44	65	63	0	59	82	46	56	43	40	3	89	45	26	94	54	12

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
37	97	75	44	65	63	0	59	82	46	56	43	40	3	89	45	26	94	54	12

• • • •

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
37	97	75	44	65	63	0	59	82	46	56	43	40	3	89	45	26	94	54	12

탐색키워드가 존재할 때 **(**해당 **position** 반환**)**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
37	97	75	44	65	63	0	59	82	46	56	43	40	3	89	45	26	94	54	12

탐색키워드가 존재할 때 **(-1** 반환**)**

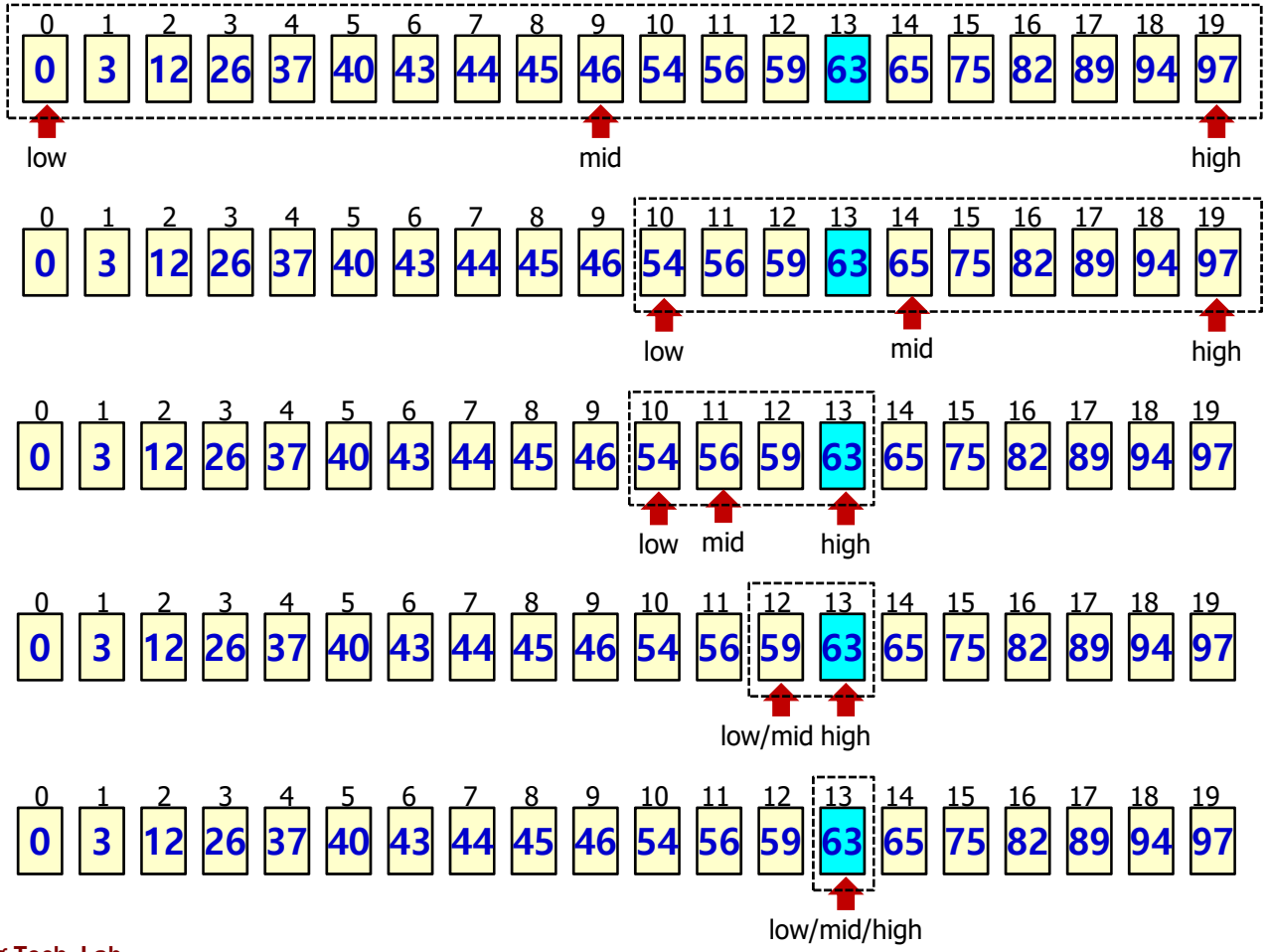
sequential_search()

```
int sequential_search(int data_array[], int size, int key_to_search)
{
    for (int pos = 0; pos < size; pos++)
    {
        if (data_array[pos] == key_to_search)
        {
            return pos;
        }
    }
    return -1;
}
```

이진 탐색 (binary search)

◆ 이진 탐색(binary search)

- 사전에 정렬되어 있는 배열의 중앙에 위치한 원소와 비교 되풀이



binary_search()

```
int binary_search(int array[], int n, int key)
{
    int low, high, mid;
    int round = 0;

    low = 0;
    high = n - 1;
    round++;
    while (low <= high)
    {
        printf("%2d-th round: Search range: [%2d, %2d]\n", round, low, high);
        mid = (low + high) / 2;
        if (key == array[mid])
            return mid;
        else if (key < array[mid])
            high = mid - 1;
        else
            low = mid + 1;
        round++;
    }
    return -1;
}
```

Student.h

```
/* Student.h */

#include <iostream>
using namespace std;

typedef struct s_Student
{
        int st_id;
        char name[16];
        char dept[16];
        int grade;
        double gpa;
} Student;

void initStudents(Student students[], int st_ids[], int num_students);
void fprintStudent(ostream& fout, Student *pSt);
void fprintStudentIDs(ostream& fout, Student students[], int num_students);
void fprintBigArrayOfStudent_IDs(ostream& fout, Student students[], int num_students, int num_first_last);
void sortStudentsByID(Student students[], int num_students);
void sortStudentsByGPA_ID(Student students[], int num_students);
void fprintBigArrayOfStudent_GPA_IDs(ostream& fout, Student students[],
        int num_students, int num_first_last);
Student *searchStudentByID(Student students[], int num_students, int st_ID);
```

main()

```
/* main.cpp (1) */
#include <iostream>
#include <iomanip>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include "Student.h"
using namespace std;
#define NUM_STUDENTS 5000
#define NUM_SEARCH 5
#define NUM_FIRST_LAST_BLOCK 100
void genBigRandArray(int *randArray, int num_rands);

void main()
{
	ofstream fout;

	int *student_ids;
	Student *students, *pSt;
	int studentID_search[NUM_SEARCH] = { 1, 123, 999, 2500, 4999 };

	fout.open("output.txt");
	if (fout.fail())
	{
		cout << "Error in opening output.txt" << endl;
		exit;
	}
```

```
/* main.cpp (2) */
      student_ids = (int *)malloc(sizeof(int)* NUM_STUDENTS);
      students = (Student *)malloc(sizeof(Student)* NUM_STUDENTS);
      genBigRandArray(student_ids, NUM_STUDENTS);
      initStudents(students, student_ids, NUM_STUDENTS);

      fout << endl << endl;
      fout <<"Initialized array of students : " << endl;
      fprintBigArrayOfStudent_IDs(fout, students, NUM_STUDENTS, NUM_FIRST_LAST_BLOCK);

      fout << endl << endl;
      fout << "Sorting array of students by non-decreasing order of ID : " << endl;
      sortStudentsByID(students, NUM_STUDENTS);
      fprintBigArrayOfStudent_IDs(fout, students, NUM_STUDENTS, NUM_FIRST_LAST_BLOCK);

      fout << endl << endl;
      fout << "Sorting array of students by (decreasing order of GPA, increasing order of ID) : " << endl;
      sortStudentsByGPA_ID(students, NUM_STUDENTS);  // non-increasing order
      fprintBigArrayOfStudent_GPA_IDs(fout, students, NUM_STUDENTS, NUM_FIRST_LAST_BLOCK);

      fout << endl << endl;
      fout << "Sorting array of students by non-decreasing order of ID : " << endl;
      sortStudentsByID(students, NUM_STUDENTS);
      fprintBigArrayOfStudent_IDs(fout, students, NUM_STUDENTS, NUM_FIRST_LAST_BLOCK);
      fout << endl << endl;
```

```
/* main.cpp (3) */
      fout << "Searching student with student_ID : " << endl;
      for (int i = 0; i < NUM_SEARCH; i++)
      {
            pSt = searchStudentByID(students, NUM_STUDENTS, studentID_search[i]);
            fout << "Student search by ID (" << setw(4) << studentID_search[i] << ") : ";
            if (pSt != NULL)
               fprintStudent(fout, pSt);
            else
               fout << "NULL - student was not found !!";
            fout << endl;
      }
      fout.close();
}
```

Result (1)

```
Initialized array of students :
  1955  4012   516   462   280  4585  3635  2539  2102  1572  2892   889   638    44  4319  4022  2666  3304  2555  2774
  3715  4629  3528  2999  1172  1740  1173   275     5  1050  3723  3762  2290  1662   400  4333   249  1510  2709  3372
   695  4466  3448  3264  2316  3431  1091  2954  1706  4013  3859   410  4602   642  3250  1873  4704  1189  4309     8
  1538  1284  3407  2614  4860   950  4412  1318  4367  4713  2263   687  1669  4270  1042  4638   352  3791  4871  2239
  3769  1347  2001  4686  1324  3252   749  2463   508  1542  3703  2962  2988  1003  3997  3147  2197  1697   898  4928
 . . . . . .
  2474   243  3209  3790  1578   572  3324  1745   288  3141   308  1827   213  4786  4772  1787   930   810  4864  2487
  4909  4836  4463  3706  2058  4070  1848  3093  1192  2592  2336  4634  2680  1552  3146  2060  3204   221  2462  4525
  4823  2181  4919  3484  3579  1601   519  2186  3491  1329  4006  1484  3206  1979  3034  1586  1064  2312   454  1883
   723  3770  1629  2384  2337  3768  1807  4644  2599  3203  2960  4979   227  4131  4708   685  1904   739  4145  3112
   164  1208  2446  2937  1886  1674  4347  4974  2326  2547  4182  4793  1455  2920   205  1729  2712  1642  3645  4965


Sorting array of students by non-decreasing order of ID :
     0     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19
    20    21    22    23    24    25    26    27    28    29    30    31    32    33    34    35    36    37    38    39
    40    41    42    43    44    45    46    47    48    49    50    51    52    53    54    55    56    57    58    59
    60    61    62    63    64    65    66    67    68    69    70    71    72    73    74    75    76    77    78    79
    80    81    82    83    84    85    86    87    88    89    90    91    92    93    94    95    96    97    98    99
 . . . . . .
  4900  4901  4902  4903  4904  4905  4906  4907  4908  4909  4910  4911  4912  4913  4914  4915  4916  4917  4918  4919
  4920  4921  4922  4923  4924  4925  4926  4927  4928  4929  4930  4931  4932  4933  4934  4935  4936  4937  4938  4939
  4940  4941  4942  4943  4944  4945  4946  4947  4948  4949  4950  4951  4952  4953  4954  4955  4956  4957  4958  4959
  4960  4961  4962  4963  4964  4965  4966  4967  4968  4969  4970  4971  4972  4973  4974  4975  4976  4977  4978  4979
  4980  4981  4982  4983  4984  4985  4986  4987  4988  4989  4990  4991  4992  4993  4994  4995  4996  4997  4998  4999
```

Result (2)

```
Sorting array of students by (decreasing order of GPA, increasing order of ID) :
   99.96  2500   99.92   932   99.92  2565   99.91  2843   99.91  2895   99.90   950   99.89  3293   99.86  3078   99.80  2318   99.77  3094
   99.74  4858   99.70  2832   99.70  4632   99.66   301   99.66    11   99.62  1803   99.60  3791   99.60  4948   99.54   380   99.54  3800
   99.53  4221   99.51   785   99.51  2531   99.48  3062   99.47    37   99.46   480   99.45  1114   99.43  4060   99.43  4776   99.41  3670
   99.41  4661   99.38   363   99.37  4149   99.33  3708   99.30  2690   99.28  2472   99.25   226   99.16   504   99.15  1530   99.15  3633
   99.12  2726   99.08  1209   99.04  1564   98.99  2023   98.99  2731   98.98    63   98.94  2820   98.89  4524   98.84   365   98.84  4531
   98.82   886   98.79   506   98.78   397   98.78  4957   98.73   526   98.71  4188   98.71  4978   98.70   199   98.70  4063   98.70  4400
   98.68  1592   98.65  1288   98.65  2966   98.62  3176   98.59  1891   98.56  4433   98.48  3425   98.48  3871   98.44    72   98.44  4191
   98.42  2835   98.37  4985   98.35  1544   98.35  4815   98.33  2014   98.27  2253   98.26   426   98.26  2801   98.25  1872   98.22  3668
   98.20   689   98.20  1538   98.15  3036   98.15  3292   98.12   264   98.11   631   98.11  4620   98.10  1919   98.09  3340   98.05  4644
   98.04  4044   98.04  4541   98.00  3530   97.97  1169   97.93  3569   97.89  4122   97.88  1674   97.85    22   97.84  1877   97.82   800
 . . . . .
    1.77  2575    1.71  1494    1.69  3827    1.69  3236    1.68   361    1.68  4833    1.67   911    1.67   159    1.66  4670    1.66  2443
    1.64  1878    1.62  2725    1.61  1649    1.60  2180    1.59  1217    1.58   606    1.52  4190    1.48  2496    1.47   872    1.43  3902
    1.40  3144    1.38   717    1.36  3160    1.35  3011    1.35  4298    1.33   468    1.32  3311    1.30  4335    1.29  4932    1.25  3183
    1.23  3112    1.21  1021    1.19  1094    1.18   908    1.16  1843    1.14  4733    1.13  2124    1.11   364    1.11   761    1.07  1377
    1.04   578    1.01  3027    1.00  1517    0.98   191    0.97  3844    0.94   988    0.93  2629    0.92   563    0.91   125    0.89  3861
    0.88  1285    0.88  1496    0.87  1654    0.84  3272    0.83  3700    0.81  4971    0.80    73    0.78  2822    0.77  2201    0.75    83
    0.74   437    0.72  4291    0.72  1998    0.71  4319    0.69  3341    0.67   145    0.66  4950    0.63  4583    0.62  4619    0.59  3662
    0.59  3928    0.58  3180    0.57  1793    0.50  4077    0.49  3076    0.49  4763    0.48  2510    0.45  4392    0.44  1400    0.35  2675
    0.34  4750    0.34  4585    0.32  4300    0.30  1963    0.23  4842    0.23  2639    0.22  1647    0.20  4363    0.17  2634    0.16  1572
    0.15  4093    0.15   821    0.10  3502    0.09  4615    0.06  3974    0.06  1876    0.03  2626    0.03  4446    0.02  2598    0.01   927
```

Result (3)

```
Searching student with student_ID :
Student search by ID (   1) : Student(ID:    1, Name:        Dcdxylgtb, Dept :  XQDPU, Grade : 73.65
Student search by ID ( 123) : Student(ID:  123, Name:        Mjnuogkit, Dept :     CQ, Grade : 25.52
Student search by ID ( 999) : Student(ID:  999, Name:  Zlivxdhbszttwlk, Dept :    PPN, Grade : 52.97
Student search by ID (2500) : Student(ID: 2500, Name:         Vqqzziwz, Dept :    TEI, Grade : 99.96
Student search by ID (4999) : Student(ID: 4999, Name:            Aqcqz, Dept :   BNRQ, Grade : 29.67
```

Oral Test

Q1.1 구조체와 배열의 차이점에 대하여 설명하라.

Q1.2 genBigRandArray(int randArray[], int num_rands) 가 0 .. num_rands-1범위의 중복되지 않는 난수들을 생성하고, 주어진 배열 randArray[]에 저장하는 기능에 대하여 설명하라.

Q1.3 구조체 배열에 대한 선택 정렬의 내부 절차에 대하여 설명하라.

Q1.4 프로그램 모듈이 실행하는 시간을 마이크로 초(micro-second) 단위로 측정하는 방법에 대하여 설명하라.