Select & Insert Sort

1. 선택 정렬

1) 정의 : 가장 작은 것을 선택, 앞으로 보내는 정렬 기법
2) 시간 복잡도

$$O(N^2)$$

(1) 가장 작은 것을 선택하는데 N번의 연산
(2) 앞으로 보내는데 N번의 연산

3-1) 배열

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

#define SIZE 100
int g_nSize = 0;

void Swap(int *pFir, int *pSec)
{
	int nTemp = *pFir;
	*pFir = *pSec;
	*pSec = nTemp;
}

void Show(int *pData)
{
	for (int i = 0; i < g_nSize; ++i)
	{
```

```
                printf("%d : %d\n", i, *(pData + i));
        }
}

int main(void)
{
        int arrData[SIZE];
        int nMinVal, nIDX;

        printf("Size : ");
        scanf("%d", &g_nSize);

        for (int i = 0; i < g_nSize; ++i)
        {
                printf("%d 번 : ", i);
                scanf("%d", &arrData[i]);
        }
        system("cls");

        printf("--- Before ---\n");
        Show(arrData);

        for (int i = 0; i < g_nSize; ++i)
        {
                nMinVal = INT_MAX;

                for (int j = i; j < g_nSize; ++j)
                {
                        if (nMinVal > arrData[j])
                        {
```

```
                                nMinVal = arrData[j];
                                nIDX = j;
                        }
                }

                Swap(&arrData[i], &arrData[nIDX]);
        }
        printf("--- after ---\n");
        Show(arrData);

        system("pause");
        return 0;
}
```

3-2) 포인터

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

#define SIZE 100
int g_nSize = 0;

void Swap(int *pFir, int *pSec)
{
        int nTemp = *pFir;
        *pFir = *pSec;
        *pSec = nTemp;
}

void Show(int *pData)
```

```
{
	int nSize = _msize(pData) / sizeof(int);
	for (int i = 0; i < nSize; ++i)
	{
		printf("%d : %d\n", i, *(pData + i));
	}
}

int main(void)
{
	int *pArrData = NULL;
	int nMinVal, nIDX;

	printf("Size : ");
	scanf("%d", &g_nSize);

	pArrData = (int*)malloc(sizeof(int) * g_nSize);
	for (int i = 0; i < g_nSize; ++i)
	{
		printf("%d 번 : ", i);
		scanf("%d", pArrData+i);
	}
	system("cls");

	printf("--- Before ---\n");
	Show(pArrData);

	for (int i = 0; i < g_nSize; ++i)
	{
		nMinVal = INT_MAX;
```

```
                for (int j = i; j < g_nSize; ++j)
                {
                        if (nMinVal > *(pArrData+j))
                        {
                                nMinVal = *(pArrData + j);
                                nIDX = j;
                        }
                }

                Swap(pArrData + i, pArrData + nIDX);
        }
        printf("--- after ---\n");
        Show(pArrData);

        system("pause");
        return 0;
}
```

3-3) 포인터 & Swap 함수 호출 오버헤드 제거

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

#define SIZE 100
int g_nSize = 0;

void Show(int *pData)
{
        int nSize = _msize(pData) / sizeof(int);
```

```
        for (int i = 0; i < nSize; ++i)
        {
                printf("%d : %d\n", i, *(pData + i));
        }
}

int main(void)
{
        int *pArrData = NULL;
        int nMinVal, nIDX;

        printf("Size : ");
        scanf("%d", &g_nSize);

        pArrData = (int*)malloc(sizeof(int) * g_nSize);
        for (int i = 0; i < g_nSize; ++i)
        {
                printf("%d 번 : ", i);
                scanf("%d", pArrData+i);
        }
        system("cls");

        printf("--- Before ---\n");
        Show(pArrData);

        int nTemp;
        for (int i = 0; i < g_nSize; ++i)
        {
                nMinVal = INT_MAX;
```

```
                for (int j = i; j < g_nSize; ++j)
                {
                        if (nMinVal > *(pArrData+j))
                        {
                                nMinVal = *(pArrData + j);
                                nIDX = j;
                        }
                }

                nTemp = *(pArrData + i);
                *(pArrData + i) = *(pArrData + nIDX);
                *(pArrData + nIDX) = nTemp;
        }
        printf("--- after ---\n");
        Show(pArrData);

        system("pause");
        return 0;
}
```

---

2. 삽입정렬

1) 정의 : 각 숫자를 적절한 위치에 삽입하는 정렬기법
2) 시간 복잡도

$$O(N^2)$$

(1) 들어갈 위치를 선택하는데 N번 연산
(2) 선택하는 횟수로 N번 연산
3) 포인터

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <                    .h>

template <typename T>
void Show(T *pSize)
{
        int nSize = _msize(pSize) / sizeof(T);
        for (int i = 0; i < nSize; ++i)
        {
                printf("%d : %d\n", i+1, *(pSize + i));
        }
}
```

☐ int main(void)

```
{
        int *pArrData = NULL;

        int nSize;
        printf("Size : ");
        scanf("%d", &nSize);

        pArrData = (int*)malloc(sizeof(int) * nSize);
        for (int i = 0; i < nSize; ++i)
        {
                printf("%d : ", i + 1);
                scanf("%d", pArrData + i);
        }
        system("cls");

        printf("--- Before ---\n");
```

```
        Show(pArrData);

        int nTemp;

        // case 0
        /*
        for (int i = 1; i < nSize; ++i)
        {
                for (int j = i; j > 0; --j)
                {
                        if (*(pArrData + (j - 1)) > *(pArrData
+ j))
                        {
                                nTemp = *(pArrData + (j - 1));
                                *(pArrData + (j - 1)) = *(pArr
Data + j);
                                *(pArrData + j) = nTemp;
                        }
                }
        }
        */

        //case 1
        for (int i = 0; i < nSize - 1; i++)
        {
                int j = i;
                while (j >= 0 && *(pArrData + j) > *(pArrData
+ (j + 1)))
                {
                        nTemp = *(pArrData + j);
```

```
                        *(pArrData + j) = *(pArrData + (j +
1));
                        *(pArrData + (j + 1)) = nTemp;
                        --j;
                }
        }
        printf("--- After ---\n");
        Show(pArrData);

        system("pause");
        return 0;
}
```