Sequential Search (순차 탐색) & Binary Search (이진 탐색)

[Sequential Search]

1. 개념

- 특정한 원소를 찾기 위해 원소를 순차적으로 하나씩 탐색하는 방법
- 시간 복잡도 : O(n)

2. 설명

찾을 문자열	강종구

인덱스	0	1	2	3	4
원소	나동빈	이태일	박한울	강종구	이상욱

- 인덱스 0번 부터 차례대로 탐색

찾을 문자열	강종구

인덱스	0	1	2	3	4
원소	나동빈	이태일	박한울	강종구	이상욱

3. code

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

```
#define STR_LENGTH 100

int main(void)
{
	int nSize;
	printf("Data Size : ");
	scanf("%d", &nSize);

	char** pDataSet = (char**)malloc(sizeof(char*) * nSiz
e);
	for (int i = 0; i < nSize; ++i)
	{
		*(pDataSet + i) = (char*)malloc(sizeof(char) *
STR_LENGTH);
		printf("[%d] Data = ", i + 1);
		scanf("%s", *(pDataSet + i));
	}

	char* strTargetName = (char*)malloc(sizeof(char) * STR
_LENGTH);

	while (strcmp("exit", strTargetName))
	{
		printf("Target Name : ");
		scanf("%s", strTargetName);

		bool bCheck = false;
		for (int i = 0; i < nSize; ++i)
		{
```

```
                        if (!strcmp(*(pDataSet + i), strTarget
Name))
                        {
                                printf("[%d]번째 원소입니다.\n",
i);
                                bCheck = true;
                                break;
                        }
                }

                if (!bCheck)
                        printf("원소를 찾을 수 없습니다.\n");
        }

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

[Binary Search]

1. 개념

- 배열 내부 데이터가 이미 정렬되어 있는 상황에서 사용 가능한 알고리즘
- 탐색 범위를 절반씩 좁혀가며 데이터를 탐색하는 특징
- 시간복잡도 : O(logN)

2. 설명

찾을 원소	37

MID 위치에 있는 원소와 반복적으로 비교

인덱스	0	1	2	3	4	5	6	7	8
원소	15	27	37	46	57	69	73	85	98

START MID END

찾을 원소	37

인덱스	0	1	2	3	4	5	6	7	8
원소	15	27	37	46	57	69	73	85	98

찾을 원소	37

인덱스	0	1	2	3	4	5	6	7	8
원소	15	27	37	46	57	69	73	85	98

찾을 원소	37

인덱스	0	1	2	3	4	5	6	7	8
원소	15	27	37	46	57	69	73	85	98

찾을 원소	37

인덱스	0	1	2	3	4	5	6	7	8
원소	15	27	37	46	57	69	73	85	98

3. code

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

#define STR_LENGTH 100

int Binary_Search(int *pDataSet, int nStart, int nEnd, int nTa
rget)
{
        if (nStart > nEnd)
                return -9999;

        int nMid = (nStart + nEnd) / 2;

        if (pDataSet[nMid] == nTarget)
                return nMid;
        else if (pDataSet[nMid] > nTarget)
        {
                return Binary_Search(pDataSet, nStart, nMid -
1, nTarget);
        }
```

```
        else
                Binary_Search(pDataSet, nMid + 1, nEnd, nTarge
t);
}

int main(void)
{
        int nSize;
        printf("Data Size : ");
        scanf("%d", &nSize);

        int* pDataSet = (int*)malloc(sizeof(int) * nSize);
        for (int i = 0; i < nSize; ++i)
        {
                printf("[%d] Data = ", i + 1);
                scanf("%d", pDataSet + i);
        }

        int nTarget = 0;
        do
        {
                bool bCheck = false;

                printf("Target : ");
                scanf("%d", &nTarget);

                int nIdx = Binary_Search(pDataSet, 0, nSize, n
Target);
                if (-9999 != nIdx)
                {
```

```
                        printf("[%d]번째 원소입니다.\n", nIdx);
                        bCheck = true;
                }

                if (!bCheck)
                        printf("원소를 찾을 수 없습니다.\n");
        } while (nTarget != 0);

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