알고리즘 설계와 구현(2)

Top-down approach에 의한 설계

Stepwise Refinement에 의한 변수 도입과 처리과정 도출

알고리즘 설계와 구현 단계

1. 문제정의
2. 알고리즘구상 => 세분화시키기 위한 변수설정 => Block diagram으로 정리
3. Simulation
4. 알고리즘 작성
5. 구현

선택정렬-C함수 구현 예

```
void selection_sort(int list[], int n)
{
        int s, m, j, temp;

        for (s = 0; s < n-1; s++) {
             m = s;
             for (j = s+1; j < n; j++)
                  if (list[j] < list[m]) m = j;
             temp = list[s];
             list[s] = list[m];
             list[m] = temp;
         } /* for */
} /* sort */
```

```
prepare input data into list
for (s=0; s < n-1; s++) {
     m = min_index(list[s] . . . list[n-1])
     swap(list[m], list[s])
}
print the result
```

이진검색(Binary Search) - 문제정의

가. 문제 정의

오름차순으로 정렬된 n개의 데이터로 부터 주어진 어떤 값이 있는지 검색한다.

"중앙값과 비교하여 검색 구간을 줄여간다."

입력 : 정렬된 n개의 정수 배열(list), list[0], list[1], . . . , list[n-1]
찾으려는 값(key)

출력 : 검색결과

	0	1	2	3	4	5	6	7	8	
예) list = {	10,	20,	30,	40,	50,	60,	70,	80,	90 },	key = 70
										key = 25

이진검색(Binary Search) – 알고리즘 구상

나. 알고리즘 구상

1. 입력데이터를 준비한다.
2. 검색구간을 설정한다.
3. 검색구간에 데이터가 있으면 다음을 수행하고 없으면 단계 4.로 간다.
   - 중앙값과 찾으려는 key값이 같으면 그 위치를 출력한다.
   - 같지 않으면 단계2로 간다.
4. 주어진 데이터 집합 안에 key가 존재하지 않으므로 -1을 출력한다.

⇓ 구체화시키기 위한 준비

(1) 단계2에서 데이터의 검색구간을 표시할 변수가 필요하다. (left, right)

left와 right의 초기값은? left = 0, right = n-1

단계3 에서 비교 후 새로운 left와 right의 설정방법은?

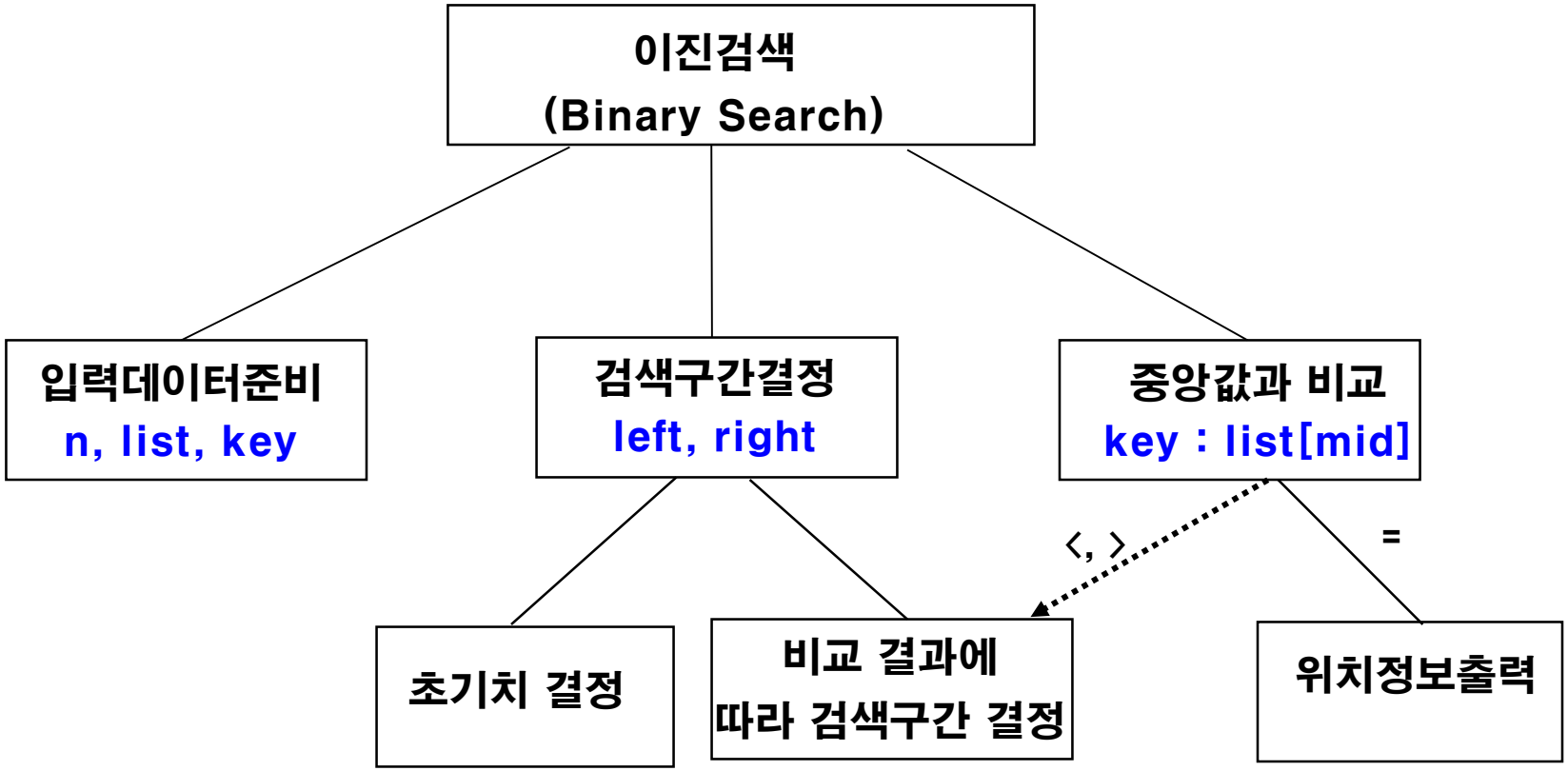
key < list[mid] : right = mid - 1

key > list[mid] : left = mid + 1

(2) 단계3에서 "검색구간에 데이터가 있으면" 조건을 변수로 표현하면? (left <= right)

(3) 중앙값의 인덱스는? mid = (left + right)/2

이진검색(Binary Search) – 알고리즘 구상



key < list[mid] : right = mid – 1

key > list[mid] : left = mid + 1

이진검색(Binary Search) – Simulation

다. Simulation

가능한 입력데이터를 준비하여 구상한 방법을 따라 실행시켜본다.

0 1 2 3 4 5 6 7 8

예) n=9, list = { 10, 20, 30, 40, 50, 60, 70, 80, 90 }

case1) key = 70

left	right	mid	key :list[mid]
0	8	4	70 : 50
5	8	6	70 : 70

case2) key = 25

left	right	mid	key :list[mid]
0	8	4	25 : 50
0	3	1	25 : 20
2	3	2	25 : 30
2	1		

이진검색(Binary Search) – 알고리즘 작성

라. 알고리즘 작성

단계 1: 입력데이터를 준비한다.

n, list[0], . . . , list[n-1], key

단계 2: 검색구간을 설정한다.

초기치 left=0, right = n-1

단계 3: 검색구간에 데이터가 있는 동안 left <= right

다음을 수행하고 없으면 단계 4 로 간다.

3.1 mid = (left + right)/2

3.2 key와 list[mid] 비교

key < list[mid] : right = mid - 1

key > list[mid] : left = mid + 1

key = list[mid] : return mid

단계 4: 주어진 데이터 집합 안에 key가 존재하지 않으므로 -1을 출력한다.

이진검색(Binary Search) – 구현

마. C함수 구현

```
int bsearch(int a[], int n, int key)
{
        int mid;
        int left = 0, right = n-1;
        while (left <= right)  {
            mid = (left + right) / 2;
            if (key > a[mid]) left = mid + 1;
            else if (key < a[mid]) right = mid – 1;
            else return mid;
        } /* while */
        return -1;
   }
```

정리 및 실습

1. 이진검색의 문제정의는 (　　)과 찾으려는 데이터를 비교하여
(　　)을 줄여가며 데이터를 검색하는 것으로 대강의 알고리즘은 다음과 같다.
(1) 입력데이터를 준비한다.
(2) 검색구간을 설정한다.
(3) 검색구간에 데이터가 있으면 다음을 수행하고, 없으면 단계(4)로 간다.
(3.1) 중앙값과 찾으려는 key값이 같으면 그 위치를 출력한다.
(3.2) 같지 않으면 단계(2)로 간다.
(4) 주어진 데이터에 찾으려는 key 값이 존재하지 않으므로 -1을 출력한다.

2. 1.의 알고리즘을 구체화 시키기 위하여 변수를 도입하는 과정에서 다음의 물음에 답하시오.
(1) 단계(2)에서 데이터의 검색 구간을 표시할 변수가 필요하다. (left, right라고 하자)
- left와 right의 초기값은?
- 새로운 left와 right의 설정방법은 ?
key < list[mid] =>
key > list[mid] =>
(2) 단계(3)에서 "검색구간에 데이터가 있으면" 의 조건을 조건식으로 표현하시오.
(3) 중앙값의 인덱스(mid)를 구하기 위한 문장을 쓰시오.

정리 및 실습

3. 내림차순으로 정렬한 데이터를 가지고 있는 경우 이진검색알고리즘의 어떤 부분을 바꾸어야하는지 정리하고

case '>' : right = mid - 1
case '<' : left = mid + 1

이진검색(binary search) 알고리즘에 따라

다음의 예에 대하여 표를 채워가며 시뮬레이션 하시오.

0 1 2 3 4 5 6 7 8 9 10

n=11, list={95, 90, 85, 80, 75, 70, 65, 60, 50, 40, 30}, key=40

left	right	mid	key :list[mid]

정리 및 실습

4. 수업동영상에서 설명한 오름차순으로 정렬되어 준비된 데이터를 가지고
   이진 검색을 수행하는 C함수를 코딩으로 정리하시오.
   int bsearch(int a[], int n, int key)

5. main() 함수를 작성하여 실행 결과를 얻어보자(bsearch.c).

   배열 list에 입력데이터를 받고 찾고자 하는 key 값을 입력 받는다.

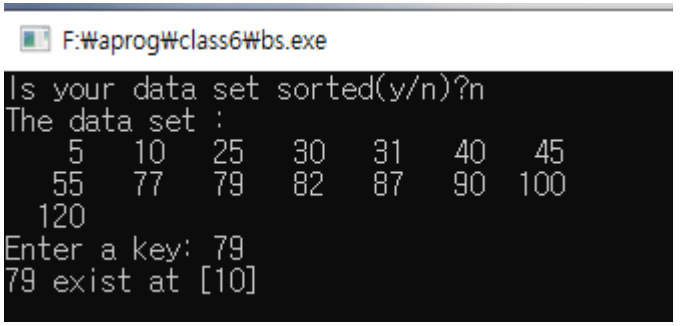
   또한 정렬되었는지 물어 정렬되지 않은 데이터이면

   selection_sort(int list[], int n)에 의해 정렬하고 (정렬된 경우면 스킵한다)

   bsearch(int a[], int n, int key)에 의하여

   주어진 key 값이 몇 번째 데이터인지를 출력한다.

   존재하지 않으면 “not exist"라고 출력한다.



Magic Square 만들기

이차원 배열을 이해하고 정해진 규칙을 프로그램 코드로 바꾸는 연습

가. 문제 정의

n*n 행렬로서 각 행의 합, 열의 합, 두 대각선의 합이 같도록 1부터 n^2의 값을 가지는 경우

예) n이 3인 경우 (합 15)

6	1	8
7	5	3
2	9	4

Magic Square 만드는 규칙

Coxeter's Rule(when n is odd)

1. 첫번째 행의 중앙에 1을 놓는다.
2. 왼쪽 대각선 방향으로 올라가면서 빈자리에 1씩 큰 수를 놓는다.
   이때 행렬의 밖으로 벗어나면 그 방향의 반대편에서 계속하라.
3. 만약 이동하려는 자리에 숫자가 이미 채워져 있으면 지금 위치의 바로 아래에
   숫자를 놓는다.

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

규칙을 프로그램 코드로 쓰기

입력 : 0으로 초기화된 n*n의 2차원 배열 table[n][n]

출력 : n*n의 magic square

행을 움직이는 변수 i, 열을 움직이는 변수 j

Rule 1 : 첫 번째 행의 중앙에 1을 놓는다.

i=0, j=(n-1)/2

table[i][j] = 1

Rule 2 : 왼쪽 대각선 방향으로 올라가면서
빈자리에 1씩 큰 수를 놓는다.
이때 행렬의 밖으로 벗어나면
그 방향의 반대편에서 계속하라.

Rule 3 : 만약 이동하려는 자리에 숫자가
이미 채워져 있으면
지금 위치의 바로 아래에 숫자를 놓는다.

```
for  count = 2 to n*n
    row = i-1, col = j-1
    if (row < 0) row = n-1
    if (col < 0) col = n-1
    if (table[row][col] != 0) i++
    else i=row, j=col
    table[i][j] = count
```

Magic Square 만들기 프로그램 작성

다음의 순서에 따라 프로그램을 작성한다.

- 필요한 변수를 선언한다.
- n을 입력 받는다.
  n이 정해진 범위에 있는지, 홀수인지를 체크한다.
- n*n의 이차원배열 table을 모두 0으로 초기화한다.
- 규칙1, 2,3에 따라 table안에 숫자를 놓는다.
- 결과 table을 출력한다.

Magic Square 만들기 프로그램 작성

```
#define MAX_SIZE 15

void main()
{
    int n, i, j, sum=0;
    int count, row, col;
    int table[MAX_SIZE][MAX_SIZE];

    printf("Enter a number: ");
    scanf("%d", &n);

    if ( (n < 1) || n > MAX_SIZE) {
        printf("Error! size is out of range.\n");
        exit(0);
    }
    if (!(n%2)) {
        printf("Error! size is even.\n");
        exit(0);
    }

    for (i=0; i < n; i++)
        for (j=0; j < n; j++)
            table[i][j] = 0;
```

```
i = 0; j = (n-1)/2;
table[i][j] = 1;

for (count = 2; count <= n*n; count++) {
    row = (i-1 < 0) ? (n-1) : (i-1); /* up */
    col = (j-1 < 0) ? (n-1) : (j-1); /* left */
    if (table[row][col])  i++;
    else {
        i = row; j = col;
    }
    table[i][j] = count;
}
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++)
        printf("%5d", table[i][j]);
    printf("\n");
}
```

```
for (j=0; j < n; j++)
    sum += table[0][j];
printf("Row/Column/Diagonal Sum = %d\n", sum
```

정리 및 실습

1. Magic Square 만들기 알고리즘에서 숫자를 놓기 전에 이차원 배열 table을 모두 0으로 초기화한 이유는 무엇인가?

2. 다음은 magic square를 완성하기 위한 Coxeter의 규칙을 코딩한 내용이다.
이를 이해하고 물음에 답하시오.

```
i = 0; j = (n-1)/2;
table[i][j] = 1;
for (count = 2; count <= n*n; count++) {
    row = (i-1 < 0) ? (n-1) : (i-1); /* up */
    col = (j-1 < 0) ? (n-1) : (j-1); /* left */
    if (table[row][col]) i++;
    else {
        i = row; j = col;
    }
    table[i][j] = count;
}
```

n이 3인 경우 count가 6일 때 i=0, j=0 이다.
그 다음 스텝으로 count=7인 경우 row, col, i, j 에 할당되는 값은?

정리 및 실습

3.Magic Square 만들기 프로그램을 다음과 같이 두개의 함수로 분리하여 다시 작성하여 실행시켜보자.

```
void make_msquare(int table[][MAX_SIZE], int n)
void display(int table[][MAX_SIZE], int n)
void main()
{
      int n, i, j, sum=0;
      int table[MAX_SIZE][MAX_SIZE];
      printf("Enter a number: ");
      scanf("%d", &n);
      if ( (n < 1) || n > MAX_SIZE) {
            printf("Error! size is out of range.\n");
            exit(0);
      }
      if (!(n%2)) {
            printf("Error! size is even.\n");
            exit(0);
      }
      for (i=0; i < n; i++)
          for (j=0; j < n; j++)
              table[i][j] = 0;
      make_msquare(table,n);
      display(table, n);
      for (j=0; j < n; j++)
          sum += table[0][j];
      printf("Row/Column/Diagonal Sum = %d\n", sum);
} /* main */
```

```
void make_msquare(int table[][MAX_SIZE], int n){
    int I,j, count, row, co;
    i = 0; j = (n-1)/2;
    table[i][j] = 1;

    for (count = 2; count <= n*n; count++) {
       row = (i-1 < 0) ? (n-1) : (i-1); /* up */
       col = (j-1 < 0) ? (n-1) : (j-1); /* left */
       if (table[row][col])  i++;
       else {
          i = row; j = col;
       }
       table[i][j] = count;
    }
}
void display(int table[][MAX_SIZE], int n) {
    int i,j;
    for (i = 0; i < n; i++) {
       for (j = 0; j < n; j++)
          printf("%5d", table[i][j]);
       printf("\n");
    }
}
```