7. 探索

7.1 逐次探索

順不同の配列中に特定のデータが入っているかどうかを探索します。配列に順不同にデータが入っている場合，最も単純には以下のように処理します。見つかった場合，配列添字の値を，見つからなかった場合，－1 を返却します。

```
int search(int N, int A[], int V){
    for(int i = 0; i < N; i++) if (A[i] == V) return i;
    return -1;
}
```

7.2 番兵法

For 文の判定（i＜N）を省略するには，最後に探索するデータ（番兵：sentinel）を格納しておきます。この方法を番兵法（sentinel method）といいます。

```
int searchSentinel(int N, int A[], int V){
        A[N] = V; int i = 0; while(A[i] != V) i++;
        if(i == N) return -1; else return i;
}
```

番兵を先頭に置くことで,見つからなかった場合 0 を返すことにすれば簡略化できます。現在の添え字のひとつ前を比較することにすれば，より短いステップで表現できます。

```
int searchSentinel2(int N, int A[], int V){
       A[0] = V; int i = N; while(A[--i] != V); return i;
}
```

7.3 2分探索

配列中のデータが大きさの順に並んでいる場合，注目している範囲の中央値と比較することで，徐々に探索幅を狭めていく 2 分探索法（binary search）を使うことができます。例えば，以下は配列内データが小さい順（昇順）に並んでいる場合です。

```
int binarySearch(int N, int A[],int V){
    int  p1, p2, p; p1 = 0; p2 = N - 1;
    while(p1 <= p2){
            p = p1 + p2;
            if     (A[p]<V)  p1 = p + 1;
            else if(A[p]>V)  p2 = p - 1;
            else return p;
    }
    return -1;
}
```