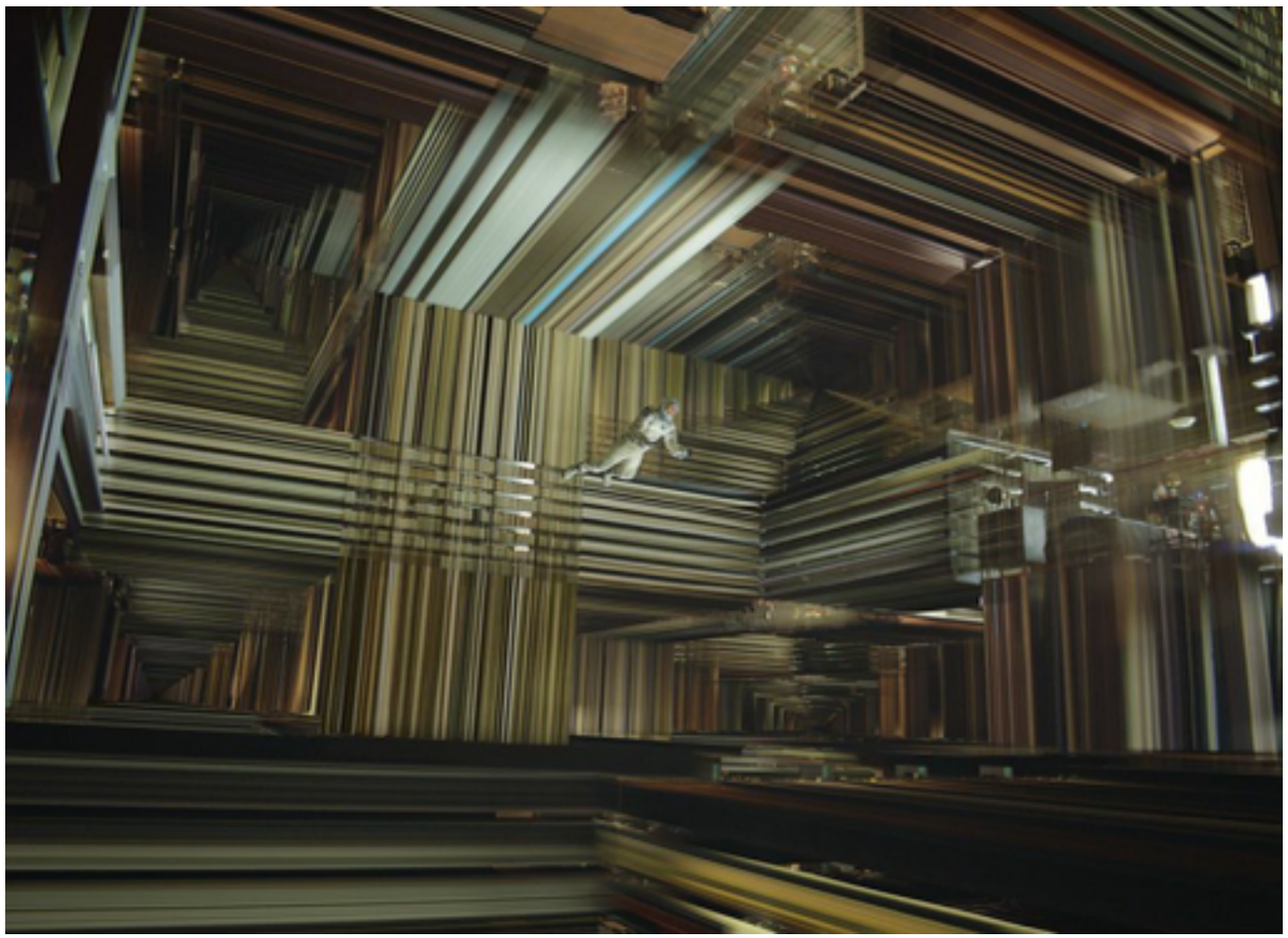


제2장

배열(array)

1D Array

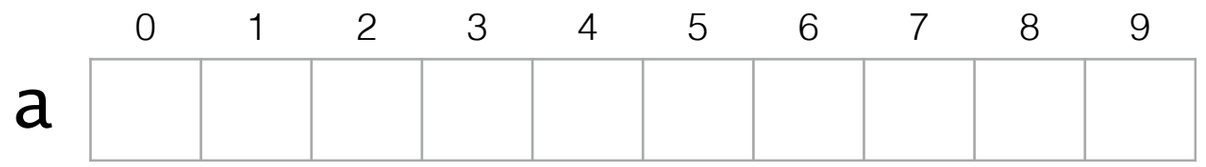
동일한 타입의 여러 개의 데이터

```
double a0, a1, a2, a3, a4, a5, a6, a7, a8, a9;
a0 = 0.0;
a1 = 0.0;
a2 = 0.0;
a3 = 0.0;
a4 = 0.0;
a5 = 0.0;
a6 = 0.0;
a7 = 0.0;
a8 = 0.0;
a9 = 0.0;
…
a4 = 3.0;
…
a4 = 8.0;
…
double x = a4 + a8;
```

10개의 실수를 저장하기 위해서 10개의 double형 변수를 사용하였다.

지루하고 실수할 가능성이 높은 코드이다.

동일한 타입의 여러 개의 데이터




```
double a[10];
…
a[0] = 1.0;
…
a[4] = 3.0;
…
a[8] = 8.0;
…
double x = a[4] + a[8];
```

크기가 10인 실수형 배열을 만들어서
각 칸에 하나의 실수를 저장한다.

배열의 각 칸은 [ ]연산자를 이용하여 엑세스한다.
배열 인덱스는 0부터 시작한다.

훨씬 간결하고 실수할 가능성도 적다.

배열

- 여러 개의 동일한 타입의 데이터를 저장하는 변수
- 배열의 선언

```
array_type array_name [array_size];
```

  - array_type: 배열의 각 칸에 저장될 데이터의 타입
  - array_name: 배열의 이름
  - array_size: 배열의 크기이며 상수여야 함

- 예:

```
int data[100];
double numbers[10];
char word[20];
```

ISO C99에서는 variable length array를 지원함. Visual C에서는 미지원

code01.c

사용자로부터 10개의 정수를 입력받은 후 합을 구하여 출력하는 코드이다.

```
#include <stdio.h>
int main(void)
{
    int data[10];
    printf("Please enter 10 integers: ");
    for (int i=0; i<10; i++)
        scanf(“%d”, &data[i]);

    int sum = 0;
    for ( int i=0; i<10; i++)
        sum += data[i];
    printf("The sum is %d.\n”, sum);

    return 0;
}
```

scanf는 키보드로 부터 입력을 받는다. 입력한 데이터가 저장될 변수명 앞에 &를 붙여야 한다.

code02.c

사용자로부터 10개의 정수를 입력받아 배열에 저장한 그 중 최대값을 구하여 출력하는 코드이다.

```
#include <stdio.h>
int main(void)
{
    int data[10];
    printf("Please enter 10 integers: ");
    for (int i=0; i<10; i++)
        scanf(“%d”, &data[i]);
    int max = 0;
    for (int i=0; i<10; i++)  {
        if (data[i] > max)
            max = data[i];
    printf("The maximum is %d.\n”, max);

    return 0;
}
```

correct ?

code02_2.c

사용자로부터 10개의 정수를 입력받아 배열에 저장한 그 중 최대값을 구하여 출력하는 코드이다.

```
#include <stdio.h>
int main(void)
{
    int data[10];
    printf("Please enter 10 integers: ");
    for (int i=0; i<10; i++)
        scanf(“%d”, &data[i]);

    int max = data[0];
    for (int i=1; i<10; i++)  {
        if (data[i] > max)
            max = data[i];

    printf("The maximum is %d.\n”, max);


    return 0;
}
```

code02_3.c

사용자로부터 10개의 정수를 입력으면서 그 중 최대값을 구하여 출력하는 코드이다. 즉 배열에 저장하지 않는다.

```
#include <stdio.h>
int main(void)
{
    printf("Please enter 10 integers: ");
    int tmp;
    int max = ?;
    for (int i=0; i<10; i++) {
        scanf(“%d”, &tmp);
        if (tmp > max)
            max = tmp;
    }
    printf("The maximum is %d.\n”, max);
    return 0;
}
```

해결방법은 ?

예

최소값	double min = a[0];
	for (int i=1; i<N; i++)
	    if (a[i]<min)
	        min = a[i];

평균	double sum = 0.0;
	for (int i=0; i<N; i++)
	    sum += a[i];
	double average = sum / N;

배열 복사	double b[N];
	for (int i=0; i<N; i++)
	    b[i] = a[i];

연습 1

1. 입력으로 10개의 정수를 받아 순서대로 배열에 저장한다. 그 중 가장 작은 정수와 2번째로 작은 정수를 찾아서 출력하는 프로그램을 작성하라. 정수들은 입력된 순서대로 배열에 저장되어 있어야 한다. 즉 정렬하지 않고 해결하라.

배열의 초기화

- 배열의 초기화

```
int a[10];
for (int i=0; i<10; i++)
    a[i] = 0;
```

- Compile-time 초기화

```
int a[] = { 8, 7, 2, 1, 1, 0, 4, 6 };
double b[] = { 0.0, 2.5, 3.4 };
```

- 모두 0으로 초기화

```
int c[10] = { 0 };
double d[100] = { 0.0 };
```

- memset으로 초기화하기: 사용중인 배열을 다시 초기화할 때 유용함

```
int data[1024];
memset(data, 0, 1024);  // string.h를 include해야함
```

반복적인 코드의 단순화

```
int month;
scanf(“%d”, &month);
int howManyDays;
if (month==1)        howManyDays=31;
else if (month==2)   howManyDays=28;
else if (month==3)   howManyDays=31;
else if (month==4)   howManyDays=30;
else if (month==5)   howManyDays=31;
else if (month==6)   howManyDays=30;
else if (month==7)   howManyDays=31;
else if (month==8)   howManyDays=31;
else if (month==9)   howManyDays=30;
else if (month==10) howManyDays=31;
else if (month==11) howManyDays=30;
else if (month==12) howManyDays=31;
printf(“%d\n”, month);
```

반복적인 코드의 단순화

```
int howManyDays[] =
        {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int month;
scanf("%d", &month);
printf(howManyDays[month]);
```

반복적인 코드의 단순화

```
char * months[] =
{
    "", "Jan", "Feb", "Mar", "Apr", "May", "Jun",
        "Jul", "Aug", "Sep", "Oct", "Nov", “Dec"
};

printf(months[m]);
```

연습 2

2. 2차원 평면에서 동, 서, 남, 북의 4가지 방향을 각각 0부터 3까지의 정수로 나타내자. 입력으로 한 점의 좌표 (x,y)와 이동할 방향을 나타내는 정수 d (0≤d≤3)를 받은 후, 점 (x,y)에서 d방향으로 거리 1만큼 이동한 점의 좌표를 출력하는 프로그램을 작성하라. x와 y는 정수이고, 북쪽이 y-좌표가 증가하는 방향이고, 동쪽이 x-좌표가 증가하는 방향이다.

배열들 간의 치환문과 비교

- 두 배열이 있다고 하자.

```
int a[10];
int b[10];
```

- 두 배열간의 치환문?

```
b=a;
```

- 혹은 두 배열간의 동일성 검사?

```
if (a==b)
    printf("equal\n");
```

code03.c

사용자로부터 최대 100개의 정수를 입력받은 후 서로 "약수-배수" 관계의 모든 쌍을 찾아 출력한다.

```
#include <stdio.h>
int main(void)
{
    int data[100];
    int n=0;
    printf("Please Enter at most 100 positive integers
                                and press Ctl-z.\n");
    int tmp;
    while (scanf("%d", &tmp) != EOF)
        data[n++] = tmp;
    for ( int i=0; i<n-1; i++) {
        for ( int j=i+1; j<n; j++ ) {
            if (data[i] % data[j] == 0 || data[j] % data[i] == 0 )
                printf("(%d, %d)\n", data[i] , data[j]);
        }
    }
}
```

Control-Z를 누르면 scanf는 EOF을 반환하고 따라서 while문이 종료한다. 미리 지정되지 않은 개수의 입력을 받을 때 유용하다.

code04.c: 순차검색

사용자로부터 최대 100개의 정수를 입력받아 배열에 저장한다. 그런 다음 또하나의 정수를 입력받아서 배열에 동일한 수가 있는지 검색한다.

```
#include <stdio.h>
int main(void)
{
    int data[100];
    int n=0;
     printf("Enter at most 100 positive integers and press Ctl-z.”);
     int tmp;
     while (scanf(“%d”, &tmp) != EOF)
         data[n++] = tmp;

    int target;
    scanf(“%d”, &target);
     for ( int i=0; i<n; i++) {
        if (data[i]==target)  {
            printf(“Found at %d\n”, i);
            break;
        }
     }
}
```

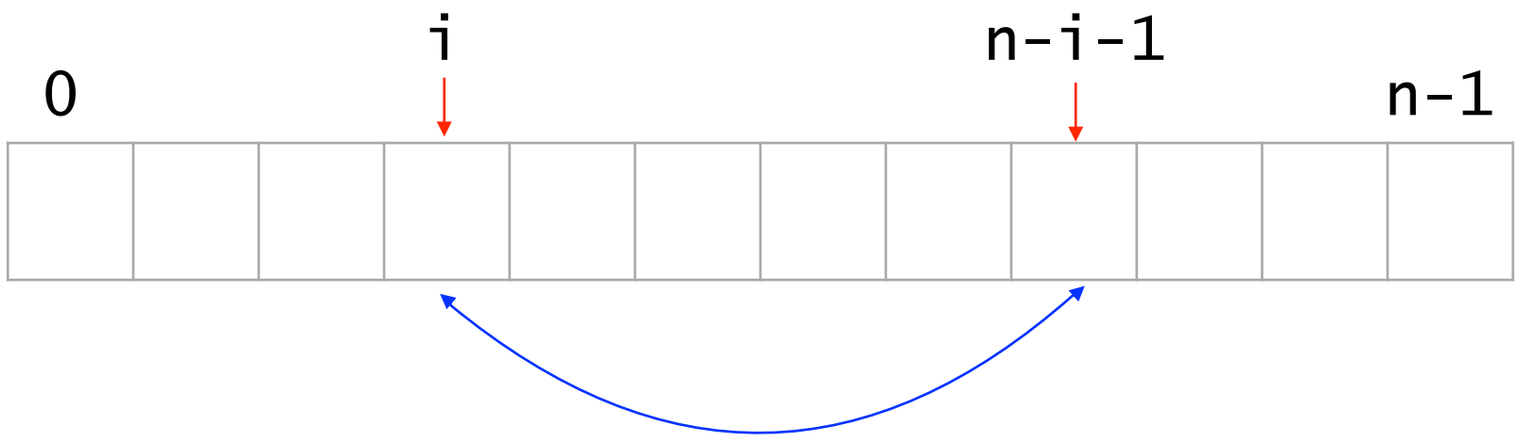
이렇게 순차적으로 값을 비교하여 원하는 값을 찾는 방법을 순차검색이라고 부른다.

code05.c: 배열 뒤집기

```
#include <stdio.h>
int main(void)
{
    int data[100];
    int n=0;
    printf("Enter at most 100 positive integers and press Ctl-z.”);
    int tmp;
    while (scanf(“%d”, &tmp) != EOF)
        data[n++] = tmp;
    for (int i=0; i<n/2; i++) {
        double tmp = data[i];
        data[i] = data[n-1-i];
        data[n-i-1] = tmp;
    }
    for (int i=0; i<n; i++)
        printf(“%d “, data[i]);
}
```

swap data[i] and data[n-i-1]

code05.c: 배열 뒤집기



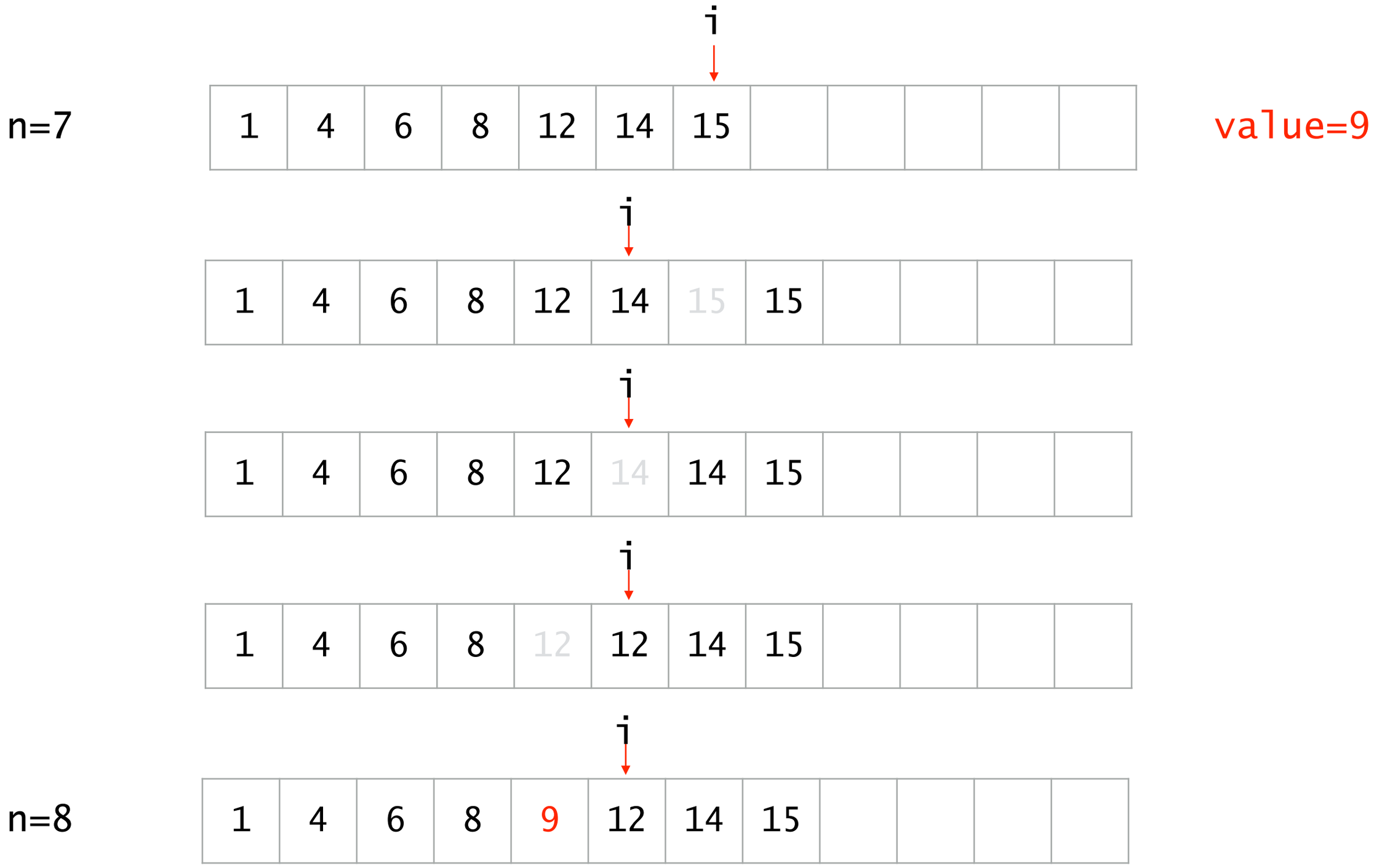
swap data[i] and data[n-i-1]

code06.c: Ordered List

입력된 정수들을 배열에 크기 순으로 저장한다.

```
#include <stdio.h>
int main(void)
{
    int data[100];
    int n=0;
    printf("Enter at most 100 positive integers and press Ctl-z.”);
    int value;
    while (scanf(“%d”, &value) != EOF)   {
        int i=n-1;
        while (i>=0 && data[i]>value) {
            data[i+1] = data[i];
            i--;
        }
        data[i+1] = value;
        n++;
    }
    for (int i=0; i<n; i++)
        printf(“%d “, data[i]);
}
```

code06.c: Ordered List



3. 입력으로 최대 100개의 정수를 받아 오름차순으로 배열에 저장한다. Control-Z를 누르면 입력을 종료한다. 다시 추가로 하나의 정수를 입력 받아서 만약 그 정수가 배열에 있으면 배열로 부터 삭제한다. 이때 삭제된 자리를 빈 칸으로 두어서는 안되며 배열에 저장된 정수들은 삭제한 후에도 여전히 오름차순으로 정렬되어 있어야 한다. 삭제한 후 배열에 저장된 정수들을 순서대로 출력하라.

code07.c: shuffle

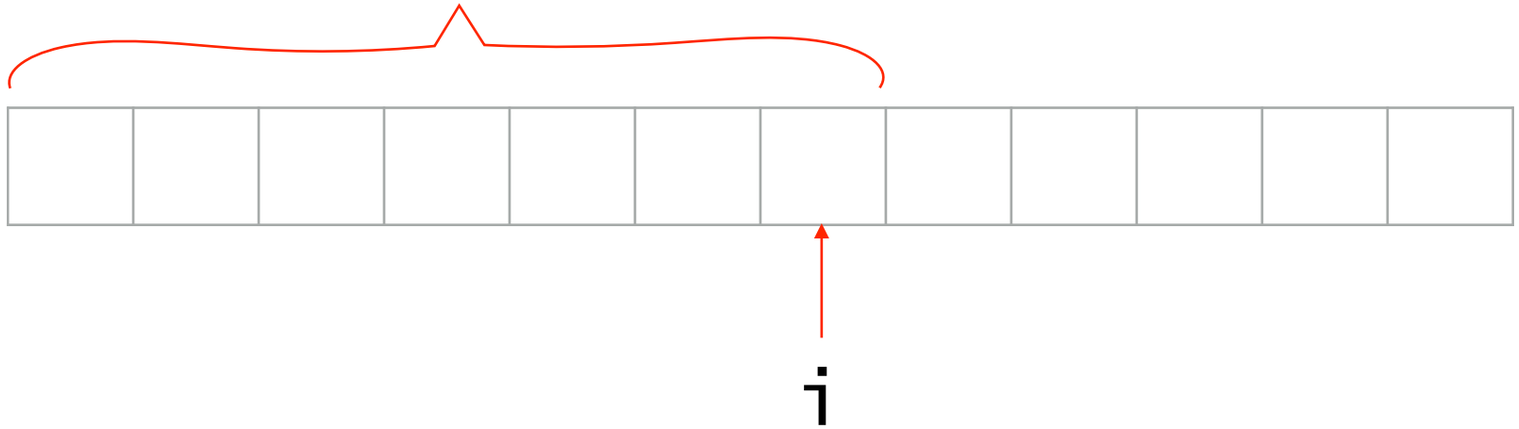
배열에 저장된 정수들을 random한 순서가 되도록 shuffle한다.

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int numbers[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
    int N = 8;
    for (int i = 0; i < N; i++) {
        int r = rand() % (i+1);

        int tmp = numbers[i];
        numbers[i] = numbers[r];
        numbers[r] = tmp;
    }

    for (int i = 0; i < N; i++)
        printf("%d ", numbers[i]);
    return 0;
}
```

stdlib가 제공하는 rand()함수는 0에서 RAND_MAX 사이의 정수를 랜덤하게 생성하여 반환한다. 그것을 (i+1)으로 나누어 나머지를 구하면 0에서 i사이의 정수가 된다.

Knuth's shuffling

이들 중 하나를 랜덤하게 선택하여 i번째 원소와 swap한다.



code08.c: Sampling without replacement

0~N-1 사이의 정수들 중 반복없이 M개를 랜덤 샘플링한다.

```
#include <stdio.h>
#define MAX 100

int main(void)  {
    int perm[MAX];
    int M, N;
    scanf("%d %d", &N, &M);

    for (int j = 0; j < N; j++)
        perm[j] = j;

    for (int i = 0; i < M; i++)  {
        int r = i + (rand()%(N-i));
        int t = perm[r];
        perm[r] = perm[i];
        perm[i] = t;
    }

    for (int i = 0; i < M; i++)
        print(perm[i]);
}
```

code09.c: 가장 긴 평지

- N개의 정수를 입력받아 순서대로 배열에 저장한다. 연속해서 등장하는 동일한 값의 최대 개수를 세는 프로그램을 작성하라. 예를 들어 입력이 다음과 같다면

6 3 2 2 2 5 5 5 5 7 8 1 1

연속해서 등장하는 동일한 값의 최대 개수는 밑줄친 4개이다.

code09.c: 가장 긴 평지

```
#include <stdio.h>
#define MAX 100

int main(void)  {
    int numbers[MAX];
    int N;

    // N개의 정수를 입력받아 배열 numbers에 저장한다.

    int count = 1;
    int max = 1;
    for (int i = 1; i < N; i++)  {
        if (numbers[i]==numbers[i-1])  {
            count++;
            if (count > max)
                max = count;
        }
        else
            count=1;
    }
    print(max);
}
```

연습

4. 가장 긴 평지 문제에서 개수와 함께 가장 긴 구간의 시작 인덱스와 끝 인덱스를 출력하는 프로그램을 작성하라. 예를 들어 입력이 다음과 같다면

6 3 2 2 2 5 5 5 5 7 8 1 1

출력은 4(5, 8)로 한다. 4는 개수이고, (5,8)은 구간이다.

연습: Left-to-Right Minima

5. 입력으로 먼저 하나의 정수 N(≤100)을 받는다. 이어서 N개의 정수를 입력받아 순서대로 배열에 저장한다. 이 정수들 중에서 left-to-right minima의 개수를 구해 출력하는 프로그램을 작성하라. left-to-right minima란 자신 보다 앞쪽에 있는 수들보다 작거나 같은 수를 말한다. 예를 들어 N=8이고 입력된 정수들이 다음과 같다면

4 3 5 6 2 2 1 9

minima의 개수는 밑줄 친 5개이다.

연습: Inverse permutation

6. 입력으로 하나의 정수 N(≤100)을 받는다. 그런 다음 0에서 N-1까지의 정수들을 뒤섞인 순서로 입력받아 배열에 저장한다. 즉 0에서 N-1사이의 정수들의 하나의 순열(permutation)을 입력받는 것이다. 그런 다음 이 순열의 역순열(inverse permutation)을 구해 출력하는 프로그램을 작성하라. 정수들이 저장된 배열을 a라고 했을 때 역순열이란 다음과 같은 조건을 만족하는 다른 배열 b를 말한다.

$$a[b[i]] = b[a[i]] = i$$

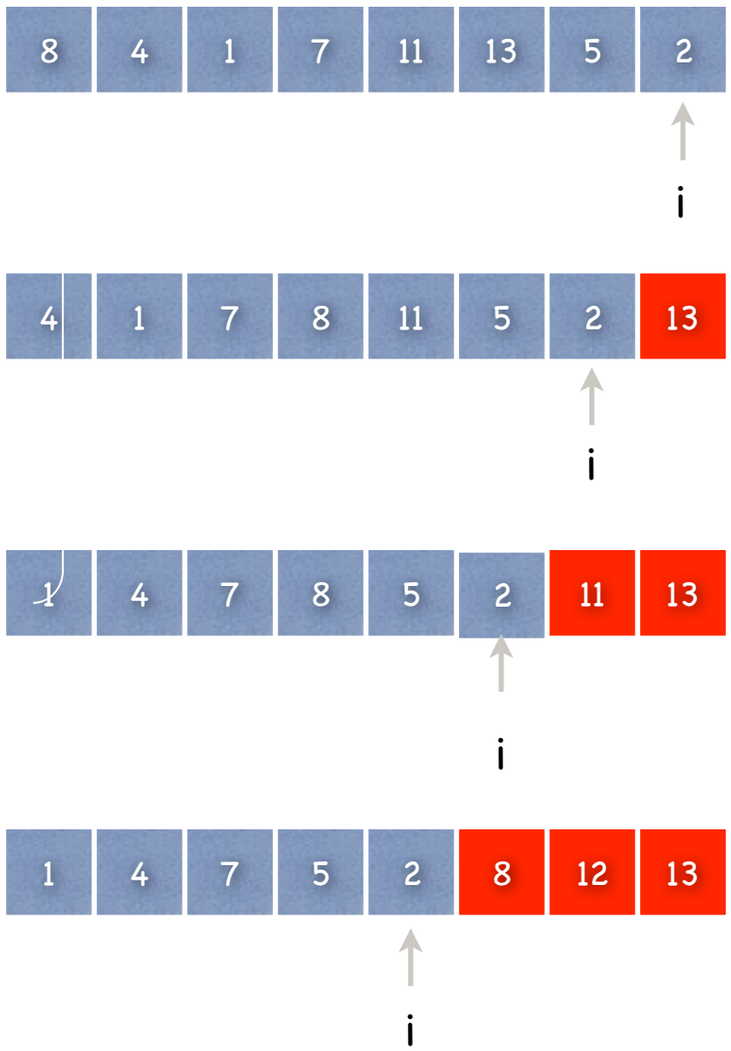
가령 N=5이고 입력된 순열이 “1 3 0 2 4”라면 역순열은 “2 0 3 1 4”이다.

code10.c

사용자로부터 최대 100개의 정수를 입력받은 후 오름차순으로 정렬(sort)하여 출력한다.

```
#include <stdio.h>

int main(void)
{
    int data[100];
int n=0;

printf("Enter at most 100 integers and press Ctl-z.”);
int tmp;
while (scanf(“%d”, &tmp) != EOF)
    data[n++] = tmp;
```

Bubble sort

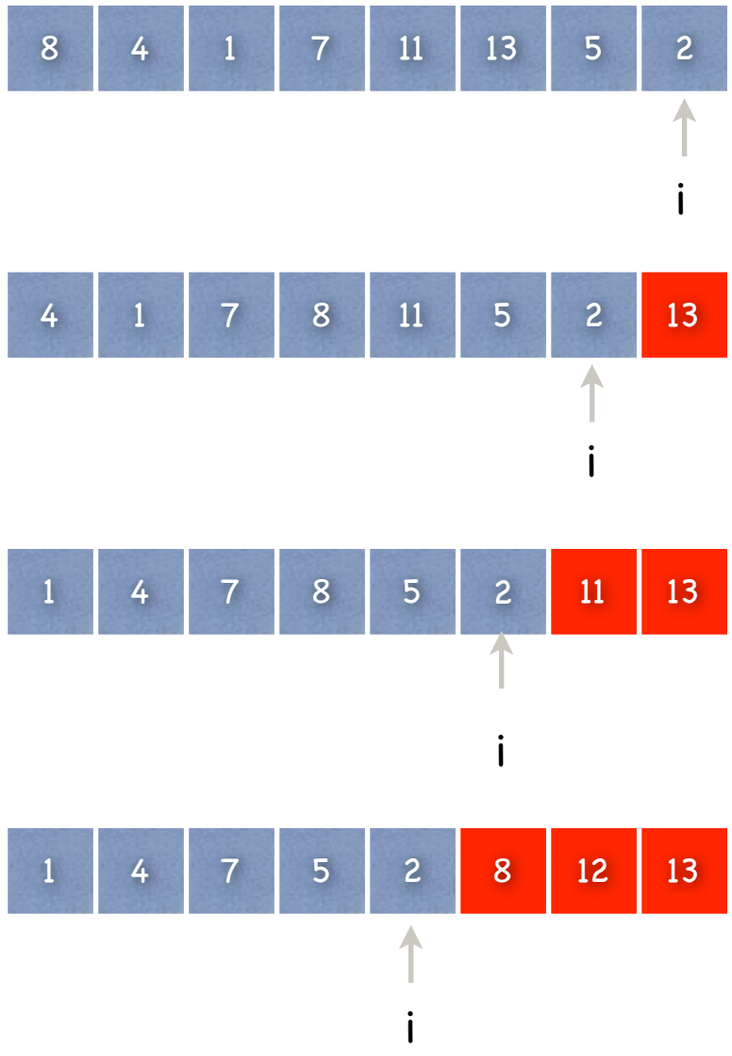


code10.c (continued)

```
// bubble sort
for ( int i=n-1; i>0; i--) {
    for ( int j=0; j<i; j++ ) {
        if (data[j] > data[j+1]) {
            int tmp = data[j];
            data[j] = data[j+1];
            data[j+1] = tmp;
        }
    }
}
printf("Sorted data:\n”);
for ( int i=0; i<n; i++)
    printf(“%d\n”, data[i]);
return 0;
}
```

data[0] ~ data[i] 중에서 최대값을 data[i] 위치로 몰아가는 일

swap



연습: Find a duplicate

7. 입력으로 하나의 정수 N(≤100)을 받은 후 N개의 정수들을 입력받는다. 입력된 정수들 중에 중복된 값이 있는지 검사하는 프로그램을 작성하라. 중복이 있으면 YES, 없으면 NO라고 출력하라. 배열에 저장된 데이터의 순서를 그대로 유지할 필요는 없다. 하지만 추가 배열을 사용해서는 안된다.

code11.c: Coupon collector

- 0~5 사이의 정수들을 연속해서 입력받는다. 중복된 값들이 입력될 수 있다. 0~5까지의 모든 정수들이 각각 적어도 한 번 이상 입력되면 그때까지 입력된 정수의 개수를 출력하고 종료한다. 예를 들어서 다음과 같이 입력되면 12를 출력하고 종료한다.

```
5 5 2 2 1 0 0 2 5 2 3 4
```

code11.c: Coupon collector

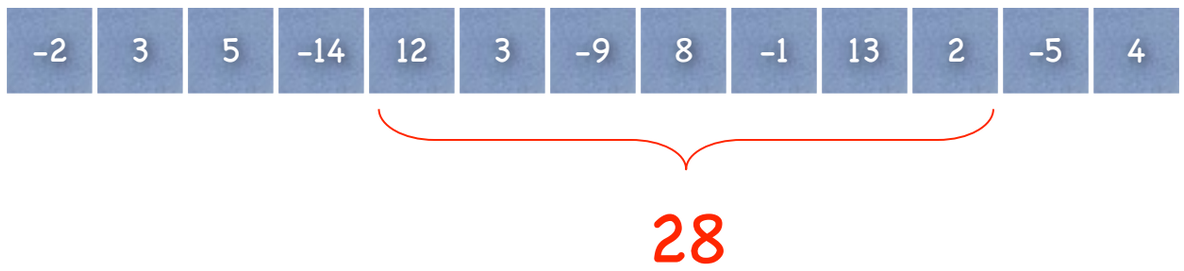
```
#include <stdio.h>

int main(void) {
    int cardcnt = 0;     // number of cards collected
    int valcnt = 0;      // number of distinct cards
    int found[6] = {0};
    while (valcnt < 6) {
        int val;
        scanf(“%d”, &val);
        cardcnt++;
        if (found[val]==0) {
            valcnt++;
            found[val]=1;
        }
    }
    printf(“%d\n”, cardcnt);
}
```

code11.c: Coupon collector

val	found						valcnt	cardcnt
	0	1	2	3	4	5		
	F	F	F	F	F	F	0	0
2	F	F	T	F	F	F	1	1
0	T	F	T	F	F	F	2	2
4	T	F	T	F	T	F	3	3
0	T	F	T	F	T	F	3	4
1	T	T	T	F	T	F	4	5
2	T	T	T	F	T	F	4	6
5	T	T	T	F	T	T	5	7
0	T	T	T	F	T	T	5	8
1	T	T	T	F	T	T	5	9
3	T	T	T	T	T	T	6	10

code12.c: 최대 부분합 구하기

- 입력으로 n개의 정수가 주어진다. 1개 이상의 연속된 정수들을 더하여 구할 수 있는 최대값을 찾아라.



code12.c: 최대 부분합 구하기

```
#include <stdio.h>
int main(void)
{
   int data[1000];
   int n=0;
   // n개의 정수들을 입력 받아 배열 data에 저장한다.
   int maxSum = data[0];
   for ( int begin=0; begin<n; begin++ )  {
      for ( int end=begin; end<n; end++) {
         int sum = 0;
         for (int i=begin; i<=end; i++)
            sum += data[i];
         if (sum > maxSum)
            maxSum = sumInterval;
      }
   }
   printf("The maximum sum is %d.\n", maxSum);
   return 0;
}
```

code12_2.c: 최대 부분합 구하기

```
#include <stdio.h>
int main(void)
{
    int data[1000];
    int n=0;
    // n개의 정수들을 입력 받아 배열 data에 저장한다.
    int maxSum = data[0];
    for ( int begin=0; begin<n; begin++ )  {
        int sum = 0;
        for ( int end=begin; end<n; end++) {
            sum += data[end];
            if (sum > maxSum)
                maxSum = sumInterval;
        }
    }
    printf("The maximum sum is %d.\n", maxSum);
    return 0;
}
```

2D Arrays

선언과 초기화

- 2차원 배열의 선언과 초기화

```
int a[N][N];
for (int i=0; i<N; i++)
    for (int j=0; j<N; j++)
        a[i][j] = 0;
```

- Compile-time 초기화

```
int b[][] = {  {1, 2, 3, 4},
               {3, 1, 2, 4},
               {4, 3, 2, 1} };
```

- 0으로 초기화

```
int c[10][10] = {0};
```

code13.c: 전치행렬

- 입력으로 두 정수 M과 N을 입력받는다. 각각은 100이하이다. 그런 다음 M*N 행렬의 원소들을 행우선 순서로 입력받는다. 그런 다음 이 행렬의 전치행렬을 구한 후 출력하는 프로그램을 작성하라.

code13.c: 전치행렬

```
#include <stdio.h>
#define MAX 100

int main(void)
{
    int matrix[MAX][MAX];
    int M, N;
    scanf("%d %d", &M, &N);
    for (int i=0; i<M; i++)
        for (int j=0; j<N; j++)
            scanf("%d", &matrix[i][j]);

    int transposed[MAX][MAX];
    for (int i=0; i<M; i++)
        for (int j=0; j<N; j++)
            transposed[j][i] = matrix[i][j];
    …
    return 0;
}
```

code14.c: 행렬 곱셈

```
#include <stdio.h>
#define MAX 100

int main(void)
{
   int A[MAX][MAX];
   int B[MAX][MAX];
   // p*q행렬 A와 q*r행렬 B를 입력받는다.

   int C[MAX][MAX];
   for (int i=0; i<p; i++)  {
      for (int j=0; j<r; j++) {
         C[i][j] = 0;
         for (int k=0; k<q; k++)
            c[i][j] += A[i][k]*B[k][j];
      }
   }
   …
   return 0;
}
```

code15.c: Minesweeper

- 입력으로 아래 왼쪽과 같은 모양의 N*N 크기의 minesweeper 게임의 모양을 받는다. 폭탄은 *로 표시되고 폭탄이 아닌 자리는 .표시된다. 오른쪽 그림과 같이 출력하는 프로그램을 작성하라.

*	*	.	.
.	.	.	.
.	*	.	.
*	.	*	*

*	*	1	0
3	3	2	0
2	*	3	2
*	3	*	*

code15.c: Minesweeper

```
#include <stdio.h>
#define MAX 100

int main(void)
{
    char grid[MAX][MAX];
    int N;
    scanf(“%d”, &N);

    for (int i=0; i<N; i++)
        for (int j=0; j<N; j++)
            scanf(“%c”, &grid[i][j]);
```

code15.c: Minesweeper

```
for (int i=0; i<N; i++)   {
   for (int j=0; j<N; j++)  {
      if (grid[i][j]==‘*’)
         printf(“* “);
      else  {
         int count = 0;
         if (i-1>=0 && j-1>=0 && grid[i-1][j-1]==‘*’)   count++;
         if (i-1>=0 && grid[i-1][j]==‘*’)               count++;
         if (i-1>=0 && j+1<N && grid[i-1][j+1]==‘*’)    count++;
         if (j-1>=0 && grid[i][j-1]==‘*’)               count++;
         if (j+1<N && grid[i][j+1]==‘*’)                count++;
         if (i+1<N && j-1>=0 && grid[i+1][j-1]==‘*’)    count++;
         if (i+1<N && grid[i+1][j]==‘*’)                count++;
         if (i+1<N && j+1<N && grid[i+1][j+1]==‘*’)     count++;
         printf(“%d “, count);
      }
   }
}
```

code15_2.c: Minesweeper

```
#include <stdio.h>
#define MAX 102

int main(void)
{
    char grid[MAX][MAX] = {'.'};
    int N;
    scanf("%d", &N);

    for (int i=1; i<=N; i++)
        for (int j=1; j<=N; j++)
            scanf("%c", &grid[i][j]);
```

code15_2.c: Minesweeper

```
for (int i=1; i<=N; i++)   {
   for (int j=1; j<=N; j++)  {
      if (grid[i][j]==‘*’)
         printf(“* “);
      else  {
         int count = 0;
         if (grid[i-1][j-1]==‘*’)  count++;
         if (grid[i-1][j]==‘*’)    count++;
         if (grid[i-1][j+1]==‘*’)  count++;
         if (grid[i][j-1]==‘*’)    count++;
         if (grid[i][j+1]==‘*’)    count++;
         if (grid[i+1][j-1]==‘*’)  count++;
         if (grid[i+1][j]==‘*’)    count++;
         if (grid[i+1][j+1]==‘*’)  count++;
         printf(“%d “, count);
      }
   }
}
```

code15_3.c: Minesweeper

```
int offset[][] = { {-1,-1}, {-1,0}, {-1,1}, {0,-1},
                   {0,1},   {1,-1}, {1,0},  {1,1}};

for (int i=1; i<=N; i++)   {
   for (int j=1; j<=N; j++)  {
      if (grid[i][j]==‘*’)
         printf(“* “);
      else  {
         int count = 0;
         for (int k=0; k<8; k++)
            if (grid[i+offset[k][0]][j+offset[k][1]]==‘*’)
               count++;
         printf(“%d “, count);
      }
   }
}
}
```

연습: Verifying Knuth Shuffling

- Knuth's shuffling 알고리즘이 제대로 random shuffling을 수행하는지 통계적으로 테스트하는 프로그램을 작성하라. 입력으로 두 정수 N과 M을 받는다. 그런 다음 0에서 N-1까지의 정수를 순서대로 저장한 배열을 만들고 그것을 Knuth's shuffling 알고리즘으로 shuffle하는 일을 M번 반복한다. shuffle된 결과에서 특정한 수가 특정한 위치에 올 확률은 1/N로 동일하다. 따라서 M번 반복하면 기대값은 M/N이 된다. 실제 실험 결과를 저장할 N*N행렬을 만들어 출력하라. 이 행렬의 i-행, j-열의 값은 i가 j번째 자리에 위치한 빈도이다.