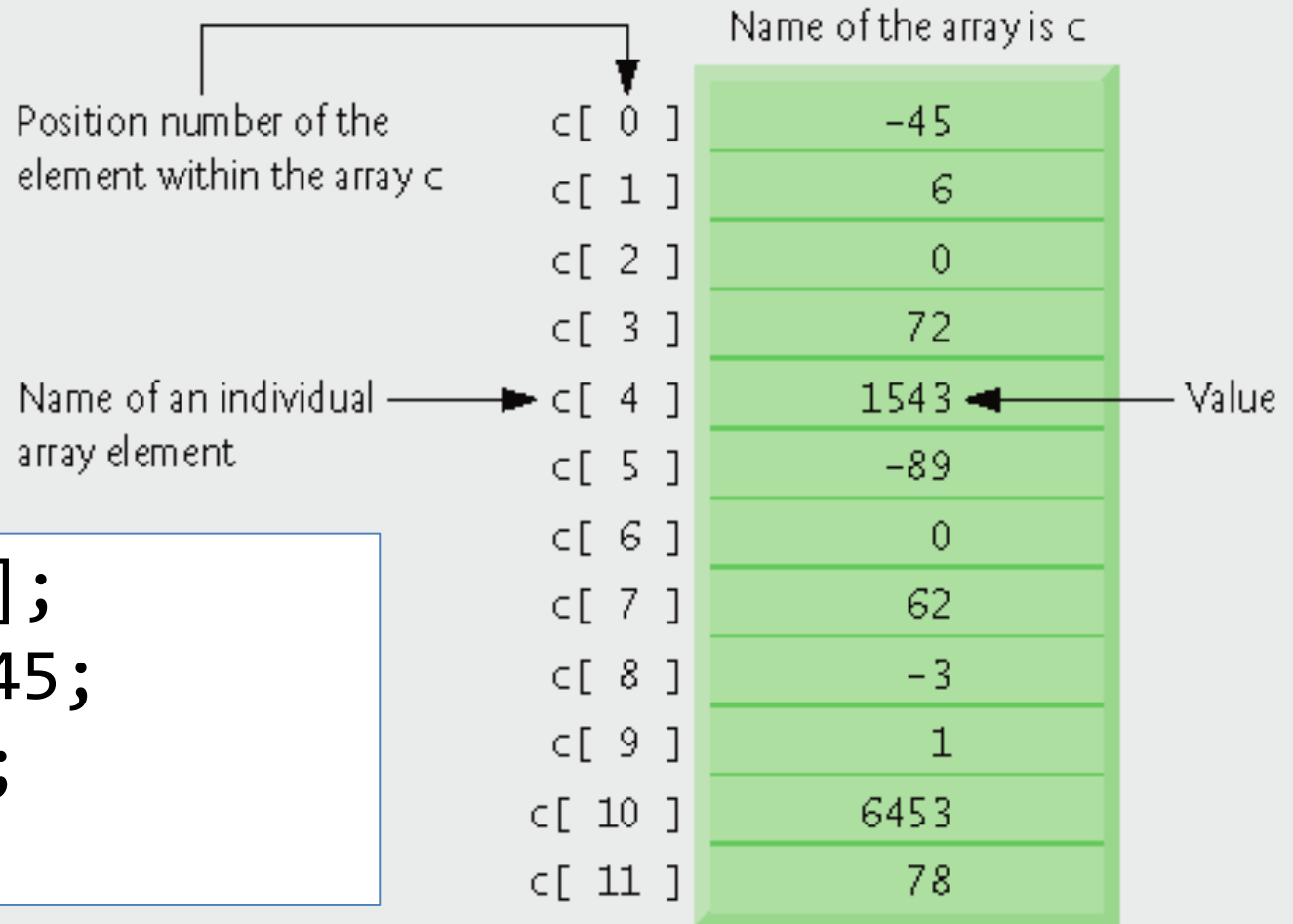


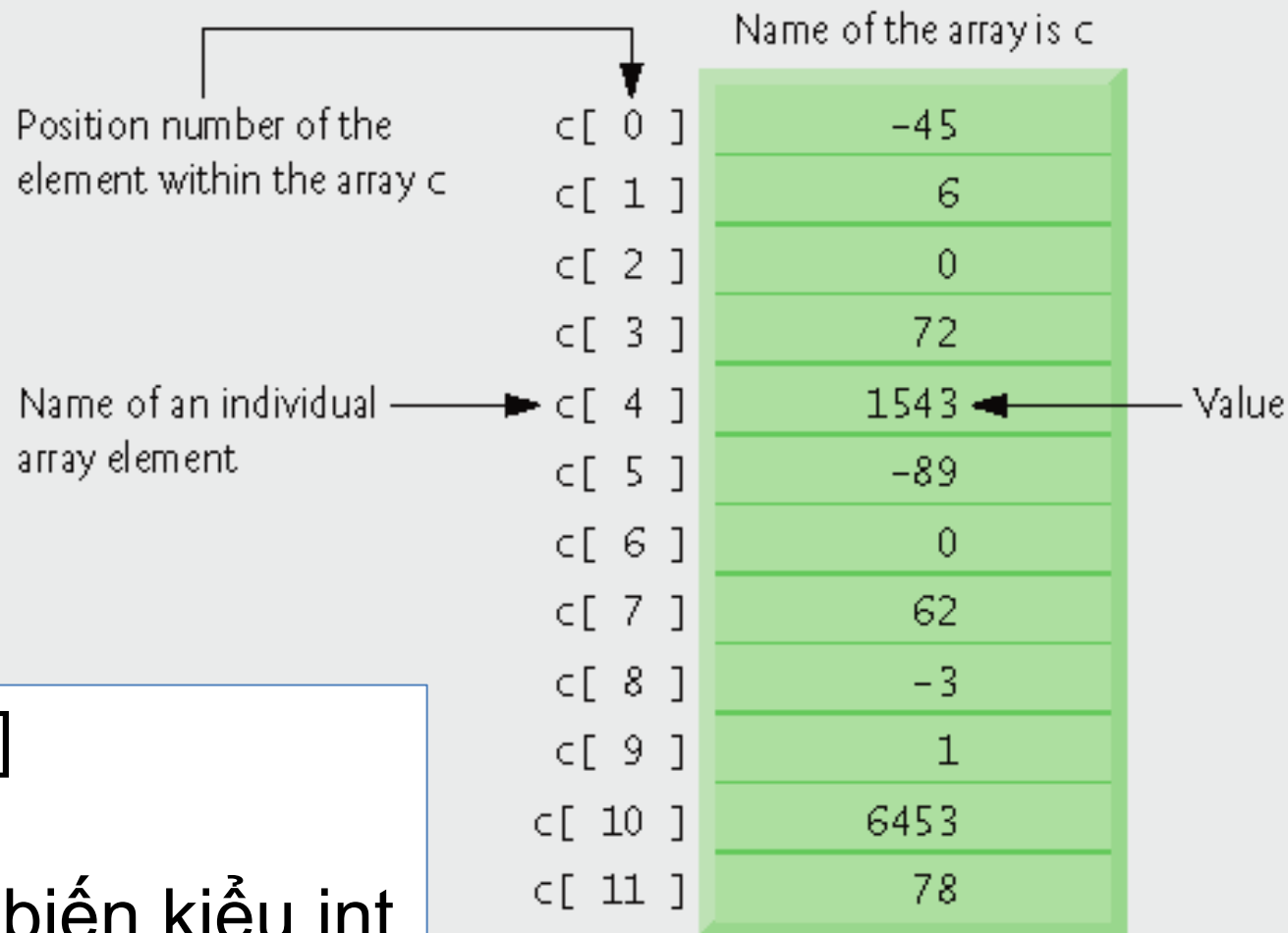
# Mảng

Lập trình nâng cao



```
int c[12];  
c[0] = -45;  
c[1] = 6;  
...
```

```
int c[] = {-45, 6, 0, 72, 1543, -89,  
           0, 62, -3, 1, 6453, 78};
```



```
int c[12]
```

- chuỗi 12 biến kiểu int
- chỉ số xuất phát từ 0



# Sử dụng

```
const int N = 12;
```


```
int c[N] = {1, 2, 3};
```

```
C[4] = 5;
```

```
int sum = 0;
```

```
for (int i = 0; i < N; i++) {  
    sum += c[i];  
}
```

Có thể dùng **hằng** để  
khai báo kích thước  
mảng



# Lỗi thường gặp: quên khởi tạo

- Nếu không khởi tạo, biến địa phương sẽ có giá trị không xác định, dẫn đến lỗi logic

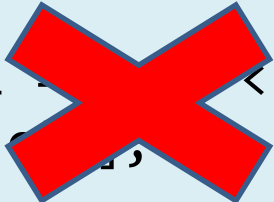
```
int main() {  
    int c[N];  
  
    int sum = 0;  
    for (int i = 0; i < N; i++) {  
        sum += c[i];  
    }  
}
```

# Good Program Practice

- Dùng hằng thay vì giá trị trực tiếp để khai báo kích thước mảng
- Lí do: tránh *magic number* lặp đi lặp lại trong code

```
int c[12];
```

```
for (int i = 0; i < 12; i++) {  
    sum += c[i];  
}
```



```
const int SIZE = 12;  
int c[SIZE];  
for (int i = 0; i < SIZE; i++) {  
    sum += c[i];  
}
```

# Chú ý!!!!

- **C++ không kiểm tra giới hạn mảng**  
(no array bounds checking)  
a[-1] a[100], a[164]... trong mảng 100 phần tử
- Gây lỗi logic xuất hiện trong thời gian chạy
- Hậu quả của việc truy nhập ngoài mảng nghiêm trọng tùy trường hợp và tùy hệ thống
  - Một biến không liên quan bị truy nhập
  - Lỗi nghiêm trọng làm sập chương trình



# Xâu kí tự - mảng char **null-terminated character sequences** (còn gọi là **C – string**)

```
char s[] = "Hi";
```

tương đương

```
char s[] = {'H', 'i', '\0'};
```

Mảng chứa các kí tự


**Chặn cuối** là một kí tự null – mã bằng 0 ( '\0' )

# Ví dụ

```
#include <iostream>
using namespace std;

int main () {
    char question[] = "What is your name?";
    char answer[80];
    cout << question;
    cin >> answer;
    cout << "Hello, " << answer;
    return 0;
}
```

Input/output  
như biến thường



# Thư viện xử lý chuỗi ký tự

```
#include <iostream>
#include <cstring>
```

Enter a string: **abcd**  
\_abcd\_ length: 4

```
using namespace std;
```

```
int main()
{
```

```
    char s[100];
```

```
    cout << "Enter a string: ";
```

```
    cin >> s;
```

```
    cout << "_" << s << "_ length: " << strlen(s);
```

```
}
```

Thư viện *string.h*  
còn gọi là *cstring*.  
Cung cấp *strlen* và  
nhiều hàm khác

# Thư viện xử lý chuỗi ký tự

- So sánh, nối, chép...
- Danh sách các hàm, mô tả, ví dụ sử dụng:
  - <http://www.cplusplus.com/reference/cstring/>
- Lưu ý
  - `#include <cstring>` *tương đương* `#include <string.h>`  
nhưng nên dùng `cstring`

# Lỗi thường gặp: đọc sâu quá kích thước

- Dùng cin đọc một chuỗi vào một mảng có kích thước ngắn hơn chuỗi đó.
  - Chuỗi input bị ghi tràn ra ngoài mảng, đè lên các vùng dữ liệu khác
- Lỗi tương tự có thể xảy ra đối với
  - strcpy(destinationString, sourceString)
  - gets(str) đọc một chuỗi ký tự từ standard input và ghi vào chuỗi str (đã bị loại khỏi C++11)
- Ghi nhớ: **C++ không kiểm tra tràn mảng!!!**

# C++ string

```
#include <iostream>
using namespace std;
```

← Không cần include

```
int main () {
    string question = "What is your name?";
    string answer;
    cout << question;
    cin >> answer;
    cout << "Hello, " << answer;
    return 0;
}
```

← Input/output  
như biến thường

# C++ string

```
#include <iostream>
using namespace std;

int main () {
    string question = "What is your name?";
    string answer;
    cout << question;
    cin >> answer;
    cout << "Hello, " << answer;
    return 0;
}
```

```
char myntcs[] = "some text";
string mystring = myntcs; // convert c-string to string
cout << mystring; // printed as a library string
cout << mystring.c_str(); // printed as a c-string
```

# So sánh C-string và C++ string

## C-string

- kích thước cố định, xác định khi biên dịch
- Bản chất là mảng
- Thư viện tiện ích `<cstring>`

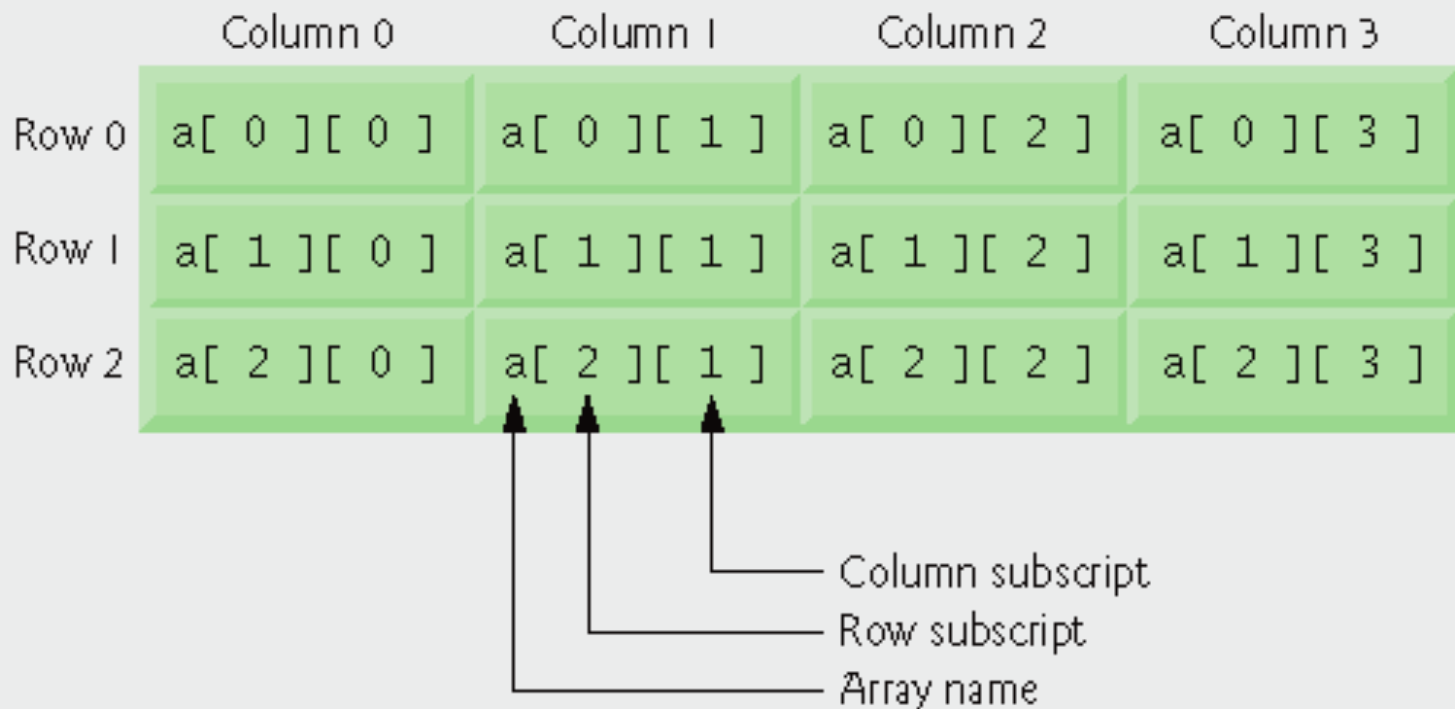
## C++ string

- kích thước động, xác định trong khi chương trình chạy
- Bản chất là đối tượng (học sau)
- Thư viện tiện ích `<string>` (không phải `<string.h>`)



# Mảng hai chiều

```
int a[3][4]; //mảng gồm 3 mảng một chiều độ dài 4
```



# Hàng hay cột?

- “*chỉ số thứ nhất là hàng, chỉ số thứ hai là cột*” chỉ là quy ước thông dụng. Không phải quy tắc!
- Tùy bài toán, hãy tự quy ước theo cách bạn muốn.

|         |         |
|---------|---------|
| a[0][0] | a[0][1] |
| a[1][0] | a[1][1] |
| a[2][0] | a[2][1] |

... còn gì nữa?

```
int a[3][2];
```

|         |         |         |
|---------|---------|---------|
| a[0][1] | a[1][1] | a[2][1] |
| a[0][0] | a[1][0] | a[2][0] |

|         |         |
|---------|---------|
| a[2][1] | a[2][0] |
| a[1][1] | a[1][0] |
| a[0][1] | a[0][0] |

# Hàng hay cột?

- ... hoặc chẳng phải hàng hay cột mà chỉ đơn giản là một cặp chỉ số

$a[i][j]$  có thể là:

- Khoảng cách giữa điểm thứ  $i$  và điểm thứ  $j$
- Người thứ  $i$  và người thứ  $j$  có quen nhau hay không
- Giá phải trả nếu thực hiện công việc  $i$  vào ngày  $j$
- Điểm môn học  $j$  của sinh viên  $i$
- Điểm môn học  $i$  của sinh viên  $j$
- ...

- `int a[3][4]` mảng gồm 3 mảng một chiều, mỗi mảng có độ dài 4
  - `a[0] ... a[2]` là các phần tử - các mảng một chiều [4]
- `int b[10][3][4]`
  - mảng gồm 10 mảng hai chiều loại [3][4]
    - `b[0].. b[9]` là các phần tử - các mảng hai chiều [3][4]
  - mảng hai chiều [10][3] gồm các mảng loại [4]
  - mảng ba chiều [10][3][4] gồm các ô nhớ kiểu int
- Không giới hạn số chiều

# Lưu ý về kích thước mảng

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

- Lập trình viên tự nhớ, tự quản lý giá trị 10
  - thêm một lý do nên dùng hằng thay cho 10.
  - một lần nữa, C++ không lưu trữ hoặc kiểm tra
  - không có `a.size()` hay `a.length`
- Làm thế nào để lấy (tính) kích thước mảng?

```
int a[...]      :      sizeof(a) / sizeof(int)
```

```
double b[...]   :      sizeof(b) / sizeof(double)
```

```
long c[..][...] :      ???
```

# Cách tránh truy nhập ngoài mảng?

1. Tự kiểm tra cẩn thận, cẩn thận, và cẩn thận  
... nếu dùng mảng
  2. Tránh dùng mảng:
    - thay C-string bằng C++ string
    - thay mảng bằng `std::vector` (chưa học)
- Mảng và C-string chạy nhanh hơn, nên có những trường hợp đó vẫn là lựa chọn tốt nhất
  - Môn học của ta: phải học và phải dùng

# Tài liệu tham khảo

- <http://www.cplusplus.com/doc/tutorial/arrays/>
- <http://www.cplusplus.com/doc/tutorial/ntcs/>

Một số thuật toán dùng mảng



# Sắp xếp nổi bọt – bubble sort

6 5 3 1 8 7 2 4

# Sắp xếp nổi bọt – bubble sort

- Nổi bọt đoạn 0..7
- Nổi bọt đoạn 0..6
- Nổi bọt đoạn 0..5
- Nổi bọt đoạn 0..4
- Nổi bọt đoạn 0..3
- ...

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 6 | 5 | 3 | 1 | 8 | 7 | 2 | 4 |

# Sắp xếp nổi bọt – bubble sort

Với k chạy từ 7 xuống 1:  
nổi bọt đoạn (0..k)

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 6 | 5 | 3 | 1 | 8 | 7 | 2 | 4 |



```
for (int k = n-1; k > 0; k--) {  
    nổi bọt đoạn (0..k)  
}
```

# Sắp xếp nổi bọt – bubble sort

nổi bọt đoạn (0..k)?

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 6 | 5 | 3 | 1 | 8 | 7 | 2 | 4 |

1. Nếu  $a[0] > a[1]$  thì đổi chỗ  $a[0]$  và  $a[1]$
2. Nếu  $a[1] > a[2]$  thì đổi chỗ  $a[1]$  và  $a[2]$
3. Nếu  $a[2] > a[3]$  thì đổi chỗ  $a[2]$  và  $a[3]$
- ...
- k. Nếu  $a[k-1] > a[k]$  thì đổi chỗ  $a[k-1]$  và  $a[k]$

# Sắp xếp nổi bọt – bubble sort

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1. Nếu $a[0] > a[1]$ thì đổi chỗ $a[0]$ và $a[1]$     | 6 | 5 | 3 | 1 | 8 | 7 | 2 | 4 |
| 2. Nếu $a[1] > a[2]$ thì đổi chỗ $a[1]$ và $a[2]$     |   |   |   |   |   |   |   |   |
| 3. Nếu $a[2] > a[3]$ thì đổi chỗ $a[2]$ và $a[3]$     |   |   |   |   |   |   |   |   |
| ...   |   |   |   |   |   |   |   |   |
| k. Nếu $a[k-1] > a[k]$ thì đổi chỗ $a[k-1]$ và $a[k]$ |   |   |   |   |   |   |   |   |



Với  $i$  chạy từ 1 đến  $k$ :

nếu  $a[i-1] > a[i]$  thì đổi chỗ  $a[i-1]$  và  $a[i]$

# Sắp xếp nổi bọt – bubble sort

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 6 | 5 | 3 | 1 | 8 | 7 | 2 | 4 |

Với  $i$  chạy từ 1 đến  $k$ :

nếu  $a[i-1] > a[i]$  thì đổi chỗ  $a[i-1]$  và  $a[i]$



```
for (int i = 1; i <= k; i++)  
    if (a[i-1] > a[i]) {  
        temp = a[i]; a[i] = a[i-1]; a[i-1] = temp;  
    }
```

# Sắp xếp nổi bọt – bubble sort

Gộp lại:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 6 | 5 | 3 | 1 | 8 | 7 | 2 | 4 |

```
for (int k = n - 1; k > 0; k--)  
    for (int i = 1; i <= k; i++)  
        if (a[i-1] < a[i]) {  
            temp = a[i];  
            a[i] = a[i-1];  
            a[i-1] = temp;  
        }
```

# Cải tiến?

6 5 3 1 8 7 2 4



# Cải tiến: dừng khi đã xếp xong

```
for (int k = n - 1; k > 0; k--) {  
    bool swapped = false;  
    for (int i = 1; i <= k; i++) {  
        if (a[i-1] < a[i]) {  
            temp = a[i];  
            a[i] = a[i-1];  
            a[i-1] = temp;  
            swapped = true;  
        }  
    }  
    if (! swapped) break;  
}
```

Ý tưởng cải tiến: Dừng lại khi lần nổi bọt hiện tại không cần đổi chỗ lần nào.

Cài đặt: Dùng cờ swapped (đã bị đổi chỗ) để đánh dấu xem lần nổi bọt hiện tại đã xảy ra đổi chỗ hay chưa.

# Dùng điều kiện lặp thay cho break

```
bool swapped = true;
for (int k = n - 1; k > 0 && swapped; k--) {
    swapped = false;
    for (int i = 1; i <= k; i++) {
        if (a[i-1] < a[i]) {
            temp = a[i];
            a[i] = a[i-1];
            a[i-1] = temp;
            swapped = true;
        }
    }
}
```

Duyệt tổ hợp (nhỏ)

# Duyệt tổ hợp nhỏ

- Cho N bạn nam và M bạn nữ, hãy liệt kê tất cả các cách ghép được một đôi.
- Cho bảng chữ cái a..z.
  - In ra tất cả các từ độ dài độ dài 5
  - In ra tất cả các từ độ dài dưới 5

# Duyệt tổ hợp nhỏ

- Cho  $N$  bạn nam  $0 \dots N-1$   
và  $M$  bạn nữ  $0 \dots M-1$   
, hãy liệt kê tất cả các cách ghép được một đôi.
- $0\ 0, 0\ 1, 0\ 2 \dots 0\ M-1$   
 $1\ 0, 1\ 1, 1\ 2 \dots 1\ M-1$   
....  
 $i\ 0, i\ 1, i\ 2, \dots i\ M-1$   
...  
 $N-1\ 0, N-1\ 1, \dots N-1\ M-1$

# Duyệt hoán vị nhỏ

- 0 0, 0 1, 0 2 .... 0 M-1  
1 0, 1 1, 1 2 ..... 1 M -1  
....  
i 0, i 1, i 2,        i M-1  
...  
N-1 0, N-1 1, ..... N-1 M-1
- ```
for (int l = 0; l < N; l++) {  
    for (int j = 0; j < M; j++)  
        cout << boy[l] << " " << girl[j] << ",";  
    cout << endl;  
}
```

# Duyệt hoán vị nhỏ

- Cho bảng chữ cái a..z.
  - In ra tất cả các từ độ dài 3
    - 3 vòng for thô
  - Mà không có chữ cái nào xuất hiện 2 lần trở lên
  - In ra tất cả các từ độ dài không quá 3

# Duyệt tổ hợp

- Cho  $N$  số, hỏi có hai số nào bằng nhau hay không?
  - Cách 1: duyệt tất cả các hoán vị của hai phần tử

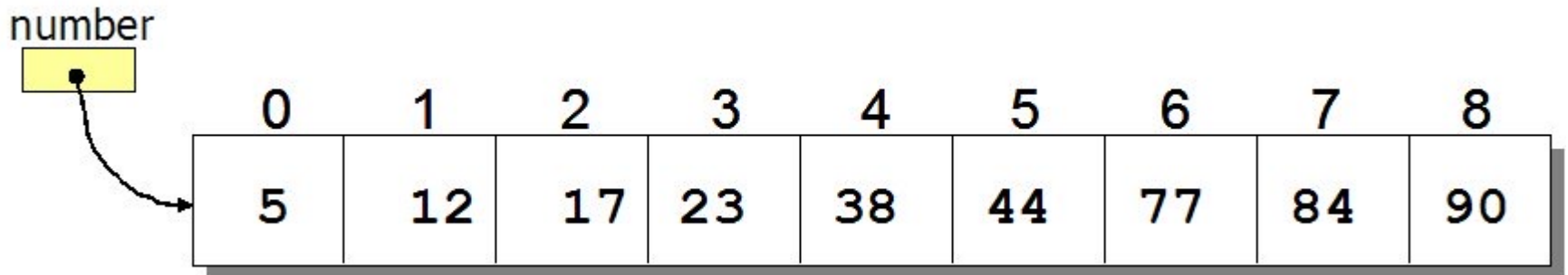


# Tìm kiếm nhị phân

Binary search

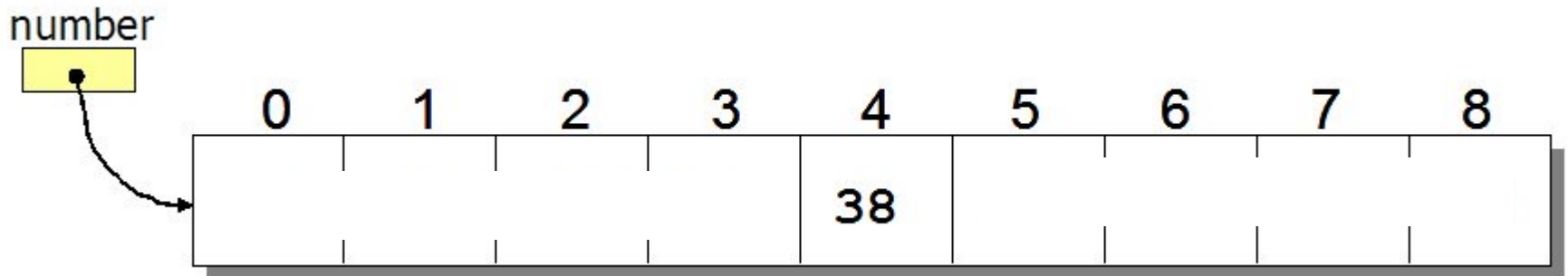
# Tìm kiếm nhị phân

- Cho một chuỗi key đã được sắp xếp, tìm một key cho trước
- Ví dụ: tìm số 44 trong dãy



# Ý tưởng thuật toán

- Nếu chỉ biết giá trị của **một** phần tử mảng, có thể suy luận gì về vị trí của 44?



# Bước 1



|    | low | high | mid |
|----|-----|------|-----|
| #1 | 0   | 8    | 4   |

```
search( 44 )
```

$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$$

| 0 | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  |
|---|----|----|----|----|----|----|----|----|
| 5 | 12 | 17 | 23 | 38 | 44 | 77 | 84 | 90 |

A diagram showing a rectangular box with the word "low" inside. An upward-pointing arrow is positioned above the box, indicating a direction of movement or a level that is low relative to something else.



high

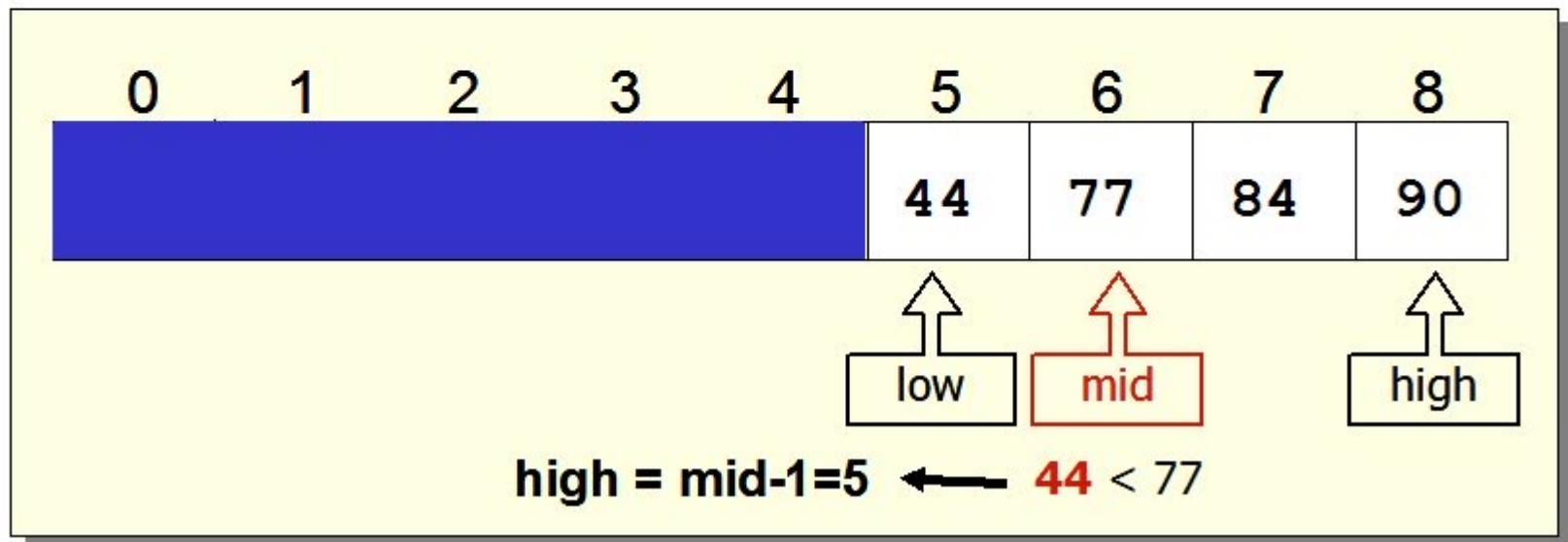
$38 < 44 \longrightarrow \text{low} = \text{mid} + 1 = 5$

## Bước 2

|    | low | high | mid |
|----|-----|------|-----|
| #1 | 0   | 8    | 4   |
| #2 | 5   | 8    | 6   |

search( 44 )

$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$$

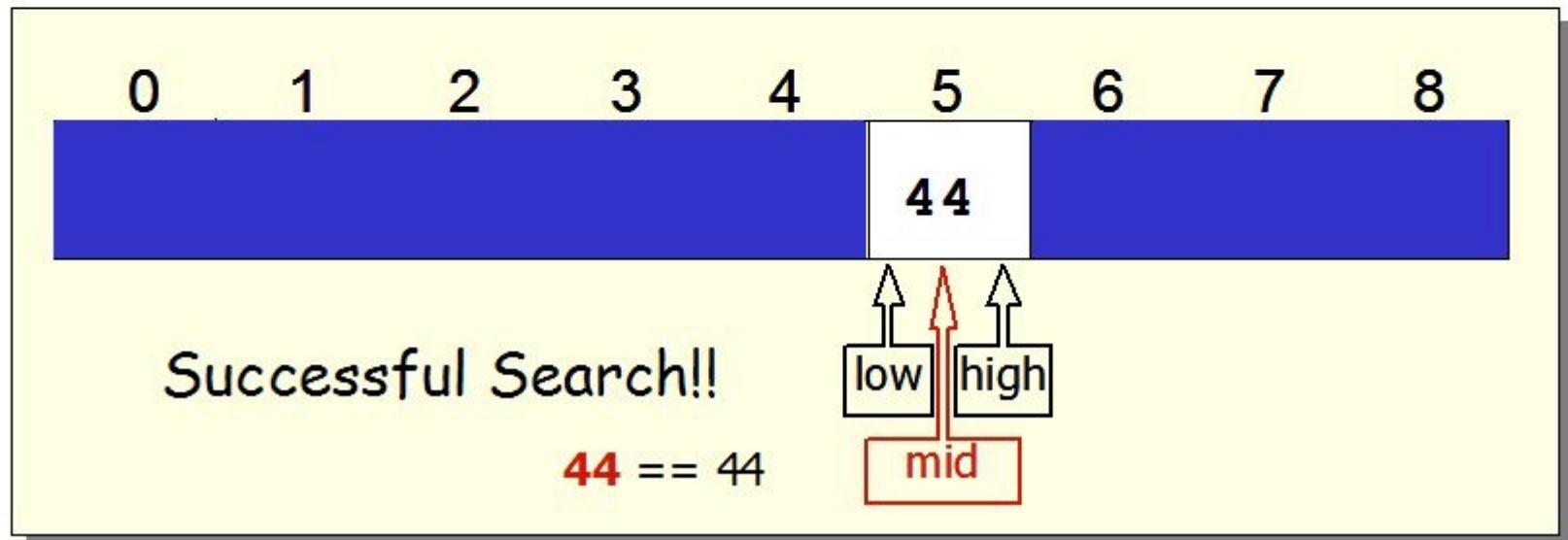


# Bước 3

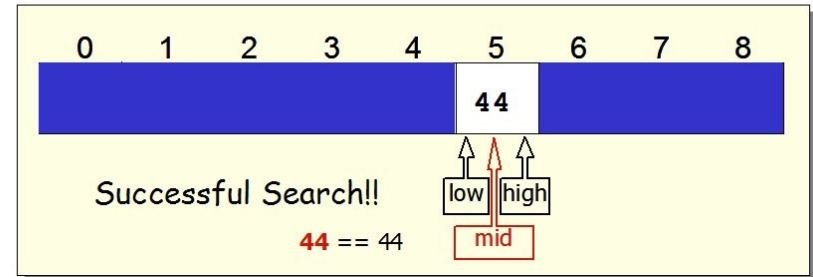
|    | low | high | mid |
|----|-----|------|-----|
| #1 | 0   | 8    | 4   |
| #2 | 5   | 8    | 6   |
| #3 | 5   | 5    | 5   |

search( 44 )

$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$$



|    | low | high | mid | search( 44 ) |
|----|-----|------|-----|--------------|
| #1 | 0   | 8    | 4   |              |
| #2 | 5   | 8    | 6   |              |
| #3 | 5   | 5    | 5   |              |

$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$$


Mảng A

Low: chỉ số đầu mảng

High: chỉ số cuối mảng

Lặp cho đến khi  $low > high$ :

$mid = TBC(low, high)$

nếu  $(a[mid] < key)$ :  $low = mid + 1$

nếu  $(a[mid] > key)$ :  $high = mid - 1$

nếu  $(a[mid] == key)$ : xong, tìm thấy, dừng  
lặp lại