

# **Nội dung 7. Mảng (Array)**

**Ts. Trần Thanh Hải**

## Nội dung

---

- Giới thiệu.
- Mảng một chiều
- Mảng nhiều chiều
- Xâu ký tự.

# Mở đầu

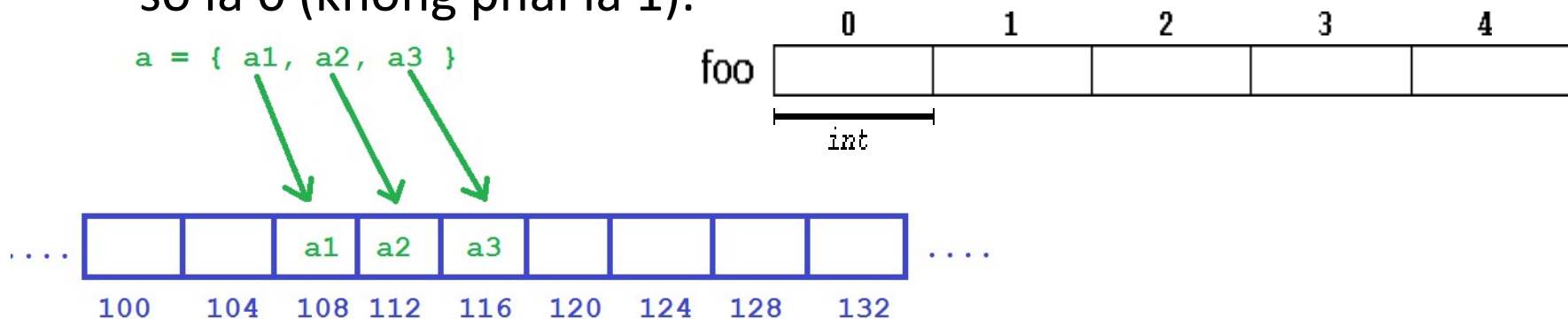
---

- Chương trình nhập 5 số nguyên:

```
int main() { noidung7 modau  
    int a, b, c, d, e;  
    cout<<"Nhập số nguyên a = ";  
    cin>>a;  
    cout<<"Nhập số nguyên b = ";  
    cin>>b;  
    cout<<"Nhập số nguyên c = ";  
    cin>>c;  
    cout<<"Nhập số nguyên d = ";  
    cin>>d;  
    cout<<"Nhập số nguyên e = ";  
    cin>>e;  
}
```

# 1. Mảng 1 chiều – định nghĩa

- **Định nghĩa** mảng một chiều (**array**): là một dãy các phần tử có **cùng kiểu dữ liệu** được đặt liên tiếp nhau trong một vùng nhớ, tham chiếu đến từng phần tử bằng **chỉ số** và **tên mảng**.
- Ví dụ: một mảng chứa 5 giá trị kiểu nguyên được đặt là **foo** được biểu diễn như sau:
- Mỗi khoảng trống biểu diễn một phần tử của mảng. Các phần tử được đánh số từ 0 đến 4. Trong đó 0 là chỉ phần tử đầu tiên và 4 là chỉ phần tử cuối cùng.
- Trong C++, phần tử đầu tiên trong một mảng luôn được đánh số là 0 (không phải là 1).



# 1. Mảng 1 chiều – cú pháp

---

- **Cú pháp 1:** type name [elements];

trong đó: **type** là kiểu hợp lệ (int, float, double, char..),

**name** tên của mảng,

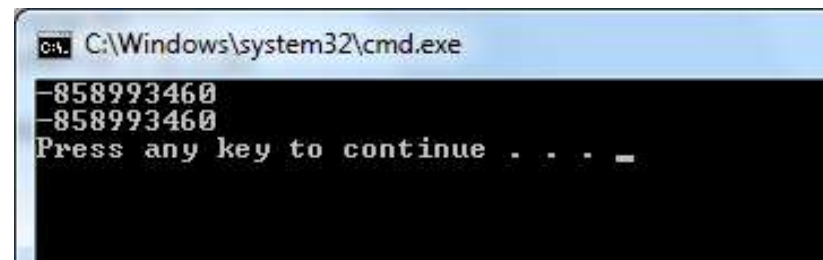
**elements** là hình thành chiều dài của mảng dưới dạng số phần tử. Ví dụ: **int foo[5];**

- Mảng foo(name) với 5 (elements) phần tử kiểu(type) int được khai báo: **int** foo [5];
- **Chú ý:** trường phần tử trong ngoặc [], biểu diễn số phần tử trong mảng cần là một biểu thức hằng (const). Do đó, mảng là khối bộ nhớ tĩnh với kích thước phải được xác định trước khi chạy.
- Tổng số byte = số byte của kiểu × số phần tử (4 byte×5=20 byte)
- Từ đó, nếu biết được tổng số byte của mảng và kiểu mảng thì ta tính được số phần tử của mảng:  $(\text{sizeof}(\text{foo})/\text{sizeof}(\text{int}) = n = 5)$ .

## 2. Khởi tạo giá trị ban đầu cho mảng

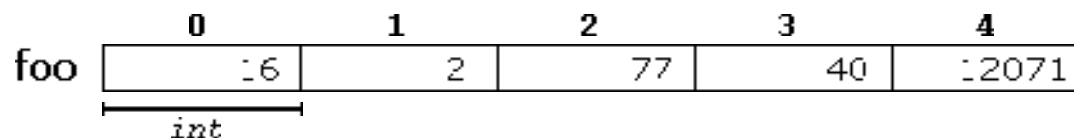
- Mặc định: mảng có phạm vi cục bộ là bỏ qua khởi tạo giá trị ban đầu (không có phần tử nào của mảng được đặt một giá trị cụ thể nào, nội dung của mảng không được xác định tại thời điểm mảng được khai báo).

```
int a[2];  
for (int i = 0; i < 2; i++){  
    cout << a[i];  
}
```



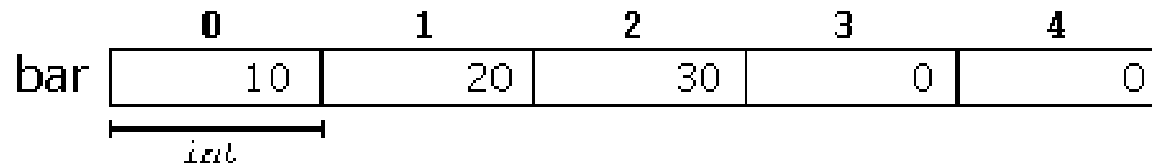
```
C:\Windows\system32\cmd.exe  
-858993460  
-858993460  
Press any key to continue . . .
```

- Phần tử trong mảng được khởi tạo một giá trị xác định bằng:
- Cú pháp 2: `type name [elements] = { value1, value2, ...valueN };`
- Ví dụ: `int foo [5] = { 16, 2, 77, 40, 12071 };`

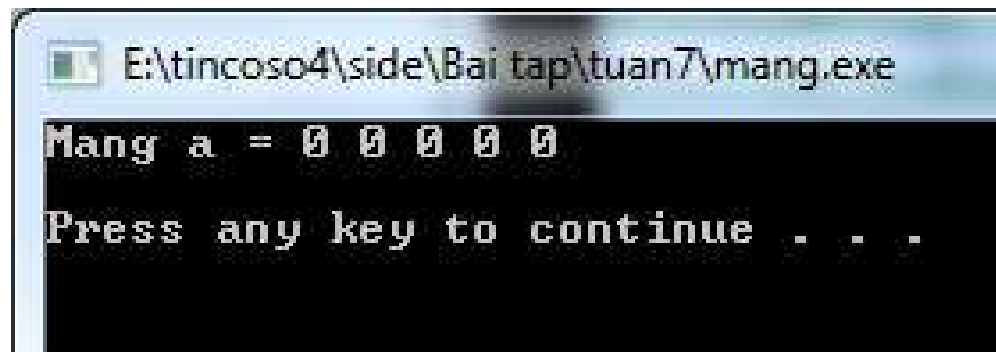


## 2. Khởi tạo giá trị ban đầu cho mảng

- Chú ý: Số lượng các giá trị trong {} không được lớn hơn số phần tử khai báo trong mảng. (nếu lớn hơn thì báo lỗi).
- Nếu khai báo ít hơn số phần tử trong ngoặc, các phần tử còn lại đặt giá trị mặc định (kiểu cơ bản chúng được điền 0)
- Ví dụ: `int bar [5] = { 10, 20, 30 }; // tùy version`



- Do đó, `int bar[5] = {};` //

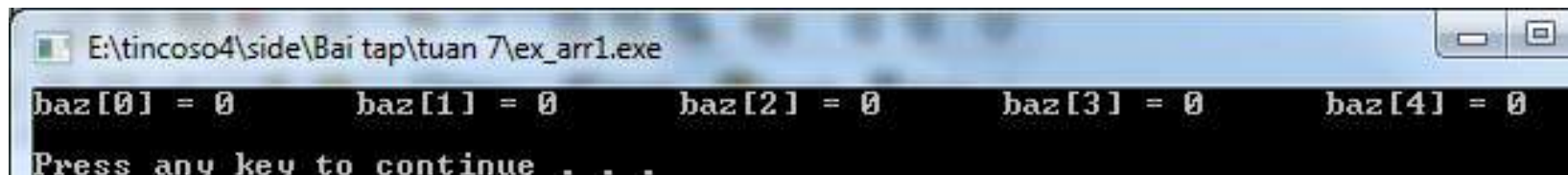


```
E:\tincoso4\side\Bai tap\tuan7\mang.exe
Mang a = 0 0 0 0 0
Press any key to continue . . .
```

## 2. Khởi tạo giá trị ban đầu cho mảng

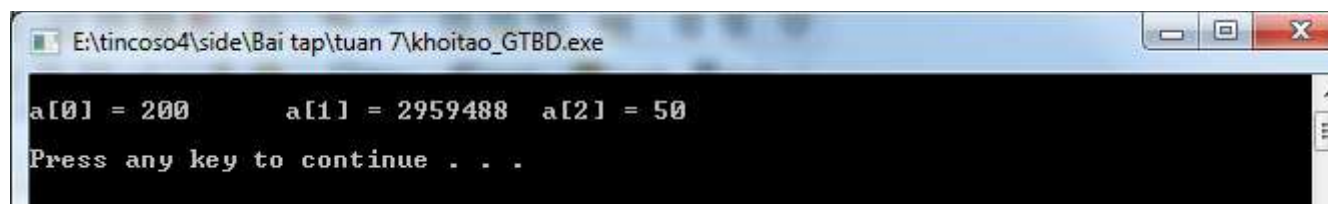
- `int baz[5] = { };` // Nếu số 5 được thiết lập thì ta có

	0	1	2	3	4
baz	0	0	0	0	0
	<u>int</u>				



```
E:\tincoso4\side\Bai tap\tuan 7\ex_arr1.exe
baz[0] = 0      baz[1] = 0      baz[2] = 0      baz[3] = 0      baz[4] = 0
Press any key to continue . . .
```

- `int baz[] = { };` // Giá trị sẽ là bất kỳ không xác định



```
E:\tincoso4\side\Bai tap\tuan 7\khoitao_GTBD.exe
a[0] = 200      a[1] = 2959488      a[2] = 50
Press any key to continue . . .
```



## 2. Khởi tạo giá trị ban đầu cho mảng

---

- Khi một khởi tạo được cung cấp cho mảng. C++ cho phép để khoảng trống trong ngoặc []. Trường hợp này, chương trình sẽ giả định tự động kích thước cho mảng phù hợp với số giá trị nằm bên trong {}. (vì giả định không biết trước – không gán =0)
- Ví dụ: `int foo [] = { 16, 2, 77, 40, 12071 };` mảng foo có 5 int
- Sự tiến hóa của C++ hai cách viết sau là tương đương
- `int foo[] = { 10, 20, 30 };`    `int foo[] { 10, 20, 30 };` - **thấp gây lỗi**
- Mảng tĩnh, và được khai báo trực tiếp bên trong namespace (bên ngoài hàm) luôn luôn được khởi tạo. Nếu không có giá trị cụ thể, toàn bộ các phần tử được mặc định giá trị = 0 (đối với các kiểu cơ bản).
- `int a[10], b[10];`  
    **Và gán: `a = b;` // sai và không hợp lệ**

# Ví dụ

---

- Tính chiều dài của mảng sử dụng biểu thức

`sizeof(arrayName)/sizeof(arrayName[0]);`

`sizeof(arrayName)`: trả về tổng số byte của mảng

`sizeof(arrayName[0])`: trả về số byte của phần tử đầu tiên.

- **C/C++ không thực hiện kiểm tra chỉ số biên của mảng**
- [Xem ví dụ:](#)
- **(khi gán `a={}`, thì tất cả không bằng 0 cũng tùy version)**

### 3. Chuỗi ký tự - xâu ký tự (Character sequences)

---

- Lớp string đã được giới thiệu đây là một lớp mạnh để xử lý và thao tác chuỗi các ký tự.

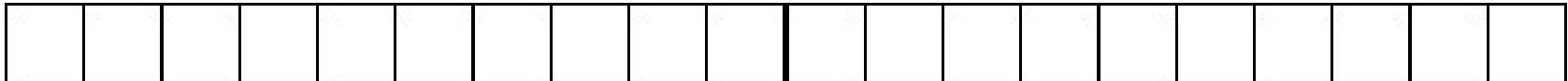
```
string mystring = "This is a string";
```

```
string mystring ("This is a string");
```

```
string mystring {"This is a string"};
```

- Tuy nhiên, chuỗi các ký tự có thể biểu diễn dưới dạng mảng các phần tử:
- `char foo[20];`

foo

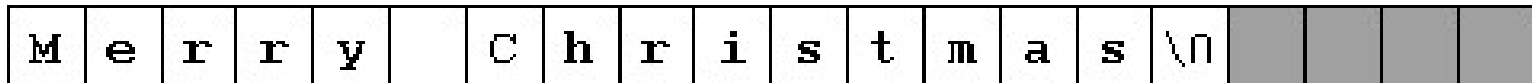


- Do đó, mảng có khả năng lưu trữ chuỗi lên đến 20 ký tự. Nhưng dung lượng **không cần phải sử dụng hết**, cũng có thể chứa chuỗi ngắn hơn.

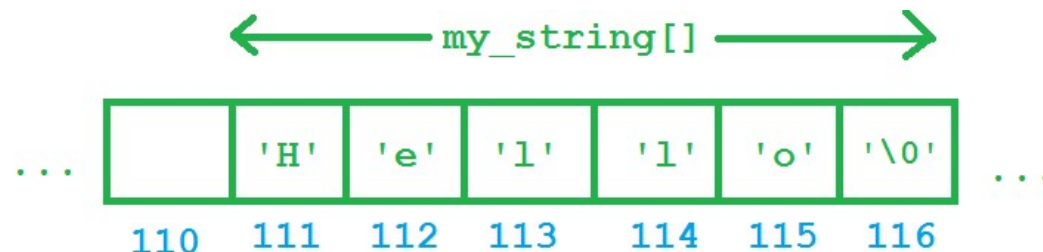
### 3. Chuỗi ký tự - xâu ký tự

- Theo quy ước, kết thúc của chuỗi được biểu diễn bằng một ký tự đặc biệt : null (`\0`)

foo



- Chú ý:** sau nội dung của chuỗi, một ký tự `\0` được cộng thêm vào mục đích là chỉ sự kết thúc của chuỗi, phần đậm là phần tử không được xác định giá trị.
- Khai báo một mảng phần tử với kiểu kí tự (1 trong hai cách)  
`char my_string[] = { 'H', 'e', 'l', 'l', 'o', '\0' }; // 6 ký tự`  
`char my_string[] = "Hello";` sử dụng dấu nháy kép null (`\0`) sẽ tự động thêm vào



### 3. Khởi tạo null kết thúc chuỗi ký tự

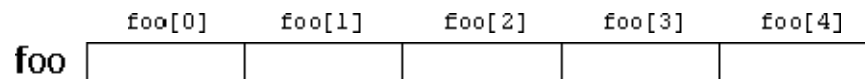
---

- Xâu ký tự (string) là: chuỗi các ký tự, có thể biểu diễn như là một mảng phần tử của kiểu ký tự (char).
- Mảng ký tự là một mảng thông thường, giống như các qui luật trước đó.(khai báo, khởi tạo...)
- `char myword[] = { 'H', 'e', 'l', 'l', 'o', '\0' };`
- *Đây là khai báo mảng 6 phần tử kiểu char, khởi đầu với các phần tử từ dạng "Hello" và cộng thêm phần tử null (\0) ở cuối.*
- `char myword[] = "Hello";` //sử dụng dấu nháy kép null sẽ tự động thêm vào
- Mảng không được gán giá trị. Ví dụ sai:  
`myword = "Bye"; myword[] = 'Bye'; myword = { 'B', 'y', 'e', '\0' };`
- Nhưng mỗi phần tử mảng có thể gán giá trị riêng lẻ  
`myword[0] = 'B'; myword[1] = 'y'; myword[2] = 'e'; myword[3] = '\0';`

### 3. Truy cập các giá trị của mảng

---

- Truy cập giá trị của một phần tử bất kỳ trong mảng sử dụng
- **Cú pháp:** `name[index]` , `int foo[5];`
- `foo` có 5 phần tử và mỗi phần tử có kiểu `int`, tên dùng để tham chiếu đến mỗi phần tử như sau:



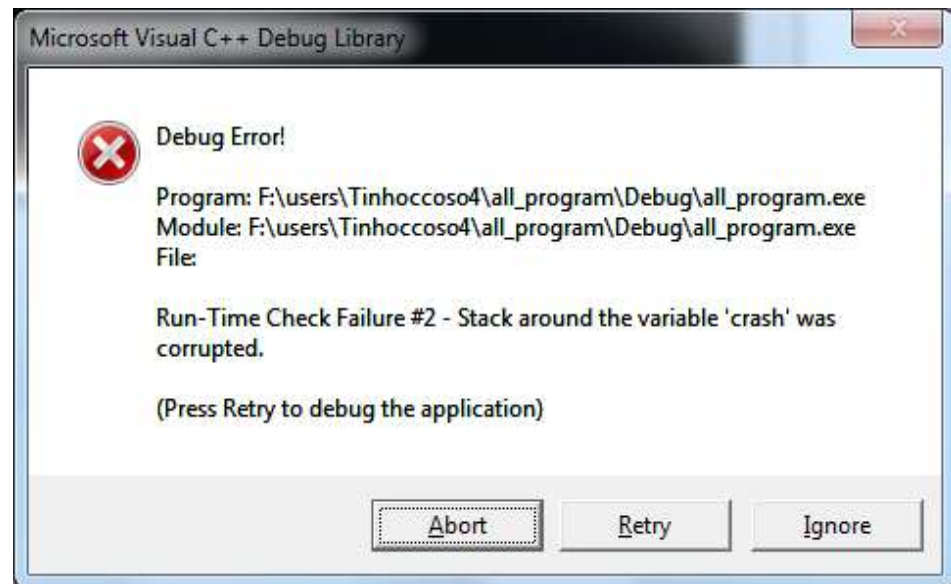
- Ví dụ: `foo [2] = 75`; lưu trữ giá trị 75 vào phần tử thứ 3 của `foo`  
`x = foo[2]`; copy **giá trị** của phần tử thứ 3 của `foo` cho biến `x`
- **Phân biệt hai cách sử dụng `[]` liên quan tới mảng:**
- (1) hình thành: kích thước của mảng khi chúng được khai báo,
- (2) hình thành: chỉ số mảng của phần tử cần truy cập.

### 3. Truy cập các giá trị của mảng

- Một số vấn đề, khi truy cập các phần tử ngoài mảng sẽ không gây ra lỗi trong than phiền cảnh báo, nhưng gây ra lỗi trong quá trình chạy.

int main() // chương trình không dung. Vuot chi so mang

```
{  
    int crash[10], i;  
    for(i = 0; i < 100; i++){  
        crash[i] = i;  
    }  
    return 1;  
}
```



## Ví dụ đếm số ký tự trong mảng ký tự

```
int main()
{
// tu them khi tu NULL
char ch[] = {'H','E','L','\0'};
    int i;
    i = 0; // biến đếm
    while(ch[i] != 0){ // '\0'
        cout<<ch[i]<<endl;
        i = i+1;
    }
} noidung7_mang2_char
```

```
int main()
{
// tu them khi tu NULL
char ch[] = "HEL" ;
    int i;
    i = 0; // biến đếm
    while(ch[i] != 0){ \\ '\0' tuong duong
        cout<<ch[i]<<endl;
        i = i+1;
    }
}
```



## Ví dụ

---

```
#include <iostream> #include <string>
using namespace std;
int main () {
    char ques1[] = "What is your name? ";
    string ques2 = "Where do you live? ";
    char ans1 [80];
    string ans2;
    cout << ques1;
    cin >> ans1;
    cout << ques2;
    cin >> ans2;
    cout << "Hello, " << ans1;
    cout << " from " << ans2 << "!\n";
    return 0;
}
```

- Chú ý: sự khác biệt trong khai báo
- Mảng: có kích thước cố định dạng ẩn hoặc hiển khi khai báo
- *ques1*: có 20 ký tự (cả \0).
- *ans1* có kích thước 80 ký tự.
- Trái lại *string* là các xâu đơn giản, không hình thành kích thước tức là không có [].
- Do đó, *string* có kích thước động được xác định trong quá trình chạy,
- Trong khi kích thước của mảng được xác định trong biên dịch, trước khi chương trình chạy.

## Ví dụ cách sử dụng xâu khai báo string

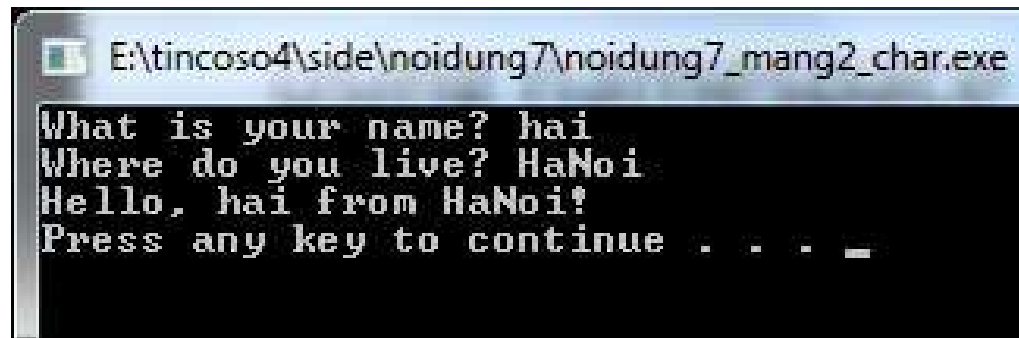
---

```
int i = 0;
int count = 0;
string ques2 = "Where do you live? ";
while(ques2[i] != 0 ) // không phải là '\n'
{
    count = count + 1;
    i = i + 1;
}
cout<<"Chieu dai cua ques2 = "<<count <<endl;
OUTPUT: cout = 19.
```

## Ví dụ: sự hoán đổi char – string

---

```
using namespace std;
int main()
{
    char question1[] = "What is your name? ";
    string question2 = "Where do you live? ";
    char answer1 [80];
    string answer2;
    cout << question1;
    cin >> answer1;
    cout << question2;
    cin >> answer2;
    cout << "Hello, " << answer1;
    cout << " from " << answer2 << "!\n";
} noidung7\_mang2\_char
```



```
E:\tincoso4\side\noidung7\noidung7_mang2_char.exe
What is your name? hai
Where do you live? HaNoi
Hello, hai from HaNoi!
Press any key to continue . . . _
```

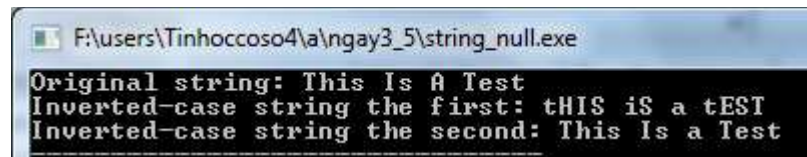
### 3. Ví dụ - Chương trình tính tổng của mảng

```
#include<iostream> #include<stdlib.h> #include<iomanip>
// setw(10) – giống \t (cố định)
using namespace std;
int main() {
    int foo [] = {16, 2, 77, 40, 12071}; // 1- Khởi tạo, khai báo
    int n, j, sum = 0;
    n = sizeof(foo)/sizeof(int); // Tính số phần tử mảng
    cout<<"Elemnts"<<"\t"<<"Value"<<endl;
    for(int i = 0; i<n; i++) {
        cout<<i<<setw(15)<<foo[i]<<endl; // truy cập phần tử thứ i
    }
    for (j = 0 ; j<n ; ++j )
        sum += foo[j];
    cout<<"Tong cua mang 1 chieu sum ="<<setw( 13 ) <<sum<<endl;
} noidung7 mang1 sum
```

# Sử dụng hàm xử lý ký tự

- Chương trình chuyển chữ thường thành chữ hoa và ngược lại.

```
for(i = 0; p[i]; i++) {  
    if(isupper(p[i]))  
        p[i] = tolower(p[i]);  
    else if(islower(p[i]))  
        p[i] = toupper(p[i]);  
}
```



```
F:\users\Tinhocoso4\ngay3_5\string_null.exe  
Original string: This Is A Test  
Inverted-case string the first: tHIS iS a tEST  
Inverted-case string the second: This Is a Test
```

```
cout << "Inverted-case string the first: " << str<<"\n";
```

```
for(i = 0; p[i]; i++) {  
    if ((p[i]>=65) && (p[i]<=97))  
        p[i] = tolower(p[i]);  
    else if((p[i]>=97) && (p[i]<=122))  
        p[i] = toupper(p[i]);  
}
```

## #include<cctype>

---

- `int isalpha(int ch);` trả về 1 nếu ch là ký tự chữ, 0 nếu không phải
- `int isdigit(int ch);` trả về 1 nếu ch là [0 – 9], trái lại bằng 0
- `int isalnum(int ch);` trả về 1 nếu chữ hoặc số, trái lại bằng 0
- `int isupper(int ch);` trả về 1 nếu chữ hoa, trái lại bằng 0
- `int islower(int ch);` trả về 1 nếu chữ thường, trái lại bằng 0
- `int toupper(int ch);` trả về chữ hoa
- `int tolower(int ch);` trả về chữ thường;
- `int isspace(int ch);` trả về 1 nếu khoảng trắng (' ', \r, \n, \t, \f, \v)
- `int ispunct(int ch);` punctuation character?
- `int iscntrl(int ch);` control character?
- `int isprint(int ch);` printable character?
- `int isgraph(int ch);` graphical representation?

## Tra từ điển – so sánh chuỗi ký tự

```
const char *dictionary[][2] = {  
    "A", "AA",  
    "B", "BB",  
    "C", "CC",  
    "D", "DD",  
    "E", "EE",  
    "", ""  
};  
  
char word[80]; int i;  
cout << "Enter word: ";  
cin >> word;  
for(i = 0; *dictionary[i][0]; i++) {  
    if(!strcmp(dictionary[i][0], word)) {  
        cout << dictionary[i][1] << "\n";  
        break;  
    }  
}  
  
if(!*dictionary[i][0])  
    cout << word << " not found.\n";
```



## Tóm lại

---

- Một mảng bao gồm không ít hơn một giá trị, có cùng kiểu dữ liệu chung, dưới cùng một tên. Mỗi giá trị có một vị trí duy nhất và được so sánh bằng cách sử dụng kỹ thuật chỉ số hóa.
- Khởi tạo mảng hay còn gọi là sự thiết lập trạng thái ban đầu của một mảng bằng các giá trị (hoặc các ký tự) được định nghĩa giữa cặp dấu ngoặc nhọn {}.
- Khai báo: có 2 cách
- Cách 1: Khai báo nhưng không khởi tạo các phần tử
- Cách 2: Khai báo và khởi tạo giá trị cho mỗi phần tử:
- `int foo[] = { 10, 20, 30 };`
- `int foo[] { 10, 20, 30 }; // C++2011 trở lên`
- `char mystring[] = {'H','\0'};`
- `float arr[10]; arr[0], arr[1], .... arr[9];`



## Ví dụ

```
#include<iostream> #include<stdio.h>
using namespace std;
int main()
{
    char ch[3] = {'a', 'b', 'c'};
    string str[]={"pen", "red","notebook"};
    int in[] = {1, 2, 3};
    double d[] = {1.2, 1.4, 1.987};
    bool b[] = {true, false, true};
    cout<<"Mot so the hien [vi tri ban dau, cuoi, boolean] \n";
    cout<<"Vi tri 1: " <<str[0]<<endl;
    cout<<"Vi tri cuoi 2: " <<str[2]<<endl;
    cout<<"Ket qua cua boolean: " <<b[0]<<endl;
    system("PAUSE");
    return 0;}

```



```
E:\tincoso4\array\array_example\array_type_data.exe
Mot so the hien [vi tri ban dau, cuoi, boolean]
Vi tri 1: pen
Vi tri cuoi 2: notebook
Ket qua cua boolean: 1
Press any key to continue . . . _

```

## 5. Mảng nhiều chiều - Multidimensional arrays

- Mảng nhiều chiều có thể được miêu tả như là “mảng của mảng”.
- Ví dụ, mảng 2 chiều có thể tưởng tượng như bảng 2 chiều chứa phần tử, **toàn bộ phần tử có cùng một kiểu dữ liệu**.
- Biểu diễn một mảng hai chiều 3 hàng và 5 cột.

a

	0	1	2	3	4
0					
1					
2					

- Khai báo: `int a [3][5];`
- Truy cập: `a[1][3]`

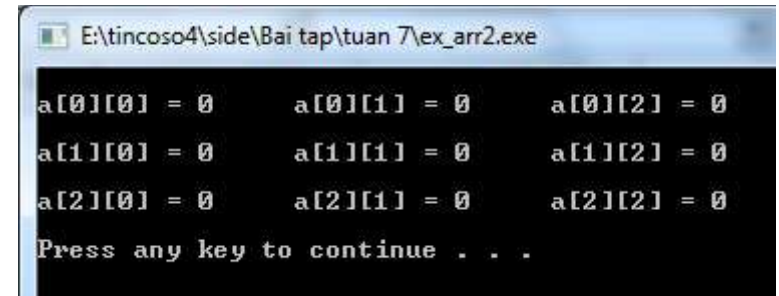
a

	0	1	2	3	4
0					
1					
2					

↓  
`a[1][3]`

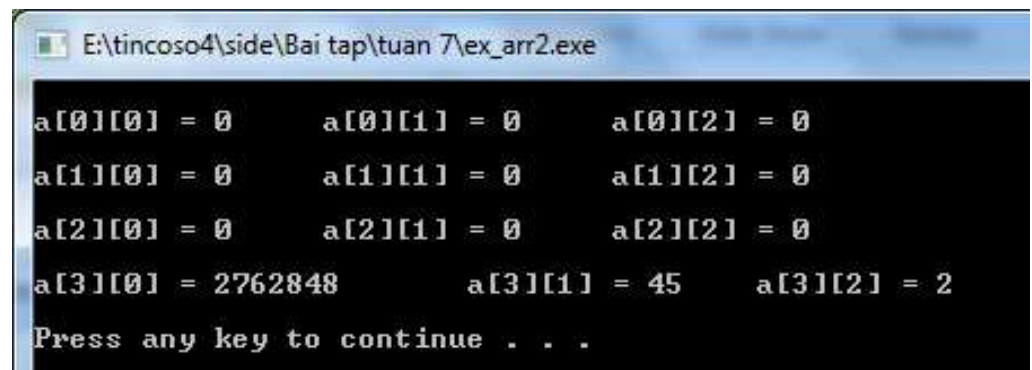
## Khởi tạo mảng ban đầu

- `int a[3][3] = {};`



```
E:\tincoso4\side\Bai tap\tuan 7\ex_arr2.exe
a[0][0] = 0      a[0][1] = 0      a[0][2] = 0
a[1][0] = 0      a[1][1] = 0      a[1][2] = 0
a[2][0] = 0      a[2][1] = 0      a[2][2] = 0
Press any key to continue . . .
```

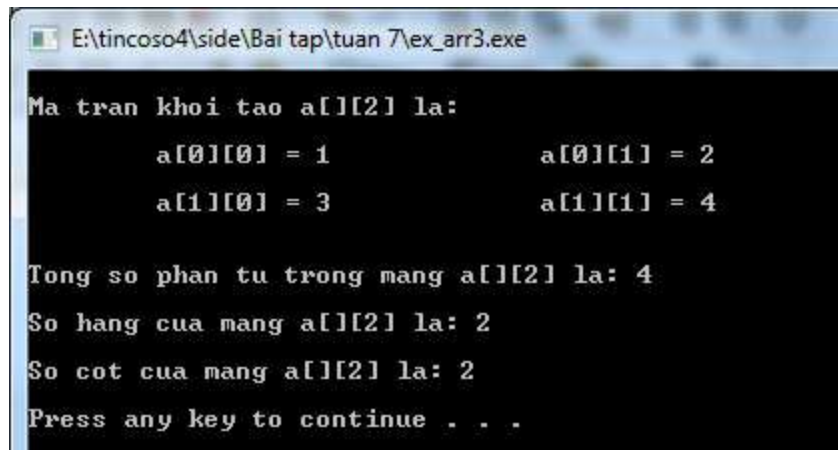
- Nếu vượt quá số kích thước qui định:



```
E:\tincoso4\side\Bai tap\tuan 7\ex_arr2.exe
a[0][0] = 0      a[0][1] = 0      a[0][2] = 0
a[1][0] = 0      a[1][1] = 0      a[1][2] = 0
a[2][0] = 0      a[2][1] = 0      a[2][2] = 0
a[3][0] = 2762848  a[3][1] = 45      a[3][2] = 2
Press any key to continue . . .
```

## Tính số hàng và số cột trong mảng hai chiều

- `int a[][2] = {{1,2},{3,4}}`
- Tổng số phần tử: `nele = sizeof(a)/sizeof(a[0][0]); // 16/4 = 4`
- Số hàng: `nrow = sizeof(a)/sizeof(*a);`  
`= sizeof(a)/sizeof(a[0]); // 4byte*nele/(2*4byte)`
- `nele = nrow × ncol ⇒ ncol = nele/nrow;`



```
E:\tincoso4\side\Bai tap\tuan 7\ex_arr3.exe
Ma tran khoi tao a[][2] la:
      a[0][0] = 1      a[0][1] = 2
      a[1][0] = 3      a[1][1] = 4

Tong so phan tu trong mang a[][2] la: 4
So hang cua mang a[][2] la: 2
So cot cua mang a[][2] la: 2
Press any key to continue . . .
```

## 4. Mảng nhiều chiều - Multidimensional arrays

---

- Mảng nhiều chiều **không giới hạn chỉ có hai chiều**, chúng có thể chứa nhiều chiều nếu cần.
- Tuy nhiên cần cẩn thận, khối lượng bộ nhớ cần thiết cho mảng sẽ tăng theo hàm mũ với mỗi chiều
- `char century [100][365][24][60][60];` // chiếm 3 gigabyte bộ nhớ.
- Mảng nhiều chiều cũng chỉ là sự trừu tượng đối với người lập trình.
- Kết quả đạt được hiệu quả với một mảng đơn giản, bằng cách nhân chỉ số của chúng
- `int a [3][5];` // tương đương với
- `int a [15];` // ( $3 * 5 = 15$ )

## Ví dụ

---

### Mảng nhiều chiều

```
#define WIDTH 5
#define HEIGHT 3
int a [HEIGHT][WIDTH];
int n,m;
int main (){
    for (n=0; n<HEIGHT; n++)
        for (m=0; m<WIDTH; m++) {
            a[n][m]=(n+1)*(m+1);
        }
}
```

### pseudo-multidimensional array (mảng ảo nhiều chiều)

```
#define WIDTH 5
#define HEIGHT 3
int a [HEIGHT * WIDTH];
int n,m;
int main (){
    for (n=0; n<HEIGHT; n++)
        for (m=0; m<WIDTH; m++){
            a[n*WIDTH+m]=(n+1)*(m+1);
        }
}
```

## 5. Khái niệm mảng nhiều chiều

---

- Mảng một chiều chứa tập hợp nhiều phần tử có cùng kiểu dữ liệu với nhau,
- Mảng nhiều chiều cũng chứa nhiều phần tử, mỗi phần tử lại là một mảng một chiều, các mảng một chiều này có kích thước bằng nhau và cũng được truy xuất thông qua một chỉ số - số thứ tự của mảng một chiều đó trong mảng nhiều chiều.

## 5. Khởi tạo mảng nhiều chiều

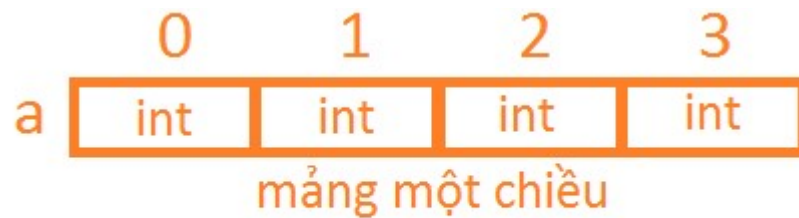
---

- `int a[2][3] = {2, 4, -5, 9, 0, 9};`
- `int a[2][3] = { {2, 4, 5}, {9, 0, 0}};`
- `int a[2][3] = { {2, 4, 5},  
                  {9, 0, 0}};`
- `int a[2][3][4] = {3, 4, 2, 3, 0, -3, 9, 11, 23, 12, 23,  
                  2, 13, 4, 56, 3, 5, 9, 3, 5, 5, 1, 4, 9};`
- `int a[2][3][4] = {  
                  { {3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2} },  
                  { {13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9} }  
                  };`



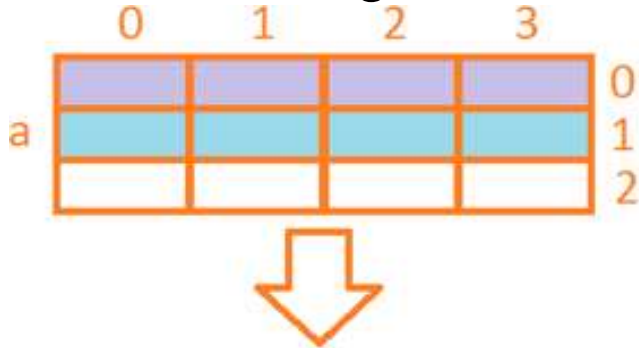
# Mảng nhiều chiều

- Mảng 2 chiều là mảng chứa n phần tử, mỗi phần tử là một mảng một chiều.
- Mảng 3 chiều sẽ chứa n phần tử, mỗi phần tử là một mảng 2 chiều chứa m mảng một chiều.



## Mảng nhiều chiều

- Khi lưu trữ, mảng nhiều chiều được “chuyển đổi” thành mảng một chiều và lưu trên một vùng nhớ liên tục tương tự như mảng một chiều.

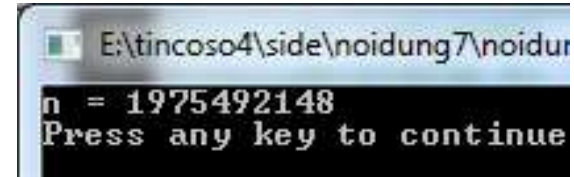


`int a[2][3][4];` (tổng số phần tử  $24 = 3 \times 2 \times 4$ )

- `a[0][0][0] = 3`, `a[0][0][1] = 4`, `a[0][0][2] = 2`, `a[0][0][3] = 3`,
- `a[0][1][0] = 0`, `a[0][1][1] = -3`, `a[0][1][2] = 9`, `a[0][1][3] = 11`,
- `a[0][2][0] = 23`, `a[0][2][1] = 12`, `a[0][2][2] = 23`, `a[0][2][3] = 2`,
- `a[1][1][0] = 6`, `a[1][1][1] = 7`, `a[1][1][2] = 8`, `a[1][1][3] = 9`,
- `a[1][2][0] = 12`, .....

# Kiểm tra điều kiện biên

```
int n;  
cout<< "n= "<<n<<endl;  
cout<<"Nhap so phan tu cua mang n =";  
cin>>n; float a[n];  
for(int i=0;i<n;i++) {  
    cout<<"a["<<i<<"]=" ";  
    cin>>a[i];  
}  
cout<<"Ket qua nhap ma tran "<<"\n";  
for(int i=0;i<n;i++) {  
    cout<<"a["<<i<<"]="<<a[i]<<endl;  
}
```



- Chương trình trên DEV++ , codeproject, vẫn chạy

# Kiểm tra điều kiện biên

---

- Tuy nhiên, Visual c++2010 không hoạt động báo lỗi
- *expected constant expression*
- *cannot allocate an array of constant size 0*
- *'a' : unknown size*
- Như vậy, đối với mảng tĩnh, ta cần cung cấp chính xác số lượng phần tử để máy cấp bộ nhớ trước khi chạy.
- Ta thêm khai báo thông tin biến n: `const int n = 2;` hoặc `float a[2];`

*int n;*

*cout<<"Nhập số phần tử của mảng n =";*

*cin>>n;*

*float \*a;*

*a = new float [n];*

- Thì lúc này chương trình hoạt động bình thường

## Kiểm tra điều kiện biên

---

```
float *j = &a[0], *k = &a[4];  
cout<<&j<<"    "<<&k<<endl;  
cout<<(k-j)<<endl;
```

- Kết quả: 4

## 6. mảng Thư viện - Library arrays

---

### language built-in array

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

int main()
{
    int myarray[3] = {10,20,30};

    for (int i=0; i<3; ++i)
        ++myarray[i];

    for (int elem : myarray)
        cout << elem << '\n';
}
```

### container library array

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

int main()
{
    array<int,3> myarray {10,20,30};

    for (int i=0; i<myarray.size(); ++i)
        ++myarray[i];

    for (int elem : myarray)
        cout << elem << '\n';
}
```