

Chương 6: Mảng

6.1. Định nghĩa:

Mảng là một tập hợp nhiều phần tử có cùng một kiểu giá trị và có cùng chung một tên

6.2. Khai báo:

Kiểu_dữ_liệu Tên_mảng[spt][spt]....;

Cụ thể:

a. Mảng một chiều:

Kiểu_dữ_liệu Tên_mảng[spt];

b. Mảng hai chiều:

Kiểu_dữ_liệu Tên_mảng[spt dòng][spt cột];

Lưu ý quan trọng:

- Chỉ số của mảng luôn luôn được đánh từ **chỉ số 0**
- Các phần tử của mảng ta có thể sử dụng như một biến bình thường

Ví dụ: Gán một phần tử của mảng cho 10

$a[2] = 10;$

Ví
du:
-
Mà
ng
1
chi

ều kiểu số nguyên có 10 phần tử: `int a[10];`

$a[0]$ $a[1]$ $a[2]$ $a[3]$ $a[4]$ $a[5]$ $a[6]$ $a[7]$ $a[8]$ $a[9]$

- Mảng 1 chiều kiểu ký tự có 5 phần tử:

`char s[5];`

//Lưu ý: Đây cũng có thể xem là một chuỗi có 5 ký tự

- Mảng 1 chiều kiểu chuỗi có 5 phần tử: `string chuoi[5];`

- Mảng 2 chiều kiểu số nguyên có 3 dòng, 4 cột: `int a[3][4];`

$a[0][0]$	$a[0][1]$	$a[0][2]$	$a[0][3]$
$a[1][0]$	$a[1][1]$	$a[1][2]$	$a[1][3]$
$a[2][0]$	$a[2][1]$	$a[2][2]$	$a[2][3]$

6.2. Bài tập:

i. Thực hiện các phép toán của mảng (1 chiều và 2 chiều):

- Khởi tạo một mảng
- Xuất các giá trị của mảng
- Thêm một phần tử vào mảng (1 chiều)
- Xoá một phần tử trong mảng (1 chiều)

ii. Nhập vào thông tin của một sinh:

- Họ tên
- Năm sinh
- Điểm các môn học (lưu vào mảng)

Xuất ra điểm trung bình của sinh viên đó

iii. Nhập vào một dãy số nguyên, tìm phần tử lớn nhất, nhỏ nhất của dãy (1 chiều, 2 chiều)

iv. Nhập vào một dãy số nguyên,

Xuất ra 2 dãy số:

- Dãy những số âm của mảng
- Dãy những số dương của mảng

v. Nhập vào một dãy số nguyên,

Xuất ra 2 dãy số:

- Dãy những số nguyên tố
- Dãy những số chính phương.

Chuyên đề: Mảng – ký tự - chuỗi

1. Ký tự

- Ví dụ: 'c' , 'b' , 'd'
- Tên kiểu dữ liệu: char
- Ví dụ:

```
char c;  
cin >> c;
```

2. Mảng

- Là một tập hợp các biến có cùng kiểu dữ liệu
- Từ phần tử của mảng ta có thể xem là một biến, ví dụ a[i], a[1], a[2][3],.... ta có thể thực hiện mọi lệnh, mọi phép toán trên biến áp dụng vào từ phần tử của mảng
- **Lưu ý: Mảng luôn luôn được đánh số từ 0**
- Cách khai báo:
 - Mảng 1 chiều:

```
Tên_dữ_liệu Tên_mảng[spt];
```
 - Mảng 2 chiều:

```
Tên_dữ_liệu Tên_mảng[số_dòng][số_cột]
```
- **Ví dụ:**

```
int n;  
int a[100]; //Khai báo mảng 1 chiều kiểu số nguyên có 10 phần tử  
  
cout<<"nhập vào số phần tử của mảng"<<endl;  
cin>>n;  
  
//Nhập vào từng phần tử của mảng  
cout<<"Mời bạn nhập vào từng phần tử của mảng"<<endl;  
for (int i = 0 ; i<n; ++i)  
    cin>>a[i];  
  
//Xuất các phần tử của mảng ra màn hình  
for (i = 0 ; i<n; ++i)  
    cout<<a[i];
```

3. Chuỗi

- Có thể nói **chuỗi là mảng 1 chiều kiểu ký tự** (từ phần tử của mảng là ký tự) – tức những thao tác về mảng, ta có thể áp dụng vào cho chuỗi
- **Chú ý:** Phần tử (ký tự) đầu tiên của chuỗi s là s[0], phần tử (ký tự) cuối cùng của chuỗi s là s[L – 1] , với L là chiều dài chuỗi

a. Chuỗi - mảng ký tự

- Khai báo: `char tên_chuỗi[chiều_dài_chuỗi];`
- Với `char s[10] = "Hello"` thì `s[0] = 'H'`, `s[1] = 'e'`, `s[2] = 'l'`, `s[3] = 'l'`, `s[4] = 'o'`, **đặc biệt s sẽ có thêm ký tự kết thúc chuỗi là `s[5] = '\0'`**. Khi đó chiều dài chuỗi là 5 (không tính ký tự kết thúc chuỗi)
- Ví dụ:
`char s[10]; //khai báo một chuỗi có tối đa 10 ký tự`
`//hay một mảng gồm 10 phần tử kiểu ký tự`
- Nhập dữ liệu vào chuỗi
 - i. **Cách 1:** Nhập từ phần tử giống như mảng
`//Với L là chiều dài của chuỗi`
`for (int i = 0; i < L; ++i)`
`cin>>s[i];`
 - ii. **Cách 2:** Nhập chuỗi **không có khoảng trắng**
`cin>>s;`
 - iii. **Cách 3:** Nhập chuỗi **có khoảng trắng**
`gets(s);`
- Cách xuất chuỗi
 - i. **Cách 1:** Xuất từ phần tử giống như mảng
`//Với L là chiều dài của chuỗi`
`for (int i = 0; i < L; ++i)`
`cout<<s[i];`
 - ii. **Cách 2:**
`cout<<s;`
- Các lệnh có liên quan đến chuỗi dạng kiểu ký tự

Cú pháp lệnh	Ý nghĩa	Thư viện phải khai báo
<code>int toupper(int c)</code>	Chuyển ký tự c thành ký tự hoa	<code>#include <ctype.h></code>
<code>int tolower(int c)</code>	Chuyển ký tự c thành ký tự thường	<code>#include <ctype.h></code>
<code>char* strupr(char* s)</code>	Chuyển s sang chuỗi hoa	<code>#include <string.h></code>
<code>char* strlwr(char* s)</code>	Chuyển s sang chuỗi thường	<code>#include <string.h></code>
<code>int strlen(const char* s)</code>	Trả về độ dài chuỗi	<code>#include <string.h></code>
<code>char* strcat(char* dest, const char* src)</code>	Cộng 2 chuỗi, kết quả trả về cho chuỗi dest	<code>#include <string.h></code>
<code>char* strcpy(char* dest, const char* src)</code>	Copy nội dung của chuỗi src về cho chuỗi dest	<code>#include <string.h></code>
<code>int strcmp(const char* s1, const char* s2)</code>	Giá trị trả về giá trị: - <0 : nếu s1 “nhỏ hơn” s2 - 0 : nếu s1 bằng s2 - >0 : nếu s1 “lớn hơn” s2	<code>#include <string.h></code>

int atoi(const char* s)	Đổi chuỗi số sang số nguyên (nếu không đổi được kết quả sẽ trả về giá trị 0) Ví dụ: <pre>char s[10] = "123"; int i = atoi(s); cout<<i; //Khi đó i sẽ bằng 123</pre>	#include <stdlib.h>
double atof(const char* s)	Đổi chuỗi số sang số thực (nếu không đổi được kết quả sẽ trả về giá trị 0.0) Ví dụ: <pre>char s[10] = "12.3"; float f = atof(s); cout<<f; //Khi đó f sẽ bằng 12.3</pre>	#include <stdlib.h>
char* itoa(int value, char *str, int hệ_cơ_số);	Đổi số value thành chuỗi, tương ứng với cơ số được khai báo Ví dụ: <pre>int i = "123"; char s[100]; itoa(i, s, 10); //đổi sang cơ số 10 cout<<s; //Khi đó s sẽ bằng "123"</pre>	#include <stdlib.h>
<u>*Định dạng chuỗi theo ngôn ngữ C</u> sprintf(char* s,chuỗi_định_dạng_của_C, tập_giá_trị); chuỗi định dạng là tập các ký tự cờ: "%c" – ký tự "%i" – số nguyên "%f" – số thực	Ví dụ: <pre>char s[100]; int i = 10; float f = 132.124; sprintf(s, "%i", i); sprintf(s, "%f", f); sprintf(s, "%.2f", f); //lấy 2 chữ số //sau dấu . //của số thực</pre>	#include <stdio.h>

b. **Chuỗi - kiểu string** (sẽ được học kỹ ở chương 14)

- Thư viện cần khai báo: **#include <string>**
- Khai báo: string tên_chuỗi[chiều_dài_chuỗi];
- Ví dụ: string s;
- Nhập dữ liệu vào chuỗi
 - i. **Cách 1**: Nhập chuỗi **không có khoảng trắng**
cin>>s;
 - ii. **Cách 3**: Nhập chuỗi **có khoảng trắng**
getline(cin,s);
- Cách xuất chuỗi
cout<<s;

Lưu ý:

- **Cách khai báo hằng:** (hằng là những biến là giá trị sẽ không bao giờ bị thay đổi trong chương trình)

+ **Cách 1: (dùng macro #define)**

```
#define biến_hằng giá_trị //Không có dấu ; cuối lệnh
```

- Ý nghĩa: nếu dùng từ khoá marco **#define**, trong khi biên dịch, chương trình sẽ thay tất cả **biến_hằng** bằng **giá_trị**

- **Ví dụ:**

```
#define bien_hang 123
```

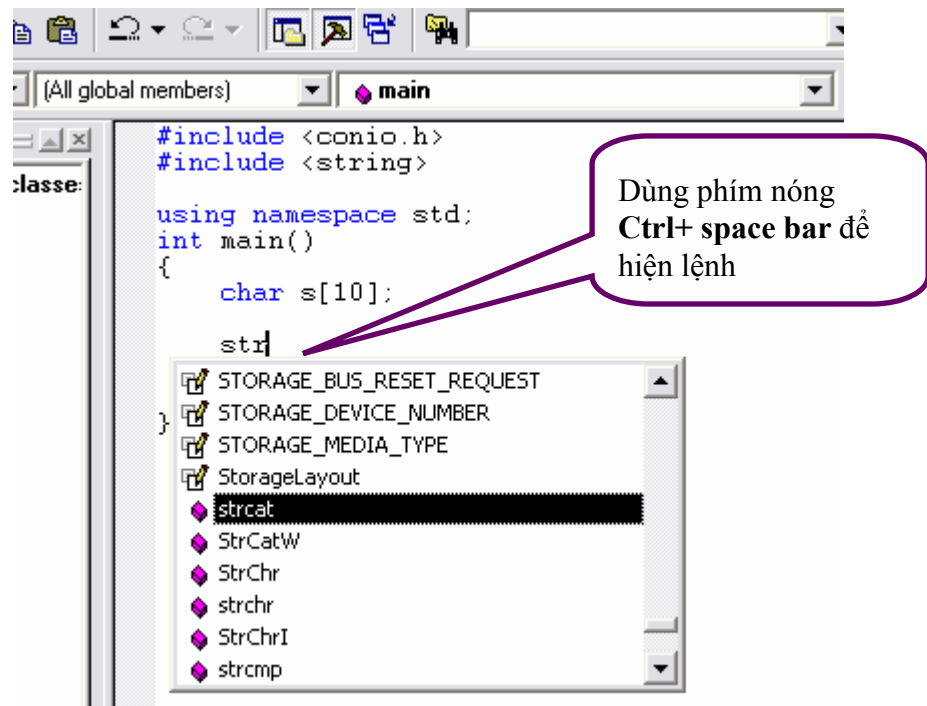
+ **Cách 2:**

```
const kiểu_dữ_liệu tên_hằng = giá_trị; //có dấu ; cuối lệnh
```

Ví dụ:

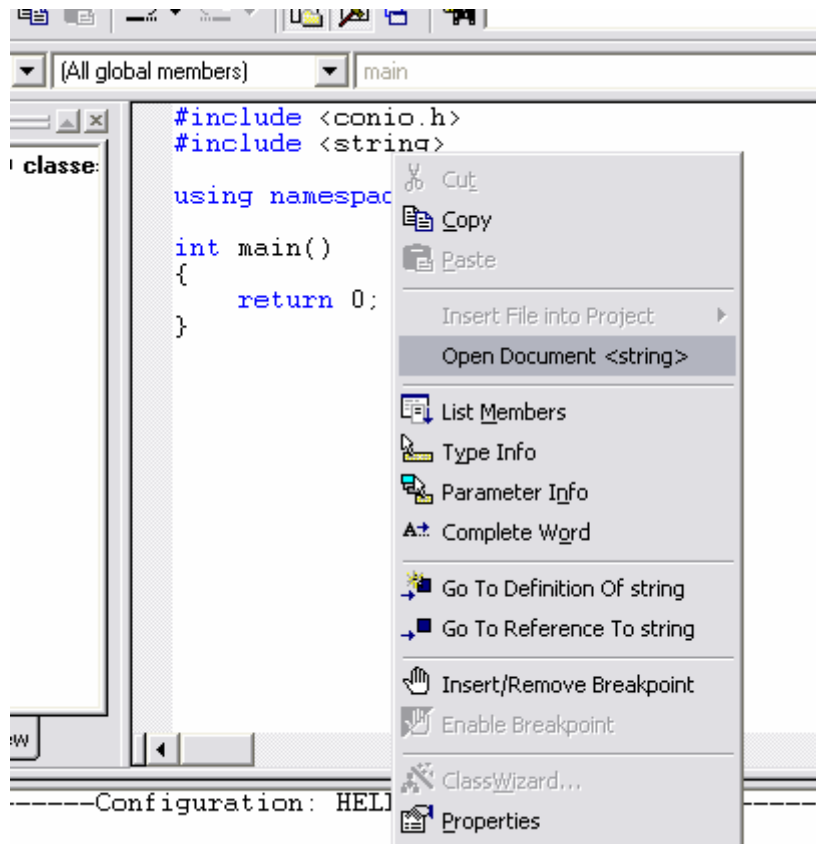
```
const int bien_hang = 123;
```

- Khi lập trình trên môi trường Visual C++, trong khi đánh lệnh ta có thể dùng phím nóng **Ctrl + Space bar** để hiển thị các lệnh của trong C++



Do hàm getline trong Visual C++ bị lỗi, ta sẽ sửa lỗi bằng cách sau:

+ Bước 1: Bấm chuột phải lên chỗ khai báo thư viện **<string>**, bấm chuột phải chọn **“Open Document <string>”**



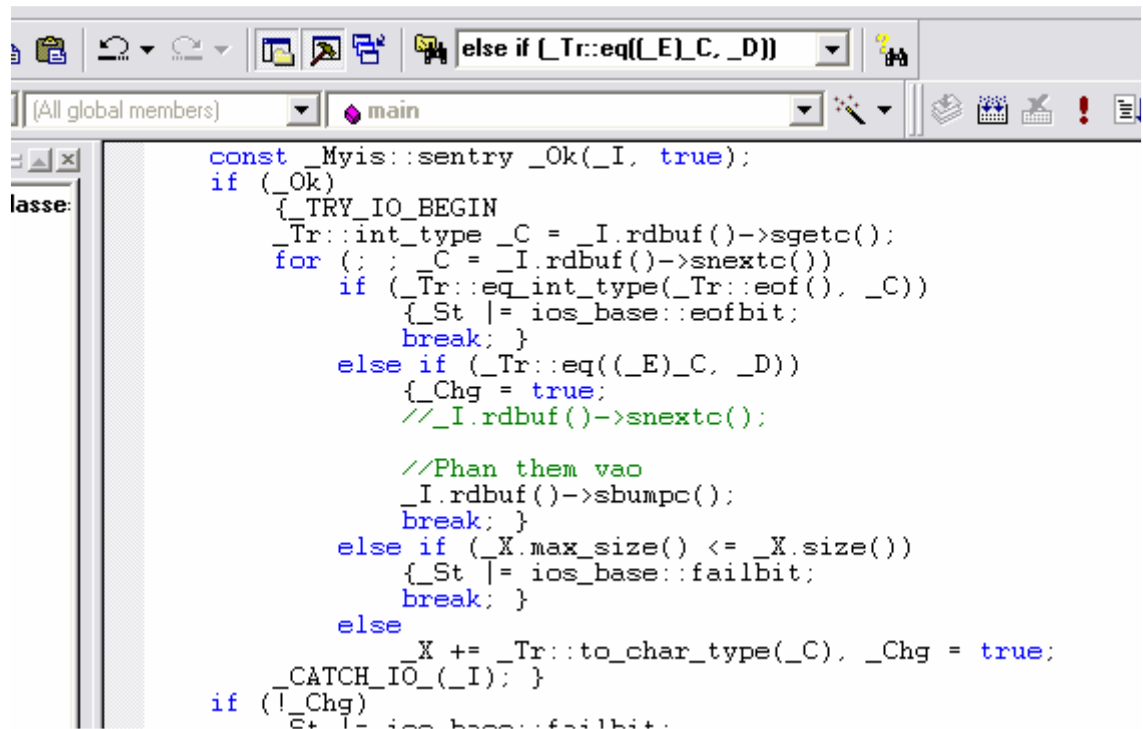
+ Bước 2: Bấm phím nóng Ctrl + F

+ Bước 3: gõ vào hộp thoại tìm kiếm dòng chữ:

else if (_Tr::eq((_E)_C, _D))

+ Bước 4: Thay đổi một số lệnh

Ban đầu	Sau khi sửa
<pre>else if (_Tr::eq((_E)_C, _D)) { _Chg = true; _I.rdbuf()->snextc(); break; } </pre>	<pre>else if (_Tr::eq((_E)_C, _D)) { _Chg = true; // _I.rdbuf()->snextc(); //Phần thêm vào _I.rdbuf()->sbumpc(); break; } </pre>



```

const _Myis::sentry _Ok(_I, true);
if (_Ok)
{
    _TRY_IO_BEGIN
    _Tr::int_type _C = _I.rdbuf()->sgetc();
    for (;;) _C = _I.rdbuf()->snextc()
        if (_Tr::eq_int_type(_Tr::eof(), _C))
        {
            _St |= ios_base::eofbit;
            break; }
        else if (_Tr::eq((_E)_C, _D))
        {
            _Chg = true;
            // _I.rdbuf()->snextc();

            //Phần thêm vào
            _I.rdbuf()->sbumpc();
            break; }
        else if (_X.max_size() <= _X.size())
        {
            _St |= ios_base::failbit;
            break; }
        else
            _X += _Tr::to_char_type(_C), _Chg = true;
    _CATCH_IO(_I); }
if (!_Chg)
    _St |= ios_base::failbit;

```

+ Bước 5: Bấm Ctrl + S (lưu nội dung đã thay đổi), sau đó đóng cửa sổ lại.

Bài tập bổ sung:

- Nhập vào một mảng 2 chiều $n \times n$ (ma trận vuông)
 - ☐ Tính tổng các số hạng trên đường chéo chính
 - ☐ Tính tổng các số hạng trên đường chéo phụ
 - ☐ Duyệt phân nửa trên của đường chéo chính
- Sắp xếp tăng dần/giảm dần các giá trị của một mảng 1 chiều
- Bài tập về phân số
 - ☐ Nhập vào 2 phân số (phân số gồm có tử và mẫu)
 - ☐ Tính tổng, hiệu, tích, thương của 2 phân số
 - ☐ Rút gọn phân số
- Xuất ra màn hình các hình sao (với chiều cao của mỗi hình là một số nguyên dương h được nhập từ bàn phím)

```

      *
     **
    ***
   ****
  *****

          *
         **
        ***
       ****
      *****

          *
         **
        ***
       ****
      *****

          *
         **
        ***
       ****
      *****
  
```

- Nhập vào một chuỗi s và một ký tự c , kiểm tra xem trong s có ký tự c không, nếu có thì xuất vị trí của c trong s , nếu không xuất ra giá trị -1
- Tương tự như bài tập 4, nhưng thay vì là ký tự c , ta sẽ thay bằng một chuỗi $s2$

Chuyên đề: Hỗ trợ kỹ thuật lập trình

A. Phạm vi hoạt động của một biến

1. Phân loại biến:

- Biến toàn cục: Là những biến có phạm vi hoạt động xuyên suốt từ đầu đến cuối chương trình
- Biến cục bộ:
 - + Là những biến có phạm vi hoạt động tại một thời điểm của chương trình.
 - + Phạm vi hoạt động của biến cục bộ là từ lúc khai báo biến cho đến **dấu đóng khối gần nhất chứa nó**.

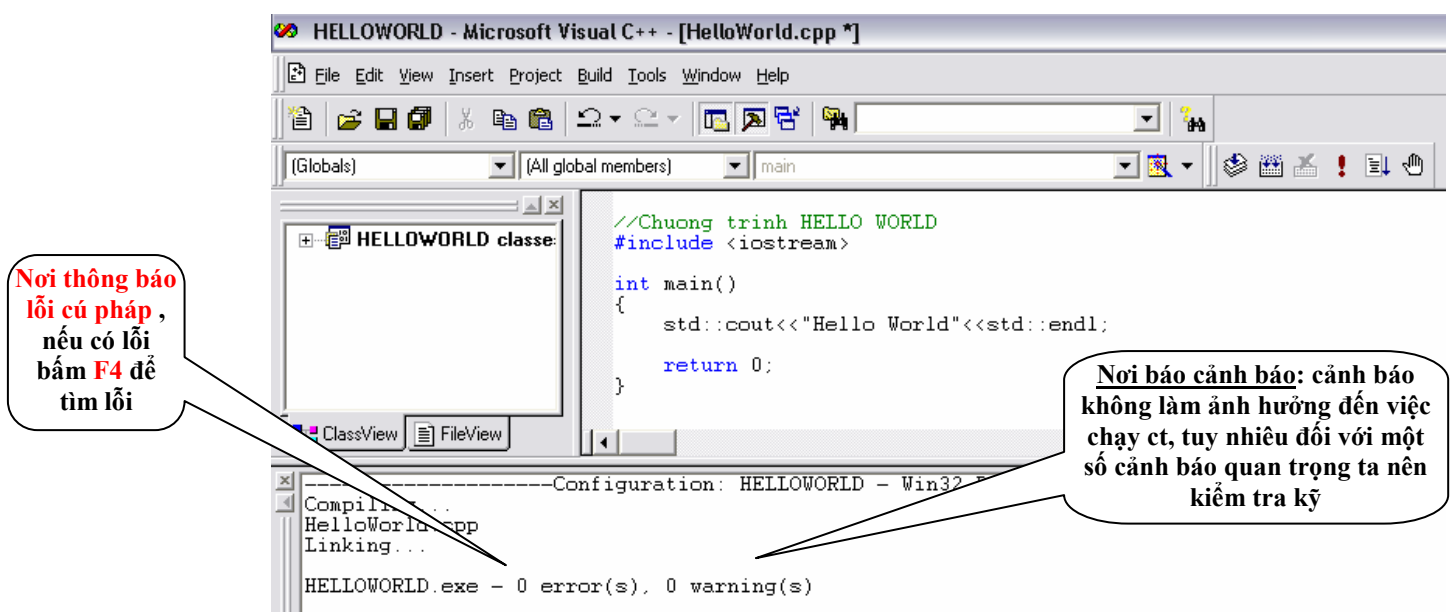
2. Nơi khai báo biến:

- Biến toàn cục: khai báo ngoài hàm main
- Biến cục bộ: khai báo trong hàm main

B. Debug

1. Phân loại các loại lỗi (Error):

- Lỗi cú pháp: Là những lỗi do người lập trình không ghi đúng cú pháp lệnh, những lỗi này sẽ được chương trình biên dịch thông báo.



- Lỗi logic: Là những lỗi chương trình do người lập trình viết không thực hiện đúng với nội dung yêu cầu được đặt ra (ví dụ: yêu cầu của bài toán, yêu cầu của khách hàng,...)

Đối với những lỗi logic chúng ta phải sử dụng **Debug** để tìm ra lỗi.

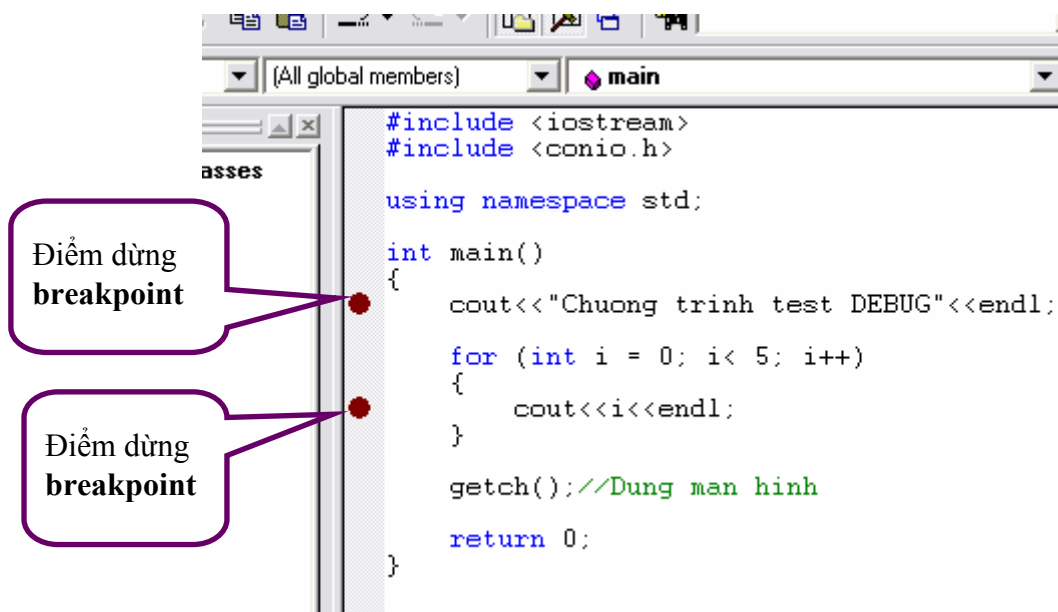
2. Debug là gì?:

- Là cách tìm những lỗi logic bằng cách để chương trình chạy từng dòng lệnh (chạy từng bước từ trên xuống), qua từng dòng lệnh ta sẽ xác định giá trị của từng biến thay đổi như thế nào có đúng với thuật toán đưa ra hay không => từ đó ta có thể xác định được lỗi logic của chương trình.

3. Cách hoạt động của Debug:

Bước 1: Đặt những điểm dừng **breakpoint** bằng bấm phím nóng **F9**

(breakpoint sẽ đánh dấu những dòng lệnh, khi đó chương trình sẽ dừng lại tại những dòng lệnh chứa breakpoint)



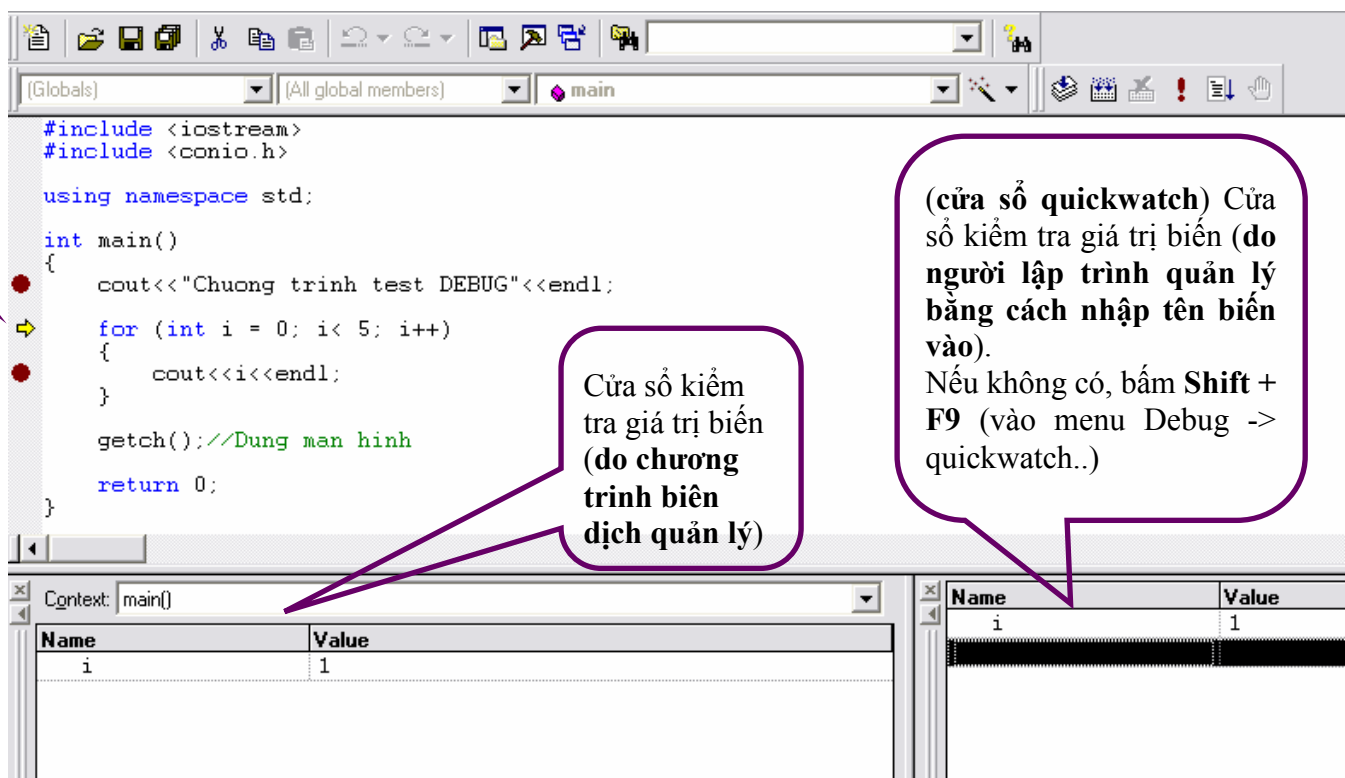
Bước 2: Bấm **F5** để chương trình sẽ bắt đầu chế độ Debug (**chương trình sẽ dừng lại ở vị trí breakpoint**).

Tuỳ theo mục đích của người kiểm tra lỗi, ta có các phím nóng để hỗ trợ

- + **F10**: chương trình chạy từng bước, bỏ qua chương trình con
- + **F11**: chương trình chạy từng bước, nếu gặp chương trình con sẽ vào kiểm tra (không bỏ qua chương trình con).
- + **F5**: đến điểm dừng breakpoint gần đó nhất
- + **F9**: thêm điểm dừng breakpoint
- + **Shift + F5**: Dừng chế độ Debug

Lưu ý:

1 .Màn hình trong chế độ debug gồm 3 cửa sổ chính



2. **Ctrl + F5** : chạy chương trình không Debug

F5: chạy chương trình ở chế độ Debug