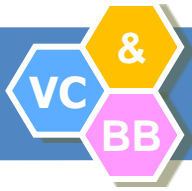


Nội dung

- 1 **Giới thiệu**
- 2 **Bộ từ vựng của C**
- 3 **Cấu trúc chương trình C**
- 4 **Một số ví dụ minh họa**
- 5 **Debug**

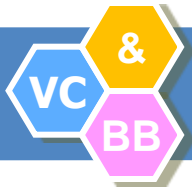


Giới thiệu

❖ Giới thiệu

- Dennis Ritchie tại Bell Telephone năm 1972.
- Tiền thân của ngôn ngữ **B**, KenThompson, cũng tại **B**ell Telephone.
- Là ngôn ngữ lập trình có cấu trúc và phân biệt chữ Hoa - thường (**case sensitive**)
- ANSI C.

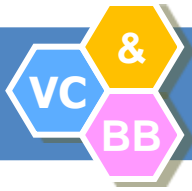




Giới thiệu

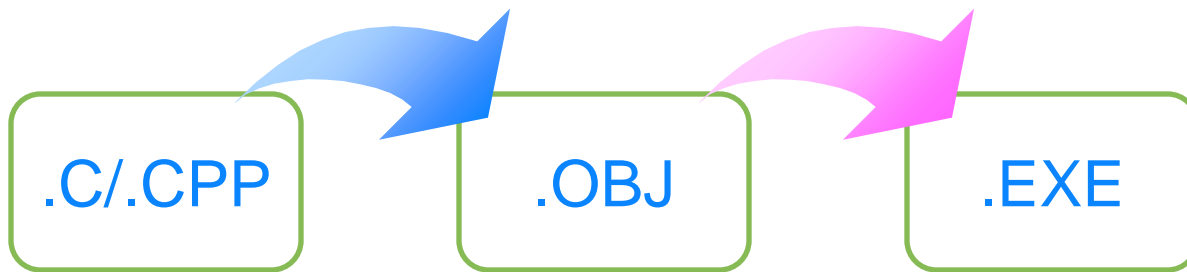
❖ Ưu điểm của C

- **Rất mạnh và linh động**, có khả năng thể hiện bất cứ ý tưởng nào.
- **Được sử dụng rộng rãi** bởi các nhà lập trình chuyên nghiệp.
- **Có tính khả chuyển**, ít thay đổi trên các hệ thống máy tính khác nhau.
- **Rõ ràng, cô đọng**.
- **Lập trình đơn thể (Module)**, tái sử dụng thông qua hàm.



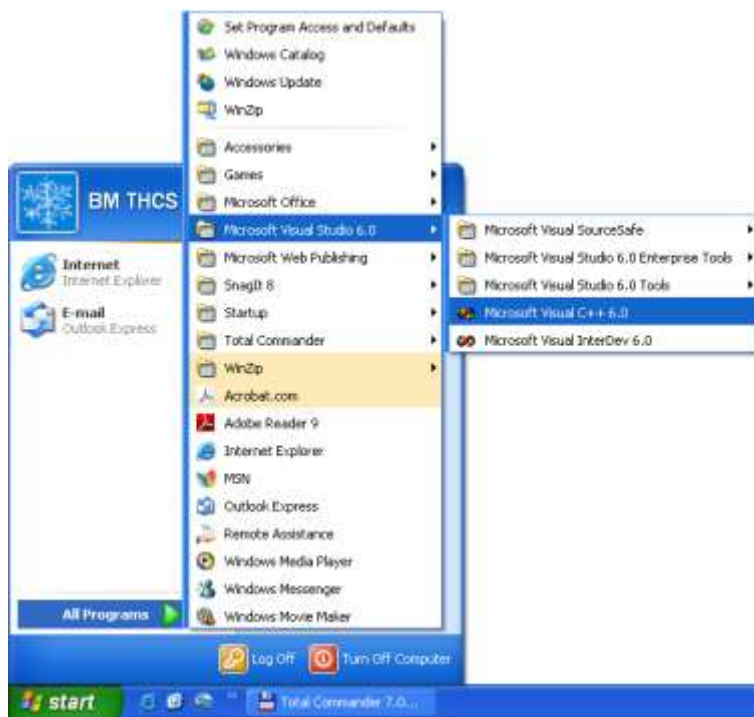
Giới thiệu

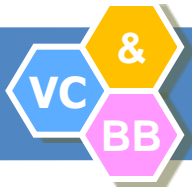
- ❖ Môi trường phát triển tích hợp IDE (Integrated Development Environment)
 - Biên tập chương trình nguồn (Trình EDIT).
 - Biên dịch chương trình (Trình COMPILE).
 - Chạy chương trình nguồn (Trình RUNTIME).
 - Sửa lỗi chương trình nguồn (Trình DEBUG).



❖ Môi trường lập trình

- Borland C++ 3.1 for DOS.
- Visual C++ 6.0, Win32 Console Application.





Bộ từ vựng của C

❖ Các ký tự được sử dụng

- Bộ chữ cái 26 ký tự Latinh **A, B, C, ..., Z, a, b, c, ..., z**
- Bộ chữ số thập phân : **0, 1, 2, ..., 9**
- Các ký hiệu toán học : **+ - * / = < > ()**
- Các ký tự đặc biệt : **. , : ; [] % \ # \$ ' "**
- Ký tự gạch nối **_** và khoảng trắng

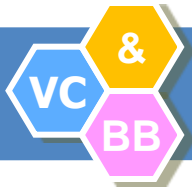




Bộ từ vựng của C

❖ Từ khóa (keyword)

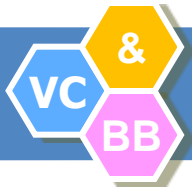
- Các từ **dành riêng** trong ngôn ngữ.
- **Không** thể sử dụng từ khóa để đặt tên cho biến, hàm, tên chương trình con.
- Một số từ khóa thông dụng:
 - `const, enum, signed, struct, typedef, unsigned...`
 - `char, double, float, int, long, short, void`
 - `case, default, else, if, switch`
 - `do, for, while`
 - `break, continue, goto, return`



Bộ từ vựng của C

❖ Tên/Định danh (Identifier)

- Một dãy ký tự dùng để chỉ tên một hằng số, hằng ký tự, tên một biến, một kiểu dữ liệu, một hàm một hay thủ tục.
- Không được trùng với các từ khóa và được tạo thành từ các chữ cái và các chữ số nhưng bắt buộc chữ đầu phải là chữ cái hoặc _.
- Số ký tự tối đa trong một tên là 255 ký tự và được dùng ký tự _ chen trong tên nhưng không cho phép chen giữa các khoảng trắng.



Bộ từ vựng của C

❖ Ví dụ Tên/Định danh (Identifier)

- Các tên hợp lệ: GiaiPhuongTrinh, Bai_Tap1
- Các tên không hợp lệ: 1A, Giai Phuong Trinh
- **Phân biệt chữ hoa chữ thường**, do đó các tên sau đây khác nhau:
 - A, a
 - BaiTap, baitap, BAITAP, bAltaP, ...





Bộ từ vựng của C

❖ Dấu chấm phẩy ;

- Dùng để phân cách các câu lệnh.
- Ví dụ: `printf("Hello World!"); printf("\n");`

❖ Câu chú thích

- Đặt giữa cặp dấu `/* */` hoặc `//` (C++)
- Ví dụ: `/*Ho & Ten: NVA*/`, `// MSSV: 0712078`

❖ Hằng ký tự và hằng chuỗi

- Hằng ký tự: `'A'`, `'a'`, ...
- Hằng chuỗi: `"Hello World!"`, `"Nguyen Van A"`
- **Chú ý:** `'A'` khác `"A"`

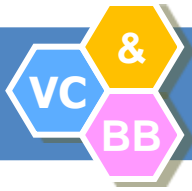


Cấu trúc chương trình C

```
#include "...";    // Khai báo file tiêu đề

int x;             // Khai báo biến hàm
void Nhap();       // Khai báo hàm

int main() // Hàm chính
{
    // Các lệnh và thủ tục
}
```



Ví dụ

```
#include <stdio.h>

int main()
{
    int x, y, tong;
    printf("Nhap hai so nguyen: ");
    scanf("%d%d", &x, &y);
    tong = x + y;
    printf("Tong hai so la %d ", tong);
}
```



Debug

Bước 1: biên dịch chương trình

- Build -> Compile

Bước 2: Đánh dấu vị trí bắt đầu Debug

- Đặt con trỏ vào dòng lệnh bắt đầu Debug
- Gõ phím F10 (hoặc nháy vào nút có biểu tượng bàn tay trên thanh công cụ)

Bước 3: bắt đầu Debug

- Vào Debug -> Start Debug (F9)

Bước 4: chuyển đến dòng lệnh tiếp theo

- Chọn Step Over (F8) để chuyển xuống dòng tiếp theo
- Quan sát kết quả trung gian trong cửa sổ





Giới thiệu một số chức năng của Debug

1. Add/Remove Breakpoint

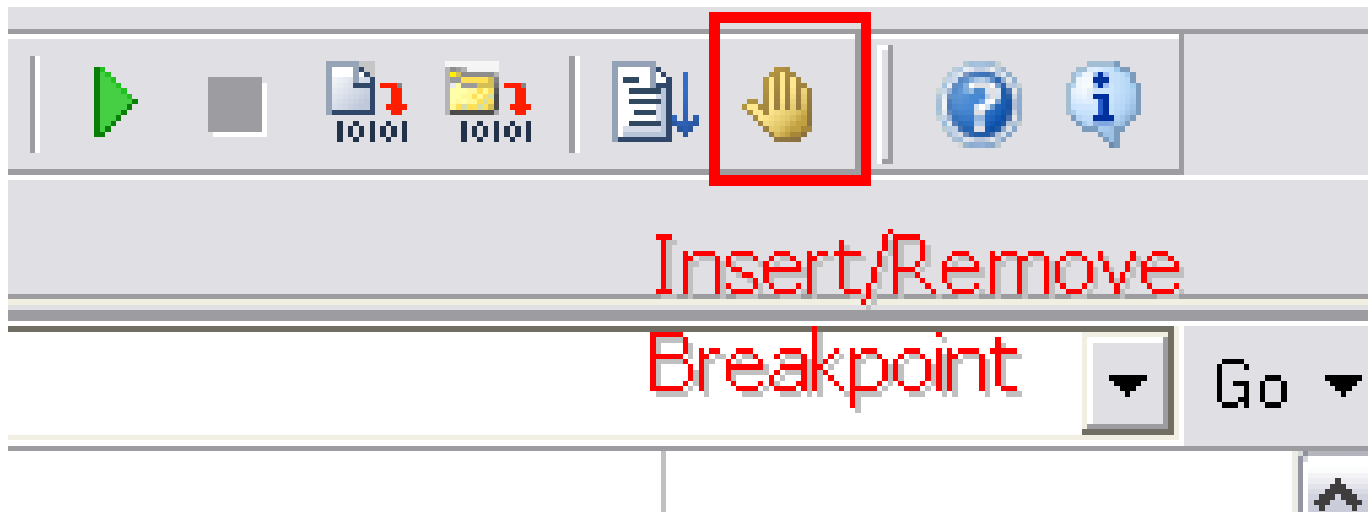
Có 2 cách để thêm một **breakpoint**:

- (1) Di chuyển chuột tới lề trái của vùng soạn thảo
- Kích chuột trái, C-Free sẽ highlights dòng này và thêm một chấm đỏ

```
17 int ComputeValue(int n)
18 {
19     int i,s=0;
20     for(i=1; i<=n; i++)
21     {
22         s += i;
23     }
24     return s;
25 }
26
```

1. Add/Remove Breakpoint

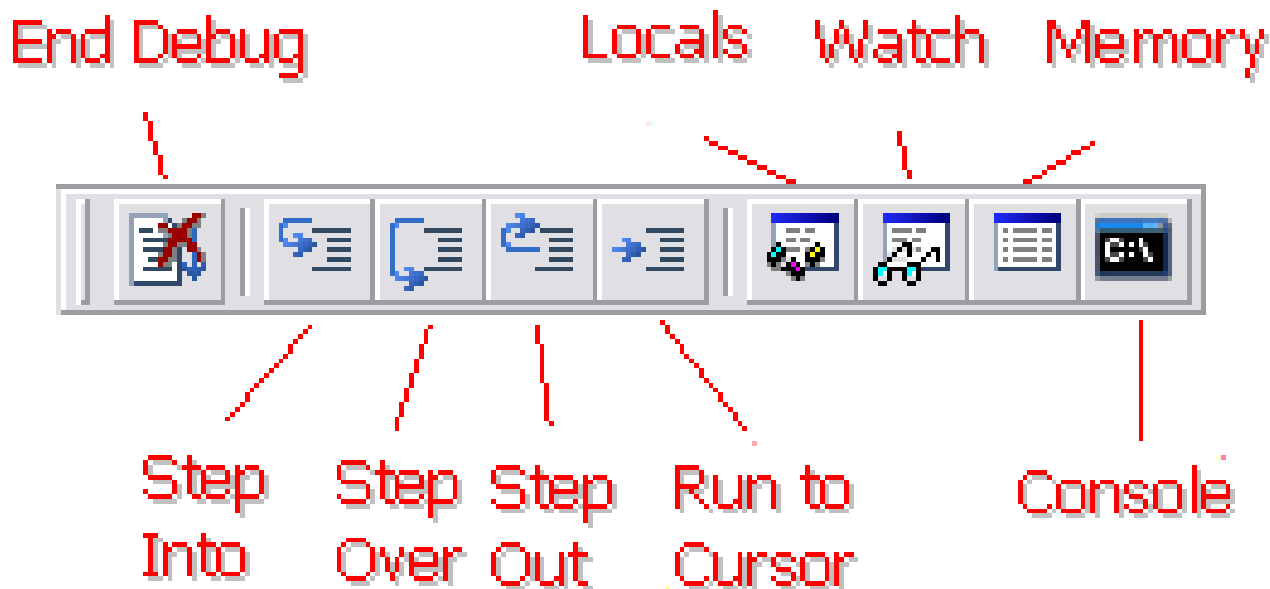
Cách thứ hai: Nháy chọn nút «Insert/ Remove breakpoint» trên thanh công cụ.



- Nháy chọn nút «Insert/ Remove breakpoint» lần 2 sẽ xóa breakpoint.

2. Start Debug

- Vào **Debug** -> **Start Debug** (F9)
- Xuất hiện thanh công cụ:



3. End Debug: dừng debug

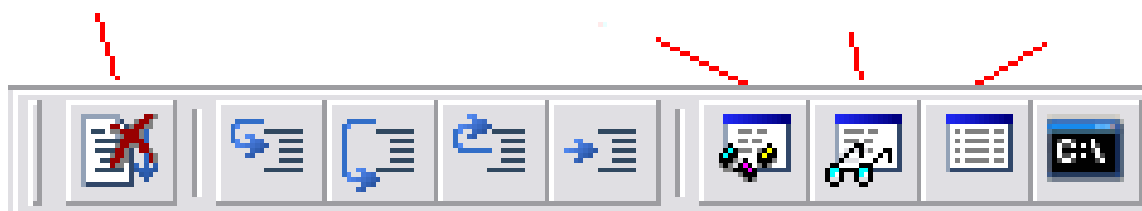
- Vào **Debug** -> **Stop Debugging** (**Ctrl + F9**)
- Hoặc nhấn nút **End Debug** trên thanh toolbar

End Debug

Locals

Watch

Memory



Step
Into

Step
Over

Step
Out

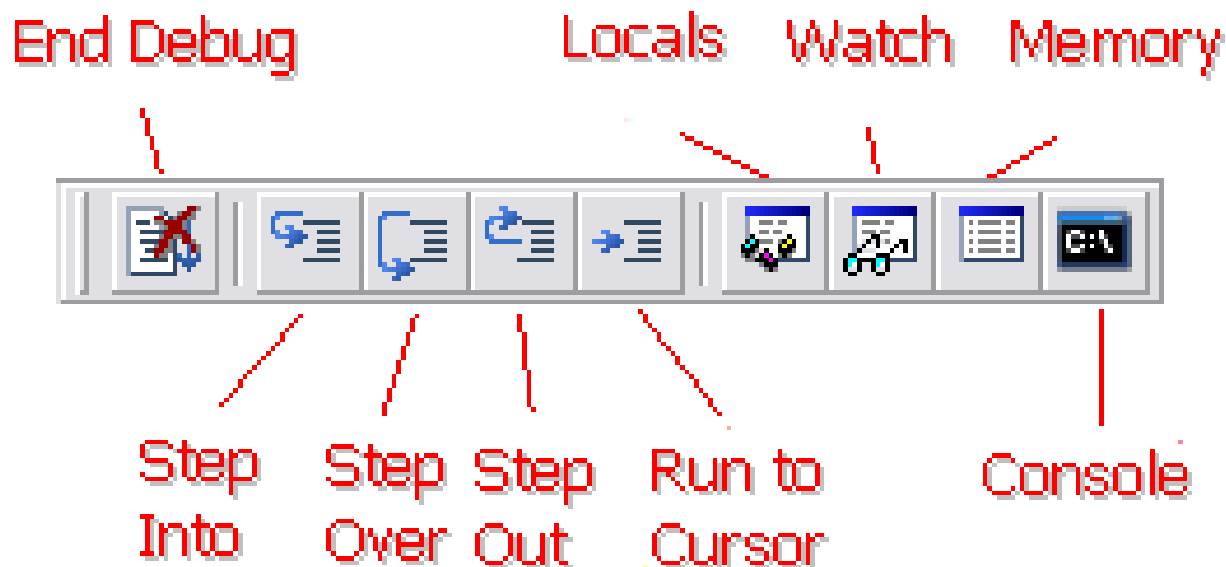
Run to
Cursor

Console

3. Step Into: vào trong hàm

Khi breakpoint đặt tại lời gọi hàm, để vào thân hàm:

- Vào **Debug** -> **Step Into (F7)**
- Hoặc nháy nút **Step Into** trên thanh toolbar



Giới thiệu một số chức năng của Debug

Ví dụ: đặt một breakpoint tại lời gọi hàm UpperCase

- Nháy nút Step Into

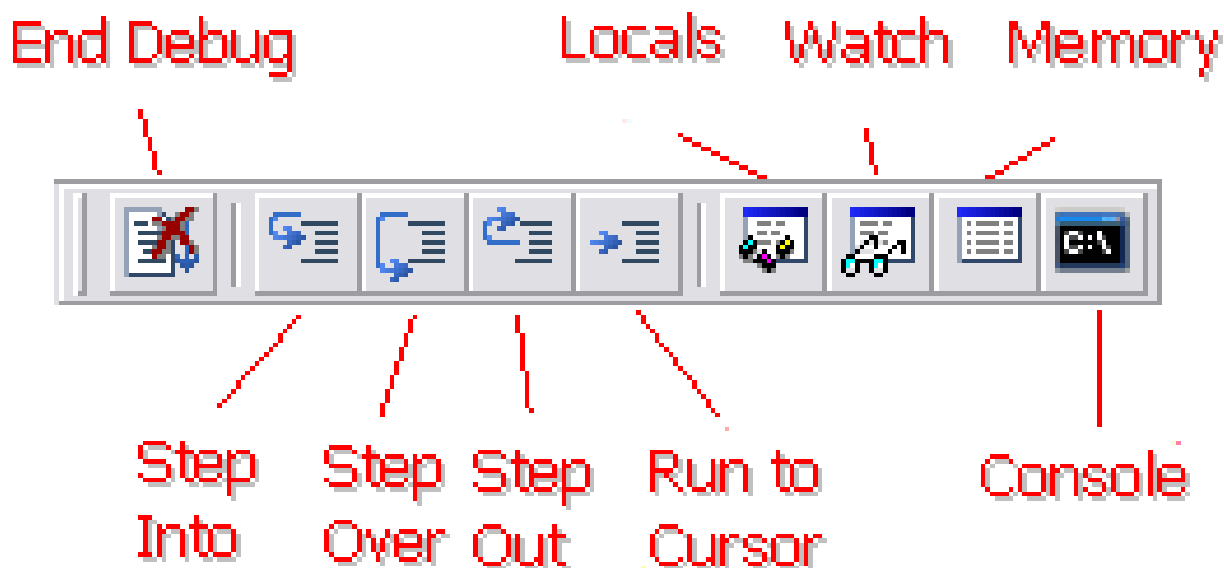
```
6 int main(int argc, char *argv[])
7 {
8     char szMyStr[] = "C-Free Debug";
9
10    printf("%d\n", ComputeValue(1000));
11    UpperCase(szMyStr);
12    printf(szMyStr);
13
14    return 0;
15 }
```

16 Nháy nút Step Into, chương trình chạy vào hàm UpperCase

```
28
29 void UpperCase(char *str)
30 {
31     char *p = str;
32     while(*p)
33     {
34         if(*p>='a' && *p<='z')
35         {
36             *p = *p - 'a' + 'A';
37         }
38         p++;
39     }
40 }
```

4. Step Out: ra khỏi hàm

Nếu đang break trong thân hàm, nhấn nút **Step Out** (**Shift + F7**) trên thanh toolbar, thì chương trình sẽ ra khỏi hàm quay về câu lệnh tiếp theo trong chương trình gọi tới hàm đó.





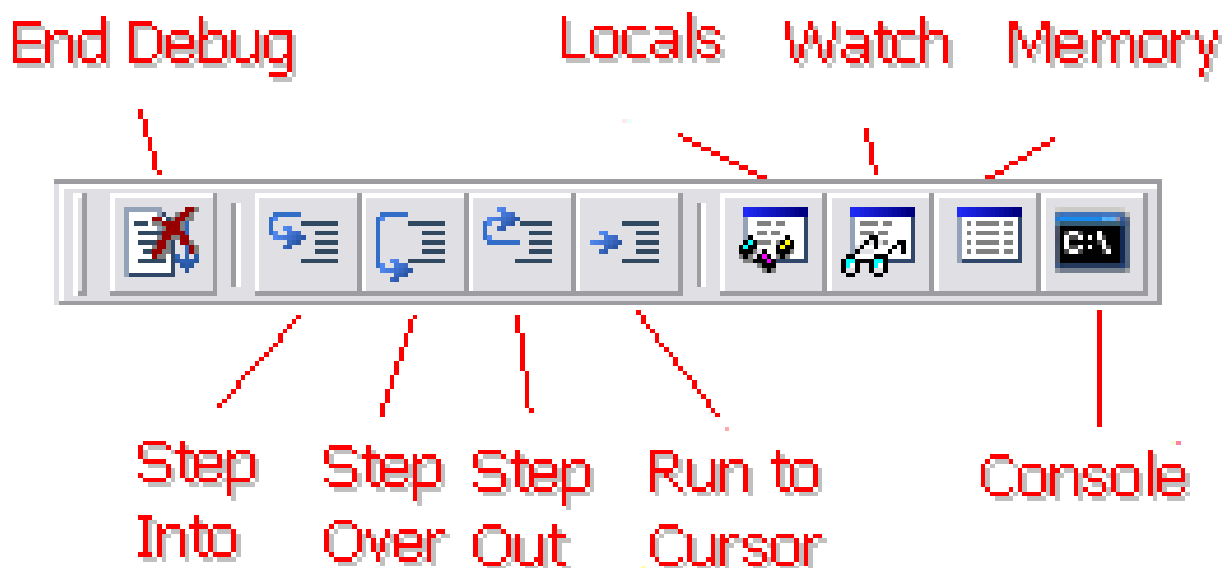
Giới thiệu một số chức năng của Debug

5. Step Over: chuyển đến câu lệnh tiếp theo

```
6 int main(int argc, char *argv[])
7 {
8     char szMyStr[] = "C-Free Debug";
9
10    printf("%d\n", ComputeValue(1000));
11    UpperCase(szMyStr);
12    printf(szMyStr);
13
14    return 0;
15 }
```

6. Run to Cursor: chương trình sẽ chạy các lệnh đến vị trí đặt con trỏ

- Vào **Debug** -> **Run to Cursor** (**Ctrl + F8**)
- Hoặc nháy nút **Run to Cursor** trên thanh toolbar hàm đó.

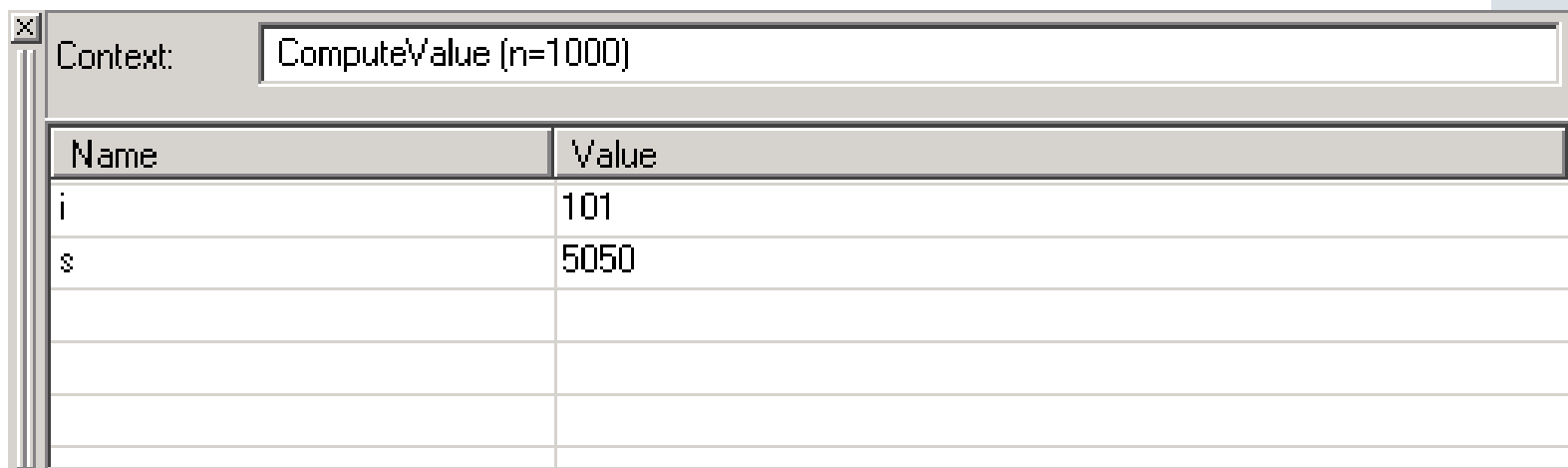




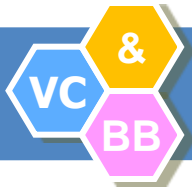
Giới thiệu một số chức năng của Debug

7. Locals: chương trình sẽ chạy các lệnh đến vị trí đặt con trỏ

- Khi chương trình đang trong trạng thái debugging, cửa sổ **Locals** hiển thị giá trị của tất cả các biến cục bộ.



Name	Value
i	101
s	5050



Giới thiệu một số chức năng của Debug

8. Inspect/Change:

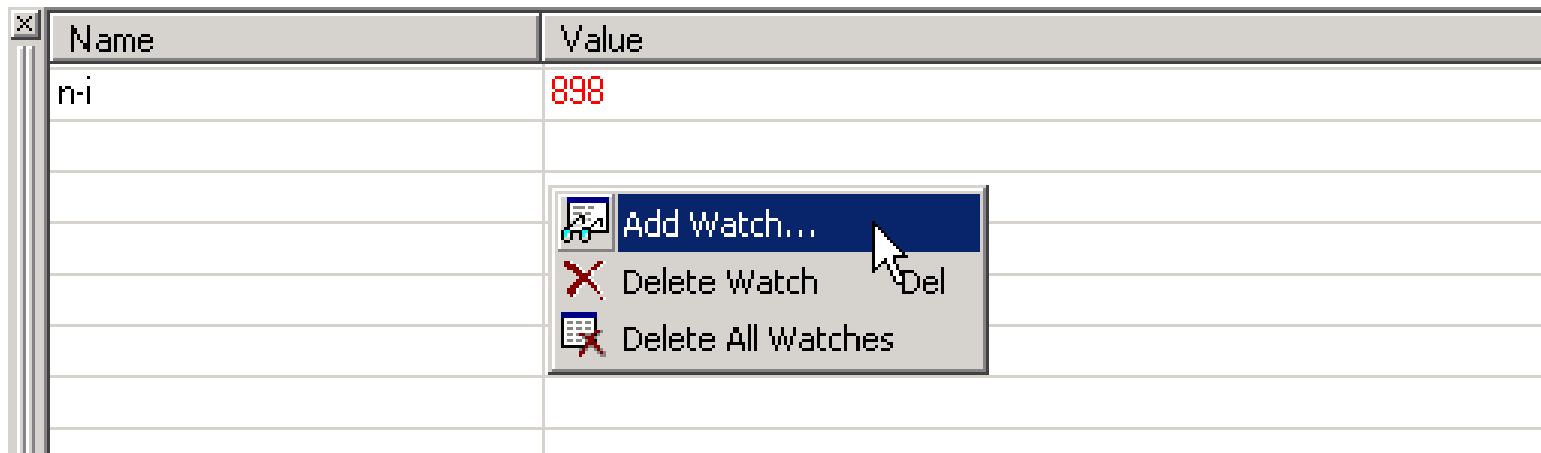
- Nháy chuột phải tại một biến trong cửa sổ **Locals**
- Chọn "**Inspect/Change**"
- Hoặc nháy đúp chuột)

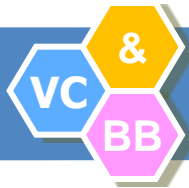
Xuất hiện một cửa sổ hiển thị biến: **array**, **structure**, hoặc **class**: trong cửa sổ **Inspector**. Các kiểu biến còn lại hiển thị trong cửa sổ **Change**.



8. Watch:

- Sử dụng cửa sổ **Watch** để truy vết giá trị của biểu thức trong thời gian **debugging**. Nháy chuột phải tại cửa sổ Watch, chọn "**Add Watch**"
- Nhập vào biểu thức cần truy vết.





Bài tập lý thuyết

1. Tên (định danh) nào sau đây đặt không hợp lệ, tại sao?
 - Tin hoc co SO A, 1BaiTapKHO
 - THucHaNH, NhapMon_L@pTrinH
2. Câu ghi chú dùng để làm gì? Cách sử dụng ra sao? Cho ví dụ minh họa.
3. Trình bày cấu trúc của một chương trình C. Giải thích ý nghĩa của từng phần trong cấu trúc.

