Lập trình trực quan

Chương 2: DỮ LIỆU, CẤU TRÚC ĐIỀU KHIỂN VÀ GỠ RỐI CHƯƠNG TRÌNH

TS. Cao Thị Luyên
luyenct@utc.edu.vn
0912403345

Nội dung

1. Biến, hằng và các kiểu dữ liệu
2. Các cấu trúc điều khiển
3. Gỡ rối chương trình

Mục tiêu

- Hiểu được khái niệm biến và hằng trong C#; Biết cách khai báo và sử dụng biến, hằng; Nắm được các kiểu dữ liệu cơ bản trong C#; Phân biệt được kiểu giá trị và kiểu tham chiếu
- Hiểu và vận dụng cấu trúc điều khiển trong C#
- Kiểm soát lỗi chương trình

1. Biến, hằng và các kiểu dữ liệu

Biến trong C#

- Là vùng nhớ dùng để lưu trữ dữ liệu
- Cần khai báo trước khi sử dụng
- Cú pháp: <kiểu dữ liệu> <tên biến>;

  Ví dụ: int tuoi;
- Có thể khởi tạo khi khai báo: int tuoi = 20;

Hằng trong C#

- Là giá trị không thay đổi trong suốt chương trình
- Khai báo với từ khóa 'const'
- Cú pháp: const <kiểu dữ liệu> <tên hằng> = <giá trị>;

  Ví dụ: const double PI = 3.14;

Các kiểu dữ liệu cơ bản trong C#

- Kiểu số nguyên: int, long, short, byte
- Kiểu thực: float, double, decimal
- Kiểu logic: bool (true/false)
- Kiểu ký tự và chuỗi: char, string

Kiểu dữ liệu

Kiểu	Biểu diễn	Dãy giá trị	Giá trị mặc định
bool	Giá trị Boolean	True hoặc False	False
byte	Kiểu unsigned integer (8 bit)	0 tới 255	0
char	Kiểu Unicode character (16 bit)	U +0000 tới U +ffff	'\0'
decimal	Kiểu thập phân (128 bit)	$(-7.9 \times 10^{28}$ tới $7.9 \times 10^{28}) / 10^{0 \text{ to } 28}$	0.0M
double	Kiểu double (64 bit)	$(+/-)5.0 \times 10^{-324}$ tới $(+/-)1.7 \times 10^{308}$	0.0D
float	Kiểu float (32 bit)	-3.4×10^{38} tới $+ 3.4 \times 10^{38}$	0.0F
int	Kiểu integer (32 bit)	-2,147,483,648 tới 2,147,483,647	0
long	Kiểu signed integer (64 bit)	-9,223,372,036,854,775,808 tới 9,223,372,036,854,775,807	0L
sbyte	Kiểu signed integer (8 bit)	-128 tới 127	0

Chuyển đổi kiểu dữ liệu

Chuyển đổi kiểu tường minh (explicit)

- Sử dụng phương thức Parse()

```
string stringValue = "123";
int a = int.Parse( stringValue );    // a sẽ mang giá trị 123

float b = float.Parse( "20.7" );     // b sẽ mang giá trị 20.7
bool c = bool.Parse( "true" );       // c sẽ mang giá trị True
```

- Sử dụng lớp hỗ trợ Convert

```
int a = Convert.ToInt32( "123" );
Console.WriteLine(a);                // a = 123

double b = Convert.ToInt32( 789 );
Console.WriteLine(b);                // b = 789

bool c = Convert.ToBoolean( 123 );
Console.WriteLine(c);                // c = True

bool d = Convert.ToBoolean(null);    // Tham số truyền vào là null -> sẽ output giá trị
Console.WriteLine(d);                // d = False
```

Chuyển đổi kiểu dữ liệu

Chuyển đổi kiểu tường minh (explicit)

- Sử dụng phương thức Parse()

```
string stringValue = "123";
int a = int.Parse( stringValue );    // a sẽ mang giá trị 123

float b = float.Parse( "20.7" );     // b sẽ mang giá trị 20.7
bool c = bool.Parse( "true" );       // c sẽ mang giá trị True
```

- Sử dụng lớp hỗ trợ Convert

```
int a = Convert.ToInt32( "123" );
Console.WriteLine(a);                // a = 123

double b = Convert.ToInt32( 789 );
Console.WriteLine(b);                // b = 789

bool c = Convert.ToBoolean( 123 );
Console.WriteLine(c);                // c = True

bool d = Convert.ToBoolean(null);    // Tham số truyền vào là null -> sẽ output giá trị
Console.WriteLine(d);                // d = False
```

Giá trị vs Tham chiếu

- Kiểu giá trị (value types): lưu trực tiếp dữ liệu ở stack

  Ví dụ: int, double, bool

- Kiểu tham chiếu (reference types): lưu địa chỉ tham chiếu đến dữ liệu ở heap

  Ví dụ: string, object, array, class

Ví dụ minh hoạ

```
int a = 5;
const double PI = 3.14;
string name = "Lan";
bool isStudent = true;
int[] a = { 1, 2, 3 }; int[] b = a;
class Person
{
    protected string Name;
        public Person(){this.Name=“Lan”;}
}
Person p1 = new Person();
Person p2 = p1;
```

2. Cấu trúc điều khiển

- Câu lệnh rẽ nhánh, lựa chọn: if, switch
- Câu lệnh lặp: for, while, do while
- Câu lệnh khác: break, continue

Các câu lệnh điều khiển

- Câu lệnh if

Cú pháp: Lệnh if có hai dạng sau

Dạng 1

if (điều kiện) câu lệnh; hoặc {khối lệnh}

Dạng 2

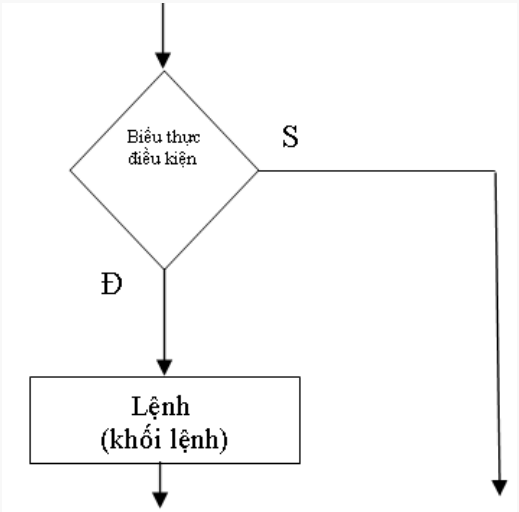
```
if (điều kiện)
        câu lệnh 1; hoặc {khối lệnh 1}
else
        câu lệnh 2; hoặc {khối lệnh 2}
```

Các câu lệnh điều khiển

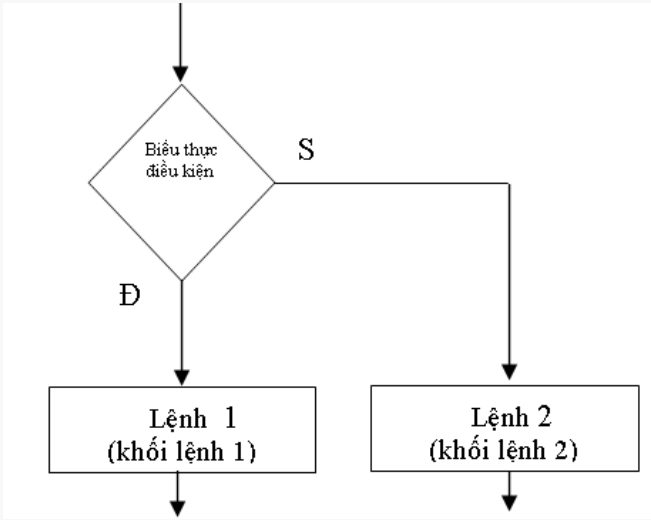
- Câu lệnh if

- Hoạt động

dạng 1



dạng 2



Các câu lệnh điều khiển

- Câu lệnh switch

Cú pháp:

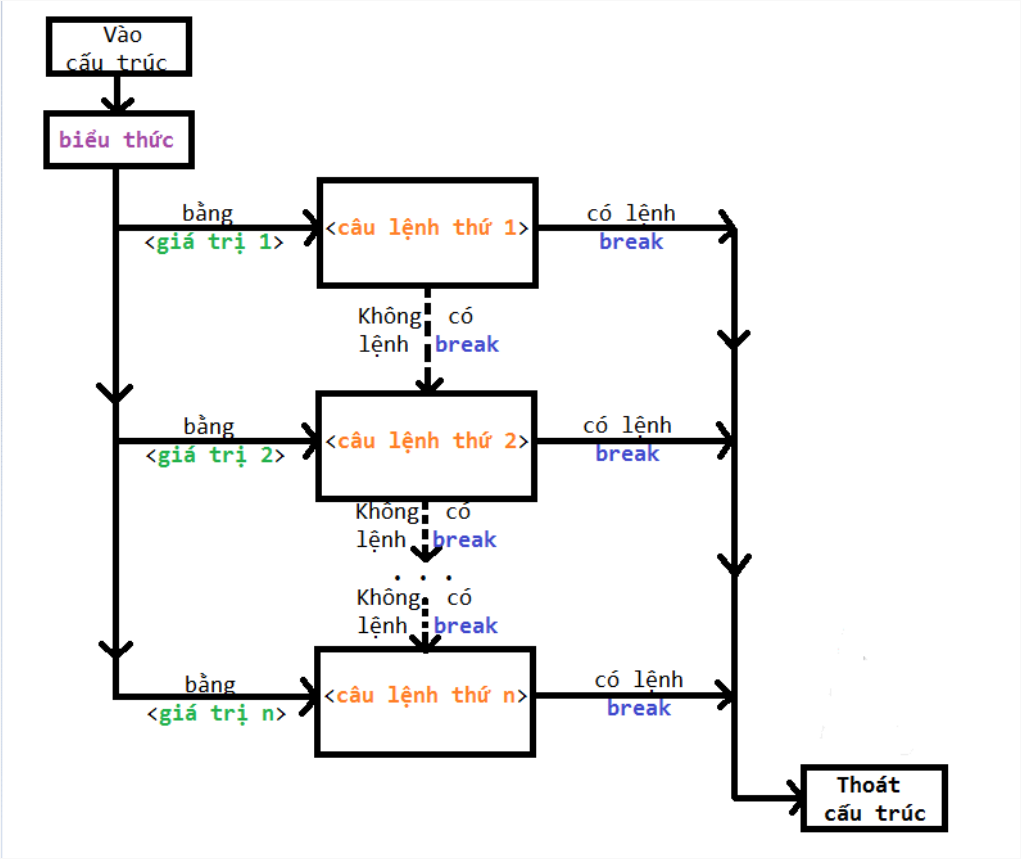
```
switch (biểu thức)
 {
 case n1:
          các câu lệnh
 case n2:
          các câu lệnh
  ..........
 case nk:
          các câu lệnh
 [default: các câu lệnh]
 }
```

- ni là các hằng số nguyên
- nếu giá trị biểu thức = ni thì thực hiện các câu lệnh sau case ni
- Nếu giá trị biểu thức khác tất cả các ni thì thực hiện các lệnh sau default nếu có, hoặc thoát khỏi switch
- sau khi thực hiện xong các câu lệnh của case ni nào đó thì sẽ thực hiện các câu lệnh thuộc case ben dưới mà không xét lại điều kiện=> muốn dừng dùng break.

Các câu lệnh điều khiển

- Câu lệnh switch

Hoạt động



Các câu lệnh điều khiển

- Câu lệnh for

- **Câu lệnh for**: Lệnh for thường dùng để giải các bài toán có tính chu trình, ví dụ như các bài toán về dãy số, về ma trận
- Cú pháp

for (bt1; bt2; bt3) **lệnh;**

Trong cú pháp trên:

for: từ khoá của lệnh for.

bt1, bt2, bt3 bắt buộc phải được trong cặp ngoặc tròn và cách nhau bởi dấu chấm phẩy. Các biểu thức này có thể vắng mặt, nhưng vẫn phải đủ các dấu chấm phẩy.

- Chú ý rằng, biểu thức ở đây có thể bao gồm nhiều biểu thức con.

Các câu lệnh điều khiển

- **Câu lệnh for:**

- Hoạt động

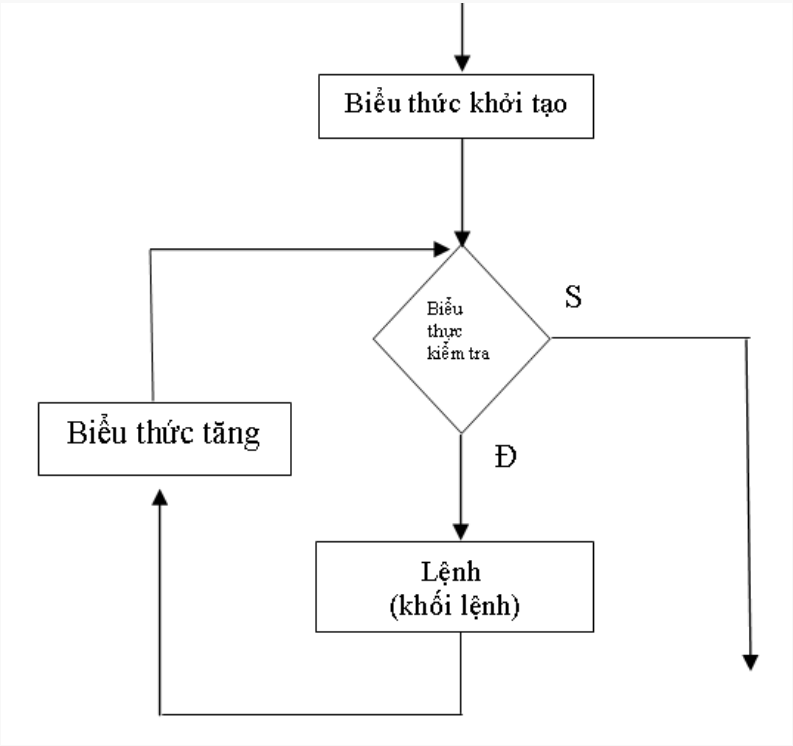
Lệnh for làm việc theo 4 bước sau:

Bước 1. Tính giá trị biểu thức khởi tạo (bt1).

Bước 2. Tính giá trị của biểu thức kiểm tra (bt2). Nếu bt2 đúng, tới bước 3, ngược lại thoát khỏi lệnh for.

Bước 3. Thực hiện lệnh.

Bước 4. Tính giá trị biểu thức tăng (bt3), sau đó quay trở lại bước 2.



Các câu lệnh điều khiển

- Câu lệnh while

- Cú pháp:

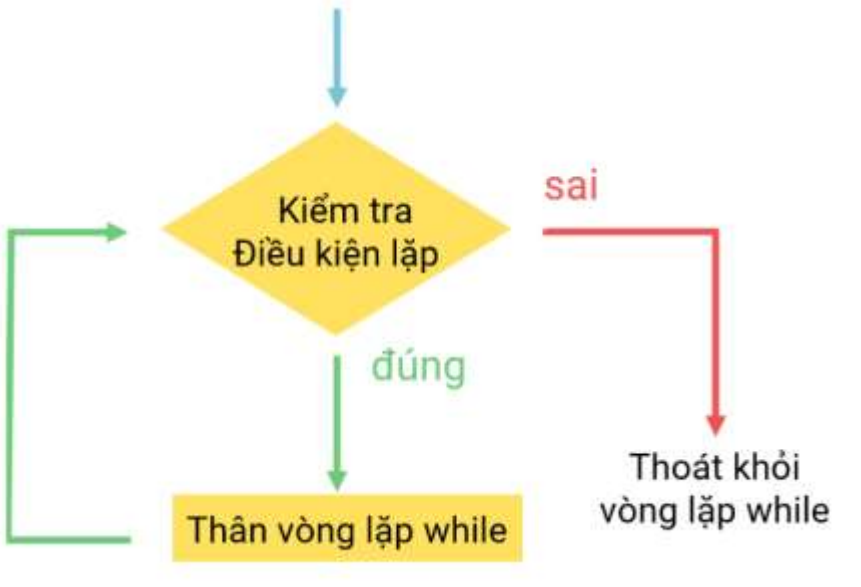
  while (biểu thức) lệnh;

trong đó while là từ khóa

- Hoạt động:

+ Bước 1: Tính giá trị của biểu thức kiểm tra. Nếu biểu thức đúng, sang bước 2, ngược lại thoát khỏi lệnh while.

+ Bước 2: Thực hiện khối lệnh, sau đó quay trở lại bước 1.



Các câu lệnh điều khiển

- Câu lệnh do while

- Cú pháp:

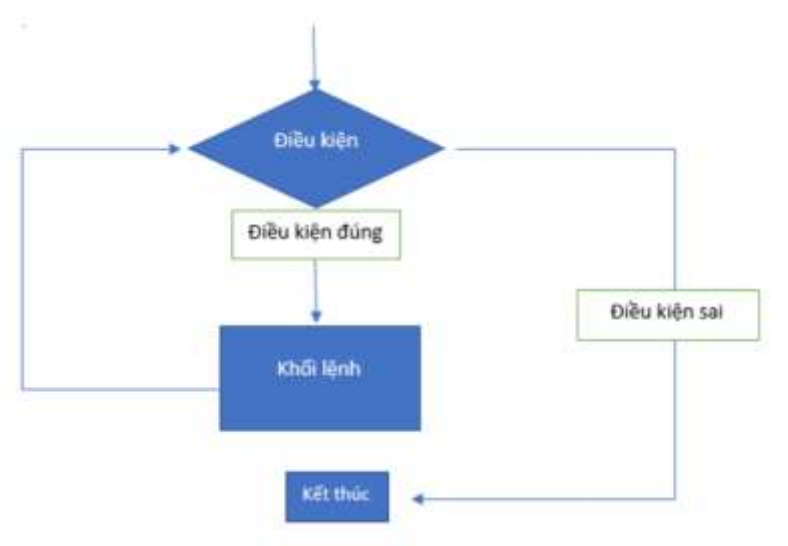
do lệnh; while (biểu thức);

trong đó: do, while là các từ khóa

- Hoạt động:

+ Bước 1: Thực hiện khối lệnh.

+ Bước 2: Tính giá trị biểu thức kiểm tra. Nếu đúng, quay trở lại bước 1, ngược lại, thoát khỏi lệnh do – while.



Các câu lệnh điều khiển

- Câu lệnh break

- Cú pháp: break;

- Hoạt động: Khi gặp break, máy sẽ bỏ qua các lệnh còn lại trong phần thân của lệnh lặp, dừng lặp và thoát ra khỏi lệnh lặp.

Ghi chú:

- Lệnh break có thể được đặt ở bất cứ đâu trong thân của các lệnh lặp.

- Khi có nhiều lệnh lặp lồng nhau, lệnh break chỉ thoát ra khỏi lệnh lặp trong cùng chứa nó.

Các câu lệnh điều khiển

- Câu lệnh continue

- Cú pháp: continue;
- Hoạt động: Khi gặp continue, máy sẽ bỏ qua các lệnh còn lại trong phần thân của lệnh lặp và quay lại bắt đầu vòng lặp mới (đối với while và do – while, vòng lặp mới bắt đầu từ bước tính biểu thức kiểm tra,đối với for, vòng lặp mới bắt đầu từ bước tính biểu thức tăng)
- Ghi chú:

    + Lệnh continue chỉ sử dụng trong các lệnh lặp.

    + Lệnh continue có thể đặt bất cứ đâu trong phần thân của lệnh lặp.

    + Khi có nhiều lệnh lặp lồng nhau, lệnh continue chỉ có tác dụng đối với lệnh lặp trong cùng

3. Gỡ rối chương trình

Các loại lỗi thông thường

- Lỗi cú pháp (Syntax errors)
- Lỗi thời gian chạy (Runtime errors)
- Lỗi logic (Logical errors)

Ví dụ	Phân loại lỗi
Console.WriteLine("Hello, World"	Lỗi cú pháp
int a = 10; int b = 0; int result = a / b;	Lỗi thời gian chạy
double area = length + width;	Lỗi logic

Công cụ gỡ lỗi trong Visual Studio

1. Điểm ngắt (Breakpoints)
2. Lệnh điều khiển thực thi (F10, F11, F5, Shift+F11)
3. Cửa sổ Watch (giám sát biến/biểu thức)
4. Cửa sổ Locals (biến cục bộ)
5. Cửa sổ Call Stack (chuỗi gọi hàm)

Ví dụ gỡ lỗi với Visual Studio

- Bước 1: đặt một điểm ngắt bằng cách nhấp vào lề màu xám bên trái của dòng mã, hoặc đặt con trỏ vào dòng đó và nhấn F9. Một chấm tròn màu đỏ sẽ xuất hiện.
- Bước 2: Nhấn F5 để chạy chương trình trong chế độ gỡ lỗi. Nhập dữ liệu và quan sát dòng tô sang màu vàng
- Bước 3: mở cửa sổ Local từ menu: Debug > Windows > Locals và quan sát giá trị các biến
  - mở cửa số Watch nếu muốn thêm biến nó từ menu: Debug > Windows > Watch > Watch 1
- Bước 4: Sử dụng phím F10 (Step Over) hoặc F11 (Step Into) để thực thi từng dòng code một và quan sát dữ liệu ở các cửa sổ
- Bước 5: Phát hiện và sửa lỗi
- Bước 6: Kiểm tra lại kết quả và xóa ngắt (nhấp lại vào chấm đỏ hoặc nhấn F9) rồi ấn F5 để chạy lại

Lời khuyên khi gỡ lỗi

- Chia nhỏ vấn đề để xử lý
- Hiển thị kết quả lên màn hình cho lỗi đơn giản
- Luôn kiểm tra các trường hợp biên
- Đọc kỹ thông báo lỗi

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

UNIVERSITY OF TRANSPORT AND COMMUNICATIONS

Trân trọng cảm ơn!