



NGÔN NGỮ LẬP TRÌNH

Bài 2: Luồng điều khiển

Giảng viên: Lý Anh Tuấn

Email: tuanla@tlu.edu.vn

Nội dung

1. Biểu thức logic

- Xây dựng, Đánh giá và các Luật Ưu tiên

2. Kỹ thuật rẽ nhánh

- if-else
- switch
- if-else lồng nhau

3. Vòng lặp

- while, do-while, for
- Vòng lặp lồng nhau

Biểu thức logic: Các toán tử so sánh

- Các toán tử logic
 - Toán tử AND (&&)
 - Toán tử OR (||)

Display 2.1 Comparison Operators

MATH SYMBOL	ENGLISH	C++ NOTATION	C++ SAMPLE	MATH EQUIVALENT
=	Equal to	==	<code>x + 7 == 2*y</code>	$x + 7 = 2y$
≠	Not equal to	!=	<code>ans != 'n'</code>	$ans \neq 'n'$
<	Less than	<	<code>count < m + 3</code>	$count < m + 3$
≤	Less than or equal to	<=	<code>time <= limit</code>	$time \leq limit$
>	Greater than	>	<code>time > limit</code>	$time > limit$
≥	Greater than or equal to	>=	<code>age >= 21</code>	$age \geq 21$

Đánh giá biểu thức logic

- Kiểu dữ liệu *bool*: trả về *true* hoặc *false*
- Bảng chân lý

Display 2.2 Truth Tables

AND

<i>Exp_1</i>	<i>Exp_2</i>	<i>Exp_1 && Exp_2</i>
true	true	true
true	false	false
false	true	false
false	false	false

OR

<i>Exp_1</i>	<i>Exp_2</i>	<i>Exp_1 Exp_2</i>
true	true	true
true	false	true
false	true	true
false	false	false

NOT

<i>Exp</i>	<i>!(Exp)</i>
true	false
false	true

Độ ưu tiên của các toán tử

Display 2.3 Precedence of Operators

::	Scope resolution operator
.	Dot operator
->	Member selection
[]	Array indexing
()	Function call
++	Postfix increment operator (placed after the variable)
--	Postfix decrement operator (placed after the variable)
++	Prefix increment operator (placed before the variable)
--	Prefix decrement operator (placed before the variable)
!	Not
-	Unary minus
+	Unary plus
*	Dereference
&	Address of
new	Create (allocate memory)
delete	Destroy (deallocate)
delete[]	Destroy array (deallocate)
sizeof	Size of object
()	Type cast

*Highest precedence
(done first)*

Độ ưu tiên của các toán tử

* / %	Multiply Divide Remainder (modulo)
+ -	Addition Subtraction
<< >>	Insertion operator (console output) Extraction operator (console input)

↓
*Lower precedence
(done later)*

Độ ưu tiên của các toán tử

Display 2.3 Precedence of Operators

All operators in part 2 are of lower precedence than those in part 1.

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal
!=	Not equal
&&	And
	Or

Độ ưu tiên của các toán tử

=	Assignment
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modulo and assign
? :	Conditional operator
throw	Throw an exception
,	Comma operator



*Lowest precedence
(done last)*

Ví dụ về độ ưu tiên

- Số học ưu tiên trước logic
 - $x+1>2 \parallel x+1<-3$ nghĩa là $(x+1)>2 \parallel (x+1)<-3$
- Đánh giá tắt
 - $(x \geq 0) \&\& (y > 1)$
 - Thận trọng với toán tử tăng: $(x > 1) \&\& (y++)$
- Giá trị logic của các số nguyên
 - Tất cả các số khác không \rightarrow true
 - Số không \rightarrow false

Kỹ thuật rẽ nhánh

- Câu lệnh if-else

- Chọn một trong hai câu lệnh dựa trên biểu thức điều kiện

- Ví dụ:

if (hrs > 40)

 grossPay = rate*40 + 1.5*rate*(hrs-40);

else

 grossPay = rate*hrs;

Cú pháp câu lệnh if-else

- Cú pháp chuẩn:
if (<biểu thức logic>
 <câu lệnh yes>
else
 <câu lệnh no>
- Lưu ý: mỗi nhánh chỉ có duy nhất một câu lệnh
- Để thực thi nhiều câu lệnh trong mỗi nhánh -> sử dụng lệnh kép
 - Nằm trong cặp dấu { }
 - Còn được gọi là khối lệnh

Ví dụ về câu lệnh kép

- Trong câu lệnh lưu ý đến việc thụt vào đầu dòng

```
if (myScore > yourScore)
{
    cout << "I win!\n";
    wager = wager + 100;
}
else
{
    cout << "I wish these were golf scores.\n";
    wager = 0;
}
```

Một số lỗi thường gặp

- Toán tử “=” và toán tử “==”
 - Toán “=” có nghĩa là phép gán
 - Toán tử “==” có nghĩa là phép so sánh bằng
- Rất khác nhau trong C++

- VD:

if (x = 12) ← Lưu ý toán tử sử dụng!

 Do_Something

else

 Do_Something_Else

Tự chọn else

- Mệnh đề *else* là tự chọn
 - Nếu bạn không muốn làm gì trong nhánh sai (*else*), thì hãy bỏ nó đi
 - Ví dụ:
if (sales >= minimum)
 salary = salary + bonus;
cout << "Salary = %" << salary;
 - Không làm gì khi điều kiện sai nên không có câu lệnh *else*
 - Tiếp tục thực hiện câu lệnh *cout*

Câu lệnh lồng nhau

- Câu lệnh if-else chứa các câu lệnh nhỏ hơn
 - Câu lệnh kép hoặc câu lệnh đơn
 - Có thể chứa bất kỳ lệnh nào, bao gồm cả câu lệnh if-else khác
 - Ví dụ:

```
if (speed > 55)
    if (speed > 80)
        cout << "You're really speeding!";
    else
        cout << "You're speeding.";
```

if-else nhiều nhánh

- Chỉ khác ở việc thụt vào đầu dòng, trong đó tránh thụt vào đầu dòng quá nhiều
 - Cú pháp:

Multiway if-else Statement

SYNTAX

```
if (Boolean_Expression_1)
    Statement_1
else if (Boolean_Expression_2)
    Statement_2
    .
    .
    .
else if (Boolean_Expression_n)
    Statement_n
else
    Statement_For_All_Other_Possibilities
```


if-else nhiều nhánh

EXAMPLE

```
if ((temperature < -10) && (day == SUNDAY))  
    cout << "Stay home.";  
else if (temperature < -10) //and day != SUNDAY  
    cout << "Stay home, but call work.";  
else if (temperature <= 0) //and temperature >= -10  
    cout << "Dress warm.";  
else //temperature > 0  
    cout << "Work hard and play hard.";
```

The Boolean expressions are checked in order until the first true Boolean expression is encountered, and then the corresponding statement is executed. If none of the Boolean expressions is true, then the *Statement_For_All_Other_Possibilities* is executed.

Câu lệnh switch

- Câu lệnh giúp kiểm soát đa nhánh
- Sử dụng biểu thức điều kiện trả về kiểu dữ liệu *bool*

Câu lệnh switch: cú pháp

switch Statement

SYNTAX

```
switch (Controlling_Expression)
{
    case Constant_1:
        Statement_Sequence_1
        break;
    case Constant_2:
        Statement_Sequence_2
        break;
        .
        .
        .
    case Constant_n:
        Statement_Sequence_n
        break;
    default:
        Default_Statement_Sequence
}
```

*You need not place a **break** statement in each case. If you omit a **break**, that case continues until a **break** (or the end of the **switch** statement) is reached.*

Câu lệnh switch: ví dụ

EXAMPLE

```
int vehicleClass;
double toll;
cout << "Enter vehicle class: ";
cin >> vehicleClass;

switch (vehicleClass)
{
    case 1:
        cout << "Passenger car.";
        toll = 0.50;
        break;
    case 2:
        cout << "Bus.";
        toll = 1.50;
        break;
    case 3:
        cout << "Truck.";
        toll = 2.00;
        break;
    default:
        cout << "Unknown vehicle class!";
}
```

*If you forget this **break**,
then passenger cars will
pay \$1.50.*

Câu lệnh switch: đa nhãn case

- Tiếp tục thực hiện cho đến khi gặp break
 - switch cung cấp một “lỗi vào”
 - Ví dụ:
case "A":
case "a":
 cout << "Excellent: you got an "A"!\n";
 break;
case "B":
case "b":
 cout << "Good: you got a "B"!\n";
 break;
 - Lưu ý rằng đa nhãn cung cấp cùng một “lỗi vào”

Câu lệnh switch: ví dụ thực đơn

- Câu lệnh switch rất thuận tiện cho việc tạo thực đơn
switch (response)
{
 case "1":
 // Execute menu option 1
 break;
 case "2":
 // Execute menu option 2
 break;
 case "3":
 // Execute menu option 3
 break;
 default:
 cout << "Please enter valid response.";
}

Toán tử điều kiện

- Cho phép nhúng điều kiện vào biểu thức
- Về cơ bản là toán tử if-else viết tắt
- Ví dụ:

```
if (n1 > n2)
    max = n1;
else
```

```
    max = n2;
```

- Có thể được viết là:

```
max = (n1 > n2) ? n1 : n2;
```

Vòng lặp

- Có ba kiểu vòng lặp trong C++
 - while
 - linh hoạt nhất
 - không bị hạn chế
 - do-while
 - kém linh hoạt nhất
 - luôn luôn thực thi thân vòng lặp ít nhất một lần
 - for
 - vòng lặp đếm tự nhiên

Vòng lặp while: cú pháp

Syntax for while and do-while Statements

A while STATEMENT WITH A SINGLE STATEMENT BODY

```
while (Boolean_Expression)  
    Statement
```

A while STATEMENT WITH A MULTISTATEMENT BODY

```
while (Boolean_Expression)  
{  
    Statement_1  
    Statement_2  
    .  
    .  
    .  
    Statement_Last  
}
```

Vòng lặp while: ví dụ

- Xét:

```
count = 0;           // Initialization
while (count < 3)     // Loop Condition
{
    cout << "Hi ";    // Loop Body
    count++;           // Update expression
}
```

- Thân vòng lặp sẽ thực thi bao nhiêu lần?

Vòng lặp do-while: cú pháp

A do-while STATEMENT WITH A SINGLE-STATEMENT BODY

```
do  
    Statement  
while (Boolean_Expression);
```

A do-while STATEMENT WITH A MULTISTatement BODY

```
do  
{  
    Statement_1  
    Statement_2  
    .  
    .  
    .  
    Statement_Last  
} while (Boolean_Expression);
```

*Do not forget
the final
semicolon.*

Vòng lặp do-while: ví dụ

- ```
count = 0; // Initialization
do
{
 cout << "Hi "; // Loop Body
 count++; // Update expression
} while (count < 3); // Loop Condition
```
- Thân vòng lặp sẽ thực thi bao nhiêu lần ?
- Vòng lặp do-while luôn luôn thực thi ít nhất một lần !

# while và do-while

- Rất giống nhau, nhưng có một khác biệt quan trọng về thời điểm kiểm tra biểu thức logic:
  - while: kiểm tra trước khi thực thi thân vòng lặp
  - do-while: kiểm tra sau khi thực thi thân vòng lặp
- while là thông dụng hơn, do tính linh hoạt không hạn chế của nó

# Toán tử phẩy

- Đánh giá danh sách biểu thức, trả về giá trị của biểu thức cuối cùng
- Thường được sử dụng trong vòng lặp for
- Ví dụ:  
$$\text{first} = (\text{first} = 2, \text{second} = \text{first} + 1);$$
  - *first* được gán giá trị 3
  - *second* được gán giá trị 3
- Không đảm bảo trật tự đánh giá các biểu thức

# Vòng lặp for: cú pháp

- Cú pháp:  
for (Init\_Action; Bool\_Exp; Update\_Action)  
    Body\_Statement
- Giống như if-else, Body\_Statement có thể là một khối lệnh

# Vòng lặp for: ví dụ

- Ví dụ:

```
for (count=0;count<3;count++)
{
 cout << "Hi "; // Loop Body
}
```

- Thân vòng lặp sẽ thực thi bao nhiêu lần?
- Cấu trúc vòng lặp for: Khởi tạo, điều kiện lặp và cập nhật
- Vòng lặp đếm tự nhiên



# Điều kiện lặp

- Biểu thức điều kiện của vòng lặp có thể là bất kỳ biểu thức logic nào
- Ví dụ:

```
while (count<3 && done!=0)
```

```
{
```

```
 // Do something
```

```
}
```

```
for (index=0;index<10 && entry!=-99)
```

```
{
```

```
 // Do something
```

```
}
```

# Một số lỗi thường gặp

- Lưu ý việc đặt sai dấu ;  

```
while (response != 0) ;←
{
 cout << "Enter val: ";
 cin >> response;
}
```

  - Kết quả là tạo ra vòng lặp vô hạn
- Điều kiện lặp phải được đánh giá là sai ở một bước lặp nào đó, nếu không sẽ tạo ra vòng lặp vô hạn  

```
while (1)
{
 cout << "Hello ";
}
```

  - Một vòng lặp luôn điều kiện luôn đúng → lặp vô hạn

# Câu lệnh break và continue

- Trong một số trường hợp có thể sửa đổi luồng tự nhiên
- break;
  - Buộc thoát khỏi vòng lặp ngay lập tức
- continue;
  - Bỏ qua phần còn lại của thân vòng lặp
- Các câu lệnh nay vi phạm luồng tự nhiên, nên chỉ sử dụng khi thật sự cần thiết

# Vòng lặp lồng nhau

- Bất cứ câu lệnh C++ đúng nào cũng có thể nằm trong thân vòng lặp
- Bao gồm cả các câu lệnh lặp khác được gọi là vòng lặp lồng nhau
- Yêu cầu thụt vào đầu dòng:

```
for (outer=0; outer<5; outer++)
 for (inner=7; inner>2; inner--)
 cout << outer << inner;
```

- Không có { } vì mỗi thân vòng lặp chỉ có một lệnh
- Tuy nhiên vẫn có thể sử dụng { } như thường

# Tóm tắt

- Các biểu thức logic: tương tự biểu thức số học nhưng cho kết quả là true hoặc false
- Các câu lệnh rẽ nhánh C++
  - if-else, switch
  - câu lệnh switch tiện lợi cho việc tạo thực đơn
- Các câu lệnh lặp C++
  - while
  - do-while
  - for

# Tóm tắt

- Vòng lặp do-while
  - Luôn luôn thực thi thân vòng lặp của nó ít nhất một lần
- Vòng lặp for
  - Một vòng lặp đếm tự nhiên
- Vòng lặp có thể thoát sớm
  - câu lệnh break
  - câu lệnh continue
  - sử dụng hạn chế