Tin học cơ sở 4

Buổi 3. Luồng điều khiển

Lệnh rẽ nhánh

Bộ môn Khoa học máy tính - 2017

Nội dung buổi học

1. Khái niệm luồng điều khiển
   - ❖ Tuần tự, rẽ nhánh
2. Lệnh if, lệnh if-else, lệnh switch
3. Biểu thức điều kiện
   - ❖ Kiểu bool (lô-gic)

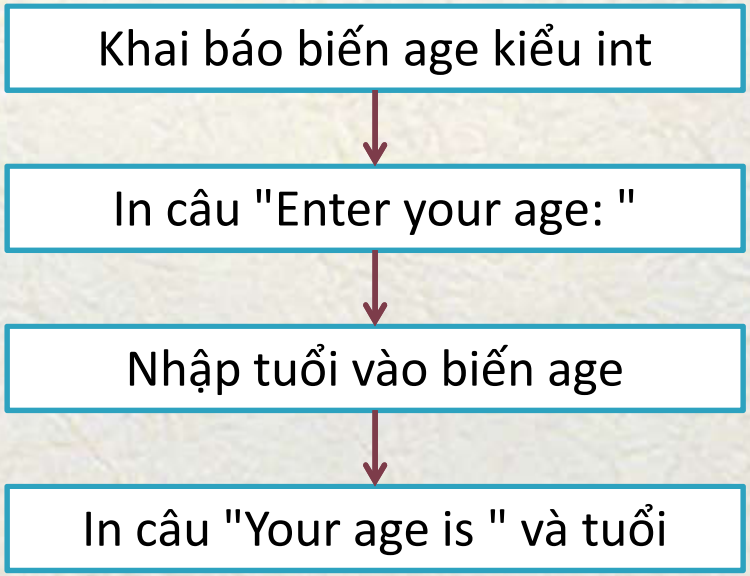
Luồng điều khiển

- Thứ tự chạy các lệnh trong chương trình
- Ví dụ:

```
int age;
cout << "Enter your age: ";
cin >> age;
cout << "Your age is " << age << endl;
```

Luồng điều khiển

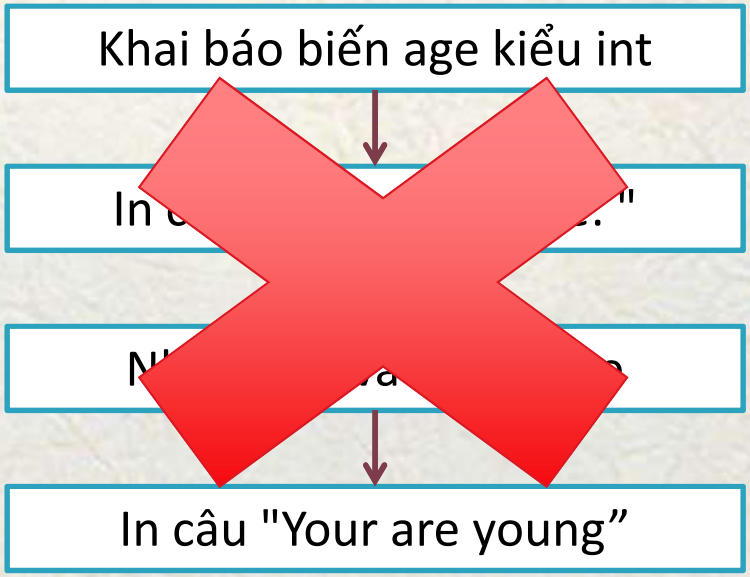
- Thứ tự chạy các lệnh trong chương trình
- Ví dụ:



Các câu lệnh được gọi lần lượt, tuần tự

Luồng điều khiển

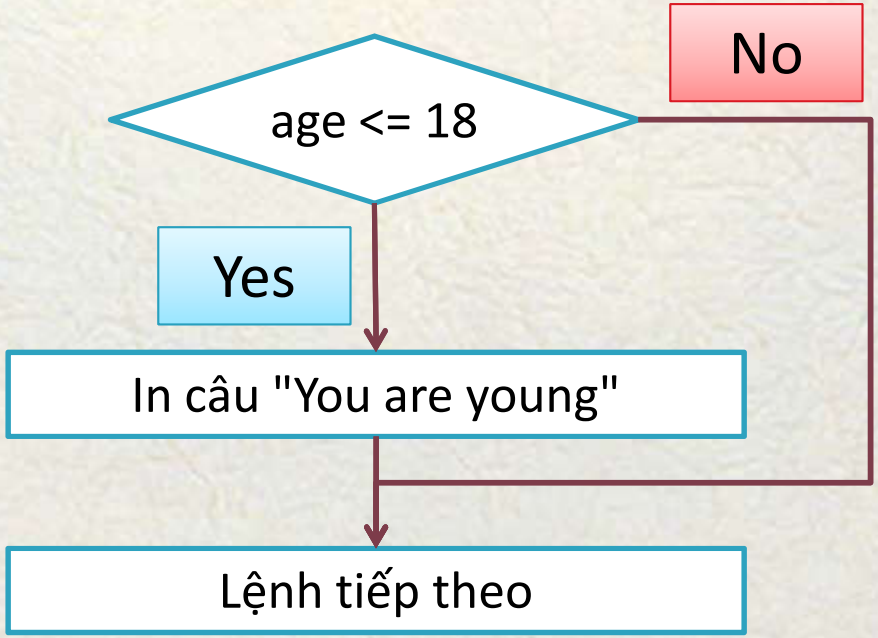
- Giả sử cần in câu “You are young” khi tuổi không lớn hơn 18
- Ví dụ:



Các câu lệnh được gọi lần lượt, tuần tự

Luồng điều khiển

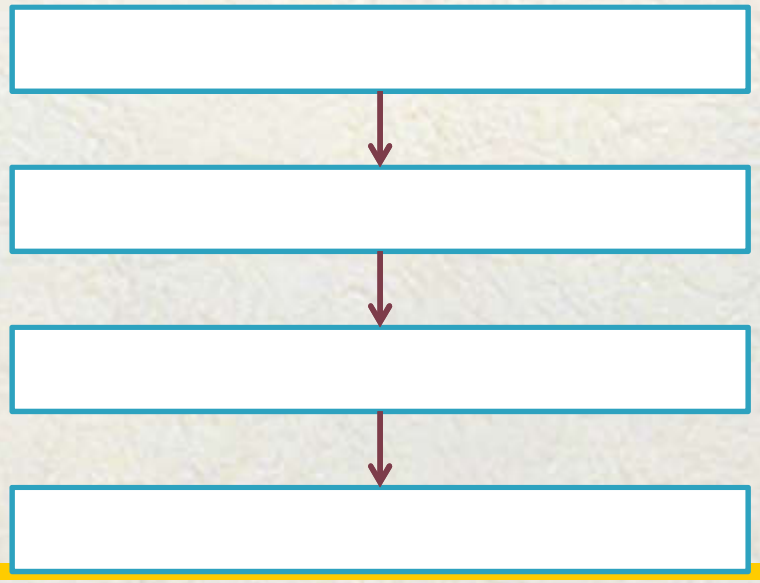
- Giả sử cần in câu “You are young” khi tuổi không lớn hơn 18



Chương trình lựa chọn nhánh để thực hiện tùy vào điều kiện đúng hay sai

Cấu trúc điều khiển tuần tự

- Các lệnh lần lượt được *chuyển quyền điều khiển* để thực hiện
  - Lệnh viết trước thực hiện trước



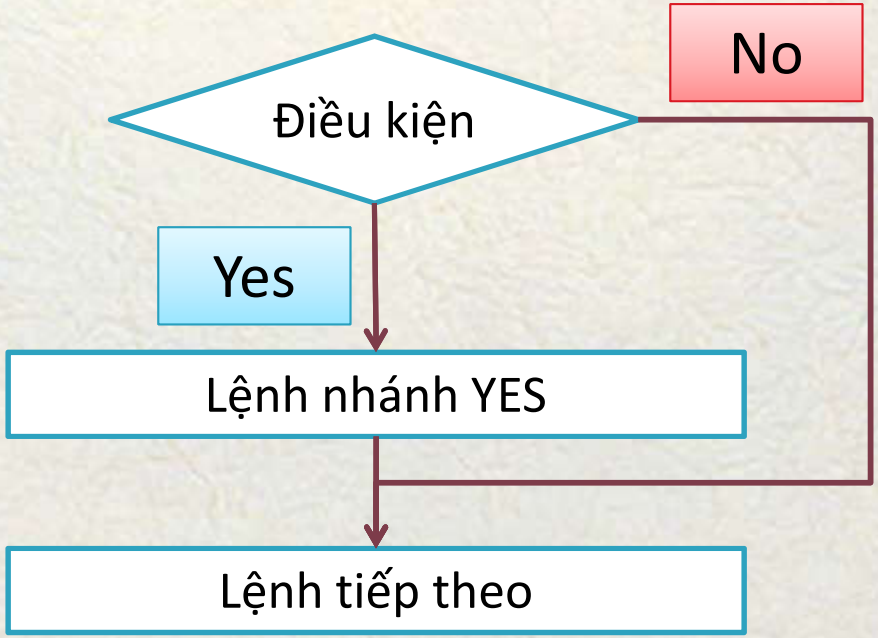
Các câu lệnh được gọi lần lượt, tuần tự

Cấu trúc điều khiển rẽ nhánh

- Chương trình lựa chọn nhánh lệnh để chuyển quyền điều khiển
  - Điều kiện đúng thì thực hiện nhánh YES
  - Điều kiện sai thì thực hiện nhánh NO

Cấu trúc điều khiển rẽ nhánh

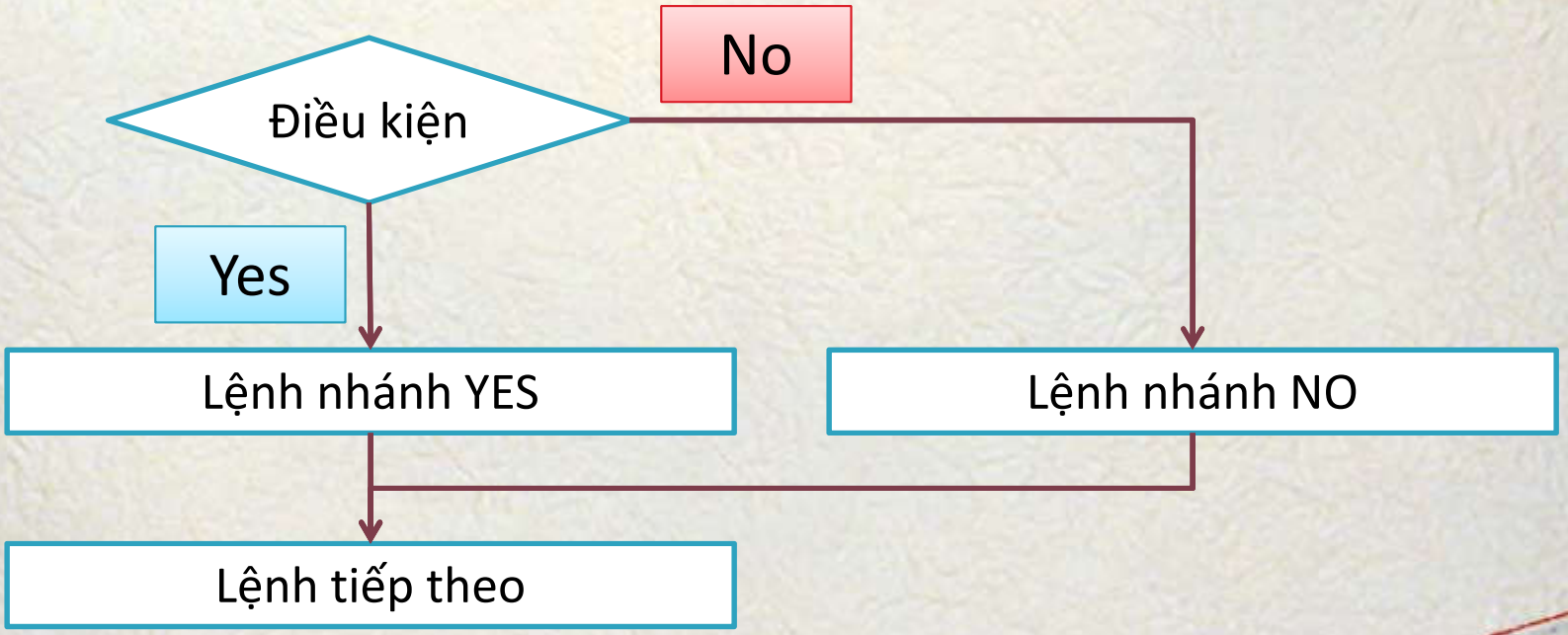
- Chương trình lựa chọn nhánh lệnh để chuyển quyền điều khiển



Chương trình lựa chọn nhánh để thực hiện tùy vào điều kiện đúng hay sai

Cấu trúc điều khiển rẽ nhánh

- Chương trình lựa chọn nhánh lệnh để chuyển quyền điều khiển



CT1. Kiểm tra tuổi trẻ

Mục tiêu: In ra câu “You are young” nếu tuổi không lớn hơn 18

```
int age;
cout << "Enter your age: ";
cin >> age;
if (age <= 18) {
    cout << "You are young." << endl;
}
```

Lệnh if

- Cú pháp

```
if (biểu thức điều kiện) {
    Dãy các lệnh trong nhánh YES
    khi điều kiện đúng
}
```

Phải có đóng mở ngoặc tròn

Dùng đóng mở ngoặc nhọn để bao khối nhiều lệnh

Ví dụ lệnh if

```
int absVal = x;
if (absVal < 0) {
    absVal = -absVal;
}
```

Tìm giá trị tuyệt đối

```
double taxRate = 0.10;
if (income < povertyIncome) {
    taxRate = 0;
}
```

Tính thuế suất thu nhập

```
int min = first, max = second;
if (first > second) {
    min = second;
    max = first;
}
```

Tìm số nhỏ nhất và số lớn nhất

Lệnh if-else

- Cú pháp

```
if (biểu thức điều kiện) {
    Dãy các lệnh trong nhánh YES
    khi điều kiện đúng
} else {
    Dãy các lệnh trong nhánh NO
    khi điều kiện sai
}
```

Ví dụ lệnh if-else

```
int absVal;
if (x < 0) {
    absVal = -x;
} else {
    absVal = x;
}
```

Tìm giá trị tuyệt đối

```
int min, max;
if (first > second) {
    min = second;
    max = first;
} else {
    min = first;
    max = second;
}
```

Tìm số nhỏ nhất và số lớn nhất

Biểu thức điều kiện

- Biểu thức có 2 giá trị: **true** (đúng) và **false** (sai)
  - Các phép so sánh
  - Các điều kiện phức hợp sử dụng toán tử **&&** (AND), **||** (OR), **!** (NOT)
- Kiểu **bool**: kiểu gồm 2 giá trị **true** và **false**

Các phép so sánh

==	**true** khi và chỉ khi toán hạng trái bằng toán hạng phải
!=	**true** khi và chỉ khi toán hạng trái không bằng toán hạng phải
>	**true** khi và chỉ khi toán hạng trái lớn hơn toán hạng phải
<	**true** khi và chỉ khi toán hạng trái nhỏ hơn toán hạng phải
>=	**true** khi và chỉ khi toán hạng trái lớn hơn hoặc bằng toán hạng phải
<=	**true** khi và chỉ khi toán hạng trái nhỏ hơn hoặc bằng toán hạng phải

Các phép so sánh

```
if (price <= 0)
{
   // Error condition
}

if (price > average_price)
{
   // Apply some discount
}
```

```
if (age < 18)
{
   // Error message - not
   // allowed to buy alcohol
}

if (age >= 65)
{
  // Apply seniors' discount
}
```

CT2. Giải phương trình bậc 1

- Các lệnh if-else lồng nhau

```
double a, b; // ax + b = 0
cin >> a >> b;

if (a == 0) {
    if (b == 0) {
        cout << "Infinitely many solutions\n";
    } else {
        cout << "No solution\n";
    }
} else {
    double x = -b / a;
    cout << "Solution x = " << x << endl;
}
```

CT3. Giải phương trình bậc 2

```
double a, b, c; // ax^2 + bx + c = 0
cin >> a >> b >> c;

if (a == 0) {
    solveLinear(b, c);
} else {
    double delta = b*b - 4*a*c;
    if (delta < 0) {
        cout << "No solution\n";
    } else if (delta == 0) {
        cout << "1 solution x = " << -b / (2*a) << endl;
    } else { // delta > 0
        cout << "2 solutions x1 = " << -b + sqrt(delta) / (2*a)
             << " x2 = " << -b - sqrt(delta) / (2*a) << endl;
    }
}
```

Phép toán && (AND)

- **true** khi và chỉ khi cả hai toán hạng điều kiện có giá trị true

&& (AND)	true	false
true	true	false
false	false	false

- 3 ≤ x ≤ 10 → (3 <= x && x <= 10)

Phép toán || (OR)

- **true** khi và chỉ khi ít nhất một trong hai toán hạng điều kiện có giá trị true

|| (OR)	true	false
true	true	true
false	true	false

- $x \notin [3,10] \rightarrow (x < 3 \,||\, x > 10)$

Phép toán ! (NOT)

- **true** khi và chỉ khi toán hạng điều kiện có giá trị false

|| (OR)	true	false
	false	true

- x không chia hết cho 3

→ (!(x % 3 == 0))

Thứ tự ưu tiên

1	`::`	Scope resolution	Left-to-right
2	`a++ a--`	Suffix/postfix increment and decrement	
	`type``() `*`type`*`{}`	Functional cast	
	`a()`	Function call	
	`a[]`	Subscript	
	`. ->`	Member access	
3	`++a --a`	Prefix increment and decrement	Right-to-left
	`+a -a`	Unary plus and minus	
	`! ~`	Logical NOT and bitwise NOT	
	(*`type`*)	C-style cast	
	`*a`	Indirection (dereference)	
	`&a`	Address-of	
	`sizeof`	Size-of[note 1]	
	`new new[]`	Dynamic memory allocation	
	`delete delete[]`	Dynamic memory deallocation	

4	`.*  ->*`	Pointer-to-member	Left-to-right
5	`a*b  a/b  a%b`	Multiplication, division, and remainder	
6	`a+b  a-b`	Addition and subtraction	
7	`<<  >>`	Bitwise left shift and right shift	
8	`<  <=` `>  >=`	For relational operators < and ≤ respectively For relational operators > and ≥ respectively	
9	`==  !=`	For relational operators = and ≠ respectively	
10	`a&b`	Bitwise AND	
11	`^`	Bitwise XOR (exclusive or)	
12	`|`	Bitwise OR (inclusive or)	
13	`&&`	Logical AND	
14	`||`	Logical OR	

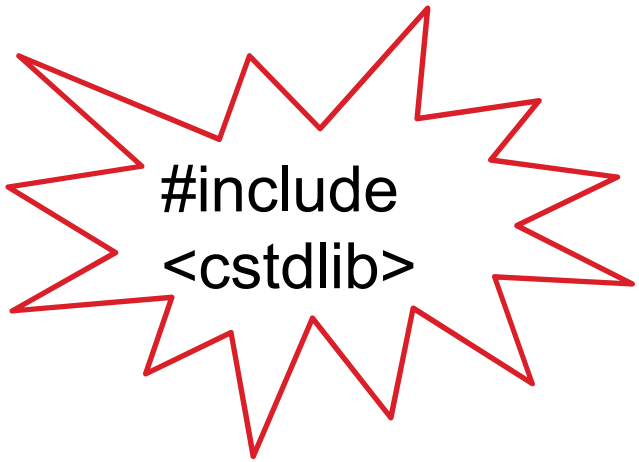
15	`a?b:c`
	`throw`
	`=`
	`+=  -=`
	`*=  /=  %=`
	`<<=  >>=`
	`&=  ^=  |=`
		Ternary conditional[note 2]
		throw operator
		Direct assignment (provided by default for C++ classes)
		Compound assignment by sum and difference
		Compound assignment by product, quotient, and remainder
		Compound assignment by bitwise left shift and right shift
		Compound assignment by bitwise AND, XOR, and OR
			Right-to-left

16	`,`	Comma	Left-to-right

CT4. Tính xác suất

Tung xúc sắc 1024 lần số ngẫu nhiên từ 1 đến 6, đếm số lần được 1 hoặc 6 điểm

```
int rollADice()
{
    int r = rand() % 6 + 1;
    return r;
}
int nextCount(int count)
{
    int r = rollADice();
    if (r == 1 || r == 6) {
        count++;
    }
    return count;
}
```



CT4. Tính xác suất

```
int next4Count(int count)
{
    count = nextCount(count);
    count = nextCount(count);
    count = nextCount(count);
    count = nextCount(count);
    return count;
}

int next16Count(int count)
{
    count = next4Count(count);
    count = next4Count(count);
    count = next4Count(count);
    count = next4Count(count);
    return count;
}
```

```
int next64Count(int count)
{
    count = next16Count(count);
    count = next16Count(count);
    count = next16Count(count);
    count = next16Count(count);
    return count;
}
```

```
int next256Count(int count)
{
    count = next64Count(count);
    count = next64Count(count);
    count = next64Count(count);
    count = next64Count(count);
    return count;
}

int next1024Count(int count)
{
    count = next256Count(count);
    count = next256Count(count);
    count = next256Count(count);
    count = next256Count(count);
    return count;
}
```

CT4. Tính xác suất

Tung xúc sắc 1024 lần số ngẫu nhiên từ 1 đến 6, đếm số lần được 1 hoặc 6 điểm

```
int main()
{
    int count = 0;
    count = next1024Count(count);
    double probability = count / 1024.0;
    cout << "Probability: "
         << probability << endl;
    return 0;
}
```

CT5. Ném bi chạy ngang, va chạm với thành sân

- Bi chạy ngang dọc theo trục Ox với vận tốc velocity (có thể âm hoặc dương)
- Mỗi bước mất 0.5 giây
- Bi va chạm với thành sân thì bắn ngược trở lại
- Kích thước sân nằm trong khoảng từ 0 đến 100

```
void display(double position)
{
    cout << "position = " << position << " (m)" << endl;
}

double step(double position, double velocity,
            double timeStep)
{
    double newPos = position + velocity * timeStep;
    display(newPos);
    return newPos;
}

double stepVelocity(double position, double velocity)
{
    if (position <= 0 || position >= 100) {
        velocity = -velocity;
    }
    return velocity;
}
```

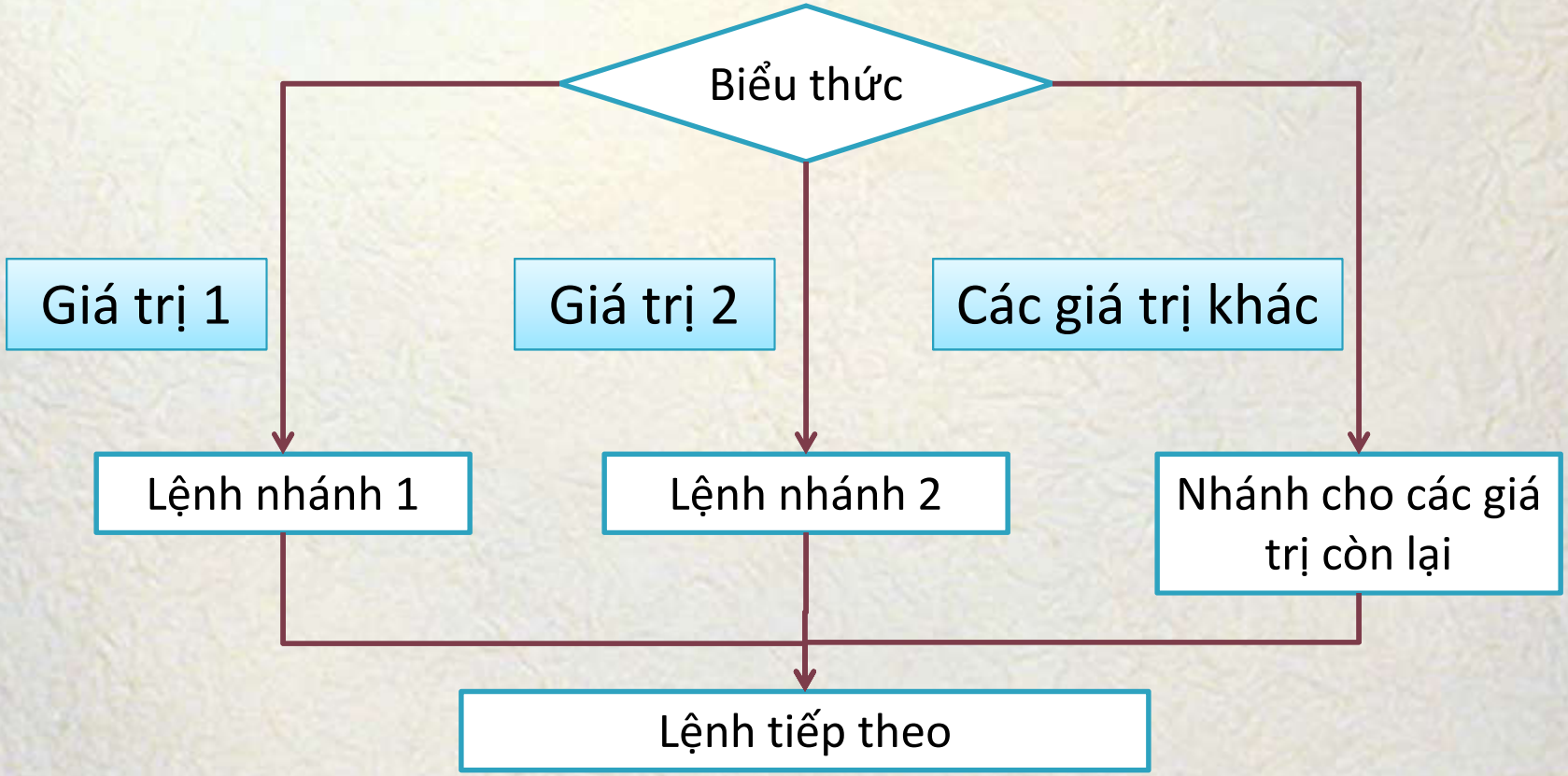
Lệnh switch

➢ Cú pháp

```
switch (biểu thức có giá trị số, kí tự, bool) {
case <giá trị 1>:
    Dãy các lệnh khi biểu thức có giá trị 1;
    break;
case <giá trị 2>:
    Dãy các lệnh khi biểu thức có giá trị 2;
    break;
…. // thêm các trường hợp khác
default:
    Dãy các lệnh khi biểu thức có các giá trị khác;
}
```

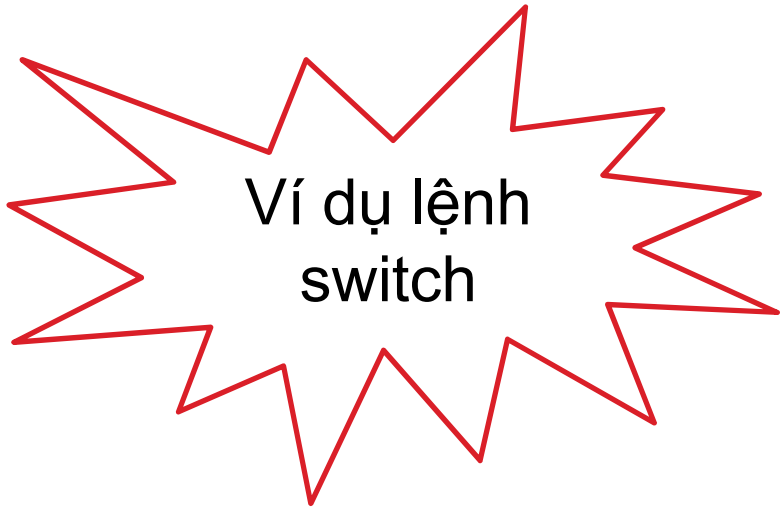
Dùng lệnh break để phân tách các trường hợp
Nếu không có break, quyền điều khiển sẽ có dạng tuần tự (lần lượt thực hiện các lệnh)

Lệnh switch


Biểu thức
Giá trị 1
Giá trị 2
Các giá trị khác
Lệnh nhánh 1
Lệnh nhánh 2
Nhánh cho các giá trị còn lại
Lệnh tiếp theo

```
int score;
cin >> score;

switch (score) {
case 0: case 1: case 2: case 3: case 4:
    cout << "Failed" << endl;
    break;
case 5:
case 6:
    cout << "Passed" << endl;
    break;
case 7:
case 8:
    cout << "Good" << endl;
    break;
case 9:
    cout << "Excellent" << endl;
    break;
case 10:
    cout << "Perfect" << endl;
    break;
default:
    cout << "Illegal score" << endl;
}
```



Câu hỏi

- Về nội dung buổi học ?
- Về nội dung khóa học ?
- Về cách tổ chức ?