Bài 6

Lập trình C
Các lệnh

Nội dung

- Các lệnh lựa chọn if và switch
- Các lệnh vòng lặp while, do while, for
- Các lệnh nhảy break, continue, return
- Các lệnh biểu thức
- Các lệnh khối

Các lệnh lựa chọn

- Lệnh lựa chọn dùng để viết mã chương trình cho cấu trúc điều khiển lựa chọn. Ngôn ngữ C hổ trợ 2 lệnh lựa chọn là
  - Lệnh **if**
  - Lệnh **switch**
- Ngoài ra, một số dạng nhất định của lệnh if có thể được thay bằng **phép toán ?**

Lệnh if

- Dạng chung của lệnh if là

  if (<biểu thức>) <lệnh>;
  else <lệnh>;

- Trong đó ***<lệnh>*** có thể là một câu lệnh đơn, một khối lệnh hoặc một câu lệnh rỗng.
- Mệnh đề ***else*** có thể có hoặc không.
- Nếu ***<biểu thức>*** có giá trị khác 0 (ĐÚNG) thì ***<lệnh>*** sau mệnh đề if được thi hành, ngược lại ***<lệnh>*** sau mệnh đề else được thi hành nếu nó tồn tại.

Lệnh if lồng nhau (1)

- **Lệnh if lồng nhau** là lệnh if mà sau mệnh đề if hoặc else của nó có lệnh if khác.
- Trong một lệnh if lồng nhau, một mệnh đề else là của một lệnh if gần nhất nằm trong cùng khối với else và chưa được liên kết với một else nào.

```
if (i)
{
    if (j) dosomething1();
    if (k) dosomething2();  // this is
    else dosomething3();    // is associated with this else
}
else
    dosomething4();         // associated with if (i)
```

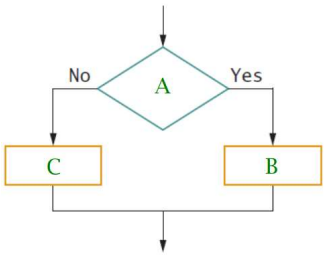

Lệnh if lồng nhau (2)

- Một dạng if lồng nhau phổ biến trong lập trình là **dạng bậc thang if - else - if**

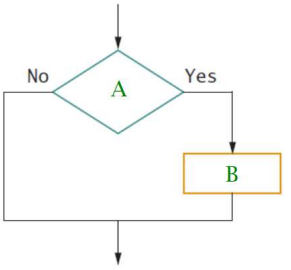
```
if (<biểu thức>) <lệnh>;
else
  if (<biểu thức>) <lệnh>;
  else
    if (<biểu thức>) <lệnh>;
    .
    .
    .
    else <lệnh>;
```

```
if (<biểu thức>)
  <lệnh>;
else if (<biểu thức>)
  <lệnh>;
else if (<biểu thức>)
  <lệnh>;
.
.
.
else
  <lệnh>;
```

Cấu trúc lựa chọn → Lệnh if (1)

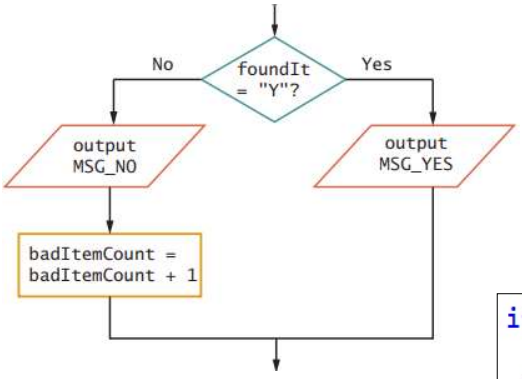



```
if (A)
   B;
else
   C;
```




```
if (A)
   B;
```


Cấu trúc lựa chọn → Lệnh if (2)




```
if (foundIt == 'Y')
    printf("%s", MSG_YES);
else
{
    printf("%s", MSG_NO);
    badItemCount++;
}
```

Phép toán lựa chọn

- **Phép toán ?** có thể được sử dụng thay cho lệnh if có dạng
  if (<biểu thức 1>) var = <biểu thức 2>;
  else var = <biểu thức 3>;
- Là một phép toán ba ngôi vì nó yêu cầu 3 toán hạng và có dạng
  (<biểu thức 1>) ? (<biểu thức 2>) : (<biểu thức 3>)

```
x = 10;
y = (x > 9) ? 100 : 200;

// equivalent to the following code

x = 10;
if (x > 9) y = 100;
else y = 200;
```


Lệnh switch (1)

- Lệnh lựa chọn nhiều nhánh switch có dạng

```
switch (<biểu thức>) {
  case <hằng 1>:
    <chuỗi lệnh>
    break;
  case <hằng 2>:
    <chuỗi lệnh>
    break;
  .
  .
  .
  default
    <chuỗi lệnh>
}
```

- Trong đó, ***<biểu thức>*** và các ***<hằng>*** có kiểu int hoặc char. **Không chấp nhận kiểu float và double.**
- Mệnh đề ***default*** có thể có hoặc không.
- Lệnh ***break*** sau ***<chuỗi lệnh>*** cuối cùng không cần thiết.

Lệnh switch (2)

- Đặc điểm của lệnh switch:
  - Cách kiểm tra điều kiện của switch là so sánh bằng.
  - Không cho phép nhiều ***case*** có ***<hằng>*** giống nhau trong cùng một lệnh switch.
  - Nếu ***<hằng>*** là kí tự thì nó sẽ được tự động chuyển kiểu thành số nguyên.
- Lệnh switch có thể được lồng bên trong một lệnh switch khác.
- Trong trường hợp thực thi xong ***<chuỗi lệnh>*** của một ***case*** mà không gặp lệnh ***break*** thì ***<chuỗi lệnh>*** của nhánh kế tiếp sẽ tiếp tục được thực thi.

Lệnh switch (3)

```
int flag, n;

flag = -1;
scanf("%d", &n);
switch (n)
{
    case 1:     // These cases have common
    case 2:     // statement sequences.
    case 3:
        flag = 3;
        break;
    case 4:
        flag = 1;
    case 5:
        error(flag);
        break;
    default:
        process(n);
}
```

Các lệnh vòng lặp

- Các lệnh vòng lặp cho phép tập hợp các chỉ thị được thực hiện lặp đi lặp lại cho đến khi một điều kiện nào đó được thỏa.
- Ngôn ngữ C cung cấp các lệnh vòng lặp cho dạng
  - Điều kiện kết thúc mở
    - Lệnh **while**
    - Lệnh **do while**
  - Điều kiện xác định trước
    - Lệnh **for**

Lệnh while (1)

- Dạng chung của lệnh while là

  while (<biểu thức>) <lệnh>;

- Vòng lặp while thực hiện lặp lại ***<lệnh>*** trong khi ***<biểu thức>*** có giá trị khác 0 (ĐÚNG).
- Khi ***<biểu thức>*** có giá trị là 0 (SAI) thì điều khiển chương trình sẽ chuyển đến dòng mã chương trình ngay sau lệnh while.

Lệnh while (2)

- ***<lệnh>*** có thể là một lệnh rỗng, một lệnh đơn hoặc một khối.

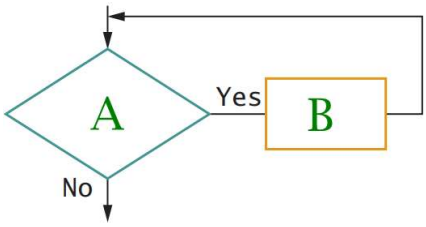
```
char ch;
int i;

// while loop with an empty statement
while ((ch = getchar()) != 'A');

// while loop with a single statement
ch = getchar();
while (ch != 'A') ch = getchar();

// while loop with a block of statements
i = 0;
while (i < 10)
{
    printf("i = %d\n", i);
    i++;
}
```


Cấu trúc vòng lặp while → Lệnh while




```
while (A) B;
```

Lệnh do while

- Dạng chung của lệnh do while là

  do
  <lệnh>;
  while (<biểu thức>);

- Vòng lặp do-while thực hiện lặp lại ***<lệnh>*** trong khi ***<biểu thức>*** có giá trị khác 0 (ĐÚNG).
- Khi ***<biểu thức>*** có giá trị là không (SAI) thì điều khiển chương trình sẽ chuyển đến dòng mã chương trình ngay sau lệnh do while.
- ***<lệnh>*** có thể là một lệnh rỗng, một lệnh đơn hoặc một khối và được thực thi ít nhất một lần.

Lệnh for (1)

- Dạng chung của lệnh for là

  for (<khởi tạo>; <điều kiện>; <gia tăng>) <lệnh>;

- Trong đó
  - ***<khởi tạo>*** là một phép gán dùng để đặt biến điều khiển vòng lặp.
  - ***<điều kiện>*** là một biểu thức quan hệ xác định thời điểm dừng vòng lặp.
  - ***<gia tăng>*** là một biểu thức thay đổi (thường là tăng) giá trị của biến điều khiển mỗi khi vòng lặp được lặp lại.
  - ***<lệnh>*** có thể là một lệnh rỗng, một lệnh đơn hoặc một khối.
- Vòng lặp for thực hiện lặp lại ***<lệnh>*** trong khi ***<điều kiện>*** có giá trị đúng. Khi ***<điều kiện>*** có giá trị sai thì điều khiển chương trình sẽ chuyển đến dòng mã chương trình ngay sau lệnh for.

Lệnh for (2)

- Các dạng phổ biến của lệnh for

```
const int DELAY_INTERVAL = 60;
int x, z;

// Time delay loop
for (x = 0; x < DELAY_INTERVAL; x++);

// The loop prints values from 1 to 100
for (x = 1; x <= 100; x++) printf("%d ", x);

// The loop prints x and its square until 65
for (x = 100; x != 65; x -= 5)
{
    z = x * x;
    printf("The square of %d is %d\n", x, z);
}
```


Lệnh for (3)

- Sử dụng phép toán dấu phẩy để **cho phép hai hay nhiều biến điều khiển vòng lặp**.

```
int x;
char y;

// Using two variables x and y to control the loops
for (x = 0, y = 0; x + y < 10; x++)
{
    y = getchar();
    y = y - '0';    // subtract the ASCII code for 0 from y
}

for (x = 0, y = 10; x <= y; x++, y--)
{
    printf("x = %d, y = %d\n", x, y);
}
```

Lệnh for (4)

- ***<điều kiện>*** không nhất thiết là một biểu thức kiểm tra biến điều khiển vòng lặp. Nó **có thể là một biểu thức có giá trị là 0 hoặc khác 0 phù hợp với một logic nào đó**.
- ***<khởi tạo>*** và ***<gia tăng>*** có thể là lệnh gọi hàm.

```
char s[] = "0123456789";
int i;

// The loop prints elements of the array s.
for (i = 0; s[i]; i++) printf("The %dth element of the array is: %c\n", i, s[i]);

float x;

// The loop takes in a number and prints its square until the input is a character.
for (printf("Enter a number = "); scanf("%f", &x); printf("\nEnter a number = "))
{
    printf("The square of %f is %f", x, x * x);
}
```


Lệnh for (5)

- **Các thành phần** của lệnh for **không nhất thiết phải tồn tại**, chúng có thể được bỏ trống.

```
int x;

/* The loop takes in an integer until the input is 0.
   Notice: <increment> is blank. */
for (x = 1; x != 0; )
{
    printf("Enter an integer (0 to stop): ");
    scanf("%d", &x);
}

/* The loop takes in an integer until the input is a character.
   Notice: <initialization>, and <increment> are blank. */
for (; printf("Enter an integer (a character to stop): "), scanf("%d", &x); );

/* An infinite loop.
   Notice: <initialization>, <condition>, and <increment> are balnk. */
for (; ; )
{
    printf("I'm in an infinite loop. Press any key to stop.\n");
    if (kbhit()) break;     // kbhit() is a function in "conio.h"
}
```

Các lệnh nhảy

- Ngôn ngữ C cung cấp 4 lệnh nhảy để thực hiện rẽ nhánh không điều kiện
  - Lệnh **break**.
  - Lệnh **continue**.
  - Lệnh **return**.
  - Lệnh **goto**.
- Lệnh return, goto sử dụng trong các hàm.
- Lệnh break, continue kết hợp với các lệnh lặp.
- Lệnh break còn được sử dụng với lệnh switch.

Lệnh break (1)

- Lệnh **break** chỉ kết hợp với lệnh **switch** và **lệnh lặp**
  - Để kết thúc **một mệnh đề case** trong lệnh **switch**.
  - Để kết thúc **một lệnh lặp**, bỏ qua việc kiểm tra điều kiện.
- Khi gặp lệnh **break** trong một lệnh lặp, **lệnh lặp sẽ kết thúc ngay lập tức** và điều khiển chương trình sẽ chuyển đến câu lệnh ngay sau lệnh lặp.

```c
#include "stdio.h"

int main(void)
{
    int t;

    for (t = 0; t < 100; t++)
    {
        printf("%d ", t);
        if (t == 10) break;
    }

    return 0;
}
```

Lệnh break (2)

- Lệnh break chỉ kết thúc lệnh lặp trong cùng khối chứa nó.
- Nếu một lệnh lặp chứa lệnh switch thì lệnh break kết hợp với lệnh switch chỉ ảnh hưởng đến switch đó chứ không ảnh hưởng đến lệnh lặp.

```c
#include "stdio.h"

int main(void)
{
    int t;

    for (t = 0; t < 100; t++)
    {
        int count = 1;

        for (;;)
        {
            printf("%d ", count);
            count++;
            if (count == 10) break;
        }
    }

    return 0;
}
```


Lệnh continue (1)

- Lệnh **continue** chỉ kết hợp với **lệnh lặp**. Khi gặp lệnh continue trong một lệnh lặp thì lần lặp hiện tại sẽ dừng ngay lập tức và vòng lặp thực hiện lần lặp tiếp theo.
- Với lệnh **for**, lệnh continue chuyển điều khiển đến thực thi ***<gia tăng>*** rồi kiểm tra ***<điều kiện>***.
- Với lệnh **while** và **do while**, lệnh continue chuyển điều khiển đến ước tính ***<biểu thức>***.

Lệnh continue (2)

- Lệnh continue kết hợp với lệnh for

```
int t, sum = 0;

printf("Enter an integer (0 to stop): ");
for (scanf("%d", &t); t; scanf("%d", &t))
{
    printf("Enter an integer (0 to stop): ");
    if (t < 0) continue;
    sum += t;
}

printf("The sum of positive integers is %d\n", sum);
```


Lệnh continue (3)

- Lệnh continue kết hợp với lệnh while

```
int t, sum = 0;

printf("Enter an integer (0 to stop): ");
scanf("%d", &t);
while (t)
{
    if (t < 0)
    {
        printf("Enter an integer (0 to stop): ");
        scanf("%d", &t);
        continue;
    }
    sum += t;
    printf("Enter an integer (0 to stop): ");
    scanf("%d", &t);
}

printf("The sum of positive integers is %d\n", sum);
```

Lệnh biểu thức

- Lệnh biểu thức (expression statement) là một biểu thức hợp lệ theo sau là dấu ;

```
func();      // a function all
a = b + c;   // an assignment statement
b + func(); // a valid, but strange statement
;            // an empty statement
```


Lệnh khối

- Lệnh khối (block statement) là một nhóm các câu lệnh **liên quan với nhau về mặt logic** bắt đầu bằng dấu mở khối { và kết thúc bằng dấu đóng khối } tương ứng.
- Còn được gọi là lệnh ghép (compound statement).