Ngôn ngữ Java

(*Các cấu điều khiển chương trình*)

Môn học: Cơ sở lập trình [*Buổi 4-5*]

GV: Nguyễn Mai Huy

Cấu trúc rẽ nhánh
(Selection constructs)

- Mặc nhiên, các lệnh trong chương trình được thực hiện một cách tuần tự, lên tục …. Cho đến khi hết lệnh thì kết thúc.
- Để chương trình xử lý "***thông minh hơn***", ta có thể ứng dụng các cấu trúc rẽ nhánh trong chương trình để quyến định khi nào thi hành tập lệnh này … và khi nào thì thi hành tập lệnh khác …
- Ngôn ngữ Java cung cấp 2 dạng cấu trúc rẽ nhánh ở dạng

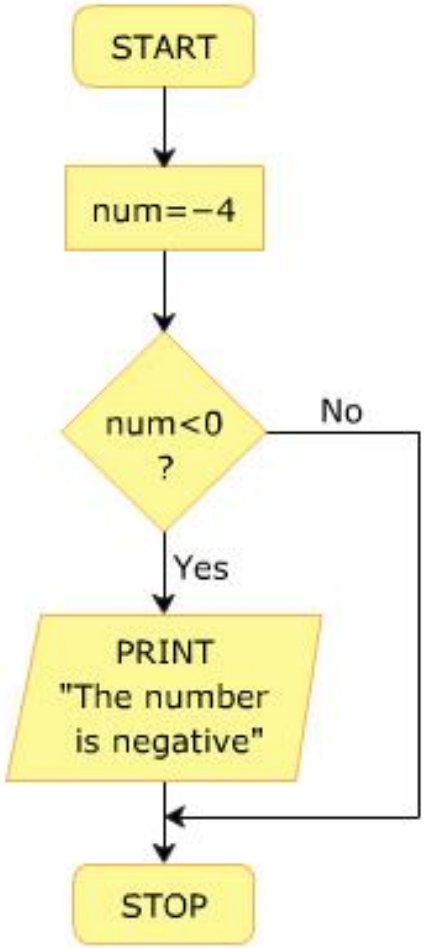
❖ **if …**

❖ **switch … case**

Cấu trúc if {...} "đơn"
(if statement)

- Cấu trúc if { … } dùng để thi hành một tập lệnh quy ước khi thoả mãn điều kiện. Cú pháp
- **if** (*bieu_thuc_dieu_kien*)

```
{
        . . .
        . . .
}
```

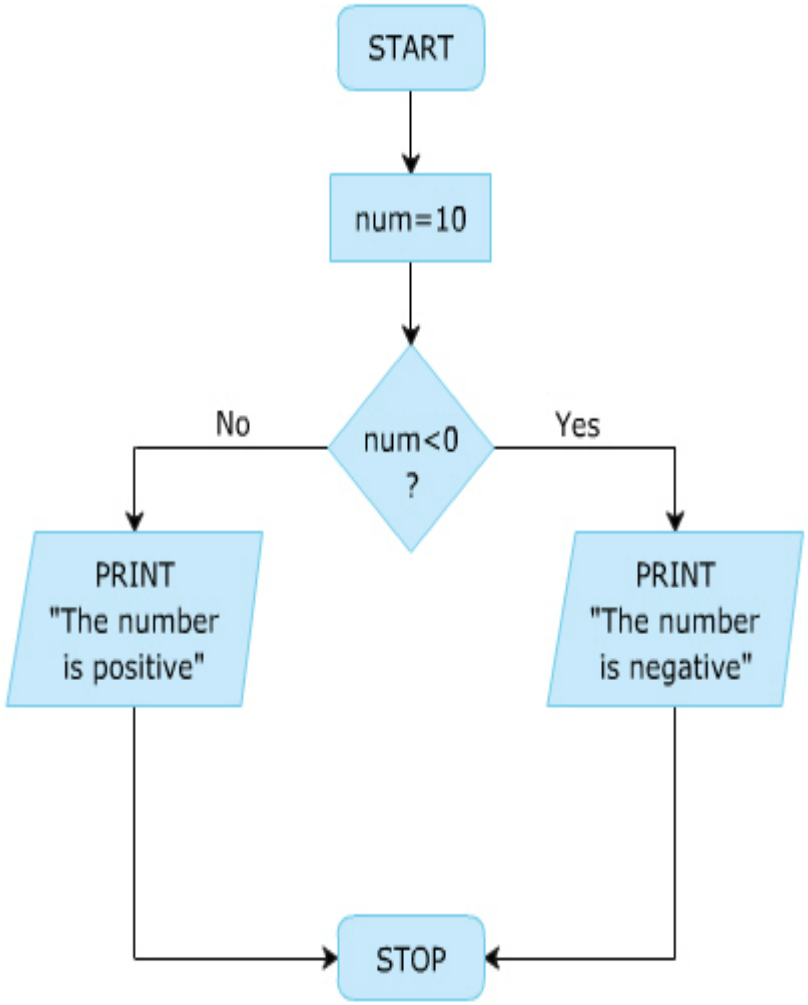


Bài tập minh hoạ

- ✓ Viết chương trình cho phép nhập vào 1 số nguyên bất kỳ, sau đó xét và in ra thông báo cho biết số đã nhập có giá trị bao nhiêu, in thêm thông báo là số dương nếu số đã nhập là số dương
- ✓ Viết chương trình cho phép nhập vào 3 số nguyên tùy ý, sau đó in ra thông báo cho biết số lớn nhất trong 3 số đã nhập có giá trị là bao nhiêu
- ✓ Viết chương trình cho phép nhập vào 2 số nguyên tuỳ ý, sau đó in ra tổng, hiệu, tích và thương của 2 số đã nhập.
  Lưu ý: chỉ in ra thương số nếu số chia 2 khác 0
  (*Số thứ nhất chia cho số thứ hai*)

Cấu trúc if {...} else {...}
(if ... else construct)

- Khi logic xử lý "*mang tính thừa kế*" ở dạng "*Nếu không phải ... thì ngược lại chắc chắn sẽ là ...*", chúng ta có thể sử dụng cấu trúc if với hai vế theo cú pháp sau
- **if** (btdk)
  {
  …
  }
  else
  {
  …
  }



Bài tập minh hoạ

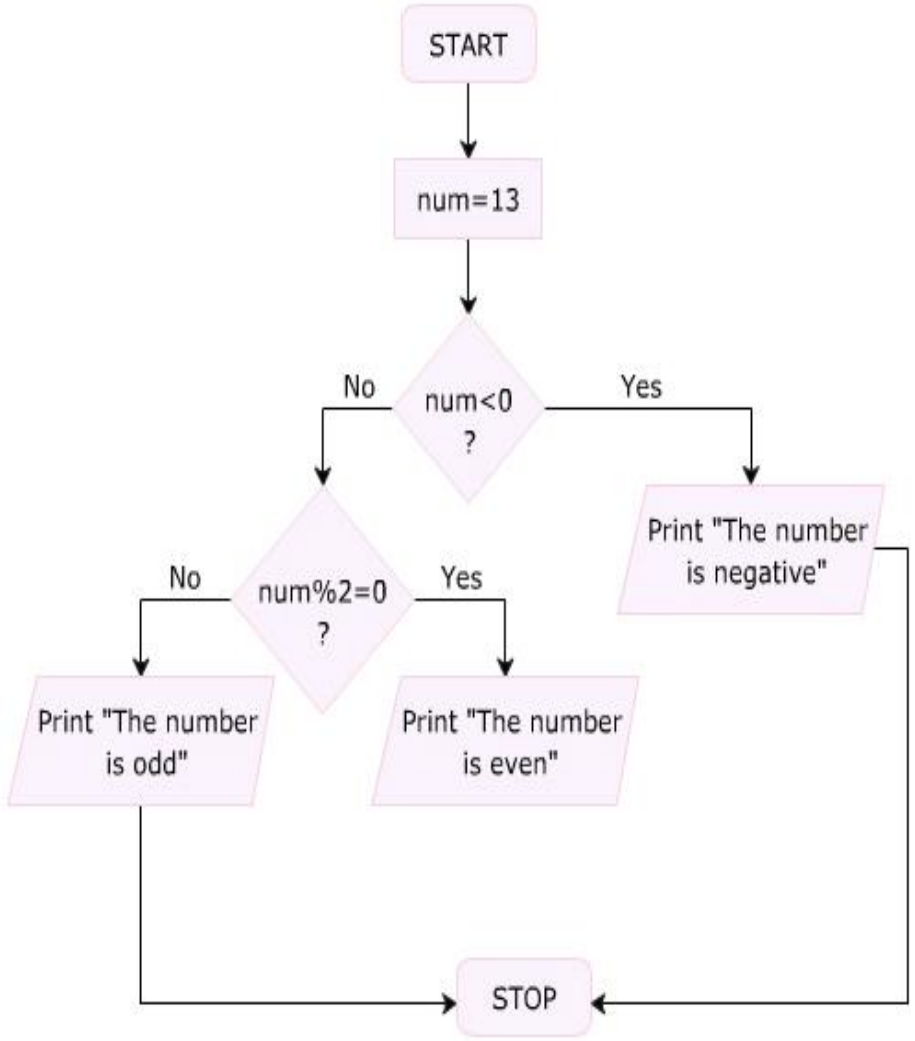
- ✓ Viết chương trình cho phép nhập vào 1 số nguyên tuỳ ý, sau đó xét và in ra thông báo cho biết số đã nhập là số âm hay số dương
- ✓ Viết chương trình cho phép nhập vào 1 số nguyên tuỳ ý, sau đó xét và in ra thông báo cho biết số đã nhập là số chẵn hay số lẻ
- ✓ Viết chương trình cho phép nhập vào điểm trung bình của 1 học sinh, sau đó in ra thông báo cho biết kết quả xét tuyển lên lớp của sinh viên đó là đạt hay không đạt (*Giả sử dtb>=5 là đạt*)
- ✓ Viết chương trình cho phép nhập vào họ tên và giới tính của 1 người, sau đó in ra lời chào dành cho người đó với danh xưng phụ thuộc vào giới tính của người đó. VD

  - "Xin chào Chị Trần Thị Y" Hoặc
  - "Xin chào Anh Nguyễn Minh Z"

Cấu trúc if {...} else if {...}

("if ... else if ..." constructs)

```
if (btdk)
{
        …
}
else if (btdk)
{
        …
}
else
{
        …
}
```



Bài tập minh hoạ

- ✓ Viết chương trình cho phép nhập vào điểm trung bình của 1 học sinh, sau đó in ra thông báo về xếp loại của học sinh tương ứng

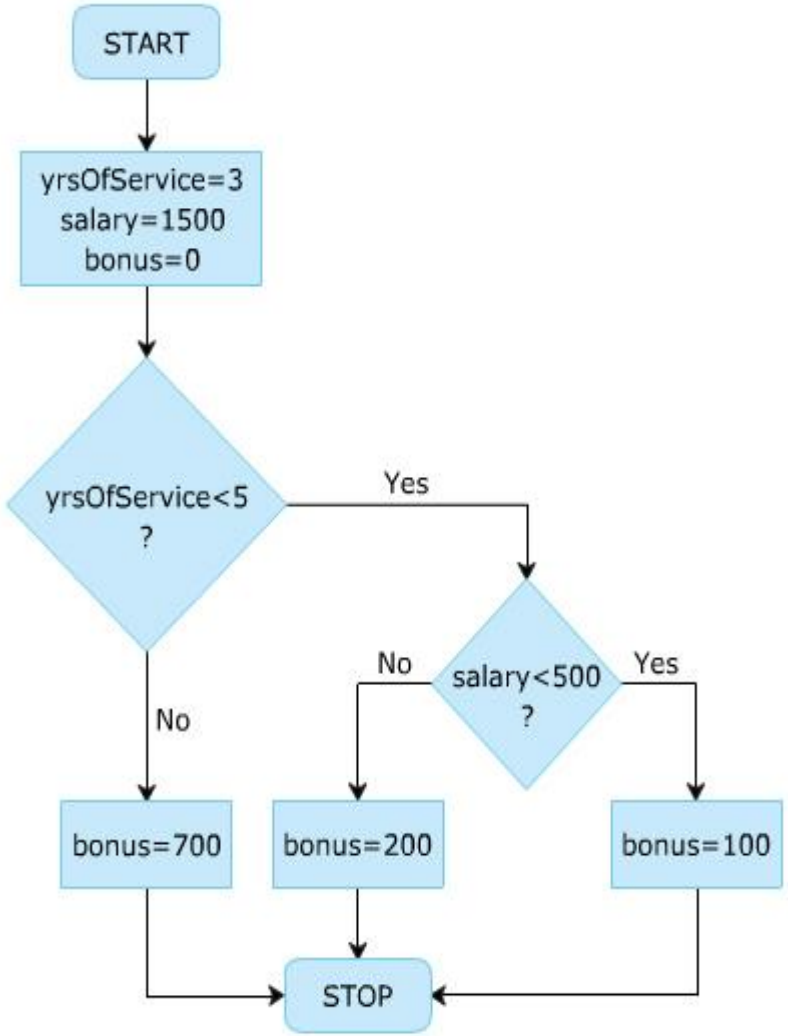
  0 … <5: Yếu
  5 … <7: Trung bình
  7 … <8: Khá
  9 … <10: Giỏi
  10: Xuất sắc

- ✓ Viết chương trình cho phép nhập vào chỉ số đọc được trên đồng hồ đo điện ở một hộ gia đình tại các thời điểm, bao gồm chỉ số cũ và chỉ số mới. Chương trình sẽ tính số kw điện đã tiêu thụ và tính tiền điện cho hộ gia đình tương ứng. Biết rằng tiền điện được tính dựa theo quy định sau

  - dưới 100kw: thì 1 kw phải trả 1200 đ
  - từ kw thứ 101 đến 249 thì 1 kw phải trả 1500 đ
  - từ kw thứ 250 đến 400 thì 1kw phải trả 2300 đ
  - từ kw thứ 401 trở đi, mỗi kw phải trả 3000đ

Sử dụng "if" lồng nhau
(*nested if construct*)

- **if** (btdk)
  {
  if (btdk)
  {
  …
  } **else**
  {
  …
  }
  }
- **else**
  {
  …
  }



Bài tập minh hoạ

- ✓ Viết chương trình giải phương trình bậc hai một ẩn số có dạng

$$ax^2 + bx + c = 0$$

Khi thi hành, chương trình cho phép nhập vào giá trị tương đương với các hệ số a, b, c. Sau đó, chương trình sẽ tính và in ra kết quả tương ứng với các tình huống giải phương trình bậc hai đã được học trước đây.

- ✓ Viết chương trình cho phép tính tiền thưởng tết cho 1 nhân viên với các thông số như sau

Nhập: Số năm làm việc, Mức lương cơ bản

Xử lý:

- Nếu số năm làm việc từ 3 năm trở xuống thì thưởng 1 triệu
- Nếu làm việc trên 3 năm thì xét tiếp
  - + Nếu mức lương dưới 5 triệu thì thưởng nửa tháng lương
  - + Nếu mức lương từ 5 triệu trở lên thì thưởng trọn tháng lương

Toán tử điều kiện – *Conditional operators*

<Điều kiện> ? <Biểu thức 1> : <Biểu thức 2>

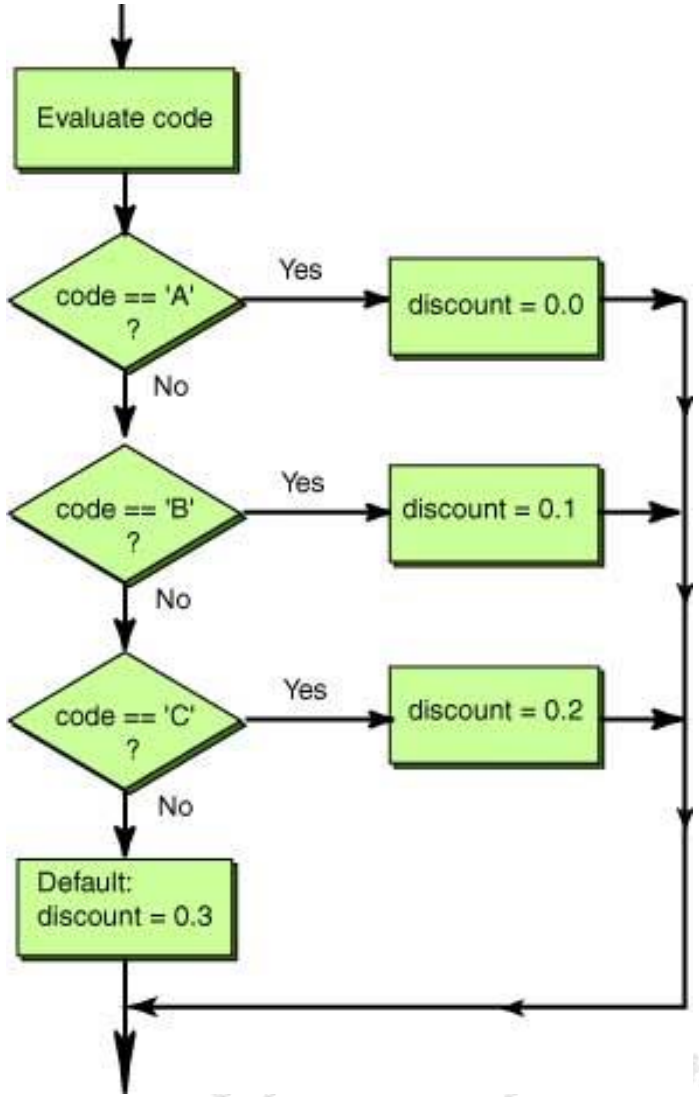
VD:

a > b ? printf(“số lớn là: %d”,a) : printf(“số lớn là: %d”, b);

printf(‘Số lớn là: %d’, a>b?a:b);

Cấu trúc switch … case
(switch … case constructs)

```
switch(bien_gia_tri)
{
        case <gia_tri>:
                <lệnh>;
                . . .
                break;
        case <gia_tri>:
                <lệnh>;
                . . .
                break;
                . . . . .
        default:
                <lệnh>;
                . . .
                break;
}
```



Bài tập minh hoạ

- ✓ Viết chương trình mô phỏng máy tính cá nhân với 4 phép toán cơ bản : + - * /. Khi thi hành, chương trình sẽ cho phép nhập vào 2 số, sau đó chọn phép toán cần thực hiện thì chương trình sẽ in ra kết quả tính được giữa 2 số đã nhập dựa trên phép toán đã yêu cầu
- ✓ Viết chương trình cho phép nhập vào Tháng và Năm cần xét, sau đó chương trình sẽ in ra thông báo cho biết Tháng của Năm đã nhập có bao nhiêu ngày.

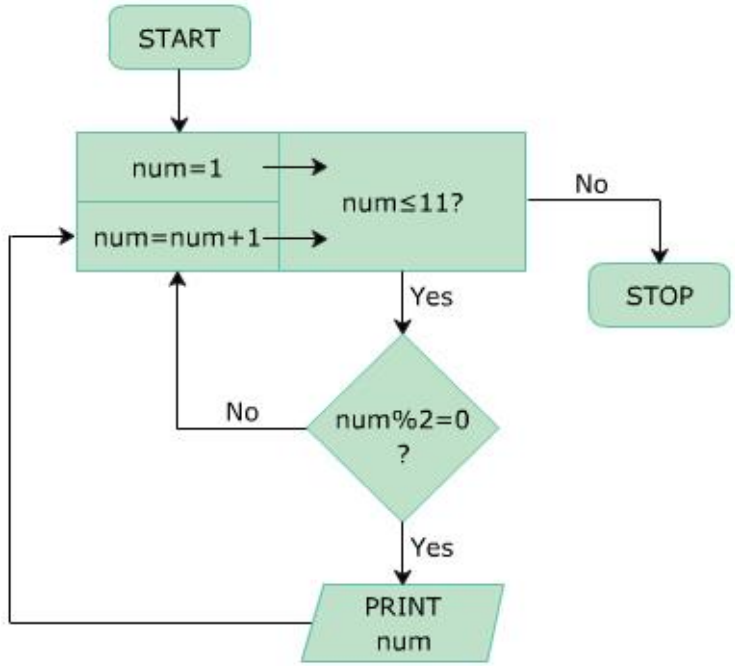
Cấu trúc lặp
(loop constructs)

- Khác với cấu trúc rẽ nhánh, cấu trúc lặp được sử dụng trong chương trình để cho phép thực hiện lệnh (*Hay nhóm lệnh*) theo cách "***Lặp đi, lặp lại***". Tức là những lệnh cùng phục vụ cho một mục đích xử lý nào đó trong chương trình sẽ được "***thi hành nhiều lần***" theo một "***chu kỳ xác định***". Nhờ đó, việc thể hiện mã lệnh trong chương trình dựa trên cấu trúc lặp sẽ ngắn gọn, rõ ràng hơn.
- Có 3 cấu trúc lặp có thể sử dụng cho việc biểu diễn mã lệnh trong chương trình

- **for** construct
- **while** … construct
- **do** … **while** construct

Cấu trúc for
(The for loop)

- Cấu trúc **For** thường được sử dụng trong trường hợp lặp với số lần xác định (*Biết trước số lần sẽ thực hiện trước khi cấu trúc lặp được gọi*)



- Cú pháp:

```
for(<KiểuDL> <biến ĐK>; <Biểu_thức_ĐK>;<Lệnh>)
{
        . . .
}
```

Bài tập minh hoạ

- ✓ Viết chương trình cho phép In ra màn hình bảng cửu chương thứ n (*n được nhập từ bàn phím trong khoảng từ 1 .. 10*)
- ✓ Viết chương trình tính n giai thừa (*n nguyên dương, được nhập từ bàn phím*).

  VD: Nhập 5

  Xuất: n! = 120
- ✓ Viết chương trình cho phép tính tổng của n số nguyên đầu tiên (*n nguyên dương, được nhập từ bàn phím*)

  VD: Nhập 10

  Xuất: S [1 + 2 + . . . + 10] = 55

Cấu trúc for lồng cấp
(Nested for loops)

- Trong những tình huống đặc biệt, chúng ta cần sử dụng các cấu trúc lặp lồng nhau (*Nested loops*), quá trình thi hành sẽ được thực hiện như sau

- Ứng với mỗi lần lặp của cấu trúc **for** cấp 1 thì toàn bộ cấu trúc **for** chứa trong nó sẽ được gọi thi hành
- Số lần lặp được xác định là tích của số lần lặp cấp 1 nhân với số lần lặp của cấu trúc cấp 2

VD: ta có cấu trúc for lồng 2 cấp như minh hoạ sau

```
int s=0;
for( int i =1; i<=10; i++)                    (Cấp 1)
{
        for (int j=1; j<=20; j++)             (Cấp 2)
                s +=5;
        printf("Hết lần lặp thứ %d, s = %d", i, s);
}
```

Bài tập minh hoạ

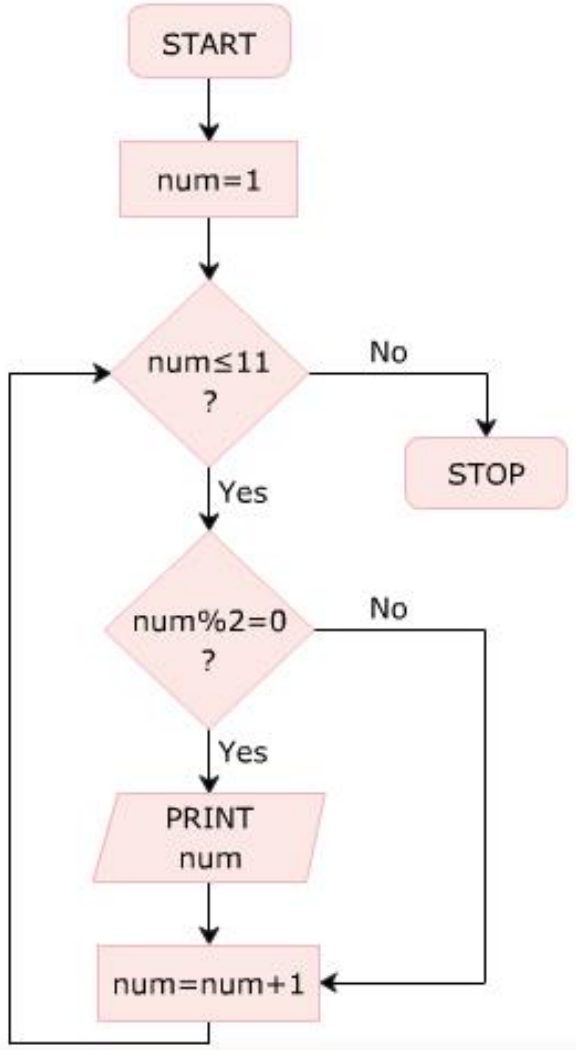
- ✓ Viết chương trình cho phép In ra màn hình các bảng cửu chương từ 1 đến 10
- ✓ Viết chương trình cho phép in ra màn hình hình chữ nhật có kích thước n x m (*n: Kích thước của chiều dài, m: Kích thước chiều rộng*). Ký hiệu vẽ sử dụng dấu '*'

Cấu trúc while
(The for loops)

- Khác với cấu trúc **for**, **while** là cấu trúc lặp không xác định trước số lần lặp (*Trước thời điểm thi hành cấu trúc lệnh*), nguyên tắc để lặp, thi hành tập lệnh trong **while** tuỳ thuộc vào biểu thức điều kiện đã quy ước. Nếu biểu thức điều kiện đúng (***true***) thì lặp, ngược lại (***false***) thì ngưng lặp
- Cú pháp

```
while (<biểu_thức_ĐK>)
{
        . . .
}
```

- * **Ghi chú:** Số lần lặp của cấu trúc while có thể là 0 lần (*Ngay lần đầu tiên, điều kiện đã không thoả mãn*)



Bài tập minh hoạ

- ✓ Viết chương trình cho phép nhập 1 số nguyên dương thuộc hệ Decimal để đổi thành số trong hệ nhị phân và in ra màn hình.

  VD: Nhập: 8 (*Decimal*)

  Xuất: 1000 (*Binary*)

- ✓ Viết chương trình cho phép nhập vào 1 số nguyên dương, sau đó kiểm tra và in ra thông báo cho biết số đã nhập có phải là số nguyên tố hay không

  VD: Nhập 7

  Xuất: 7 Là số nguyên tố

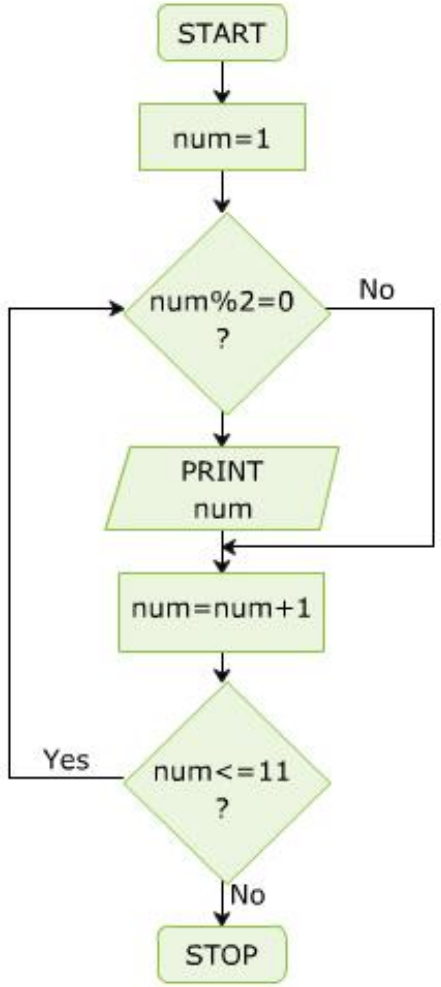
  Nhập: 9

  Xuất: 9 Không phải là số nguyên tố

Cấu trúc do … while
(The **do … while** loop)

- Về cơ bản thì cấu trúc **while** và **do … while** có cơ chế hoạt động hoàn toàn giống nhau, sự khác nhau duy nhất là ở thời điểm xét điều kiện để quyết định "***lặp***" hay "***không lặp***".
- Thời điểm mà **do** … **while** xét điều kiện được đặt phía sau tập lệnh sẽ thực hiện lặp khi điều kiện thoả mãn. Như vậy, ta thấy số lần lặp tối thiểu của cấu trúc dạng này là 1 lần
- Cú pháp

```
do
{
        . . .
} while (<bieu_thuc_dieu_kien);
```



Bài tập minh hoạ

- ✓ Viết chương trình cho phép nhập vào 1 giá trị số nguyên dương, tượng trưng cho số tiền vnđ, sau đó trên màn hình sẽ đưa ra thông báo cho phép chọn loại ngoại tệ muốn đổi (Giả sử ta có các loại ngoại tệ và tỷ giá quy đổi như sau:

```
- 1 USD = 21380 Vnđ - 1 EUR = 23555 Vnđ
- 1 GBP = 32622 Vnđ - 1 JPY =   178 Vnđ
```

Sau khi in ra kết quả, chương trình sẽ đưa ra câu hỏi xem người dùng muốn tiếp tục đổi nữa hay không, nếu không, chương trình sẽ kết thúc

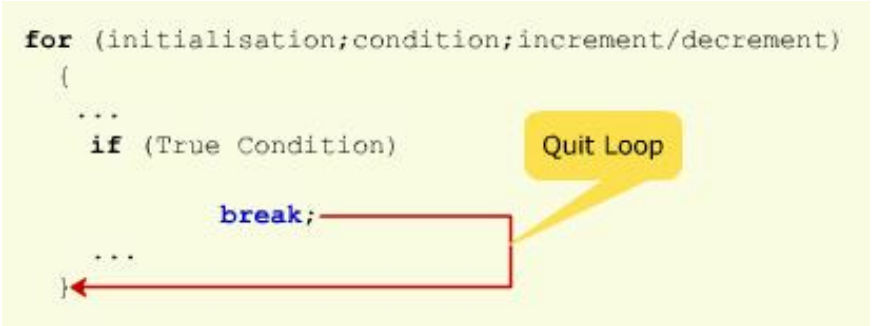
- ✓ Viết chương trình, cho phép tính diện tích và chu vi của các hình tròn, chữ nhật, hình thang. Giao diện có dạng như mô tả sau

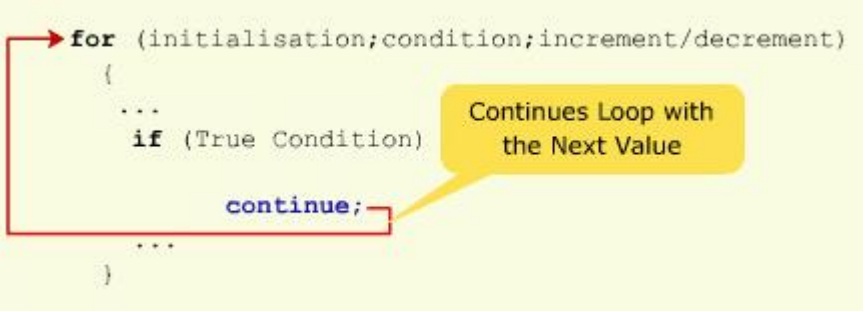
```
**************************************
*       Tính Diện tích - Chu vi      *
*    1 - Hình tròn                   *
*    2 - Hình chữ nhật               *
*    3 - Hình thang                  *
*    4 - Thoát khỏi chương trình     *
**************************************
```

Chương trình chỉ kết thúc khi người dùng nhấn phím số 4

continue - break

- **continue**: Đây là lệnh thường được sử dụng kết hợp trong các cấu trúc lặp để bỏ qua các lệnh còn lại (*Của cấu trúc lặp*) và tiếp tục lần lặp tiếp theo khi thoả mãn 1 điều kiện nào đó trong chương trình
- **break**: gần giống như **continue**, tuy nhiên khác ở chỗ là **break** không những bỏ qua các lệnh còn lại trong cấu trúc có sử dụng nó mà còn kết thúc quá trình hoạt động của cấu trúc tương ứng luôn, không thực hiện nữa





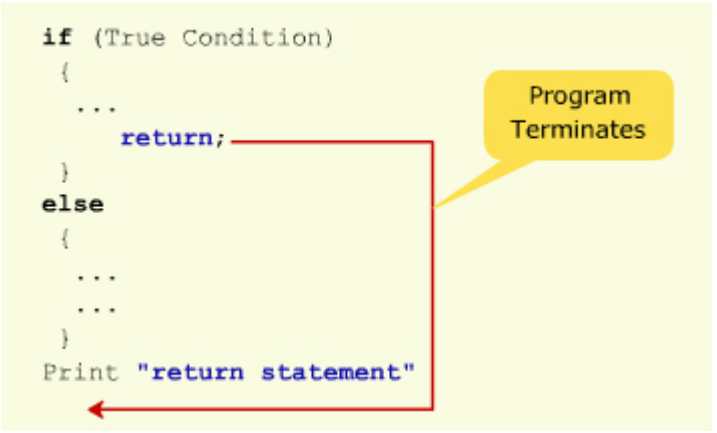
label - goto

- **label**: là lệnh dùng để khai báo một nhãn (*cột mốc*) trong thân chương trình (*Nhằm hỗ trợ cho lệnh goto*). Nhãn chính là 1 định danh (*Identify*) trong chương trình và luôn phải kết thúc bởi dấu ":"
- **goto**: dùng để cho phép chuyển điều khiển thực thi các lệnh trong chương trình tới một nhãn nào đó (*làm mất tính tuyến tính khi thực thi mã lệnh của chương trình*)

Lưu ý: Do lệnh goto kết hợp với label có nguy cơ phá vỡ tính cấu trúc của các giải thuật, chính vì thế, hiện nay, gần như rất ít được sử dụng khi viết chương trình

return

- **return**: được dùng để kết thúc chương trình con (*hàm hiện hành*), đồng thời có thể trả về giá trị (*hoặc không*) cho nơi đã gọi hàm
- cú pháp:
  return [giá_trị];



- **Ghi chú:**
  - Lệnh **return** được sử dụng phổ biến trong các hàm (*function*) được định nghĩa trong chương trình

Nhớ gì ?!!!

- Cấu trúc rẽ nhánh **dùng cho mục đích gì** trong chương trình ?, Có những cấu trúc nào ?.
- Toán tử điều kiện
- Hiểu thế nào về các cấu trúc lặp ?. Phân biệt các tình huống sử dụng cấu trúc lặp trong lập trình. VD: Khi nào dùng **for**, khi nào **while**, khi nào nên dùng **do … while** ?.
- Phân biệt **continue** – **break**. Lệnh **return** dùng để làm gì ?

- **The for Statement**

  (https://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html)

- **The while and do-while Statements**

  (https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html)

- **Java - Loop Control**

  (https://www.tutorialspoint.com/java/java_loop_control.htm)