



## Chương 2 (tiếp)

# CÁC CẤU TRÚC ĐIỀU KHIỂN

Giảng viên: Nguyễn Chiến Thắng  
Email: [thangnc.hai@gmail.com](mailto:thangnc.hai@gmail.com)

# Nội dung

1. Cấu trúc rẽ nhánh
2. Cấu trúc lặp
3. Các lệnh chuyển điều khiển

## 2.2. Cấu trúc lặp

1. Tìm hiểu về cấu trúc lặp **for (...)** và cách sử dụng cấu trúc lặp **for (...)** trong C.
2. Tìm hiểu về cấu trúc lặp lồng nhau.
3. Tìm hiểu về cấu trúc lặp **while (...)**.
4. Tìm hiểu về cấu trúc lặp **do ... while (...)**.
5. Sử dụng các toán tử **break** và **continue**.
6. Các bài tập áp dụng.

## 2.2.1. Đặt vấn đề

- Viết đoạn chương trình hiển thị lên màn hình dãy số tự nhiên 1 2 3 4 5 với giá trị các số lần lượt được lưu trong biến *i*.

```
int main() {  
    int i = 1;  
    cout<<i<<" "; i++;  
    cout<<i<<" "; i++;  
    cout<<i<<" "; i++;  
    cout<<i<<" "; i++;  
    cout<<i<<" "; i++;  
}
```

## 2.2.2. Cấu trúc lặp là gì?

- Một đoạn mã lệnh trong chương trình điều khiển máy tính thực hiện lặp đi lặp lại một lệnh hoặc một khối lệnh cho đến khi một điều kiện xác định được thỏa mãn.

## 2.2.3. Cấu trúc lặp for ...

- Cú pháp:

```
for ([BT_1]; [BT_2]; [BT_3]) {  
    Lệnh_S;  
}
```

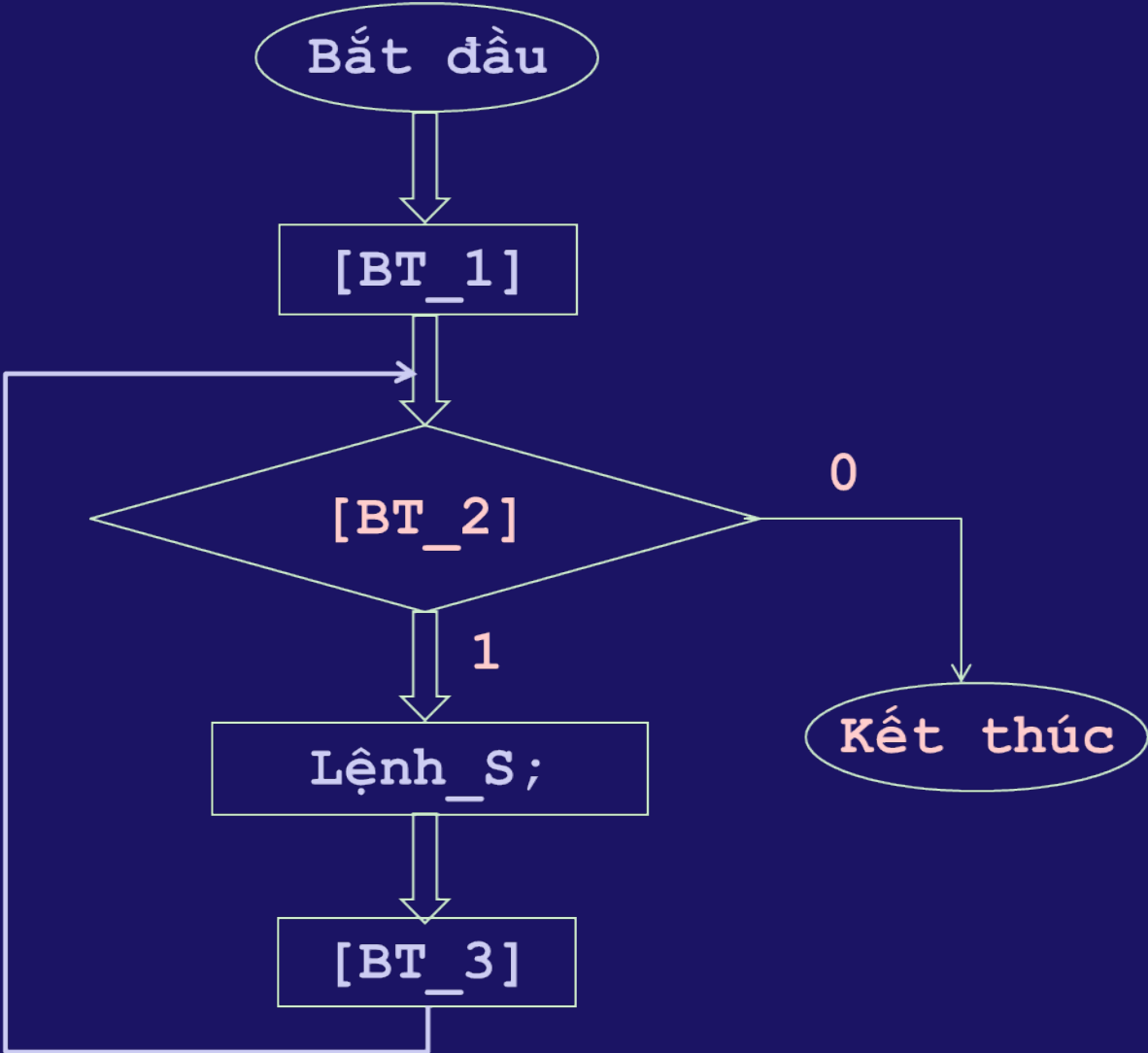
- Trong đó:

- Các [BT\_1], [BT\_2], [BT\_3] có thể khuyết, nhưng bắt buộc phải có dấu ;
- [BT\_1] là một lệnh gán để khởi tạo giá trị ban đầu cho biến điều khiển của vòng lặp.
- [BT\_2] là một biểu thức quan hệ để chỉ định khi nào vòng lặp sẽ kết thúc.
- [BT\_3] định nghĩa cách thức thay đổi giá trị biến điều khiển khi vòng lặp được thực thi.
- Lệnh\_S là một câu lệnh hoặc một khối lệnh.

# Cấu trúc lặp for ... (tt)

- Hoạt động:
  - **Bước 1:** [BT\_1] (Biến điều khiển được gán giá trị khởi tạo)
  - **Bước 2:** [BT\_2] Kiểm tra giá trị của biểu thức quan hệ.
    - Biểu thức quan hệ cho giá trị "đúng" sang bước 3
    - Biểu thức quan hệ cho giá trị "sai" sang bước 4
  - **Bước 3:**
    - Thực hiện Lệnh\_S
    - [BT\_3] (thay đổi giá trị của biến điều khiển)
    - Quay lại bước 2
  - **Bước 4:** Kết thúc vòng lặp

# Cấu trúc lặp for ... (tt)





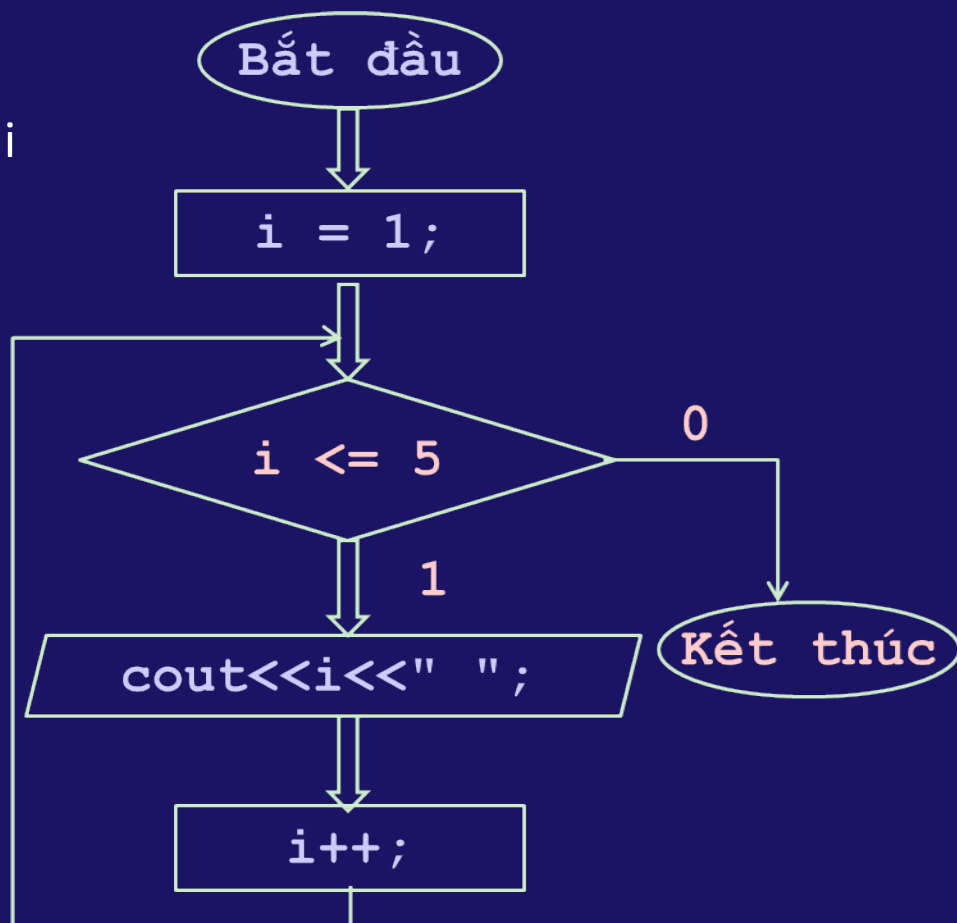
# Cấu trúc lặp for ... (tt) – Ví dụ 1

- Ví dụ 1: Trở lại ví dụ hiển thị lên màn hình các số **1 2 3 4 5** biết các số lần lượt được lưu trong biến **i**.

- Thiết kế vòng lặp:

- Biến điều khiển vòng lặp là biến **i**
- Biểu thức 1:  **$i = 1$**
- Biểu thức 2:  **$i \leq 5$**
- Biểu thức 3:  **$i++$**
- Lệnh\_S: **`cout<<i<<" "`**;

```
for (i=1; i<=5; i++)  
{  
    cout<<i<<" "  
}
```



## Cấu trúc lặp for ... (tt) – Ví dụ 2

- Ví dụ 2: Viết đoạn chương trình hiển thị lên màn hình dãy số: 5, 4, 3, 2, 1, 0, -1, -2, -3, -4, -5
- Thiết kế vòng lặp:
  - Biến điều khiển vòng lặp là biến `i`
  - Biểu thức 1: `i = 5`
  - Biểu thức 2: `i >= -5`
  - Biểu thức 3: `i--`
  - Lệnh\_S: `cout<<i<<" "`;

```
for (i=5; i>=-5; i--)  
{  
    cout<<i<<" "  
}  
}
```

## 2.2.4. Cấu trúc lặp for lồng nhau

- Các vòng lặp for lồng nhau khi nó có dạng như sau:

```
for (i = 1; i <= m; i++) {  
    Lệnh_A;  
    for (j = 1; j <= n; j++) {  
        Lệnh_B;  
    }  
    Lệnh_C;  
}
```

## Cấu trúc lặp for lồng nhau (tt)

- Ví dụ 1: Viết chương trình giải quyết bài toán:
  - Vừa gà vừa chó
  - Bó lại cho tròn
  - Ba mươi sáu con
  - Một trăm chân chẵn
  - Hỏi có bao nhiêu gà bao nhiêu chó?
- Ví dụ 2: Viết chương trình hiển thị ra màn hình tam giác số nguyên n. Chẳng hạn với  $n = 5$  ta có:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

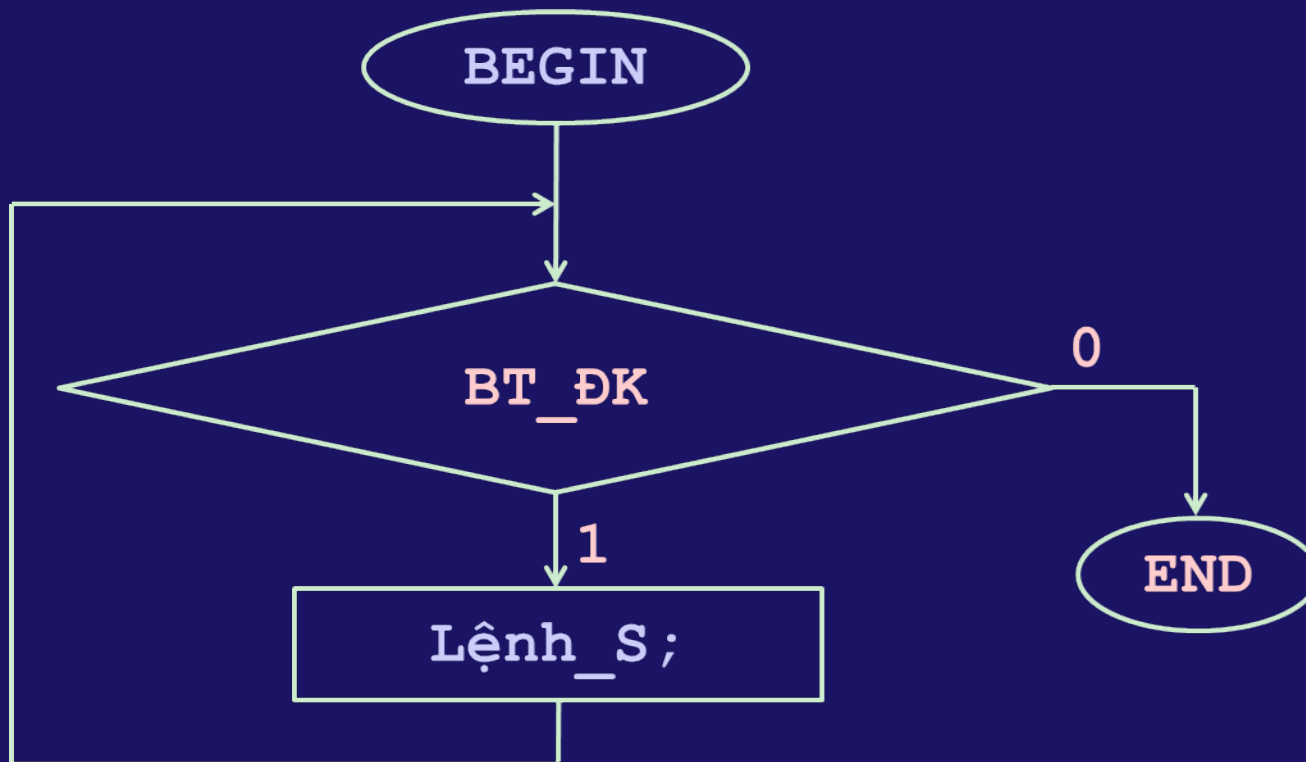
## 2.2.5. Cấu trúc lặp while ...

```
while (BT_ĐK)
{
    Lệnh_S;
}
```

Vòng lặp while lặp lại Lệnh\_S trong khi BT\_ĐK mang giá trị 1 (true).

## Cấu trúc lặp while ... (tt)

- Hoạt động:



## Cấu trúc lặp while ... (tt) – Ví dụ 1

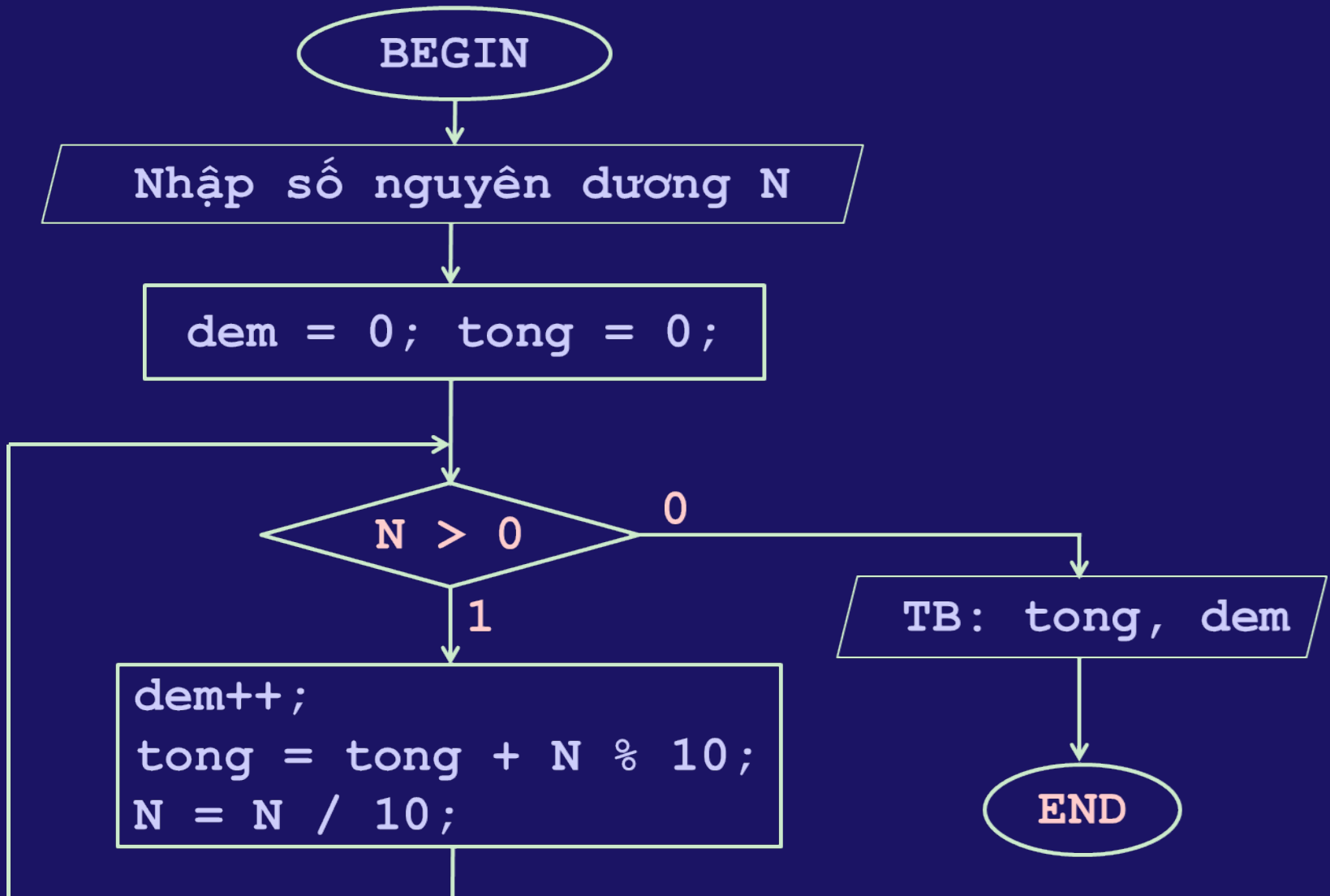
```
/*Chương trình in ra màn hình các số tự
nhiên từ 1->10*/
#include <iostream>
using namespace std;
int main() {
    cout<<"Day so tu nhien: ";
    int i = 1;
    while (i <= 10) {
        cout<<i<<" ";
        i++;
    }
    cout<<"\nKet thuc lap, i = "<<i;
}
```

## Cấu trúc lặp while ... (tt) Ví dụ 2

- Bài toán:
  - Nhập số nguyên dương N.
  - Cho biết N có bao nhiêu chữ số, tổng các chữ số của N.
- Cách giải quyết:
  - Chia liên tiếp N cho 10 cho đến khi được kết quả bằng 0 thì dừng lại.
  - Số lần chia 10 là số chữ số của N.
  - Tổng các số dư trong mỗi lần chia là tổng các chữ số của số nguyên dương N.



## Cấu trúc lặp while ... (tt) – Ví dụ 2



## 2.2.6. Cấu trúc lặp do ... while ...

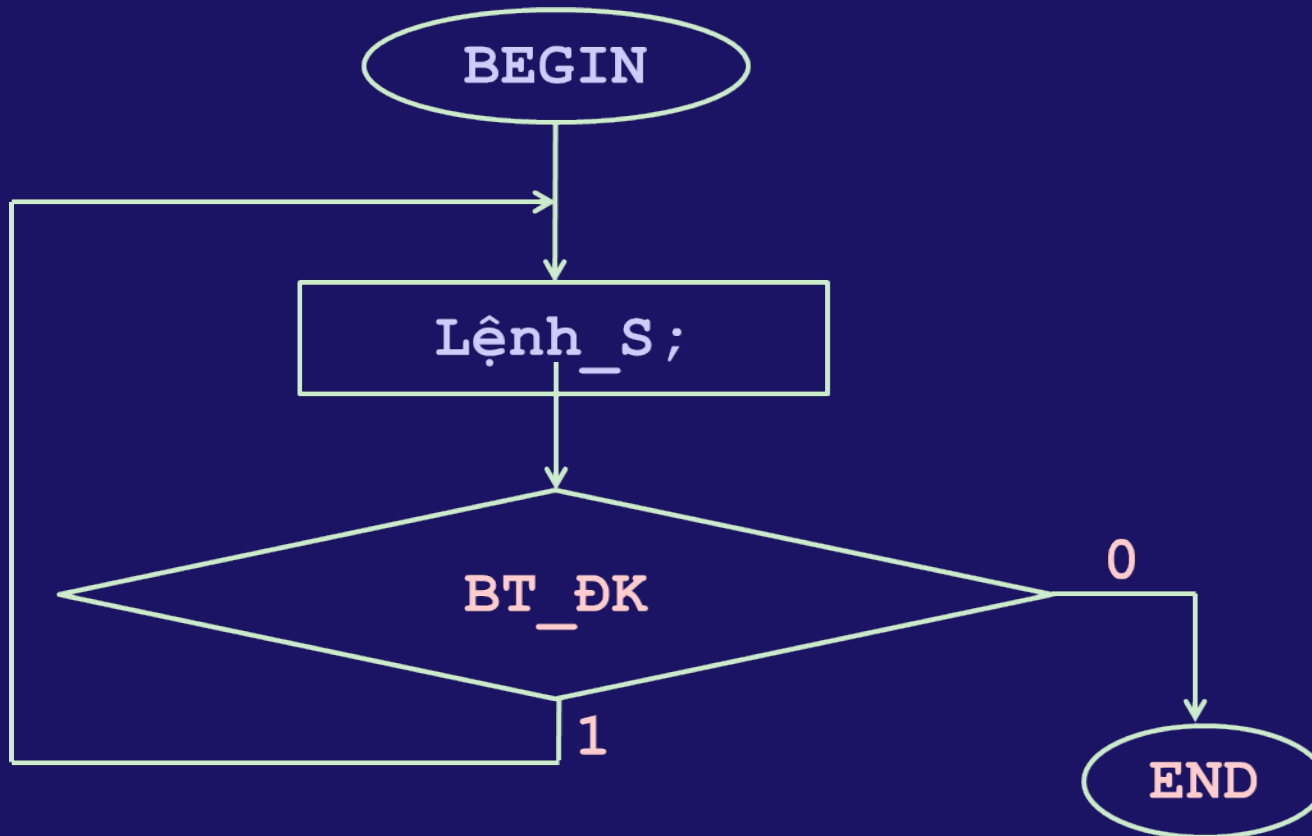
- Cú pháp:

```
do {  
    Lệnh_S;  
} while (BT_ĐK);
```

- Trong đó:
  - Lệnh\_S trong thân của cấu trúc lặp được thực thi trước khi BT\_ĐK kiện được kiểm tra.
  - Khi BT\_ĐK mang giá trị 0 (false), vòng lặp do ... while (...) sẽ kết thúc, và điều khiển chuyển đến lệnh xuất hiện ngay sau lệnh while (...).

## Cấu trúc lặp do ... while ... (tt)

- Hoạt động:



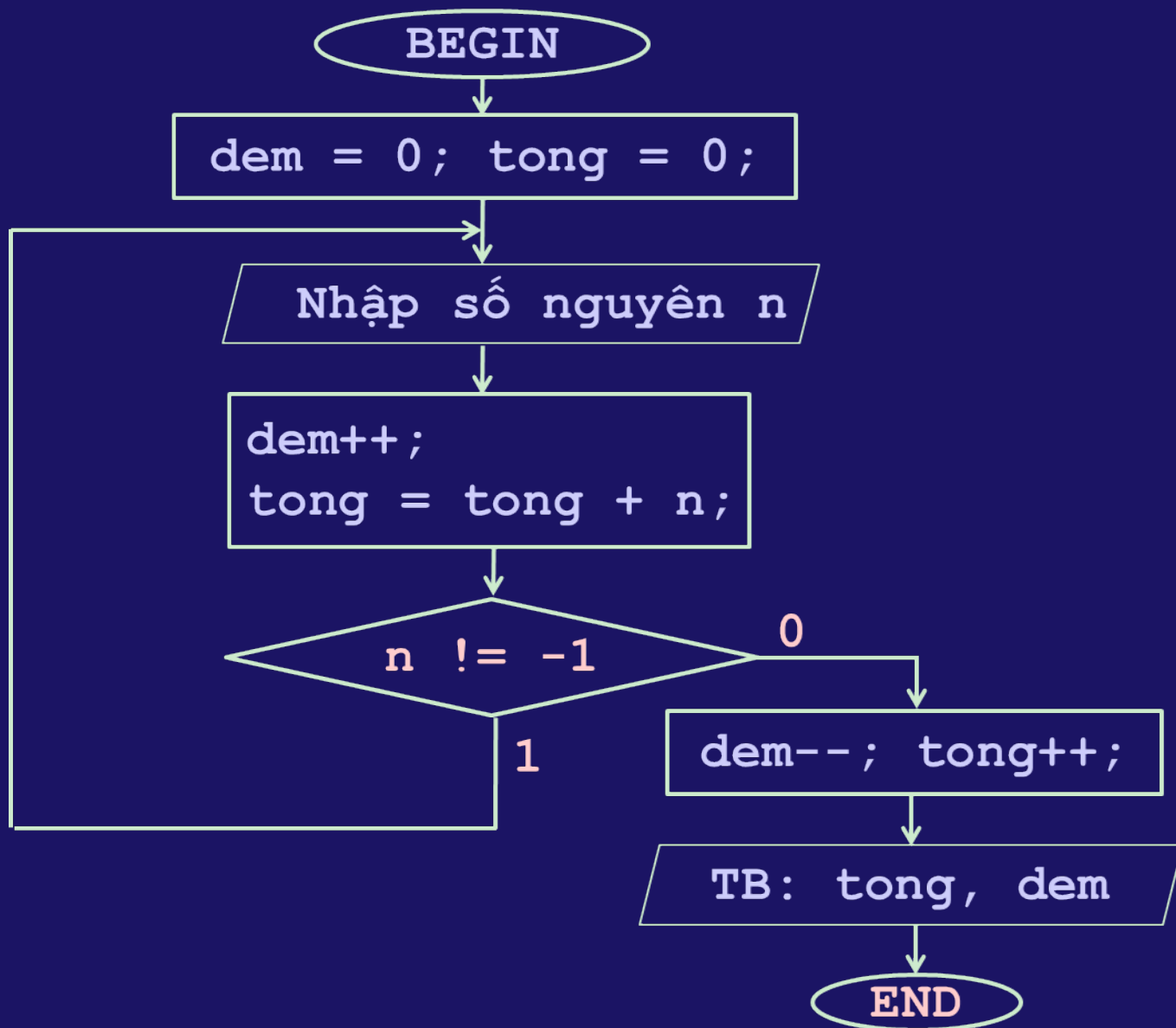
# Cấu trúc lặp do ... while ... (tt) \_ Ví dụ 1

```
/*Chương trình in ra màn hình các số tự
nhiên từ 1->10*/
#include <iostream>
using namespace std;
int main(){
    cout<<"Day so tu nhien: ";
    int i = 1;
    do{
        cout<<i<<" ";
        i++;
    } while(i <= 10);
    cout<<"\nKet thuc lap voi i = "<<i;
}
```

## Cấu trúc lặp do ... while ... (tt) \_ Ví dụ 2

- Viết chương trình thực hiện:
  - Nhập liên tiếp các số nguyên, việc nhập kết thúc khi gặp số nhập vào có giá trị là -1.
  - Cho biết có bao nhiêu số đã được nhập (không tính -1) và tổng các số đó.

# Cấu trúc lặp do ... while ... (tt) \_ Ví dụ 2



## Cấu trúc lặp do ... while ... (tt) \_ Ví dụ 2

```
#include <iostream>
using namespace std;
int main (){
    int n, dem = 0, tong = 0;
    cout<<"Nhap day so nguyen (nhap -1 de dung)\n";
    do {
        cout<<"Nhap so thu "<<dem+1; cin>>n;
        dem++;
        tong = tong + n;
    } while (n != -1);
    cout<<"Co "<<dem - 1<<" so da duoc nhap\n";
    cout<<"Tong cac so da nhap la "<<tong + 1;
}
```

## 2.2.7. Các lệnh chuyển điều khiển

# break;

- Lệnh **break** được sử dụng để kết thúc một mệnh đề **case** trong câu lệnh **switch (...)**.
- Nó cũng có thể được sử dụng để kết thúc ngang giữa vòng lặp.
- Khi gặp lệnh **break**, vòng lặp sẽ kết thúc ngay và điều khiển được chuyển đến lệnh kế tiếp bên ngoài vòng lặp.



# Các lệnh chuyển điều khiển (tt) - break

```
#include <iostream>
using namespace std;
int main (){
    int i, j = 0;
    for (i = 1; i <= 100; i++)
    {
        cout<<"Nhap j: ";
        cin>>j;
        if (j == 100) break;
    }
}
```

## 2.2.7. Các lệnh chuyển điều khiển

# continue;

- Lệnh **continue** dùng để bắt đầu thực hiện lần lặp kế tiếp của vòng lặp.
- Khi gặp lệnh **continue**, các câu lệnh còn lại trong thân vòng lặp bị bỏ qua và điều khiển được chuyển đến lần lặp kế tiếp.

# Các lệnh chuyển điều khiển (tt) - continue

```
#include <iostream>
using namespace std;
int main ()
{
    int i;
    for (i = 1; i <= 100; i++)
    {
        if (i % 9 == 0)
            continue;
        cout<<i<<" ";
    }
}
```

## 2.2.7. Các lệnh chuyển điều khiển

# return;

- Lệnh **return** dùng để kết thúc một hàm, chuyển điều khiển chương trình về nơi gọi hàm.
- Khi gặp lệnh **return**, hàm sẽ kết thúc ngay.

# Các lệnh chuyển điều khiển (tt) - continue

```
#include <iostream>
using namespace std;
int giai_thua (int n)
{
    if (n == 0) return 1;
    int gt = 1;
    for (int i = 1; i <= n; i++)
    {
        gt *= i;
    }
    return gt;
}
```

**THANK YOU!**