CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

BÀI 1 - TỔNG QUAN VỀ CTDL & GT

Giới thiệu môn học

Thông tin học phần

1. Tên học phần: Cấu trúc dữ liệu và giải thuật
2. Mã học phần: 2101409
3. Số tín chỉ: 03LT+1TH
4. Học phần tiên quyết: Kỹ thuật lập trình

Mục tiêu học phần

- **C**ung cấp các PP tổ chức và các thao tác cơ sở trên các CTDL, song song là sự kết hợp hai thành phần trên để hình thành nên một chương trình máy tính.
- **C**ủng cố và phát triển kỹ năng lập trình được học trong giai đoạn trước.
- **H**ình thành thái độ làm việc chăm chỉ, có cường độ cao và chú ý đến chi tiết. Hình thành phong cách lập trình chuyên nghiệp.

Tóm tắt nội dung

- Học phần này giúp sinh viên hiểu được tầm quan trọng của giải thuật và cách thức tổ chức dữ liệu, là hai thành tố quan trong nhất cho một chương trình.
- Cung cấp các phương pháp tổ chức và những thao tác cơ sở trên từng CTDL, kết hợp với việc phát triển tư duy giải thuật để hình thành nên chương trình máy tính. Công cụ sử dụng trong môn học là ngôn ngữ lập trình C.

Phương pháp đánh giá

(1) Điểm quá trình

- Điểm bài tập nhóm/ kiểm tra trong quá trình học
- Điểm danh
- Điểm thực hành
- Kiểm tra GK: THỰC HÀNH

(2) Điểm thi kết thúc môn

- **Thi** tự luận/trắc nghiệm

Tài liệu tham khảo

1. Slide bài giảng, Lab thực hành
2. Giáo trình chính:

- Nguyễn Trung Trực (2019). Cấu trúc dữ liệu và giải thuật. NXB Trường Đại học Bách Khoa TP.HCM.
- G A Vijayalakshmi Pai(2017). Data Structures And Algorithms: Concepts, Techniques And Applications, First Edition. Mc Graw Hill

Nội dung

Bài 1. Tổng quan về CTDL & GT

1.1. Vai trò của cấu trúc dữ liệu trong một đề án tin học

1.2. Các tiêu chuẩn đánh giá cấu trúc dữ liệu

1.3. Trừu tượng hóa dữ liệu

1.4. Kiểu dữ liệu cơ bản

1.5. Kiểu dữ liệu có cấu trúc

1.6. Độ phức tạp của giải thuật*

1.7. Bài tập

Nội dung

Bài 2. Các kỹ thuật tìm kiếm

2.1. Khái quát về tìm kiếm

2.2. Tìm tuyến tính (Linear Search)

2.3. Tìm nhị phân (Binary Search)

Nội dung

Bài 3. Các kỹ thuật sắp xếp

3.1. Tổng quan

3.2. Các phương pháp sắp xếp thông dụng

Nội dung

Bài 4. Cấu trúc danh sách

4.1. Khái niệm

4.2. Cấu trúc danh sách

4.3. Các phương pháp cài đặt danh sách

4.4. Hiện thực danh sách kề

4.5. Hiện thực danh sách liên kết đơn

4.6. Các loại danh sách khác

4.7. Bài tập

Nội dung

Bài 5. Cấu trúc Stack và Queue

5.1. Giới thiệu về Stack

5.2. Hiện thực Stack

5.3. Một số bài toán ứng dụng Stack

5.4. Giới thiệu về Queue

5.5. Hiện thực Queue

5.6. Hàng đợi có ưu tiên

5.7. Bài tập

Nội dung

Bài 6. Cấu trúc Cây - Cây nhị phân – Cây nhị phân tìm kiếm

6.1 Cấu trúc cây tổng quát

6.2 Cây nhị phân

6.3 Mô tả cây nhị phân

6.4 Hiện thực cây nhị phân tổng quát

6.5. Định nghĩa cây nhị phân tìm kiếm

6.6. Cài đặt cây nhị phân tìm kiếm

6.7. Bài tập

Nội dung

Bài 7. Cây nhị phân tìm kiếm cân bằng - AVL

7.1. Định nghĩa cây nhị phân tìm kiếm cân bằng

7.2. Các tác vụ xoay

7.3. Thêm một nút vào cây AVL

7.4. Cài đặt cây AVL

7.5. Bài tập

Bài 1 - Tổng quan về CTDL & GT

Nội dung bài học

1. Giới thiệu vai trò của tổ chức dữ liệu
2. Mối quan hệ giữa GT & CTDL
3. Các khái niệm và yêu cầu về CTDL
4. Nhắc lại các kiểu dữ liệu trong C
5. Tổng quan về đánh giá độ phức tạp GT
6. Nhắc lại các cấu trúc điều khiển trong C
7. Bài tập

Vai trò của tổ chức dữ liệu

Bài toán thực tế

Bài toán giải quyết
trong máy tính

Đối tượng
dữ liệu

Xử lý trên
đối tượng DL

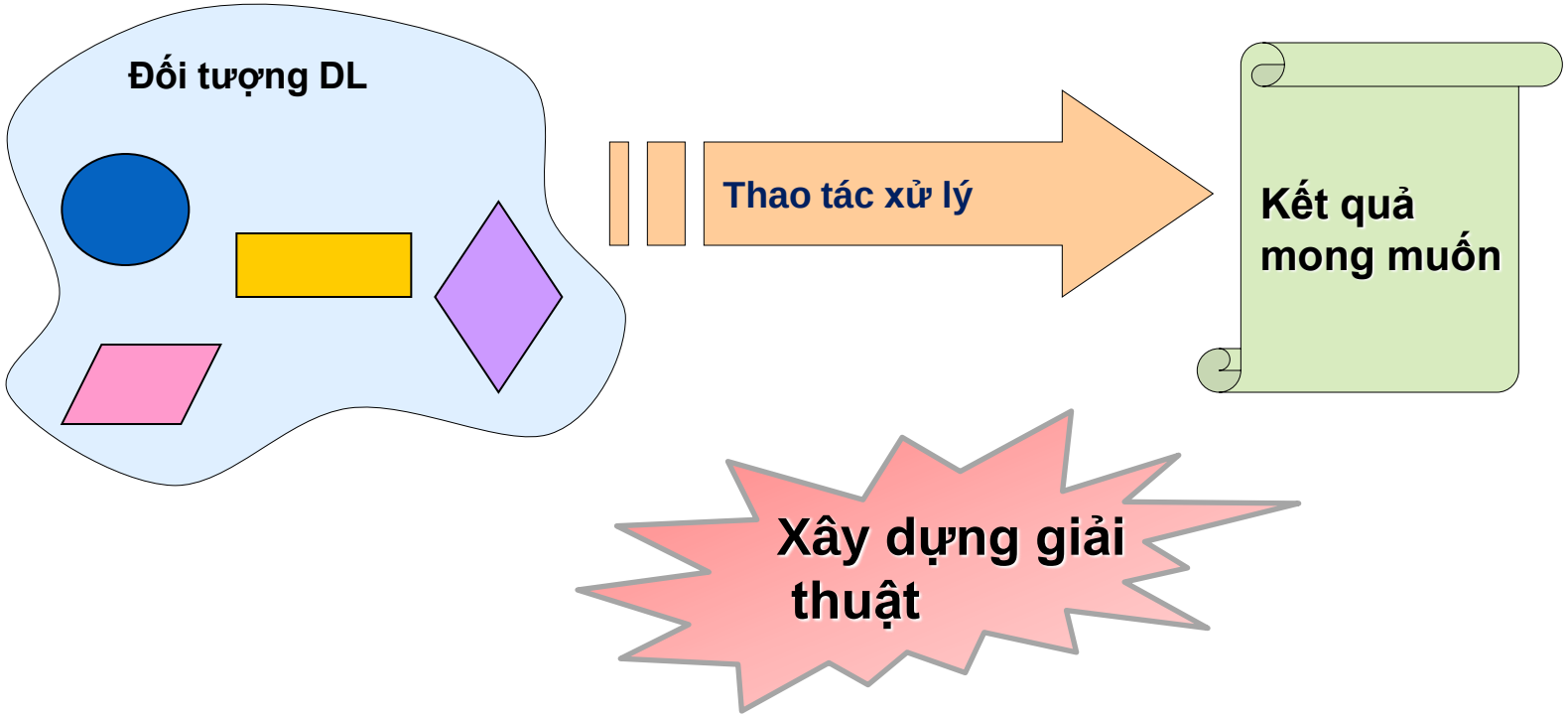
Vai trò của tổ chức dữ liệu

- Dữ liệu thực tế
  - ✓ Muôn hình vạn trạng, đa dạng, phong phú
  - ✓ Thường có chứa đựng các mối quan hệ với nhau
- Cần phải tổ chức biểu diễn thành cấu trúc thích hợp nhất
  - ✓ Phản ánh chính xác dữ liệu thực tế
  - ✓ Dễ dàng xử lý trong máy tính

Vai trò của tổ chức dữ liệu

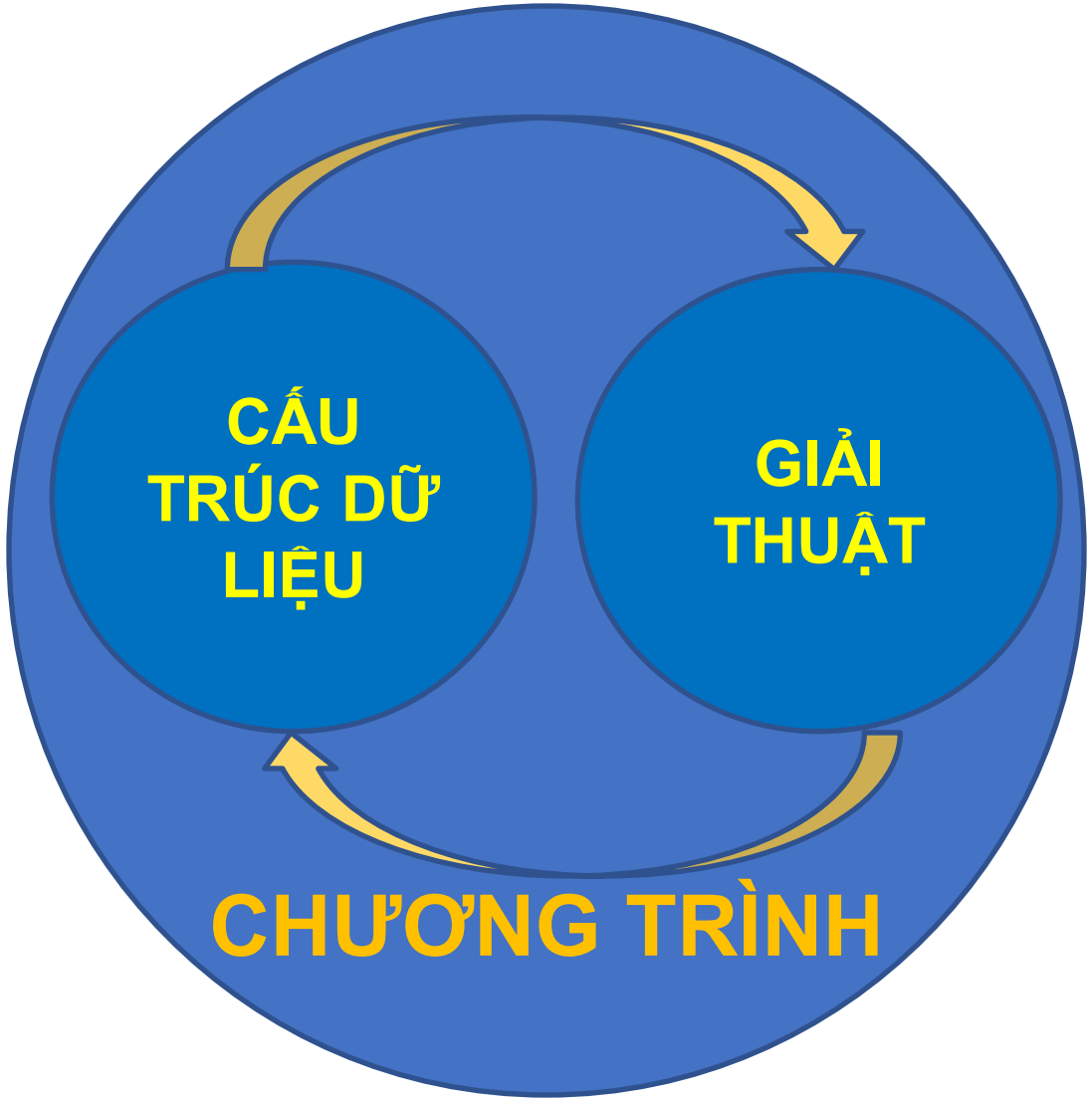
Xây dựng thao tác xử lý

Dựa trên Y/C cụ thể, xác định các trình tự giải quyết vấn đề trên máy tính để đưa kết quả mong muốn



Vai trò của tổ chức dữ liệu

Niklaus Wirth



Cấu trúc dữ liệu?

- Cách thức liên kết/ tổ chức các DL hợp lý để sử dụng một cách hiệu quả (*phương pháp lưu trữ trên máy tính*)
- Tập các thao tác để truy cập/ xử lý các thành phần DL

→ **Cần giải thuật**

- *Danh sách liên kết (Linked List)*
- *Ngăn xếp (Stack)/ Hàng đợi (Queue)*
- *Cây (Tree)*
- *…*

Các tiêu chuẩn đánh giá CTDL

(1) Phản ánh đúng thực tế

- Thể hiện được đầy đủ thông tin nhập/xuất của bài toán

(2) Phù hợp với thao tác xử lý

- Tăng tính hiệu quả của chương trình → hiệu quả của dự án tin học

(3) Tiết kiệm tài nguyên hệ thống

- Sử dụng tài nguyên vừa đủ để thực hiện chức năng & nhiệm vụ.

Trừu tượng hóa dữ liệu

- Trừu tượng hoá
  - Làm đơn giản hóa, sáng sủa, dễ hiểu hơn
  - Che đi phần chi tiết, làm nổi bật cái tổng thể
- Trừu tượng hoá dữ liệu: đưa ra các KDL trừu tượng (Abstract Data Type – ADT) gồm:
  - Mô tả dữ liệu (data)
  - Tác vụ xử lý liên quan (operation)

Trừu tượng hóa dữ liệu

Ví dụ: mô tả kiểu dữ liệu trừu tượng về số hữu tỉ a/b với các tác vụ: +,*, /

- Mô tả dữ liệu
  - Tử số
  - Mẫu số (≠0)
- Mô tả tác vụ

  Cộng (Số_Hữu_Tỉ 1, Số_Hữu_Tỉ 2)
  - Nhập:
    - a, b là tử số và mẫu số của Số_Hữu_Tỉ 1
    - c, d là tử số và mẫu số của Số_Hữu_Tỉ 2
  - Xuất:
    - ad+bc là tử số của số hữu tỉ kết quả
    - bd là mẫu số của số hữu tỉ kết quả

Khái niệm về kiểu dữ liệu

- Thể hiện các dạng/ cấu trúc lưu trữ dữ liệu
- Các loại kiểu dữ liệu
  - Kiểu dữ liệu cơ bản: Cơ sở, mảng, cấu trúc cơ bản
  - Kiểu dữ liệu có cấu trúc hướng giải quyết vấn đề: Danh sách liên kết, hàng đợi, ngăn xếp, cây, bảng băm, …

Nhắc lại các kiểu dữ liệu C

- Kiểu cơ sở: Số nguyên, số thực
- Kiểu dãy (mảng), xâu ký tự
- Kiểu có cấu trúc

Kiểu số nguyên – số thực

Data type	Size	Value range
char	1 byte	-128 đến 127 hoặc 0 đến 255 (Ký tự dạng mã ASCII)
unsigned char	1 byte	0 đến 255
int	2 bytes	-32,768 đến 32,767
unsigned int	2 bytes	0 đến 65,535
long	4 bytes	-2,147,483,648 đến 2,147,483,647
unsigned long	4 bytes	0 đến 4,294,967,295
float	4 bytes	3.4E-38 đến 3.4E38
double	8 bytes	1.7E-308 đến 1.7E308

Kiểu dữ liệu có cấu trúc

- Kết hợp nhiều kiểu dữ liệu cơ bản để phản ánh bản chất của đối tượng thực tế
- Đa số các ngôn ngữ đều cài đặt sẵn một số kiểu có cấu trúc cơ bản: mảng, chuỗi, tập tin, bản ghi…
- Ngoài ra cung cấp cơ chế cho người lập trình cài đặt các dữ liệu cấu trúc khác

Kiểu mảng một chiều có n phần tử

Giá trị	5	7	10	…	3	11	2
Vị trí	*0*	*1*	*2*	*…*		*n-2*	*n-1*

- **Khai báo**

<KDL> <Tên mảng> [<Số phần tử max>];

VD: int a[100];

- **Gán giá trị ban đầu**

VD1: int a[100] = {0};

VD2: int a[5] = {3, 6, 2, 10, 17};

hoặc: int a[] = {3, 6, 2, 10, 17};

Kiểu mảng một chiều

Phát sinh ngẫu nhiên <time.h>

- Khởi tạo phát sinh ngẫu nhiên: srand((unsigned int)time(NULL));
- Phát sinh giá trị ngẫu nhiên: int x = rand()%k;
  - *k: Số nguyên dương*
  - *$x \in [0..k-1]$*

VD: Phát sinh 1 số nguyên có giá trị từ 0 đến 50

srand((unsigned int)time(NULL));

int x = rand()%51;

Kiểu xâu ký tự

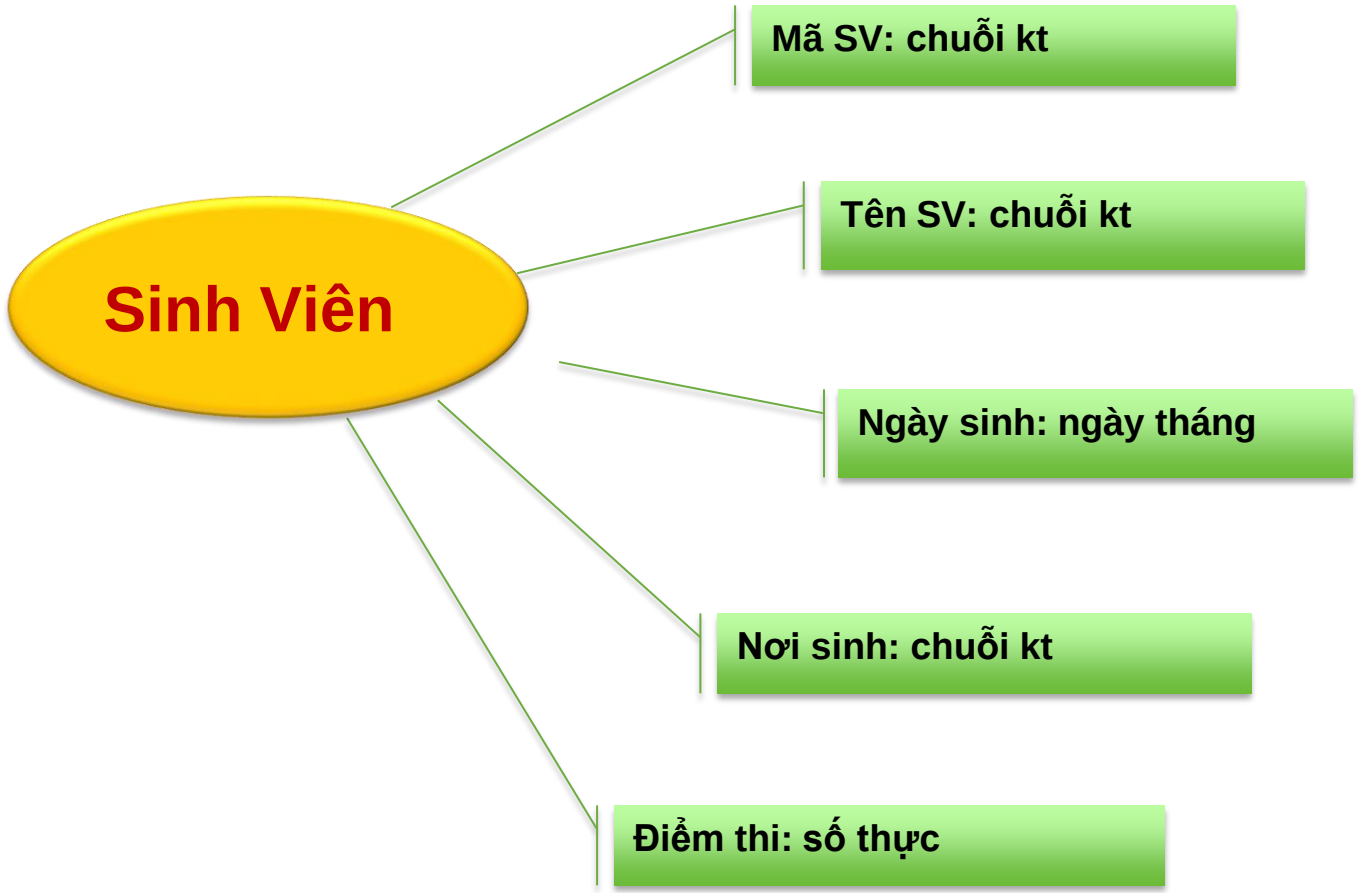
- **Khai báo**

char <Tên xâu> [<Số ký tự max>] ;

VD: char hoten[50];

- **Xem lại các hàm**
  - *gets, fflush*
  - *strcpy, strcat, strlen*
  - *strcmp, stricmp*
  - *strchr, strstr*
  - …

Kiểu dữ liệu có cấu trúc



Kiểu dữ liệu có cấu trúc

```
typedef struct tên_struct
{
        khai báo các thuộc tính;
} tên_kiểu;
```

Ví dụ kiểu dữ liệu có cấu trúc

```
typedef struct ttDate
{
        int ngay;
        int  thang;
        int nam;
} Date;
```

Truy cập thành phần có cấu trúc

Biến cấu trúc kiểu tĩnh

<tên biến>.thành phần cấu trúc

VD:

```
Date d;

d.nam = 2012;
```

Khái niệm về giải thuật

- Là một thủ tục tính toán xác định (well-defined) nhận các giá trị hoặc một tập các giá trị (input) và sinh ra một vài giá trị hoặc tập giá trị (output)

➔ Cách thức/ quy trình thực hiện hoàn thành một công việc xác định cụ thể nào đó.

VD Cộng 2 số, tính tổng dãy Fibonaci, …

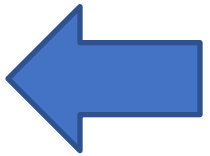
Các phương pháp mô tả giải thuật

1. Lưu đồ
2. Mã giả
3. Mã tự nhiên

Các ký hiệu lưu đồ

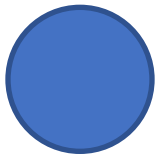
Bắt đầu/ kết thúc

Rẽ nhánh



Giá trị trả về

Luồng xử lý



Điểm nối

Khối xử lý

Nhập/ Xuất

Ký hiệu mã giả

- IF <điều kiện> THEN …ENDIF
- IF <điều kiện> THEN … ELSE … ENDIF
- WHILE <điều kiện> DO … ENDWHILE
- DO … UNTIL <điều kiện>
- DISPLAY …
- RETURN …

Ví dụ mô tả giải thuật

Tìm ước số chung lớn nhất của 2 số nguyên dương a và b

- Đầu vào: 2 số nguyên dương a và b
- Đầu ra: ước số chung lớn nhất của a và b

Mô tả bằng mã tự nhiên

Bước 1: Nếu a = b thì kết luận a là ước số chung lớn nhất, kết thúc

Bước 2: Nếu a > b thì a = a – b;

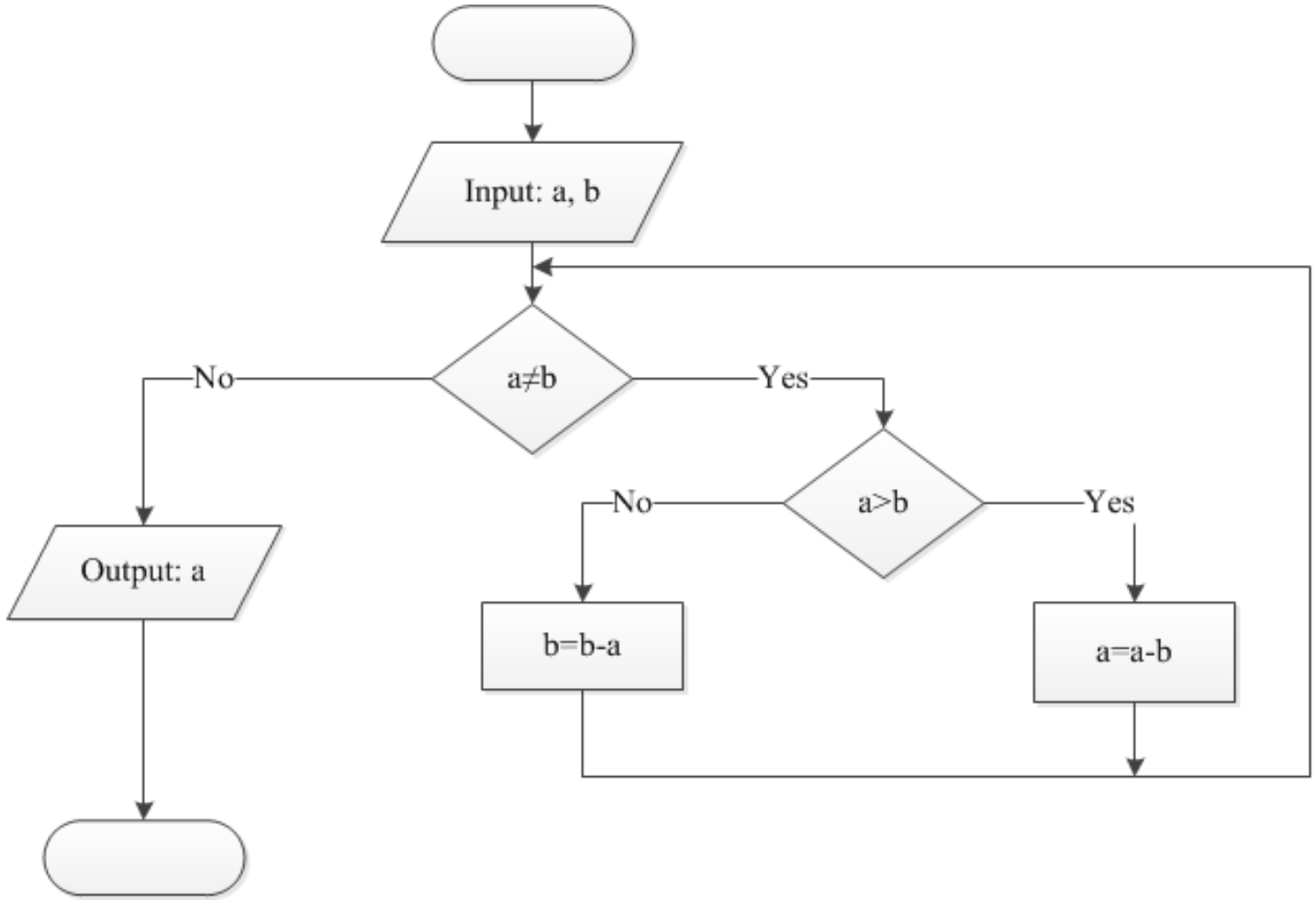
Ngược lại thì b = b – a;

Bước 3: Quay trở lại Bước 1

Mô tả bằng mã giả

```
WHILE a ≠ b DO
      IF a>b THEN
            a=a-b
      ELSE
            b=b-a
      ENDIF
ENDWHILE
DISPLAY a
```

Mô tả bằng lưu đồ



Bài tập

Bài toán: In ra tất cả các ước số của số nguyên dương

1. Mô tả giải thuật
2. Viết hàm

Đặc trưng của giải thuật

1. Tính đúng đắn
2. Tính dừng
3. Tính xác định
4. Tính hiệu quả
5. Tính phổ quát

Đặc trưng của giải thuật

[1] Tính đúng đắn *

Đảm bảo kết quả đúng sau khi thực hiện đối với bộ dữ liệu đầu vào

[2] Tính dừng

Dừng → Sau một vài bước thực hiện

Đặc trưng của giải thuật

[3] Tính xác định

- Rõ ràng, cụ thể

- Không nhập nhằng, gây hiểu lầm → hiểu, cài đặt

[4] Tính hiệu quả

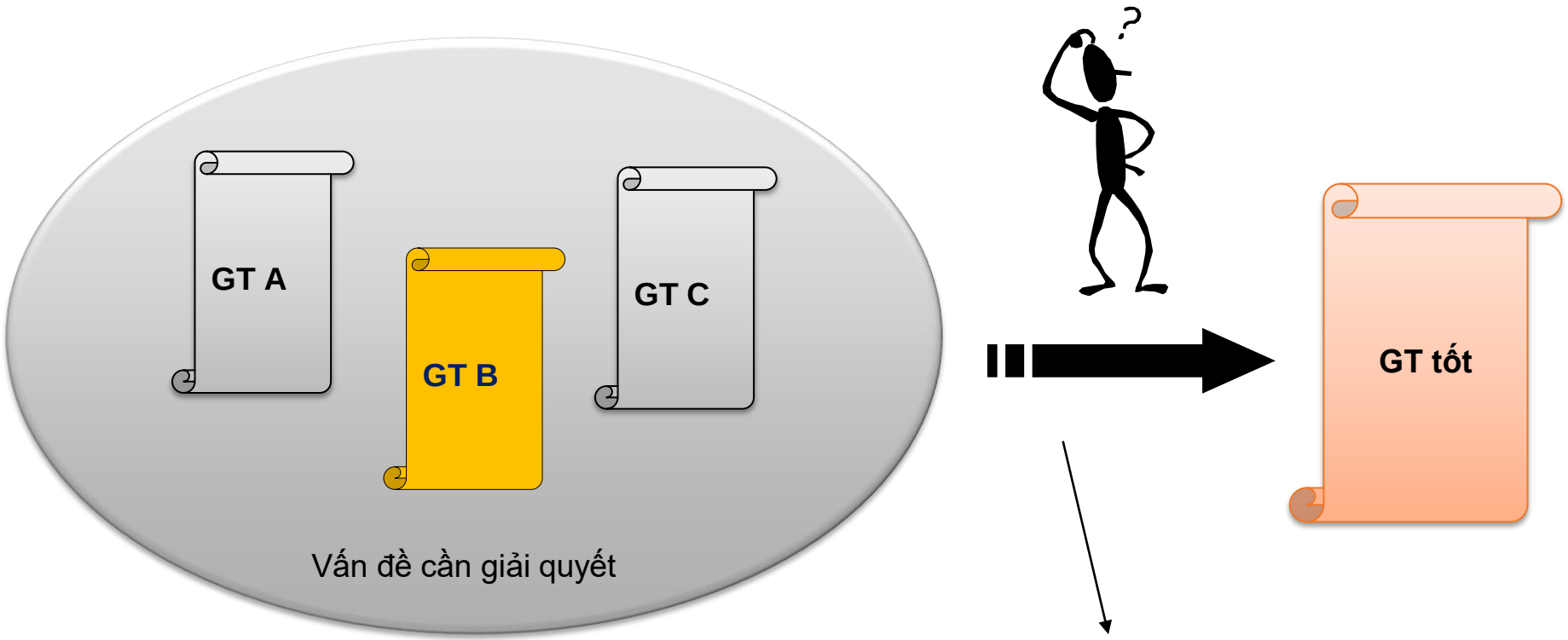
- Giải quyết trong thời gian, điều kiện cho phép

- Đáp ứng yêu cầu người dùng

[5] Tính phổ quát

Có thể giải quyết được một lớp bài toán tương tự

Đánh giá
độ phức tạp của giải thuật



1. Giải thuật đúng
2. Giải thuật đơn giản
3. Giải thuật thực hiện nhanh và tài nguyên sử dụng

Đánh giá
độ phức tạp của giải thuật

Những khó khăn

1. Phụ thuộc vào ngôn ngữ lập trình
2. Phụ thuộc vào người lập trình
3. Phụ thuộc vào bộ dữ liệu thử
4. Phụ thuộc vào phần cứng

Đánh giá
độ phức tạp của giải thuật

Thông thường so sánh các thuật toán dựa vào độ phức tạp về thời gian thực thi: độ phức tạp của giải thuật (algorithm complexity)

→ *ước lượng số phép tính cần thực hiện*

Đánh giá
độ phức tạp của giải thuật

- *Input: Số nguyên dương (n)*
- *Output: Tổng từ 1 đến n (s)*
- *Algorithm:*

```
i=1, s=0
WHILE i<=n DO
    s=s+i;
ENDWHILE
RETURN s
```

Thời gian thực thi chương trình

- Thời gian thực hiện một chương trình là một hàm của kích thước dữ liệu vào, ký hiệu T(N) trong đó N là kích thước (độ lớn) của dữ liệu vào
- *VD: Chương trình tính tổng của N số có thời gian thực hiện (số phép tính) là T(N) = c.N trong đó c là một hằng số*
- Thời gian thực thi chương trình: $T(n) \geq 0, \forall n \geq 0$

Khái niệm
độ phức tạp giải thuật

- Hàm T(N) có **tỷ suất tăng (growth rate)** là f(n) nếu tồn tại các hằng số c và N_0 sao cho $T(N) \leq f(N)$ với mọi $N \geq N_0$
- VD: Giả sử T(0) = 1, T(1) = 4 và tổng quát $T(N) = (N+1)^2$. Đặt $N_0 = 1$ và c = 4 thì với mọi $N \geq 1$ chúng ta dễ dàng chứng minh rằng $T(N) = (N+1)^2 \leq 4N^2$ với mọi $N \geq 1$, tức là tỷ suất tăng của T(N) là N^2
- VD: Tỷ suất tăng của hàm $T(N) = 3N^3 + 2N^2$ là N^3. Đặt $N_0 = 0$ và c = 5 ta dễ dàng chứng minh rằng với mọi $N \geq 0$ thì $3N^3 + 2N^2 \leq 5N^3$

Khái niệm
độ phức tạp giải thuật

- Hàm T(N) là O(f(N)) (hay T(N)=O(f(N) – T(N) có **độ phức tạp/ có tỷ suất tăng** là f(N)) nếu tồn tại hằng số c và N_0

$$\forall N > N_0; T(N) < c.f(N)$$

- Độ phức tạp giải thuật là một hàm chặn trên của hàm thời gian

VD: **$\log_2 n$, n, $n\log_2 n$, n^2, n^3, n!, n^n**

Quy tắc tính
độ phức tạp giải thuật

(1) Quy tắc cộng

Nếu T1(N) và T2(N) là thời gian thực hiện của hai đoạn chương trình P1 và P2; và T1(N)=O(f(N)), T2(N)=O(g(N)

→ thời gian thực hiện của đoạn hai chương trình đó **nối tiếp nhau** là

$$T(N)=O(\mathbf{max(f(N), g(N))})$$

Quy tắc tính
độ phức tạp giải thuật

(1) Quy tắc cộng

- *Input: Số nguyên dương n,m*
- *Output:* $p = \frac{1+2+3+\cdots+n}{m!}$

Algorithm:

```
i=1, s=0, p=1
WHILE i<=n DO
    s=s+i;
    i=i+1;
ENDWHILE
i=1;
WHILE i<=m DO
    p=p*i;
    i=i+1;
ENDWHILE
RETURN s/p
```

Quy tắc tính
độ phức tạp giải thuật

(2) Quy tắc nhân

Nếu T1(N) và T2(N) là thời gian thực hiện của hai đoạn chương trình P1và P2 và T1(N) = O(f(N)), T2(N) = O(g(N))

→ thời gian thực hiện của đoạn hai đoạn chương trình đó **lồng nhau** là

$$T(N) = O(\mathbf{f(N).g(N)})$$

Quy tắc tính
độ phức tạp giải thuật

(2) Quy tắc nhân

- *Input: Ma trận a_{nxm}, số nguyên x*
- *Output:*
  - *True: nếu tồn tại x*
  - *False: không tồn tại x*

Algorithm:

```
i=0
WHILE i<n DO
    j=0
    WHILE j<m DO
            IF a(i,j)==x THEN
                    RETURN True
            ENDIF
            j=j+1
    ENDWHILE
    i=i+1
ENDWHILE
RETURN False
```

Các độ phức tạp của giải thuật thường gặp

- O(1) : Nếu T(n) là hằng số (T(n)=C)
- $O(\log_2 n)$: Độ phức tạp dạng logarit
- O(n) : Độ phức tạp tuyến tính
- $O(n\log_2 n)$: Độ phức tạp tuyến tính logarit
- $O(n^2)$, $O(n^3)$,…,$O(n^{\alpha})$: Độ phức tạp đa thức
- O(n!), $O(n^n)$

→ Thông thường thuật giải có độ phức tạp đa thức thì có thể cài đặt

→ Độ phức tạp ở mức hàm mũ thì phải cải tiến giải thuật!

Một số công thức thường dùng

$$\sum_{i=1}^{n} 1 = n$$

$$\sum_{i=a}^{b} 1 = \sum_{i=1}^{b} 1 - \sum_{i=1}^{a} 1 = b - a + 1$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=a}^{n} i = \frac{n(n+1) - (a-1)a}{2}$$

$$\sum_{i=1}^{n^2} i = \frac{n^2(n^2+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$$

Bài tập

Algorithm:

```
i = 0
n =1000000;
WHILE i<n DO
        print(i)
ENDWHILE
```

Số lượng phép tính T(n) = 3n + 2

Bài tập

Algorithm:

```
n =10;
x = 1
y = 2
z = 0
t = 0
WHILE i <= n DO
    WHILE j <= n DO
        z = x + y
        t = x - y
    ENDWHILE
ENDWHILE
```

Số lượng phép tính $T(n) = 4n^2 + 2n + 5$

Bài tập

Algorithm:

```
i = 1
n = 100
WHILE i <= n DO
    printf(i)
    i = i*2
ENDWHILE
```

Số lượng phép tính T(n) = 3log2(n) + 2

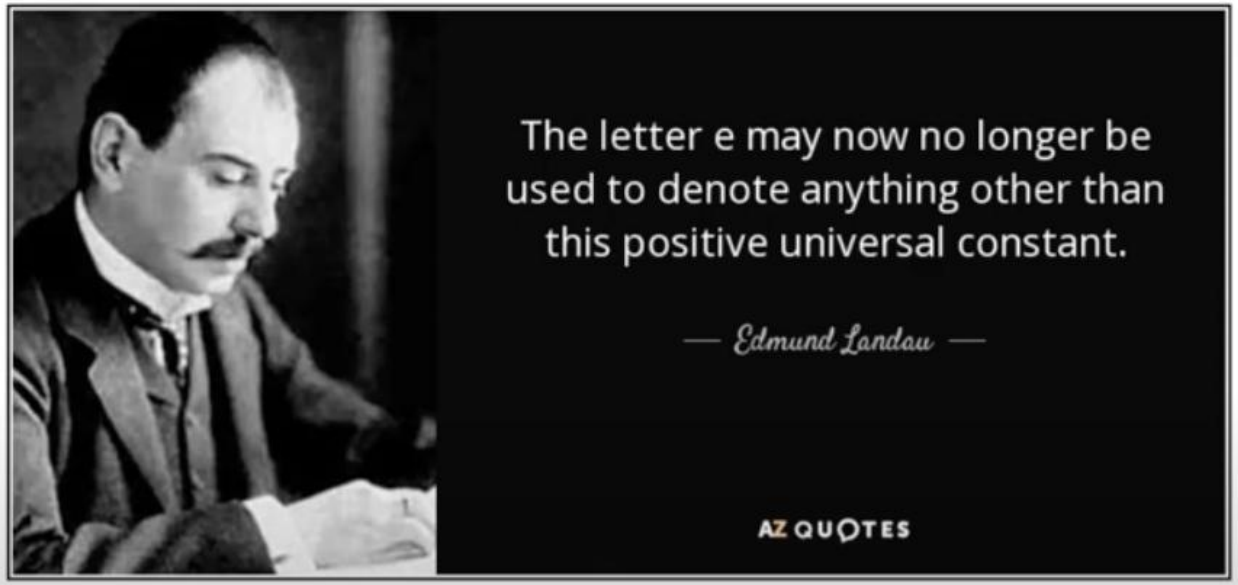
Bài tập

```
Bước 1: i = 1;
Bước 2:
      j = N;
      Trong khi (j > i) thực hiện:
            Nếu a[j]<a[j-1] thì
                  tạm = a[i];
                  a[i]=a[j];
                  a[j]=tạm;
            j = j-1;
Bước 3:
      i = i+1;
      Nếu  i = N: Dừng
      Ngược lại: Lặp lại Bước 2.
```

Phân tích độ phức tạp

Big O Notation (Ký hiệu O) - hay còn gọi ký hiệu Landau. Ký hiệu lấy từ tên nhà toán học Landau

Big O nghĩa là tốc độ tăng nhanh của hàm (The Rate of Growth of function)



Phân tích độ phức tạp

- Đơn giản hóa sự phức tạp

$$T(n) = 2 + 3n \longrightarrow T(n) = O(n)$$

$$T(n) = 4n^2 + 2n + 5 \longrightarrow T(n) = O(n^2)$$

$$T(n) = 3log_2(n) + 2 \longrightarrow T(n) = O(log_2(n))$$

- **Big0** tổng quát hóa mối quan hệ này thông qua loại bỏ những biến số đầu vào có ảnh hưởng nhỏ và giữ lại những biến số có ảnh hưởng lớn lên số lượng phép tính toán khi số lượng đầu vào tiến tới vô cùng.

Phân tích độ phức tạp

- Việc loại bỏ này có hợp lý?

$$T(n) = n^2 + 1000n + 6000$$

$$\downarrow$$

$$T(n) = O(n^2)$$

Việc loại bỏ này thể hiện rằng bigO là hàm mang tính chất tiệm cận (asymptotical)

Nên việc đo lường chỉ thể hiện đầy đủ ý nghĩa khi đầu vào rất lớn.

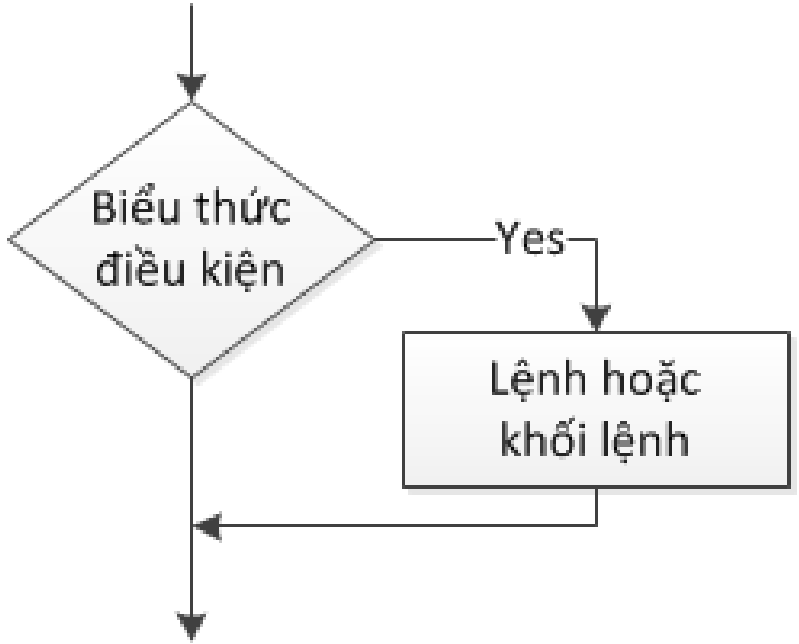
Cấu trúc điều khiển trong C

Cấu trúc rẽ nhánh

Cấu trúc rẽ nhánh chỉ cho phép thực hiện một dãy lệnh dựa vào kết quả của biểu thức điều kiện

Chỉ xét trường hợp đúng

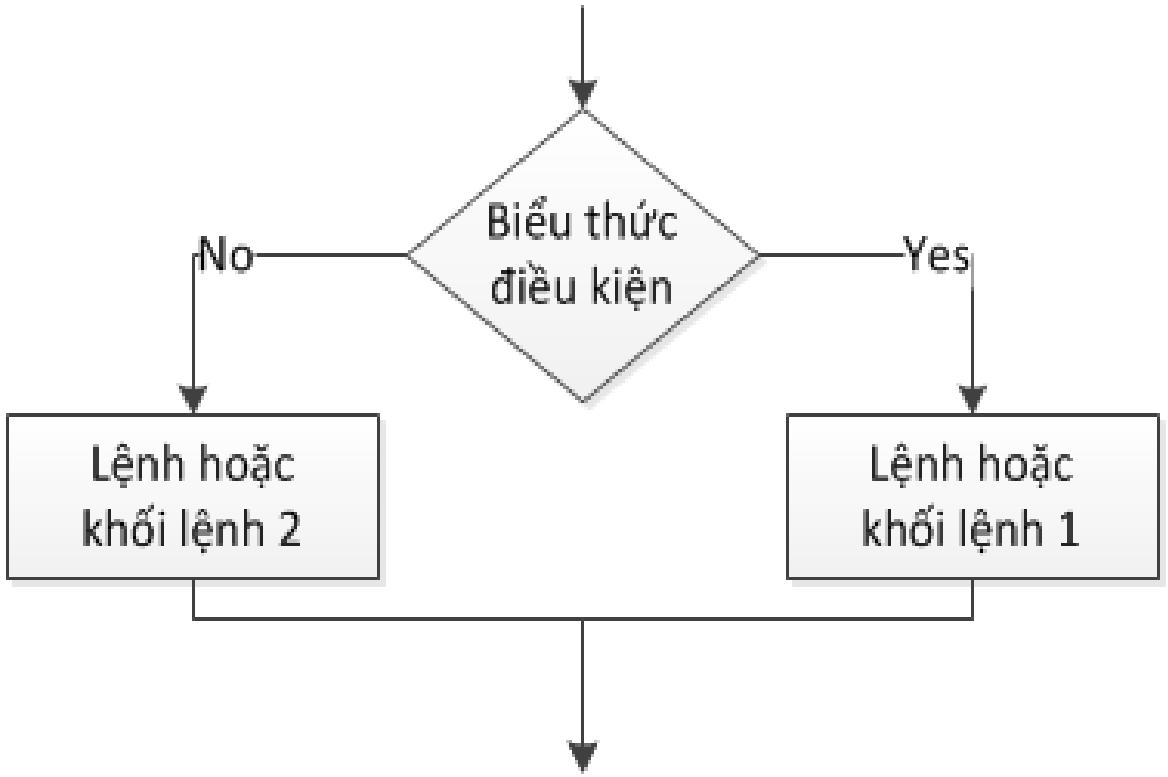
```
if (biểu thức điều kiện)
{
  <khối lệnh>;
}
```



*Nếu biểu thức điều kiện cho kết quả **true** thì thực hiện khối lệnh bên trong **if***

Cấu trúc rẽ nhánh

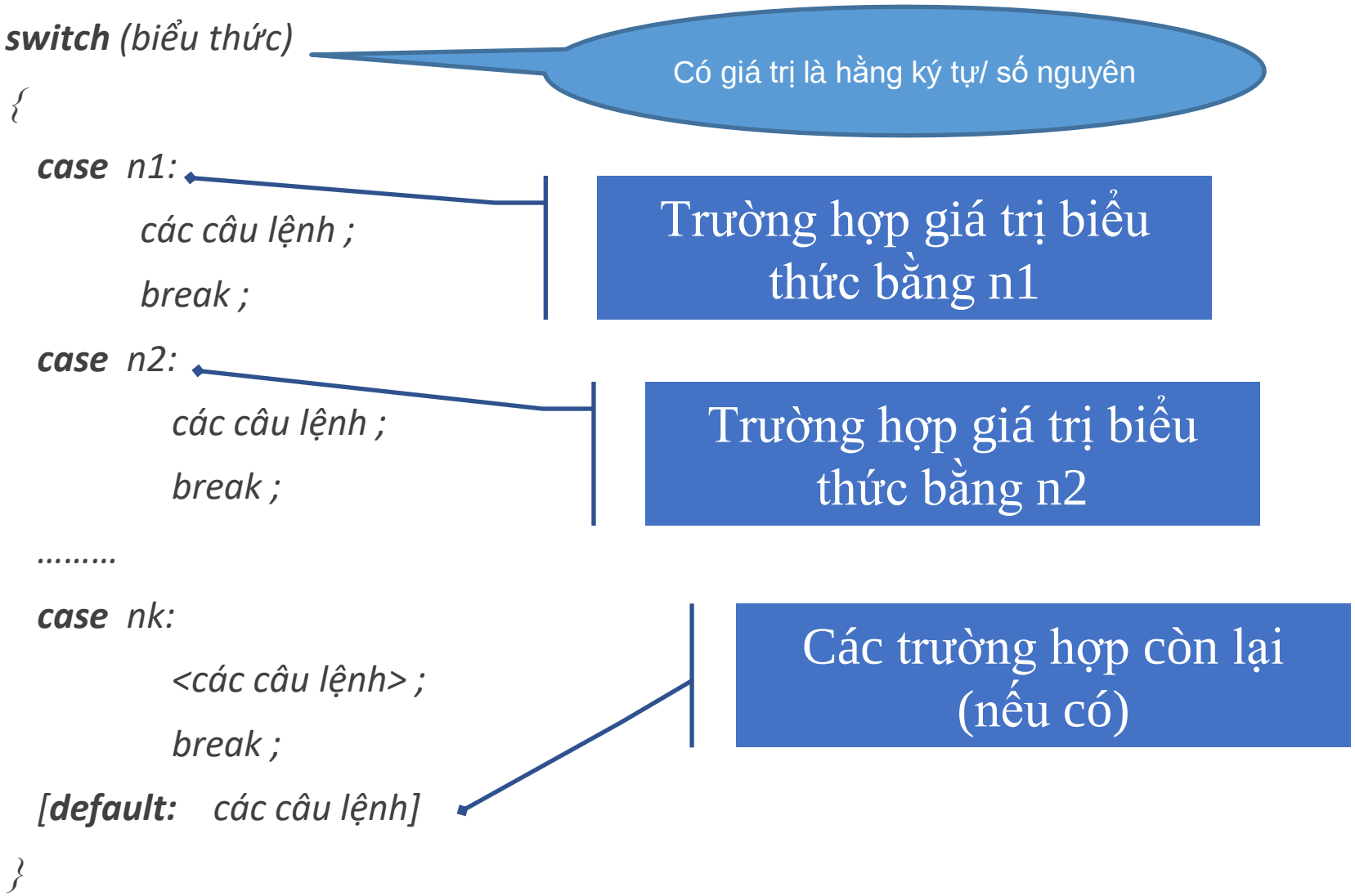
```
if (biểu thức điều kiện)
{
  <khối lệnh 1>;
}
else
{
  <khối lệnh 2>;
}
```



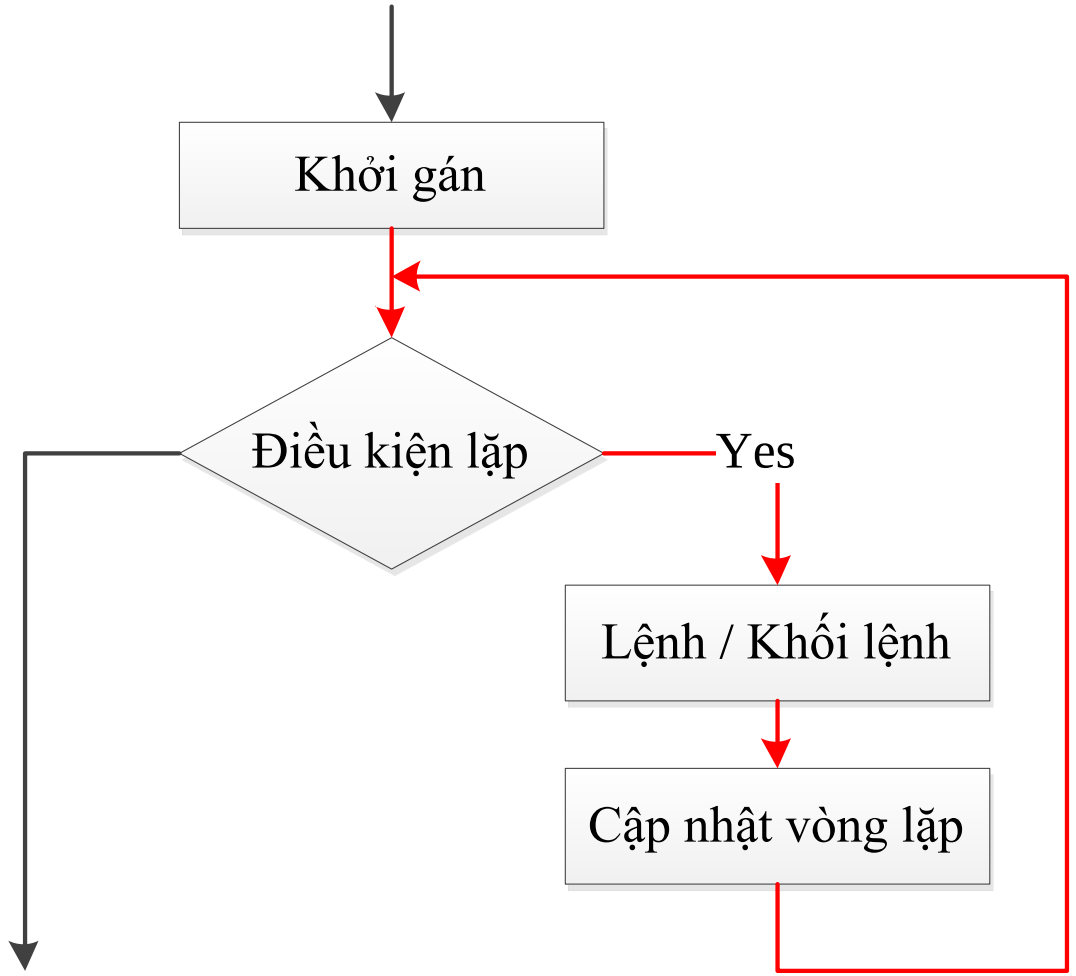
*Nếu biểu thức điều kiện cho kết quả **true** thì thực hiện khối lệnh 1, ngược lại thì cho thực hiện khối lệnh thứ 2*

Có thể lồng các cấu trúc if…else vào bên trong if/ else

Cấu trúc lựa chọn



Cấu trúc lặp



Cấu trúc lặp *while*

< Khởi gán>;

while *(<điều kiện lặp>)*

{

lệnh/ khối lệnh;

<cập nhật>;

}

Cấu trúc lặp *for*

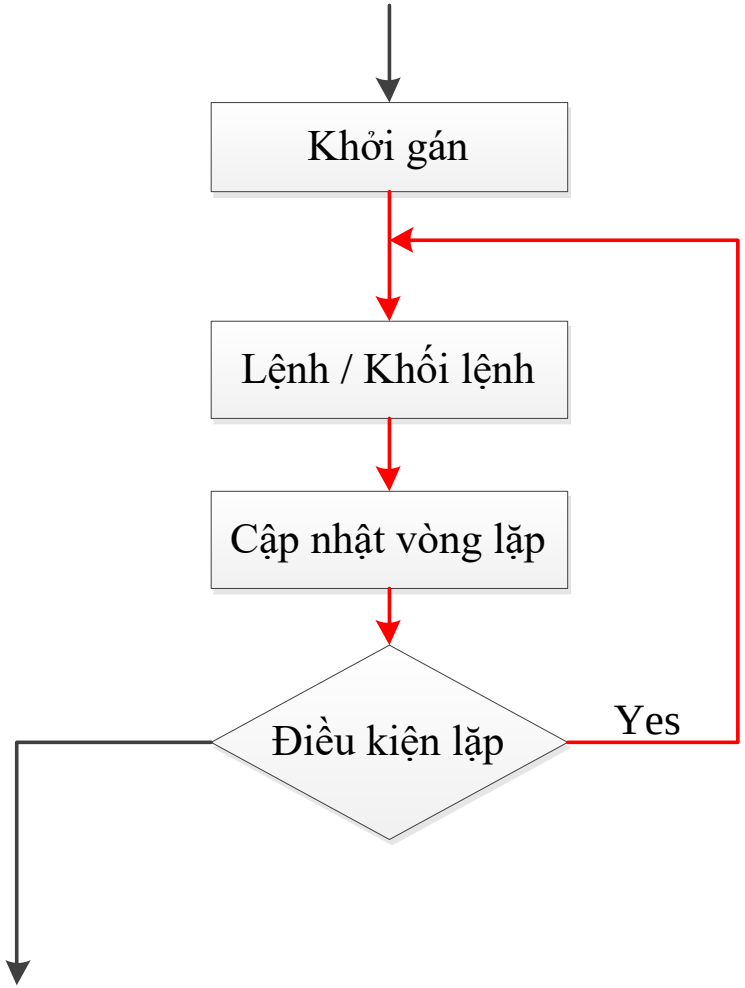
for *(<khởi gán>**;** <điều kiện lặp>**;** <cập nhật>)*

{

<khối lệnh>;

}

Cấu trúc lặp *do...while*




```
<khởi gán>;
do
{
        <khối lệnh>;
        <cập nhật>;

} while (điều kiện);
```

Bài tập

1. Viết chương trình C khai báo kiểu dữ liệu là mảng một chiều số nguyên (tối đa 100 phần tử), chương trình có các chức năng như sau:
   a) Nhập giá trị vào mảng.
   b) Xuất các phần tử trong mảng.
2. Hãy xây dựng và hiện thực kiểu dữ liệu trừu tượng cho thông tin sinh viên với các thao tác nhập, xuất thông tin của sinh viên.
3. Hãy xây dựng và hiện thực kiểu dữ liệu trừu tượng của số hữu tỉ a/b với các tác vụ cộng hai số hữu tỉ, nhân hai số hữu tỉ, chia hai số hữu tỉ.

Q&A