CHƯƠNG 1 – TỔNG QUAN VỀ CTDL VÀ THUẬT TOÁN

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Nội dung

1. Tổng quan về CTDL và thuật toán
2. Các tiêu chuẩn và vai trò của CTDL
3. Thuật toán & Độ phức tạp của thuật toán
4. Thực hiện và hiệu chỉnh chương trình
5. Tiêu chuẩn của chương trình

1. Tổng quan về CTDL và thuật toán

- Niklaus Wirth:

  CTDL + Thuật toán = Chương trình

- Cần nghiên cứu về thuật toán và CTDL!

1. Tổng quan về CTDL và thuật toán

- Theo *từ điển Tiếng Việt*: dữ liệu là số liệu, tư liệu đã có, được dựa vào để giải quyết vấn đề
- Trong *Tin học:* Dữ liệu dùng để biểu diễn các thông tin cần thiết cho bài toán.
- CTDL là cách tổ chức lưu trữ dữ liệu.

1. Tổng quan về CTDL và thuật toán

- **Thuật toán**: Là tập hợp (dãy, bước) hữu hạn các chỉ thị (hành động) được định nghĩa rõ ràng và có thể thực thi nhằm giải quyết một bài toán cụ thể nào đó. Quá trình hành động theo các bước này phải dừng và cho được kết quả như mong muốn.
- **Ví dụ**: Thuật toán tính tổng tất cả các số nguyên dương nhỏ hơn n gồm các bước sau:

Bước 1: *S=0, i=1;*

Bước 2: *nếu i<n thì s=s+i;*

Ngược lại: qua bước 4;

Bước 3:

i=i+1;

Quay lại bước 2;

Bước 4: *Tổng cần tìm là S.*

Input: Số nguyên dương N
Output: Tổng S

1. Tổng quan về CTDL và thuật toán

1. Viết thuật toán tính trung bình cộng các số trong dãy từ 1 đến n
2. Viết thuật toán in ra các số chẵn trong dãy từ 1 đến n

Bước 1: ……

Bước 2: ……

Bước 3: ……

Bước 4: ……

……

Input:……

Output:……

Sự Cần Thiết của Thuật Toán

- Tại sao sử dụng máy tính để xử lý dữ liệu?
  - Nhanh hơn.
  - Nhiều hơn.
  - Giải quyết những bài toán mà con người không thể hoàn thành được.
- Làm sao đạt được những mục tiêu đó?
  - Nhờ vào sự tiến bộ của kỹ thuật: tăng cấu hình máy ⇨ chi phí cao ☹
  - Nhờ vào các thuật toán hiệu quả: thông minh và chi phí thấp ☺

"Một máy tính siêu hạng vẫn không thể cứu vãn một thuật toán tồi!"

Các tính chất của thuật toán

- Tính chất cơ bản của thật toán:
  - Tính rõ ràng (xác định): các câu lệnh minh bạch (không mập mờ), được sắp xếp theo thứ tự nhất định và có khả năng thực thi.
  - Tính kết thúc (dừng): là sự hữu hạn các bước tính toán.
  - Tính chính xác (đúng): để đảm bảo kết quả tính toán hay các thao tác mà máy tính thực hiện được là chính xác.

Các tính chất của thuật toán

- Ngoài ra còn có những tính chất sau:
  - Tính hiệu quả: tính hiệu quả của thuật toán được đánh giá dựa trên một số tiêu chuẩn như khối lượng tính toán, không gian và thời gian khi thuật toán được thi hành.
  - Tính tổng quát: phải áp dụng được cho mọi trường hợp của bài toán.
  - Tính khách quan: được viết bởi nhiều người trên máy tính nhưng kết quả phải như nhau.
  - Tính phổ dụng: có thể áp dụng cho một lớp các bài toán có đầu vào tương tự nhau.

Biểu diễn Thuật toán

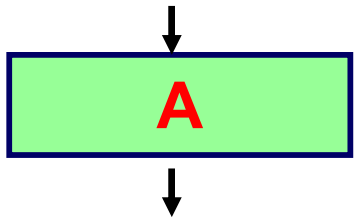
- Dạng ngôn ngữ tự nhiên
- Dạng lưu đồ (sơ đồ khối)
- Dạng mã giả
- Ngôn ngữ lập trình

Biểu diễn bằng Ngôn ngữ tự nhiên

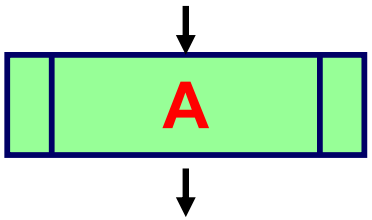
- NN tự nhiên thông qua các bước được tuần tự liệt kê để biểu diễn thuật toán.
- Ưu điểm:
  - Đơn giản, không cần kiến thức về về cách biểu diễn (mã giả, lưu đồ,...)
- Nhược điểm:
  - Dài dòng, không cấu trúc.
  - Đôi lúc khó hiểu, không diễn đạt được thuật toán.

Lưu đồ

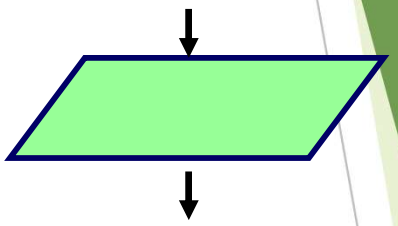
- Là hệ thống các nút, cung hình dạng khác nhau thể hiện các chức năng khác nhau.



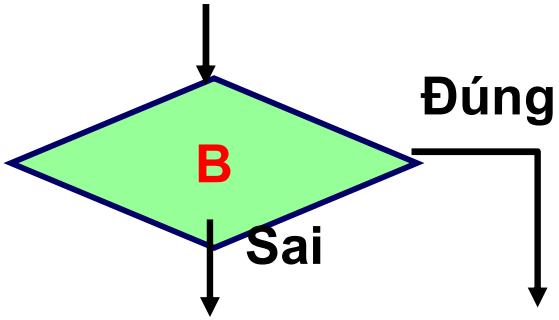
Thực hiện A



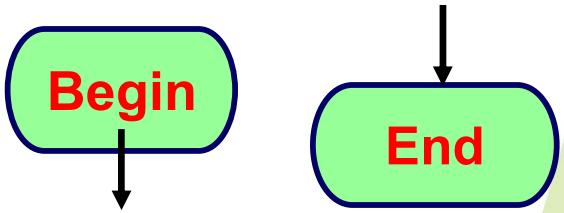
Gọi hàm A



Vào / Ra dữ liệu

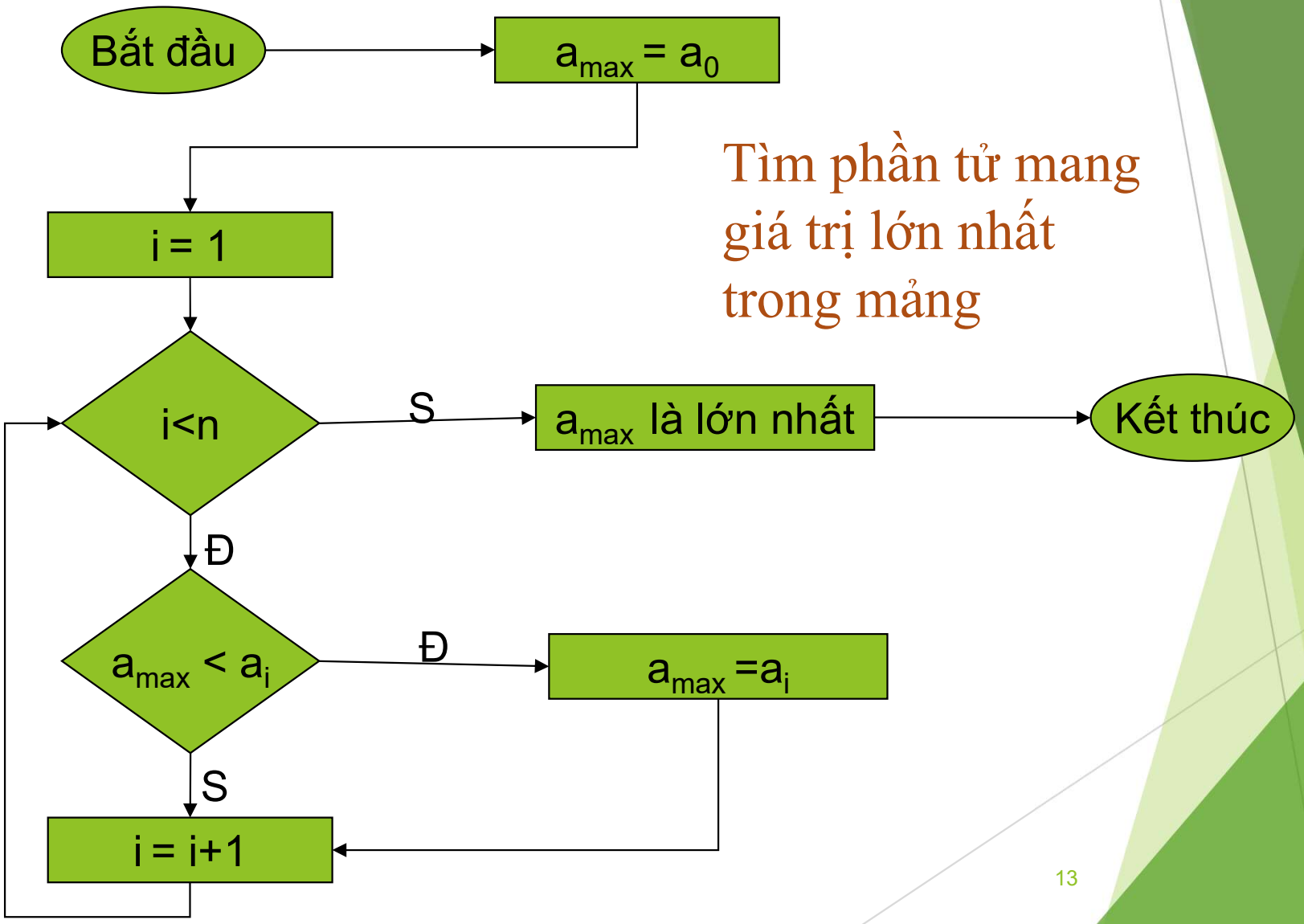


Điều kiện rẻ nhánh B



Nút giới hạn bắt đầu / kết thúc chương trình

Biểu diễn bằng Lưu đồ



Tìm phần tử mang giá trị lớn nhất trong mảng

Biểu Diễn Bằng Mã Giả

- Ngôn ngữ tựa ngôn ngữ lập trình:
  - Dùng cấu trúc chuẩn hóa, chẳng hạn tựa Pascal, C.
  - Dùng các ký hiệu toán học, biến, hàm.
- Ưu điểm:
  - Đỡ cồng kềnh hơn lưu đồ khối.
- Nhược điểm:
  - Không trực quan bằng lưu đồ khối.

Biểu diễn bằng Mã giả

- **Một số quy ước**

1. Các biểu thức toán học
2. Lệnh gán: "=" (A←B)
3. So sánh: "==", "!="
4. Khai báo hàm (thuật toán)

Thuật toán *<tên TT> (<tham số>)*

Input: *<dữ liệu vào>*

Output: *<dữ liệu ra>*

<Các câu lệnh>

End

Biểu diễn bằng Mã giả

5. Các cấu trúc:

Cấu trúc chọn:

if … **then** … [**else** …] **fi**

Vòng lặp:

while … **do**

do … **while** (…)

for … **do** … **od**

6. Một số câu lệnh khác:

Trả giá trị về: **return** [giá trị]

Lời gọi hàm: <Tên>(tham số)

Biểu Diễn Bằng Mã Giả

- **Ví dụ**: Tìm phần tử lớn nhất trong mảng một chiều.

$a_{max}=a_0$;
i=1;
while (i<n)
  if ($a_{max}<a_i$) $a_{max} = a_i$;
    i++;
end while;

Biểu Diễn Bằng Ngôn Ngữ Lập Trình

- Dùng ngôn ngữ máy tính (C, Pascal,...) để diễn tả thuật toán, CTDL thành câu lệnh.
- Kỹ năng lập trình đòi hỏi cần học tập và thực hành (nhiều).
- Dùng phương pháp tinh chế từng bước để chuyển hoá bài toán sang mã chương trình cụ thể.

2. Các tiêu chuẩn và vai trò của CTDL

- Các tiêu chuẩn của CTDL:
  - Phải biểu diễn đầy đủ thông tin.
  - Phải phù hợp với các thao tác trên đó.
  - Phù hợp với điều kiện cho phép của ngôn ngữ lập trình.
  - Tiết kiệm tài nguyên hệ thống.

2. Các tiêu chuẩn và vai trò của CTDL

- Vai trò của cấu trúc dữ liệu
  - Cấu trúc dữ liệu đóng vai trò quan trọng trong việc kết hợp và đưa ra cách giải quyết bài toán.
  - CTDL hỗ trợ cho các thuật toán thao tác trên đối tượng được hiệu quả hơn

3. Độ Phức Tạp Của Thuật Toán

- Một thuật toán hiệu quả:
  - Chi phí cần sử dụng tài nguyên thấp: Bộ nhớ, thời gian sử dụng CPU, …
- Phân tích độ phức tạp thuật toán:
  - **N** là khối lượng dữ liệu cần xử lý.
  - Mô tả độ phức tạp thuật toán qua một hàm **f(N).**
  - Hai phương pháp đánh giá độ phức tạp của thuật toán:
    - **Phương pháp thực nghiệm.**
    - **Phương pháp xấp xỉ toán học.**

Phương Pháp Thực Nghiệm

- Cài thuật toán rồi chọn các bộ dữ liệu thử nghiệm.
- Thống kê các thông số nhận được khi chạy các bộ dữ liệu đó.
- *Ưu điểm*: Dễ thực hiện.
- *Nhược điểm*:
  - Chịu sự hạn chế của ngôn ngữ lập trình.
  - Ảnh hưởng bởi trình độ của người lập trình.
  - Chọn được các bộ dữ liệu thử đặc trưng cho tất cả tập các dữ liệu vào của thuật toán: khó khăn và tốn nhiều chi phí.
  - Phụ thuộc vào phần cứng.

Phương Pháp Xấp Xỉ

- Đánh giá giá thuật toán theo hướng tiệm xấp xỉ tiệm cận qua các khái niệm O().
- *Ưu điểm*: Ít phụ thuộc môi trường cũng như phần cứng hơn.
- *Nhược điểm*: Phức tạp.
- Các trường hợp độ phức tạp quan tâm:
  - Trường hợp tốt nhất (phân tích chính xác)
  - Trường hợp xấu nhất (phân tích chính xác)
  - Trường hợp trung bình (mang tích dự đoán)

Sự Phân Lớp Theo Độ Phức Tạp Của Thuật Toán

- **Sử dụng ký hiệu BigO**
  - Hằng số : O(c)
  - logN: O(logN)
  - N : O(N)
  - NlogN : O(NlogN)
  - N^2 : $O(N^2)$
  - N^3 : $O(N^3)$
  - 2^N : $O(2^N)$
  - N! :O(N!)

Độ phức tạp tăng dần

Độ Phức Tạp Của Thuật Toán

Độ phức tạp tính toán của giải thuật: O(f(n))

Việc xác định độ phức tạp tính toán của giải thuật trong thực tế có thể tính bằng một số quy tắc đơn giản sau:

– Quy tắc bỏ hằng số:

$T(n) = O(c.f(n)) = O(f(n))$ với c là một hằng số dương

– Quy tắc cộng:

$T1(n) = O(f(n))$ $T2(n) = O(g(n))$

$T1(n) + T2(n) = O(f(n) + g(n))$

– Quy tắc lấy max:

$T(n) = O(f(n)+ g(m)) = O(max(f(n), g(m)))$

– Quy tắc nhân:

Đoạn chương trình có thời gian thực hiện $T(n)=O(f(n))$

Nếu thực hiện k(n) lần đoạn chương trình với $k(n) = O(g(n))$ thì độ phức tạp sẽ là $O(g(n).f(n))$

Độ Phức Tạp Của Thuật Toán

```
1 for (i = 0; i < n; i++)
2     for (j = 0; j < n; j++)
3         a[i][j] = 0;
4 for (i = 0; i < n; i++)
5     a[i][j] = 1;
```

$T_3 = O(1)$
$T_2 = O(n)$
$T_{23} = O(n) \times O(1) = O(n)$
$T_1 = O(n)$
$T_{123} = O(n) \times O(n) = O(n^2)$
$T_5 = O(1)$
$T_4 = O(n)$
$T_{45} = O(n) \times O(1) = O(n)$
$T_{12345} = T_{123} + T_{45} = O(n^2) + O(n) = O(n^2 + n) = O(n^2)$

Độ Phức Tạp Của Thuật Toán

```
1 for (i = 0; i < n; i++)
2     for (j = 0; j < n; j++)
3.        if (i == j)
4.           a[i][j] = 1;
5         else
6            a[i][j] = 0;
```

$T_4 = O(1)$
$T_6 = O(1)$
$T_3 = O(1)$
$T_{3456} = O(1)$
$T_2 = O(n)$
$T_{23456} = O(n) \times O(1) = O(n)$
$T_1 = O(n)$
$T_{12345} = O(n) \times O(n) = O(n^2)$

4. Thực Hiện Và Hiệu Chỉnh Chương Trình

- Chạy thử.
- Lỗi và cách sửa:
  - Lỗi thuật toán.
  - Lỗi trình tự.
  - Lỗi cú pháp.
- Xây dựng bộ test.
- Cập nhật, thay đổi chương trình theo yêu cầu (mới).

5. Tiêu Chuẩn Của Một Chương Trình

- Tính tin cậy
  - Giải thuật + Kiểm tra cài đặt
- Tính uyển chuyển
  - Đáp ứng quy trình làm phần mềm.
- Tính trong sáng
  - Dễ hiểu và dễ chỉnh sửa
- Tính hữu hiệu.
  - Tài nguyên + giải thuật

Câu hỏi và Bài tập

Cho bài toán sau:

Hãy kiểm tra N là số tăng hay giảm hay không tăng không giảm (với N là số nguyên)

Thực hiện các yêu cầu sau:

1. Trình bày Input/Output và các bước của thuật toán
2. Biểu diễn thuật toán bằng lưu đồ

Câu hỏi và Bài tập

3. Tính độ phức tạp thuật toán

```
1 sum = 0
2 for (i = 0; i < n; i++)
3    for (j = i + 1; j <= n; j++)
4       for (k = 1; k < 10; k++)
5          sum = sum + i * j * k;
```

```
1 sum = 0
2 for (i = 0; i < n; i++)
3    for (j = i + 1; j <= n; j++)
4       for (k = 1; k < m; k++) {
5          x = 2 * y
6          sum = sum + i * j * k;
7       }
```

```
1 sum = 0
2 thisSum = 0
3 for (i = 0; i < n; i++) {
4    thisSum += a[i];
5    if (thisSum > sum)
6       sum = thisSum;
7    else
8       thisSum = sum;
9 }
```

Câu hỏi và Bài tập

1. Trình bày tầm quan trọng của CTDL&GT?

2. Các tiêu chuẩn để đánh giá CTDL&GT?

3. Khi xây dựng giải thuật có cần quan tâm tới CTDL không? Tại sao?

4. Sử dụng các kiểu dữ liệu cơ bản trong C, xây dựng CTDL để lưu trữ đa thức có bậc tự nhiên n ($0 \le n \le 100$) trên trường số thực

$$(a_i, x \in R)$$

$$fn(x) = \sum_{i=0}^{n} a_i x^i$$

Với CTDL đã được xây dựng, trình bày thuật toán và cài đặt chương trình để thực hiện hàm fn(x)

Nội dung thuyết trình

Trình bày phương pháp và ví dụ cho các chiến lược thiết kế giải thuật sau:

1. Quay lui (Backtracking)
2. Nhánh và cận (Branch-and-bound)
3. Chia để trị (Divide-and-conquer)
4. Quy hoạch động (Dynamic programming)
5. Tham lam (Greedy)