CHƯƠNG 1

TỔNG QUAN VỀ CTDL VÀ THUẬT TOÁN

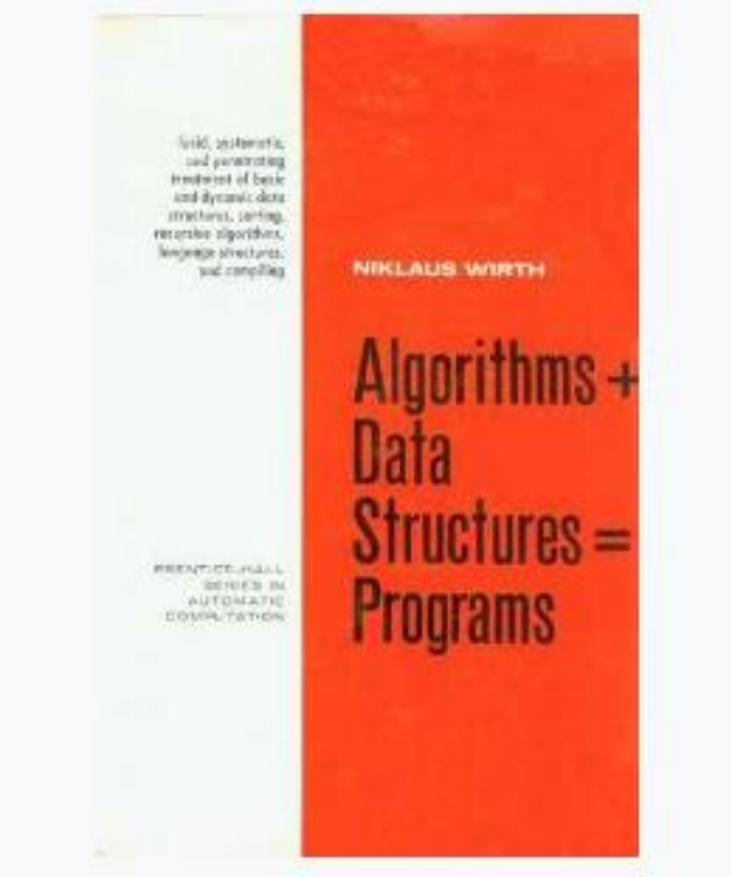
Nội Dung

- Khái niệm CTDL & thuật toán
- Các kiểu dữ liệu
- Vai trò của CTDL & thuật toán
- Các tiêu chuẩn đánh giá CTDL & thuật toán
- Độ phức tạp của thuật toán
- Thực hiện và hiệu chỉnh chương trình
- Tiêu chuẩn của chương trình

Khái Niệm Về CTDL Và Thuật Toán

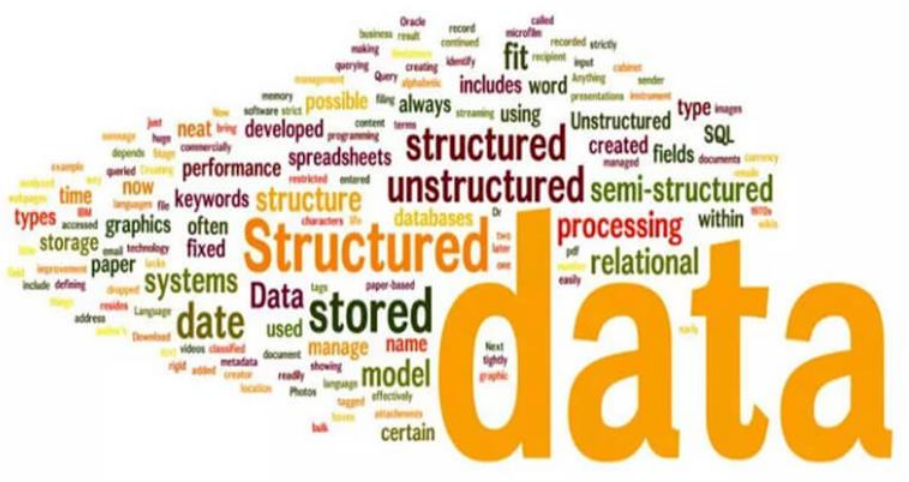
Algorithms + Data Structures = ????

Khái Niệm Về CTDL Và Thuật Toán



Algorithms + Data Structures = Programs[1] is a 1976 book written by Niklaus Wirth

Khái Niệm Về CTDL Và Thuật Toán



Dữ liệu (Data)

Kiểu dữ liệu (Data type)

Cấu trúc dữ liệu (Data structures)

Dữ Liệu

- Theo *từ điển Tiếng Việt*: Số liệu, tư liệu đã có, được dựa vào để giải quyết vấn đề
- *Tin học:* Là các thông tin cần thiết cho bài toán đã được chuyển sang dạng có hiệu quả để di chuyển hoặc xử lý .

Kiểu Dữ Liệu

Trong khoa học máy tính và lập trình máy tính, một kiểu dữ liệu (tiếng Anh: data type) hay đơn giản type là một cách phân loại dữ liệu cho trình biên dịch hoặc thông dịch hiểu các lập trình viên muốn sử dụng dữ liệu (Wikipedia).

+ Là tập hợp các giá trị mà một biến thuộc kiểu đó có thể nhận.
+ Trên đó xác định một số phép toán có thể thực hiện được.

Kiểu Dữ Liệu

- Kiểu dữ liệu cơ sở: giá trị dữ liệu của nó là đơn nhất.
  - Ví dụ: kiểu bool, integer, …
- Kiểu dữ liệu có cấu trúc: giá trị dữ liệu của nó là sự kết hợp của các giá trị khác (cơ sở).
  - Ví dụ: array.
- Kiểu dữ liệu trừu tượng: là một mô hình toán học cùng với một tập hợp các phép toán trừu tượng được định nghĩa trên mô hình đó.
  - Ví dụ tập hợp số nguyên cùng với các phép toán hợp, giao, hiệu.

Cấu Trúc Dữ Liệu

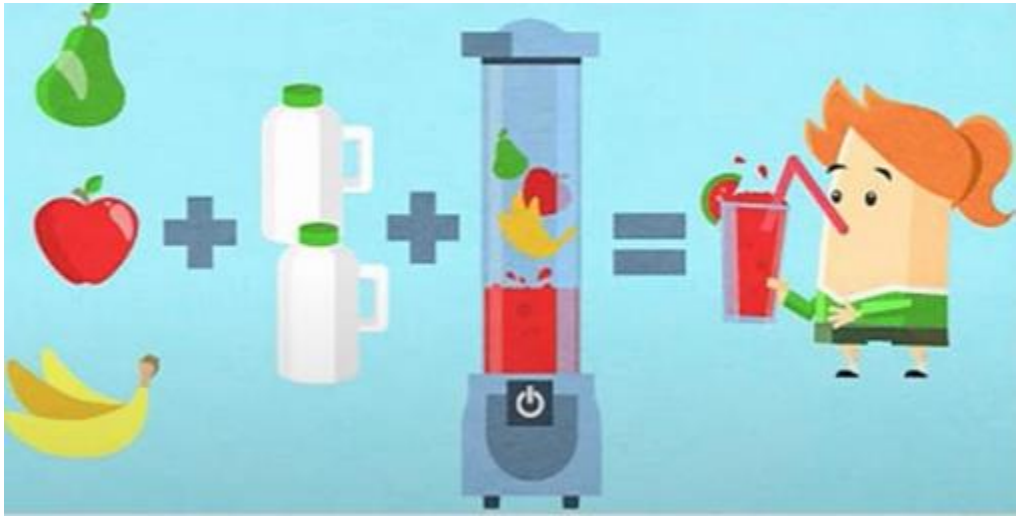
- ***Khái niệm chung***: CTDL là một cách thể hiện và tổ chức dữ liệu trong máy tính sao cho nó được sử dụng một cách có hiệu quả nhất.
- ***Khái niệm khác***: CTDL là một dữ liệu phức hợp, gồm nhiều thành phần dữ liệu, mỗi thành phần hoặc là dữ liệu cơ sở hoặc là một CTDL đã được xây dựng. Các thành phần dữ liệu tạo nên một CTDL được liên kết với nhau theo một cách nào đó.

- Các tiêu chuẩn của CTDL:
  - Phải biểu diễn đầy đủ thông tin của bài toán (input/output)→ phản ánh đúng thực tế
  - Phải phù hợp với các thao tác của thuật toán mà ta chọn để giải quyết bài toán→ phát triển thuật toán đơn giản và đạt hiệu quả
  - Phù hợp với điều kiện cho phép của NNLT.
  - Tiết kiệm tài nguyên hệ thống.

Vai Trò Của Cấu Trúc Dữ Liệu

- CTDL 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

Thuật toán là gì?



01

KHÁI NIỆM VỀ VẤN ĐỀ/BÀI TOÁN

03

CÁC BƯỚC GIẢI QUYẾT VẤN ĐỀ/BÀI TOÁN TRÊN MÁY TÍNH

04

KHÁI NIỆM THUẬT TOÁN

CÁC PHƯƠNG PHÁP BIỂU DIỄN THUẬT TOÁN

ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

05

02

01 KHÁI NIỆM VỀ VẤN ĐỀ/BÀI TOÁN

02 CÁC BƯỚC GIẢI QUYẾT VẤN ĐỀ/BÀI TOÁN TRÊN MÁY TÍNH

03 KHÁI NIỆM THUẬT TOÁN

04 CÁC PHƯƠNG PHÁP BIỂU DIỄN THUẬT TOÁN

05 ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

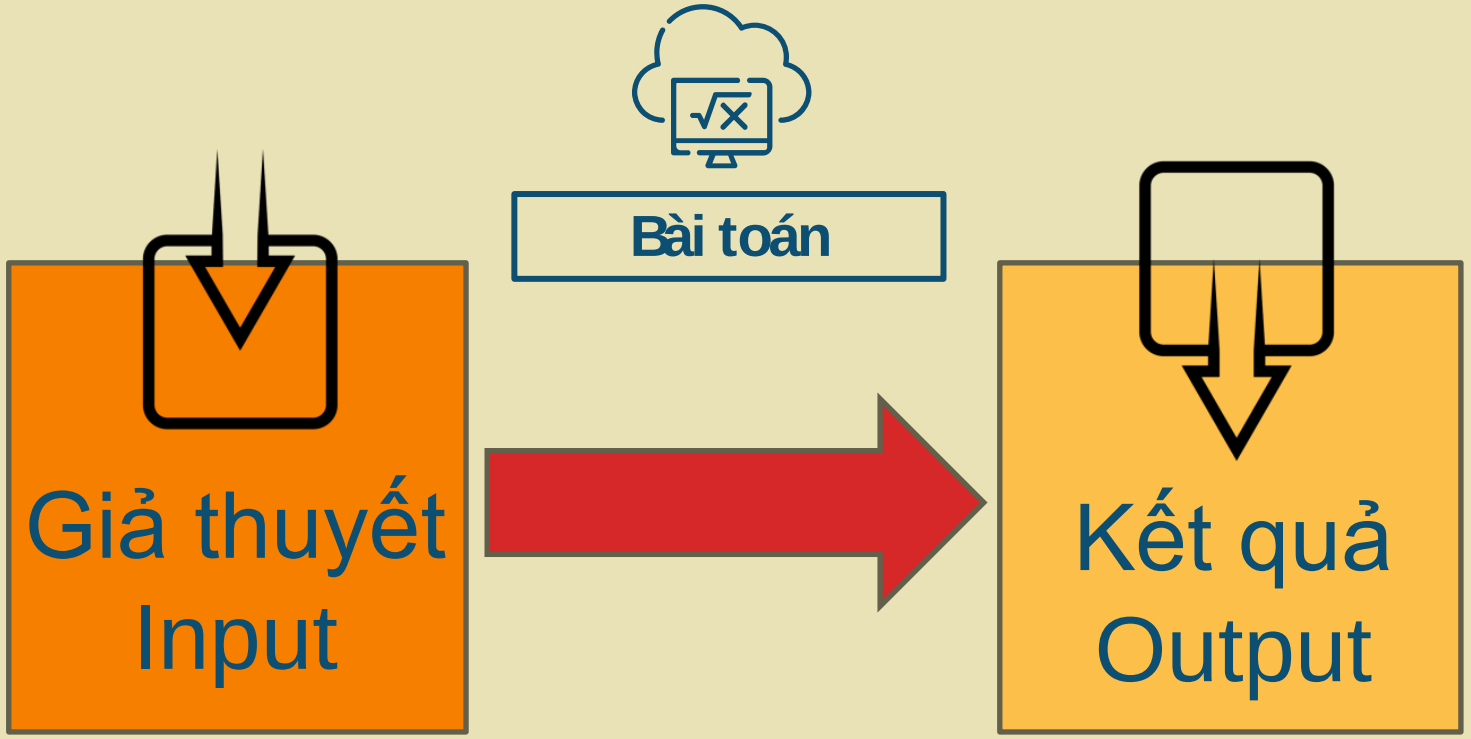
KHÁI NIỆM VỀ VẤN ĐỀ/BÀI TOÁN

KHÁI NIỆM VỀ VẤN ĐỀ/BÀI TOÁN

Bài toán

Vấn đề được giải quyết bằng phương pháp tính toán

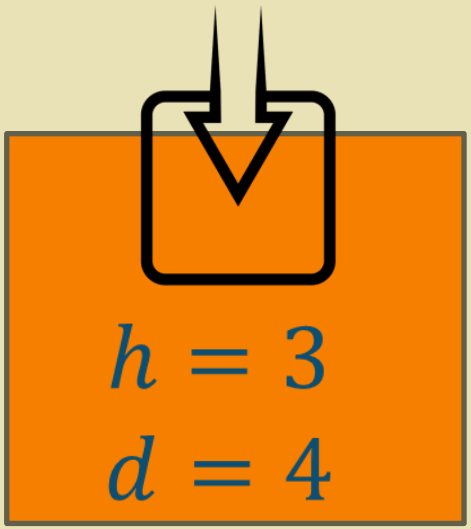
KHÁI NIỆM VỀ VẤN ĐỀ/ BÀI TOÁN



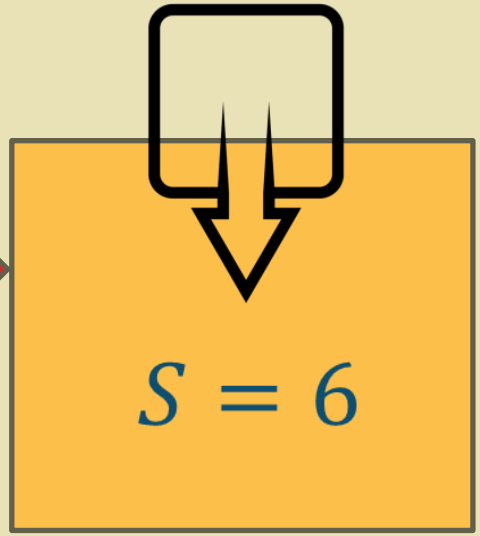
KHÁI NIỆM VỀ VẤN ĐỀ/BÀI TOÁN

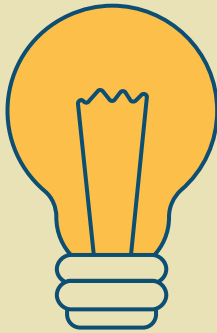
Tính diện tích hình tam giác

Chiều dài đáy và chiều cao



Diện tích tam giác

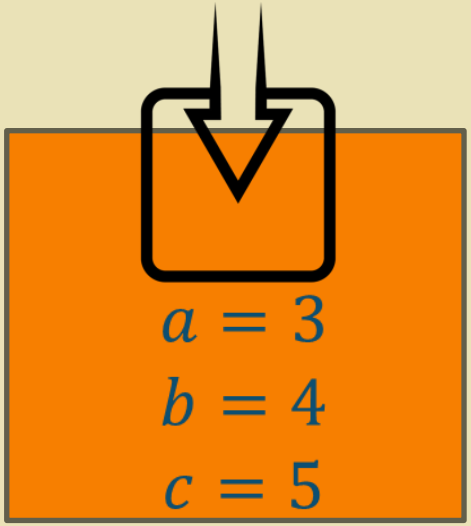




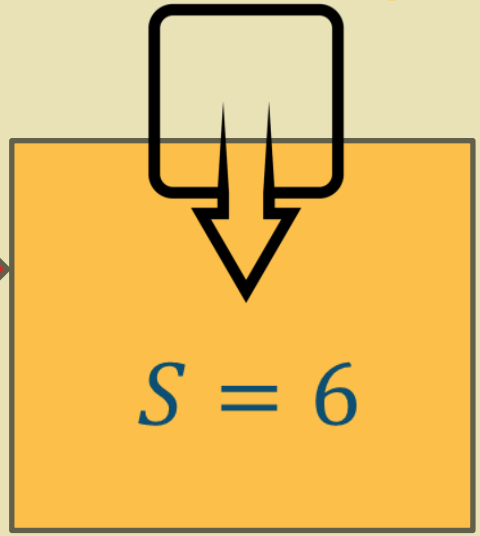
KHÁI NIỆM VỀ VẤN ĐỀ/BÀI TOÁN

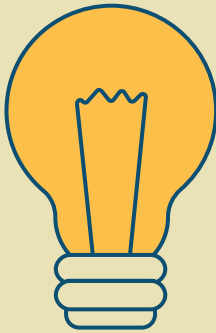
Tính diện tích hình tam giác

Độ dài 3 cạnh tam giác



Diện tích tam giác





CÁC BƯỚC GIẢI QUYẾT VẤN ĐỀ/BÀI TOÁN TRÊN MÁY TÍNH

CÁC BƯỚC GIẢI QUYẾT VẤN ĐỀ/BÀI TOÁN TRÊN MÁY TÍNH

Xác định vấn đề/ bài toán

Cho một dãy số A có n phần tử được sắp xếp tăng dần và số k. Tìm 2 số $x_1, x_2 \in A$ bất kì sao cho $x_1 + x_2 = k$.

- Dòng đầu chứa 2 số n và k
- Dòng thứ hai gồm n số thuộc dãy số a

7 18
2 5 6 8 10 12

- Hai số x_1, x_2 là kết quả của bài toán

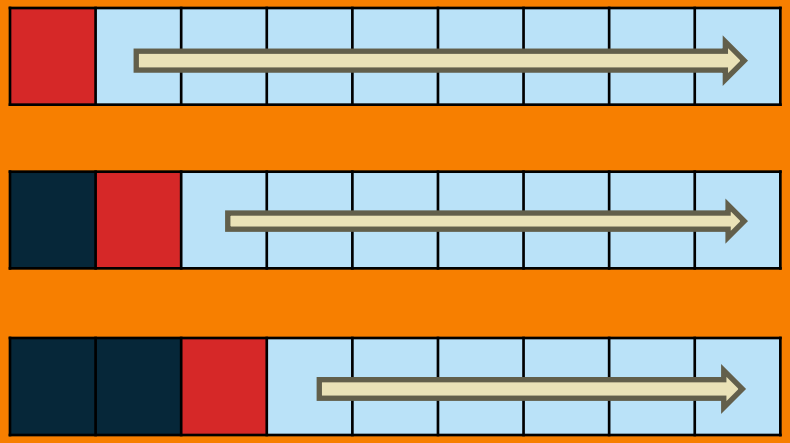
6 12

/#/ . ` / DỈL vÜồÇ ể b 7 □. "LÇh#bÇw<b a #òÇNbI

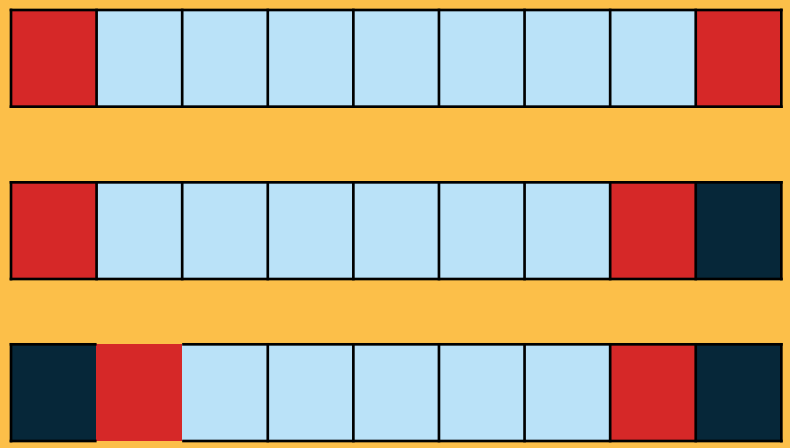
Xác định vấn đề/ bài toán

Lựa chọn giải pháp

VÉT CẠN

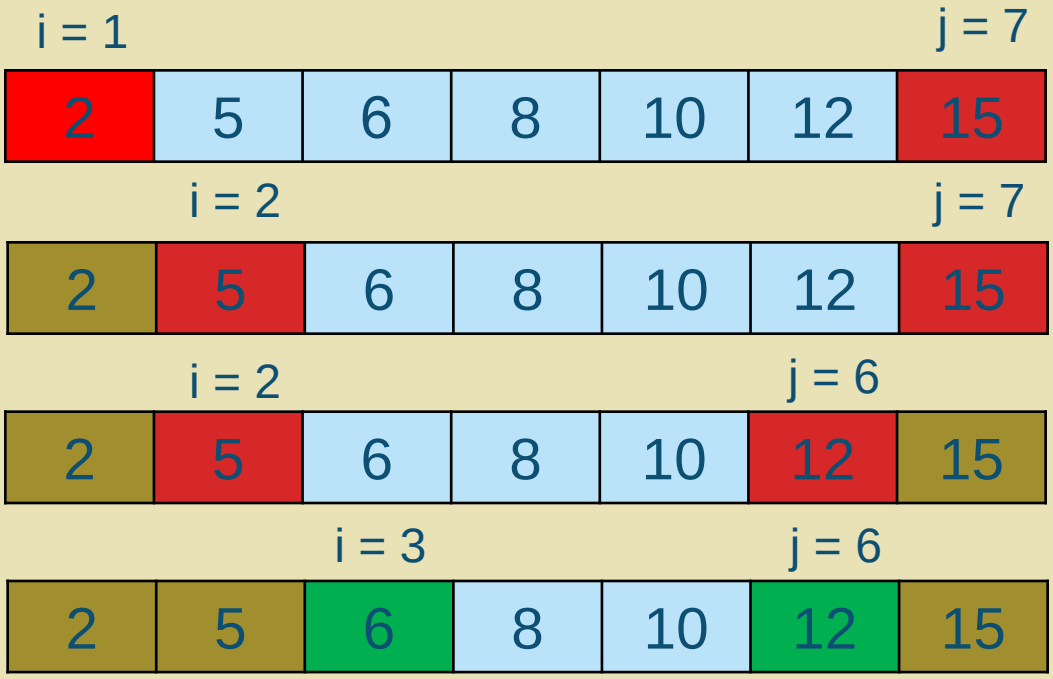


HAI CON TRỎ



CÁC BƯỚC GIẢI QUYẾT VẤN ĐỀ/BÀI TOÁN TRÊN MÁY TÍNH

Xác định vấn đề/ bài toán → Lựa chọn giải pháp → Xây dựng thuật toán



$2 + 15 < 18 \Rightarrow i = i + 1$

$5 + 15 > 18 \Rightarrow j = j - 1$

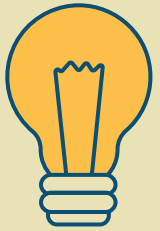
$5 + 12 < 18 \Rightarrow i = i + 1$

$6 + 12 = 18$

CÁC BƯỚC GIẢI QUYẾT VẤN ĐỀ/BÀI TOÁN TRÊN MÁY TÍNH

Xác định vấn đề/ bài toán → Lựa chọn giải pháp → Xây dựng thuật toán → Cài đặt chương trình

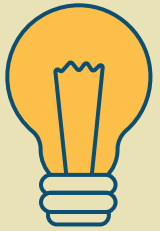
```cpp
#include <bits/stdc++.h>

using namespace std;
int a[10005];
int main()
{
    int n, k;
    cin >> n >> k;
    int i = 1, j = n;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    while (i < j) {
        if (a[i] + a[j] == k) {
            cout << a[i] << " " << a[j];
        return 0;
        }
        if (a[i] + a[j] < k)
            i += 1;
        else
            j -= 1;
    }
    return 0;
}
```



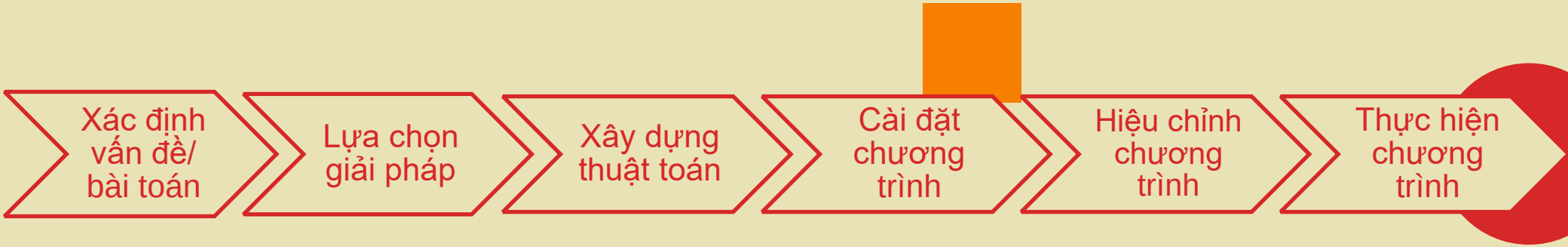
CÁC BƯỚC GIẢI QUYẾT VẤN ĐỀ/BÀI TOÁN TRÊN MÁY TÍNH

Xác định vấn đề/ bài toán → Lựa chọn giải pháp → Xây dựng thuật toán → Cài đặt chương trình → Hiệu chỉnh chương trình

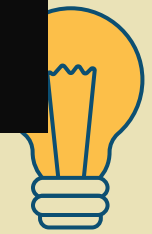
```cpp
#include <bits/stdc++.h>

using namespace std;
int a[10005];
int main()
{
    int n, k;
    cin >> n >> k;
    int i = 1, j = n;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    while (i < j) {
        if (a[i] + a[j] == k) {
            cout << a[i] << " " << a[j];
        return 0;
        }
        if (a[i] + a[j] < k)
            i += 1;
        else
            j -= 1;
    }
    return 0;
}
```

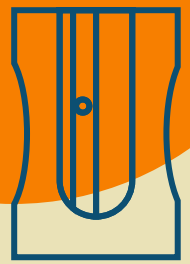


CÁC BƯỚC GIẢI QUYẾT VẤN ĐỀ/BÀI TOÁN TRÊN MÁY TÍNH



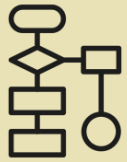

```
7 18
2 5 6 8 10 12 15
6 12
Process returned 0 (0x0)   execution time : 13.597 s
Press any key to continue.
```





KHÁI NIỆM THUẬT TOÁN

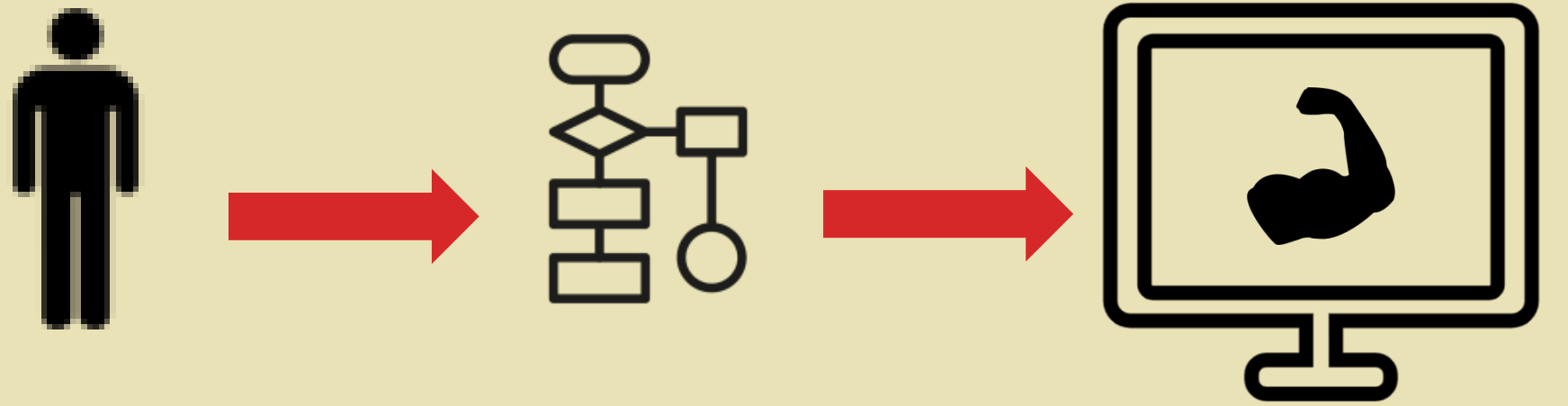
KHÁI NIỆM THUẬT TOÁN



Thuật toán

Một dãy hữu hạn các thao tác cần thực hiện để giải quyết một bài toán nào đó

SỰ CẦN THIẾT CỦA THUẬT TOÁN



Con người

Thuật toán

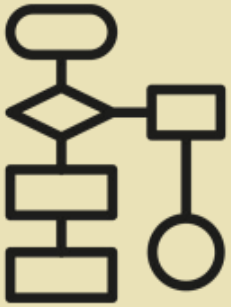
Sức mạnh máy tính

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 tiêu chuẩn của thuật toán



Thuật toán

- Tính chính xác/đúng
- Tính xác định/khách quan
- Tính hữu hạn/kết thúc
- Tính tổng quát/phổ dụng
- Tính rõ ràng/hiệu quả

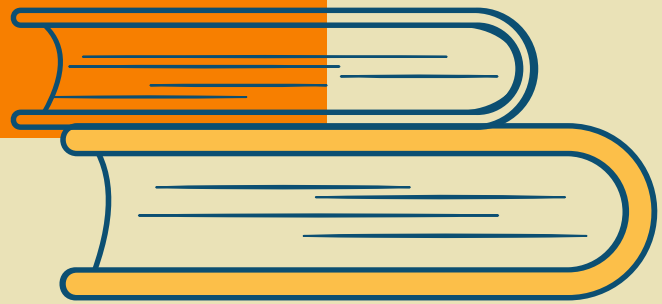
Các tiêu chuẩn của thuật toán

- Tính chính xác/đúng:
  - Quá trình tính toán hay các thao tác máy tính thực hiện là chính xác.
  - Khi kết thúc, giải thuật phải cung cấp kết quả đúng đắn.
- Tính phổ dụng/tổng quát:
  - Có thể áp dụng cho một lớp các bài toán có đầu vào tương tự nhau.
- Tính kết thúc/hữu hạn:
  - Thuật toán phải dừng sau một số bước hữu hạn.

Các tiêu chuẩn của thuật toán

- Tính rõ ràng/hiệu quả:
  - Các câu lệnh minh bạch được sắp xếp theo thứ tự nhất định.

- Tính khách quan/xác định:
  - Được viết bởi nhiều người trên máy tính nhưng kết quả phải như nhau.
  - Trong cùng một điều kiện hai bộ xử lý cùng thực hiện, thuật toán phải cho những kết quả giống nhau.



CÁC PHƯƠNG PHÁP BIỂU DIỄN THUẬT TOÁN

Biểu Diễn Thuật Toán

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

CÁC PHƯƠNG PHÁP BIỂU DIỄN THUẬT TOÁN

BÀI TOÁN MẪU

Cho một dãy số A có n phần tử được sắp xếp tăng dần và số k. Tìm vị trí của phần tử có giá trị k.

- Dòng đầu chứa 2 số n và k
- Dòng thứ hai gồm n số thuộc dãy số a

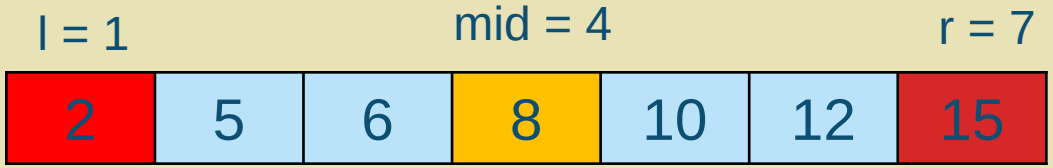
```
7 10
2 5 6 8 10 12
```

- Hai số x_1, x_2 là kết quả của bài toán

```
5
```

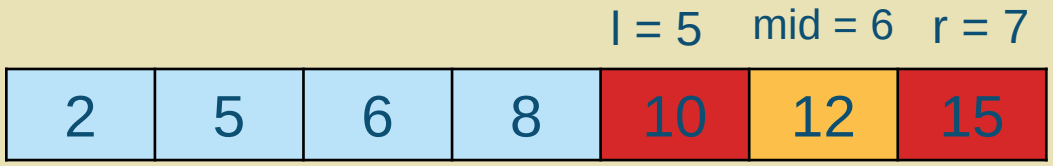
CÁC PHƯƠNG PHÁP BIỂU DIỄN THUẬT TOÁN

XÂY DỰNG THUẬT TOÁN



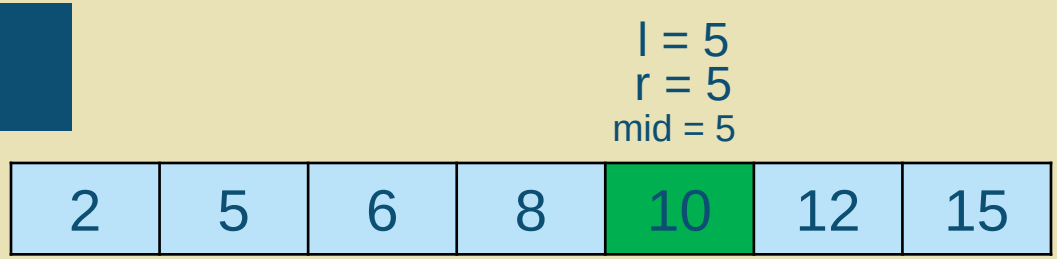
$$mid = (l + r) \div 2 = 4$$

$$\mathrm{A}[mid] < 10 \Rightarrow l = mid + 1$$



$$mid = (l + r) \div 2 = 6$$

$$\mathrm{A}[mid] > 10 \Rightarrow r = mid - 1$$



$$mid = (l + r) \div 2 = 5$$

$$\mathrm{A}[mid] = 10$$

CÁC PHƯƠNG PHÁP BIỂU DIỄN THUẬT TOÁN

NGÔN NGỮ TỰ NHIÊN

Bước 1: Xét phần tử trung vị $A[mid]$ trong đoạn $A[l, r]$. Nếu $l \geq r$, kết thúc chương trình. Nếu $A[mid] = k$, xuất ra kết quả.

Bước 2: Nếu $A[mid] < k$, lặp lại bước 1 trên đoạn $A[mid + 1, r]$.

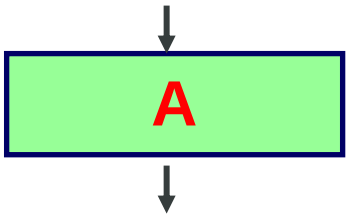
Bước 3: Lặp lại bước 1 trên đoạn $A[l, mid - 1]$.

Biểu Diễn Bằng Ngôn Ngữ Tự Nhiên

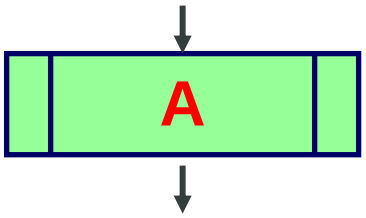
- NN tự nhiên thông qua các bước được tuần tự liệt kê để biểu diễn thuật toán.
- Ưu điểm:
  - Đơn giản, không cần kiến thức về về cách biểu diễn (mã giả, lưu đồ,...)
- Nhược điểm:
  - Dài dòng, không cấu trúc.
  - Đôi lúc khó hiểu, không diễn đạt được thuật toán.

Lưu Đồ

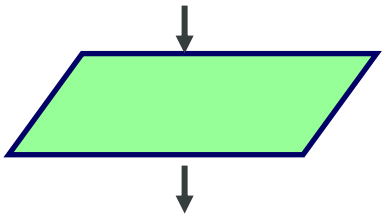
- Là hệ thống các nút, cung hình dạng khác nhau thể hiện các chức năng khác nhau.



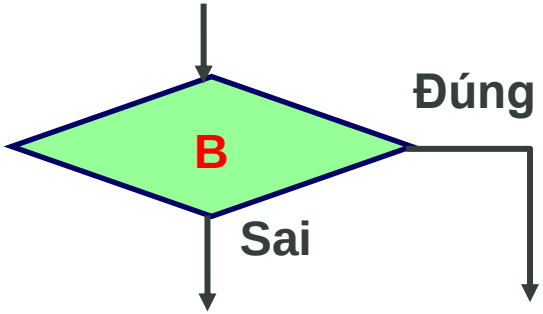
Thực hiện A



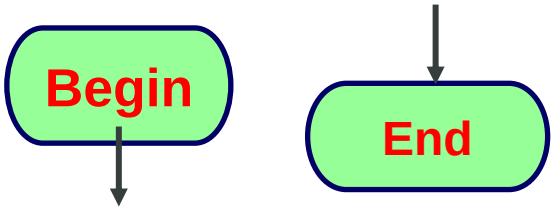
Gọi hàm A



Vào / Ra dữ liệu



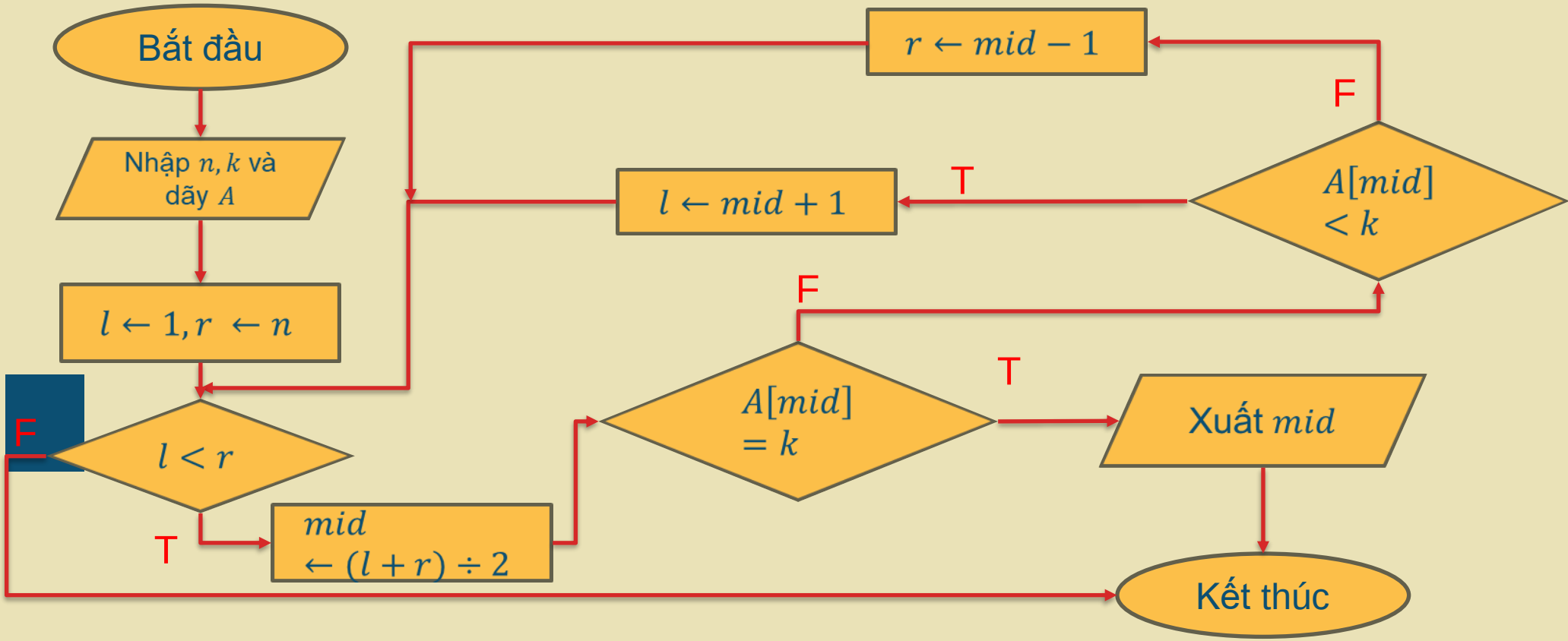
Điều kiện rẻ nhánh B



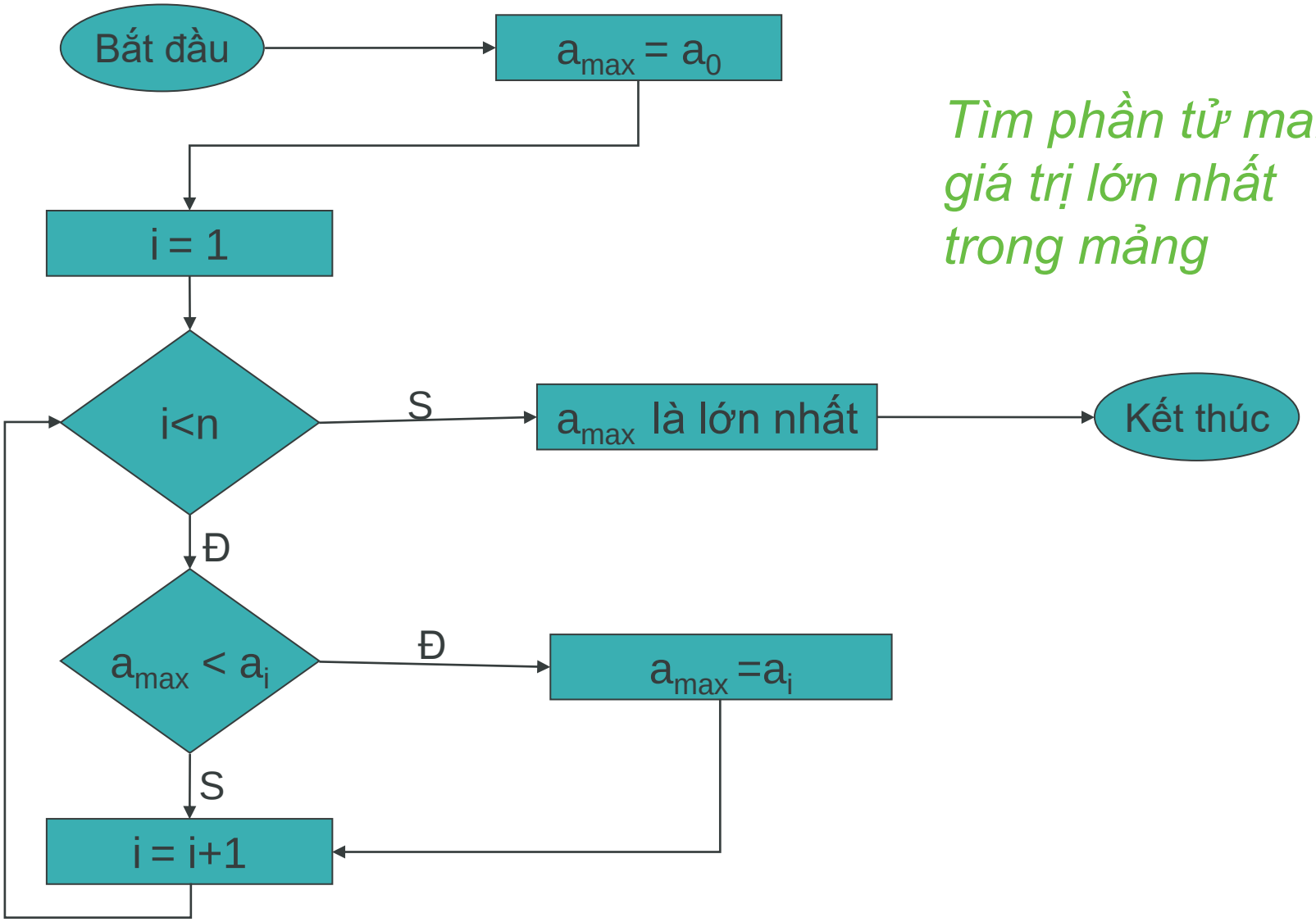
Nút giới hạn bắt đầu /
kết thúc chương trình

CÁC PHƯƠNG PHÁP BIỂU DIỄN THUẬT TOÁN

LƯU ĐỒ THUẬT TOÁN



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** …]

Vòng lặp:

while … **do**

do … **while** (…)

for … **do** …

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;

CÁC PHƯƠNG PHÁP BIỂU DIỄN THUẬT TOÁN

MÃ GIẢ

Đầu vào: n, k và dãy A
Đầu ra: vị trí phần tử của A có giá trị bằng k
$l = 1; r = n;$
While $(l < r)$
Begin
    $mid = (l + r) \div 2$
    If$(A[mid] = k)$ Then xuất mid
    Else If$(A[mid] <$ k) Then $l = mid + 1$
    Else $r = mid - 1$
End.

Biểu Diễn Bằng Ngôn Ngữ Lập Trình

- Dùng ngôn ngữ máy tính (C, C++,...) 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.

Tính hiệu quả của giải thuật

Làm thế nào để đánh giá một chương trình có tốt/hiệu quả hay không????

Tốn bao nhiêu thời gian

Chiếm bao nhiêu dung lượng bộ nhớ

……

Tính hiệu quả của giải thuật

Để giải một bài toán có thể có nhiều giải thuật khác nhau. Cần lựa chọn một giải thuật tốt theo hai tiêu chuẩn:

- Đơn giản, dễ hiểu, dễ lập trình.
- Thời gian thực hiện nhanh, **dùng ít tài nguyên** máy tính.

Tiêu chuẩn 2 là tính hiệu quả của giải thuật.

Độ phức tạp về thời gian thường được quan tâm hơn.

→ Đánh giá độ phức tạp của giải thuật là đánh giá thời gian thực hiện giải thuật đó.

Đánh giá thời gian thực hiện giải thuật

Thời gian thực hiện giải thuật phụ thuộc: Ngôn ngữ lập trình, chương trình dịch, hệ điều hành, phần cứng của máy,…

Mặt khác phải lập trình mới đo được thời gian thực hiện giải thuật.

Cần có cách đánh giá khác sao cho:

- Không phụ thuộc máy, ngôn ngữ lập trình, chương trình dịch.
- Không cần triển khai chương trình thực hiện giải thuật.
- Chỉ dựa vào phân tích bản thân giải thuật.

→ Phân tích độ phức tạp về mặt thời gian nhằm xác định mối quan hệ giữa dữ liệu đầu vào và chi phí thực hiện thuật toán thông qua các thao tác cơ bản như gán, so sánh, di chuyển.

→ Tổng số phép toán sơ cấp cần thiết để thực hiện giải thuật là cách làm đáp ứng được các yêu cầu trên.

Độ Phức Tạp Của Thuật Toán

- 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)

Phương Pháp Xấp Xỉ

- N: kích thước dữ liệu đầu vào
- O(c): hằng số, thời gian chạy không phụ thuộc N
  - Phép gán, nhập, xuất
- O(logN): logarit, chia đôi miền giá trị
  - Tìm kiếm nhị phân trên mảng 1 chiều
- O(N): tuyến tính
  - Vòng lặp for, tìm kiếm phần tử trong mảng 1 chiều
- O(NlogN): tách làm 2 bài toán nhỏ, lặp lại N lần
  - Quicksort
- $O(N^2)$
  - Hai vòng lặp for lồng nhau

Phương Pháp Xấp Xỉ

- Quy tắc cộng
  - O(max(f(n),g(n)))
    - f(n): thời gian thực hiện chương trình đầu
    - g(n): thời gian thực hiện chương trình sau

Ví dụ:

- x = x*4; //O(c)
- for (int i=0; i<n; i++) //O(N)

  cout << i;

- Độ phức tạp O(max(O(c),O(n))) = O(n)

Phương Pháp Xấp Xỉ

- Quy tắc nhân
  - Áp dụng cho các chương trình lồng vào nhau

Ví dụ:

- for (int i=0; i<n; i++) //O(N)

```
        for (int j=0; j<n; j++) //O(N)
                cout << i + j;
```

- Độ phức tạp $O(N^2)$

Ví dụ

```
cin >> n;
sum = 0;
for (int i = 1; i <= n; i++){
  sum += i;
}
```

```
cin >> n;
sum = n * (n + 1) / 2;
```

```
cin >> n;
for (int i = 1; i <= n; i++){
  for (int j = 1; j <= n; j++){
    cout << i << " " << j << endl;
  }
}
```

Ví dụ

```
1  int count_1( int N)
2  {
3    sum = 0                                  ——————————————  1
4    for (i=1; i<=n; i++) {                   ——————————————  2N
5      for (j=i; j<=n; j++) {                 ——————————————  2∑_{i=1}^{N}(N+1−i)
6        sum++                                ——————————————  ∑_{i=1}^{N}(N+1−i)
7      }
8    }
9    return sum                               ——————————————  1
10 }
```

Ví dụ

```
n = 5

i = 1, 2, 3, 4, 5

i = 1:
        j = 1, 2, 3, 4, 5
i = 2:
        j = 2, 3, 4, 5
i = 3:
        j = 3, 4, 5
i = 4:
        j = 4, 5
i = 5:
        j = 5
```

Tổng quát vòng for j sẽ chạy số lần là: n*(n+1)/2

Số phép tính của vòng for j sẽ là: 2*n*(n+1)/2

Thời gian chạy: Số phép tính của vòng for j sẽ là:

2+2n+3*n*(n+1)/2=3/2n^2+7/2n+2

j sẽ chạy: 5+4+3+2+1 lần

Tổng quát:

j sẽ chạy: n+n-1+n-2+…+1 lần

j sẽ chạy: 1+2+3+…+n-2+n-1+n lần = n*(n+1)/2

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

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).

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

Quy Trình Làm Phần Mềm

- *Bước 0:* Ý tưởng (concept).
- *Bước 1:* Xác định yêu cầu (Requirements Specification).
- *Bước 2:* Phân tích (Analysis).
- *Bước 3:* Thiết kế (Design).
- *Bước 4:* Cài đặt (Implementation).
- *Bước 5:* Thử nghiệm (Testing).
- *Bước 6:* Vận hành, theo dõi và bảo dưỡng (Operation, follow-up and Maintenance).