BÀI 1

GIỚI THIỆU CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

TÌNH HUỐNG DẪN NHẬP

Chương trình máy tính là một dãy các câu lệnh để xử lý thông tin và đem lại kết quả mong muốn cho người sử dụng. Tuy nhiên, các thông tin lấy từ thực tế thường là các thông tin trừu tượng và không thể biểu diễn một cách trực tiếp trong máy tính.

Câu hỏi đặt ra là:

Vậy thông tin lấy từ thực tế, được biểu diễn như thế nào trong máy tính

MỤC TIÊU

- Mô tả đúng đối tượng và các phương pháp nghiên cứu môn học;
- Trình bày đúng khái niệm về cấu trúc dữ liệu;
- Liệt kê và xác định đúng các kiểu dữ liệu cơ bản như kiểu dữ liệu số nguyên, số thực, logic, ký tự...;
- Mô tả kiểu đúng các kiểu dữ liệu trừu tượng như cấu trúc dữ liệu bản ghi, kiểu dữ liệu mảng... một cách chính xác;
- Mô tả ngôn ngữ diễn đạt giải thuật đúng và vận dụng nó để diễn đạt các thuật toán;
- Trình bày thuật toán đệ quy, thuật toán quay lui một cách chính xác.

NỘI DUNG

- Khái niệm
- Kiểu dữ liệu
- Ngôn ngữ diễn đạt giải thuật
- Đánh giá hiệu quả của thuật toán
- Đệ quy

1. KHÁI NIỆM

1. KHÁI NIỆM

- **Cấu trúc dữ liệu** là cách thức tổ chức sắp xếp dữ liệu để biểu diễn thông tin trên máy tính.
- **Giải thuật** để mô tả các phương pháp giải quyết vấn đề phù hợp cho việc triển khai trên máy tính.
- Cấu trúc dữ liệu và giải thuật là hai thuật ngữ luôn đi kèm với nhau. Cấu trúc dữ liệu là đầu vào và cũng là đầu ra của giải thuật.

1. KHÁI NIỆM

- Về cấu trúc dữ liệu ta quan tâm đến các vấn đề cơ bản sau:
  - Cách cài đặt cấu trúc dữ liệu đó;
  - Cách thực hiện các thao tác cơ bản với cấu trúc dữ liệu đó:
    - Tạo mới;
    - Thêm 1 phần tử;
    - Xóa 1 phần tử;
    - Tìm 1 phần tử …
- Về giải thuật ta quan tâm đến các vấn đề cơ bản sau:
  - Tư tưởng của giải thuật;
  - Nội dung của thuật toán và cách cài đặt thuật toán đó;
  - Đánh giá độ phức tạp về thời gian.

2. KIỂU DỮ LIỆU

2. KIỂU DỮ LIỆU

- Các kiểu dữ liệu cơ bản:
  - Kiểu dữ liệu số nguyên (INTEGER);
  - Kiểu dữ liệu số thực (REAL);
  - Kiểu dữ liệu logic (BOOLEAN);
  - Kiểu dữ liệu ký tự (CHAR);
  - Kiểu dữ liệu con trỏ (POINTER).
- Kiểu dữ liệu trừu tượng:
  - Cấu trúc dữ liệu bản ghi (RECORD);
  - Cấu trúc dữ liệu mảng (ARRAY).

Ví dụ 1.1. (Array)

a[3][5].

Ví dụ 1.2. (Integer)

1; –1; 2; –2 là các số nguyên

31 / 5 = 6.2

31 DIV 5 = 6 31 MOD 5 = 1

–31 DIV 5 = – 7 –31 MOD 5 = 4

(Real) 1.34; –2.7

Ví dụ 1.3. (Boolean)

Phép so sánh 5 = 9 sẽ đem lại kết quả sai (FALSE)

Phép so sánh 3 < 7 đem lại kết quả đúng (TRUE).

(Char) 't'; 'o'; 'p'; 'i', 'c'; 'a'

3. NGÔN NGỮ DIỄN ĐẠT GIẢI THUẬT

3. NGÔN NGỮ DIỄN ĐẠT GIẢI THUẬT

- Các giải thuật được diễn đạt hay biểu diễn bằng một ngôn ngữ chung;
- Ngôn ngữ đó vừa phải mang tính trừu tượng vừa gần với ngôn ngữ tự nhiên và với các ngôn ngữ lập trình hiện có;
- Các ngôn ngữ diễn đạt giải thuật thường sử dụng các thuật ngữ tiếng Anh với hai lý do:
  - Câu lệnh của các ngôn ngữ lập trình thường dựa trên các thuật ngữ tiếng Anh;
  - Phần lớn các tài liệu về công nghệ thông tin được viết bằng tiếng Anh, trong đó bao gồm cả các tài liệu về cấu trúc dữ liệu và giải thuật.

3.1. CẤU TRÚC DỮ LIỆU GIẢI THUẬT ĐƯỢC VIẾT BẰNG NGÔN NGỮ DIỄN ĐẠT GIẢI THUẬT

Một giải thuật được viết bao gồm hai phần:

- **Phần định nghĩa**: Mô tả khái quát giải thuật bao gồm dữ liệu đầu vào, điều kiện áp dụng giải thuật, và kết quả trả về mong muốn của việc thực hiện giải thuật.
- **Phần triển khai**: Bao gồm việc khai báo các biến dữ liệu sẽ được sử dụng trong giải thuật, và mô tả các lệnh cụ thể cần được thực hiện để triển khai giải thuật. Được đặt giữa hai dấu móc lệnh { và }

Cấu trúc:

```
Algorithm    name
Input: ...
Pre-condition
Post-condition
Declarations
{
    List of operations
}
```

Cấu trúc:

```
Algorithm name
        Input…..
Pre-condition
Post-condition
Output
```

3.2. KHAI BÁO BIẾN

- Khai báo một biến (Variable) để sử dụng trong giải thuật, chúng ta dùng cấu trúc:

  Type variable;

Ví dụ 1.4. Các ví dụ về khai báo biến:

```
int n; // Kiểu số nguyên  float x;
// Kiểu số thực  char c; //
Kiểu ký tự
int a, b, c; // khai báo 3 số nguyên a, b, c.
```

3.2. KHAI BÁO BIẾN

- Khai báo một biến (Variable) để sử dụng trong giải thuật, chúng ta dùng cấu trúc:

  Type variable;

- Khai báo một biến thuộc cấu trúc mảng

  Type_of_Array variable[size];

Kích thước của mảng hai chiều được khai báo theo cú pháp:

số hàng x số cột

Ví dụ 1.5.

int arr[25]; // Mảng 25 phần tử thuộc kiểu số nguyên.

int arr[10][15]; // Mảng 10x15 phần tử thuộc kiểu số nguyên.

int a,b,c[10]; // 2 số nguyên a, b và mảng số nguyên c có 10 phần tử.

3.3. CÂU LỆNH

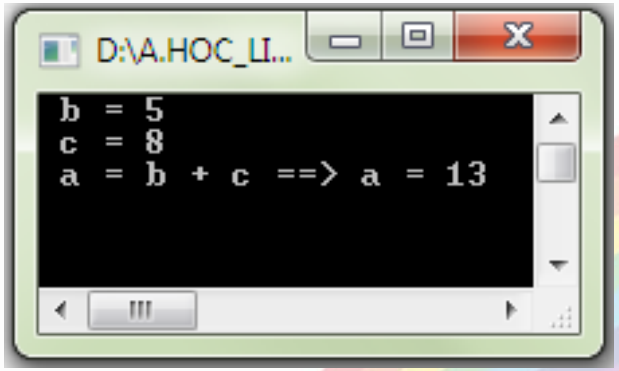
Câu lệnh gán:

Cấu trúc của một câu lệnh gán:

variable = constant/expression;

Ví dụ 1.6. Phép gán giá trị cho một biến kiểu số nguyên:

```
#include <stdio.h>
#include <conio.h>
main()
{
int a,    b,   c; b = 5; c = 8;
printf    ("   b = 5\n");
printf    ("   c = 8\n“);
printf    ("   a = b + c ==> a = %d", b+c);
getch();
}
```



3.3. CÂU LỆNH

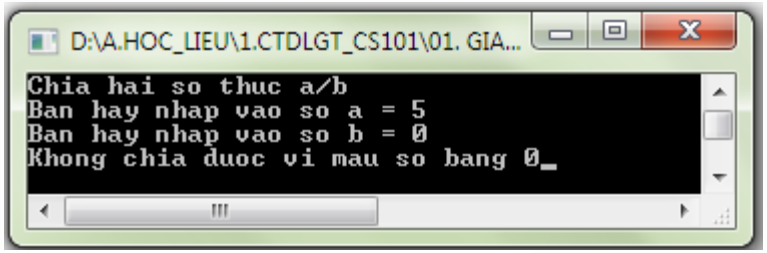
Câu lệnh điều kiện NẾU THÌ (IF THEN ELSE)
Cấu trúc của câu lệnh điều kiện:

```
if (condition)
{
        list operation…
}
else
{
        list operation…
}
```

3.3. CÂU LỆNH

Ví dụ 1.7. Phép chia hai số thực:

```
#include <stdio.h>
#include <conio.h>
main()
{
float a, b;
printf ("Chia hai so thuc a/b\n");
printf ("Ban hay nhap vao so a = "); scanf ("%f", &a);
printf ("Ban hay nhap vao so b = "); scanf ("%f", &b);
if (b != 0)
        printf (" a/b = %f", a/b);
else
        printf (" Khong chia duoc vi mau so bang 0");
getch();
}
```



3.3. CÂU LỆNH

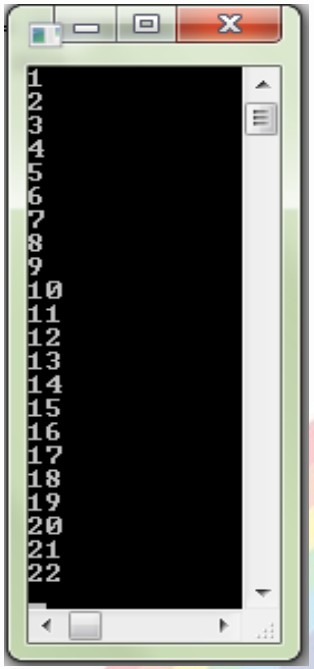
Câu lệnh vòng lặp xác định (For)

Cấu trúc câu lệnh lặp xác định được mô tả như sau:

```
for(i=start_value; i<end_value; i+=step_value)
//step_value : bước nhảy
{
        operation
}
```

Ví dụ 1.8. In ra màn hình từ 1 đến 22

```
#include <stdio.h>
#include <conio.h>
main()
{
int i;
for (i = 1; i <= 22; i++)
printf ("%d\n",i);
getch();
}
```



3.3. CÂU LỆNH

Câu lệnh vòng lặp không xác định

- Cấu trúc của câu lệnh vòng lặp không xác định như sau:

```
while (condition)
{
    list operation.
}
```

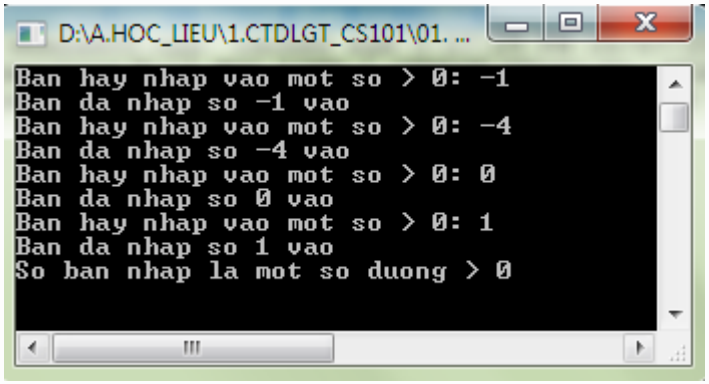
- Một biến thể của vòng lặp vô hạn trong đó số lần lặp thay đổi từ 1 đến vô cùng như sau:

```
do
{
    operation……
} while (condition);
```

3.3. CÂU LỆNH

Ví dụ 1.9. Vòng lặp không xác định khi nhập vào số <= 0

```
#include <stdio.h>
#include <conio.h>
main()
{
    int h;
    do
    {
      printf ("Ban hay nhap vao mot so > 0: "); scanf  ("%d", &h);
      printf ("Ban da nhap so %d vao\n", h);
    }
    while (h <= 0);
    printf ("So ban nhap la mot so duong");
    getch();
```



4. ĐÁNH GIÁ HIỆU QUẢ CỦA THUẬT TOÁN

4. ĐÁNH GIÁ HIỆU QUẢ CỦA THUẬT TOÁN

- Tính hiệu quả của thuật toán thông thường được đo bởi thời gian tính (thời gian được sử dụng để tính bằng máy hoặc bằng phương pháp thủ công) khi các giá trị đầu vào có kích thước xác định;
- Tính hiệu quả của thuật toán cũng được xem xét theo thước đo dung lượng bộ nhớ đã sử dụng để tính toán khi kích thước đầu vào đã xác định;
- Hai thước đo đã nêu trên liên quan đến độ phức tạp tính toán của một thuật toán, được gọi là độ phức tạp thời gian và độ phức tạp không gian (còn gọi là độ phức tạp dung lượng nhớ).

4. ĐÁNH GIÁ HIỆU QUẢ CỦA THUẬT TOÁN

- Độ phức tạp thuật toán là một hàm phụ thuộc đầu vào;
- Để ước lượng độ phức tạp của một thuật toán ta thường dùng khái niệm bậc O – lớn;
- **Bậc 0 lớn:** Gọi *n* là độ lớn đầu vào, độ phức tạp của bài toán phụ thuộc vào *n*. Ở đây ta không chỉ đặc trưng độ phức tạp bởi số lượng phép tính, mà dùng một đại lượng tổng quát là *tài nguyên cần dùng R* (*n*). Nếu tìm được hằng số C sao cho với *n* đủ lớn, các hàm *R* (*n*), *g* (*n*) đều dương và R(n) < C.g(n) thì ta nói thuật toán có độ phức tạp cỡ *O* (*g*(*n*)).

4. ĐÁNH GIÁ HIỆU QUẢ CỦA THUẬT TOÁN

Các độ phức tạp thường gặp đối với các thuật toán thông thường gồm có:

- Độ phức tạp hằng số, $O(1)$. Số phép tính/thời gian chạy/dung lượng bộ nhớ không phụ thuộc vào độ lớn đầu vào;
- Độ phức tạp tuyến tính, $O(n)$. Số phép tính/thời gian chạy/dung lượng bộ nhớ có xu hướng tỉ lệ thuận với độ lớn đầu vào;
- Độ phức tạp đa thức, $O(P(n))$, với P là đa thức bậc cao (từ 2 trở lên);
- Độ phức tạp logarit, $O(\log n)$ (chú ý: Bậc của nó thấp hơn so với $O(n)$);
- Độ phức tạp hàm mũ, $O(a^n)$. Trường hợp này bất lợi nhất và sẽ rất phi thực tế nếu thực hiện thuật toán với độ phức tạp này.

4. ĐÁNH GIÁ HIỆU QUẢ CỦA THUẬT TOÁN

- **Ví dụ:** Xét thuật toán tìm kiếm phần tử lớn nhất trong dãy số nguyên cho trước

  Input: a là mảng các số nguyên, n > 0, là số các số trong mảng a;

  Output: Max, số lớn nhất trong mảng a.

- **Giải thuật:**

```
Max = a[0];
for (int i = 0; i < n - 1; i++)
if(Max < a[i]) Max = a[i];
```

- Ta dễ dàng thấy được số các phép toán so sánh sơ cấp được sử dụng ở đây là 2(n – 1). Vì vậy ta nói rằng độ phức tạp của thuật toán nói trên là O(n)(gọi là độ phức tạp tuyến tính).

5. ĐỆ QUY

5. ĐỆ QUY

- Giới thiệu về đệ quy (Recurrence);
- Phương pháp quay lui;
- Bài toán tháp Hà Nội.

5.1. GIỚI THIỆU VỀ ĐỆ QUY (RECURRENCE)

- Bản chất của việc thiết kế giải thuật theo phương pháp đệ quy là việc gọi lại chính giải thuật đó trong quá trình thiết kế giải thuật;
- Lời gọi một giải thuật có cấu trúc như sau:

```
TÊN_GIẢI_THUẬT (Danh sách các tham số đầu vào);
```

- Giải thuật đệ quy thường được thiết kế theo cấu trúc:
  - Kiểm tra điều kiện kết thúc đệ quy và trả về kết quả tương ứng;
  - Chia nhỏ dữ liệu đầu vào;
  - Áp dụng giải thuật đệ quy trên từng phần dữ liệu con;
  - Tổng hợp các kết quả thu được.

5.1. GIỚI THIỆU VỀ ĐỆ QUY(RECURRENCE)

Ví dụ 1.10

```
#include <stdio.h>
long int Giai_thua (int h);

main()
        {
                int h;
                long int Giai_thua (int h);
                printf (" h = "); scanf ("%d", &h);
                printf (" h != %ld\n", Giai_thua(h));
                getch();
        }
    long int Giai_thua (int h)
        {
                if (h == 0) return (1);
                else return (h * Giai_thua (h - 1));
        }
```

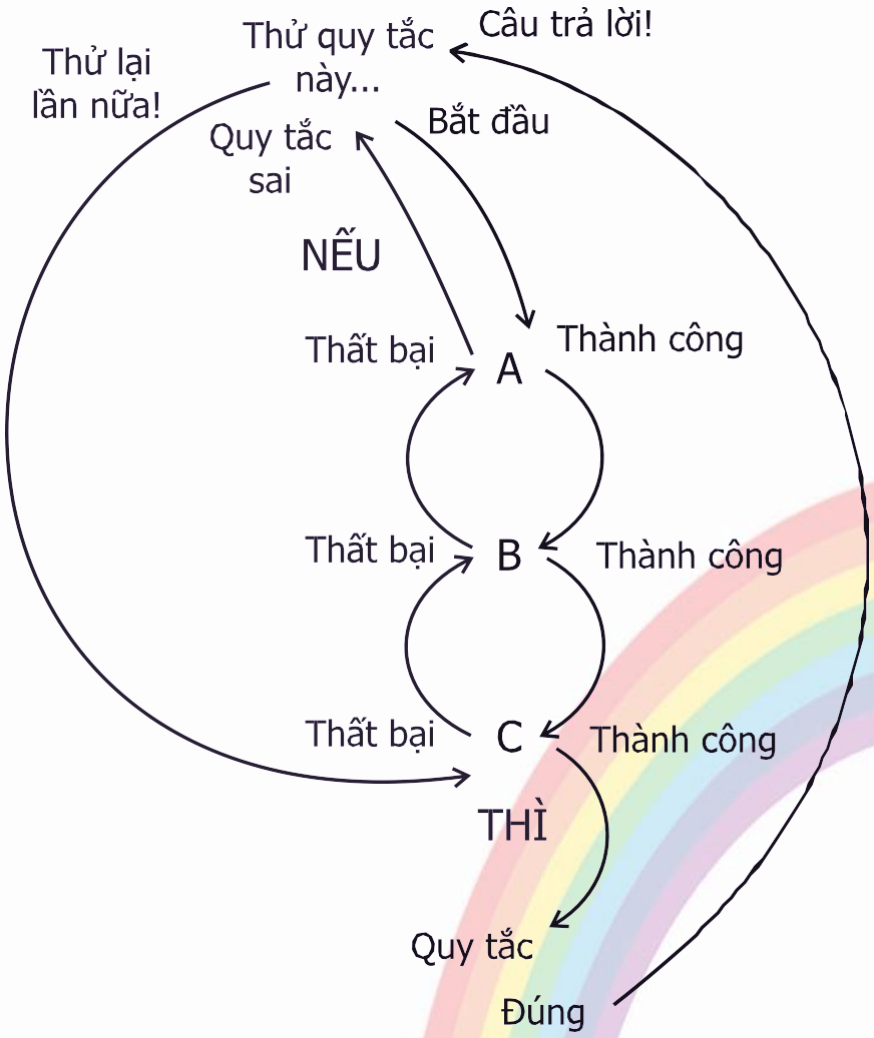
5.1. GIỚI THIỆU VỀ ĐỆ QUY (RECURRENCE)

- **Ưu điểm của giải thuật đệ quy:**
  - Tính hiệu quả trong thiết kế và lập trình: Logic của chương trình được biểu diễn hết sức đơn giản và trong sáng;
  - Đặc biệt thích hợp để giải quyết các vấn đề, để triển khai các hàm hay để xử lý các dữ liệu vốn dĩ đã được định nghĩa bằng phương pháp truy hồi.

- **Nhược điểm:**
  - Thời gian thực hiện giải thuật sẽ tăng rất nhanh khi giá trị của n tăng lên;
  - Bộ nhớ cần thiết để chạy chương trình sẽ tăng lên khi độ sâu của lời gọi đệ quy tăng lên.

5.2. PHƯƠNG PHÁP QUAY LUI

- Trong thực tế, nhiều bài toán không thể xây dựng lời giải bằng các quy tắc, các công thức tính toán nhất định;
- Với những bài toán phức tạp, ta thường chia nhỏ thành nhiều tầng theo hướng đệ quy;
- Khi việc gọi đệ quy cho một tầng nào đó không thực hiện được, ta sẽ quay lui lên tầng trên để tìm lời giải. Vì vậy phương pháp này gọi là quay lui.



5.2. PHƯƠNG PHÁP QUAY LUI

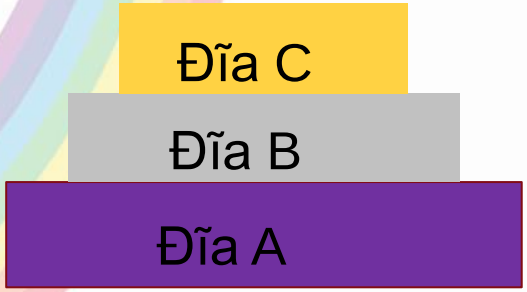
Bài toán Tháp Hà Nội

Nội dung của bài toán:

Có ba cọc 1, 2, 3. Cọc 1 xếp 3 đĩa A, B, C theo kích thước từ lớn đến bé (kích thước tăng dần). Hãy tìm cách chuyển toàn bộ đĩa từ cọc 1 sang cọc 3 theo quy tắc:

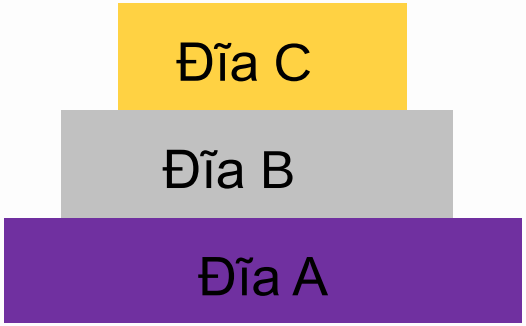
- Mỗi lần chỉ được chuyển 1 đồng xu ở trên cùng từ cọc này sang cọc khác.
- Không bao giờ được để đồng xu to hơn nằm trên đồng xu nhỏ hơn.

5.2. PHƯƠNG PHÁP QUAY LUI



Cọc 1

Cọc 2



Cọc 3

Phân tích bài toán:

- Chuyển đĩa C từ cọc 1 sang cọc 3;
- Chuyển đĩa B từ cọc 1 sang cọc 2;
- Chuyển đĩa C từ cọc 3 sang cọc 2;
- Sau đó chuyển đĩa A từ cọc 1 sang cọc 3.
- Chuyển đĩa C từ cọc 2 về cọc 1.
- Chuyển đĩa B từ cọc 2 sang cọc 3.
- Chuyển đĩa C từ cọc 1 sang cọc 3.

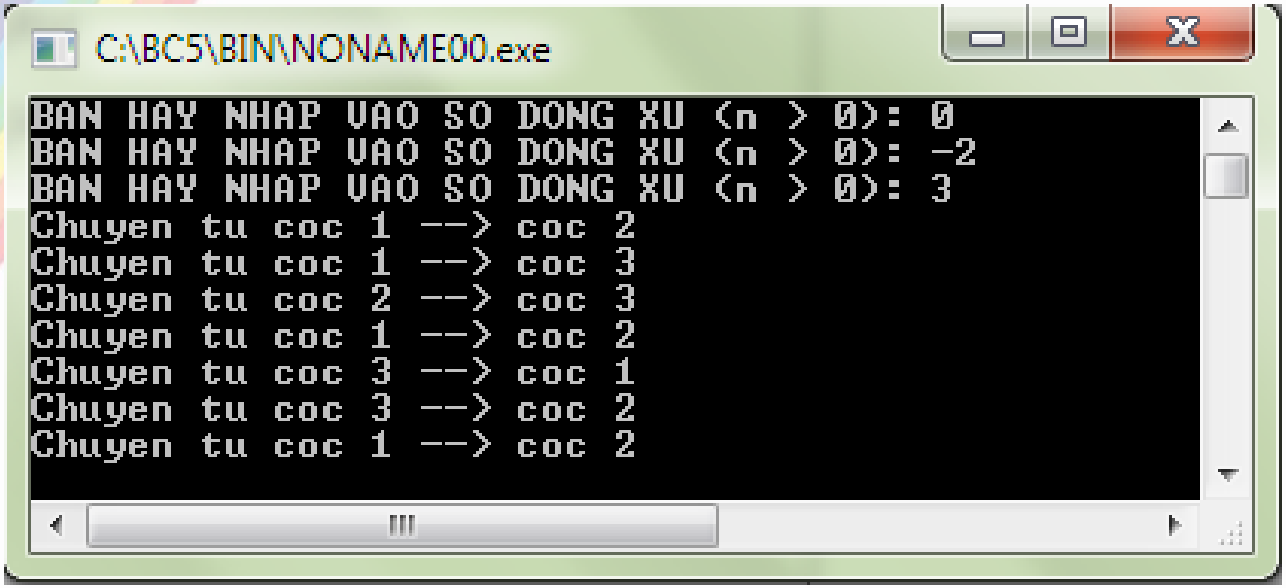
5.2. PHƯƠNG PHÁP QUAY LUI

Biểu diễn giải thuật:

```
#include <stdio.h>
#include <conio.h>
void Thap_Ha_Noi (int n, int coc_1, int coc_2, int coc_3);
main()
{   int n;
    do
    { printf ("BAN HAY NHAP VAO SO DONG XU (n > 0): ");
    • scanf ("%d", &n);}
    • while (n <=0); Thap_Ha_Noi (n, 1, 2, 3); getch();
• }
• void Thap_Ha_Noi (int n, int coc_1, int coc_2, int coc_3)
•   {   if (n == 1) printf ("Chuyen tu coc %d --> coc %d\n",  coc_1, coc_2);
    • else
        • {
            • Thap_Ha_Noi (n - 1, coc_1, coc_3, coc_2);  Thap_Ha_Noi (1, coc_1, coc_2, coc_3);  Thap_Ha_Noi (n - 1, coc_3, coc_2, coc_1);
    • };
};
```

5.2. PHƯƠNG PHÁP QUAY LUI

Biểu diễn giải thuật:



TÓM LƯỢC CUỐI BÀI

- Nắm được khái niệm về cấu trúc dữ liệu và giải thuật;
- Nắm được các kiểu dữ liệu cơ bản;
- Nắm được các kiểu dữ liệu trừu tượng;
- Nắm được ngôn ngữ diễn đạt giải thuật;
- Đánh giá hiệu quả của thuật toán;
- Nắm được thuật toán đệ quy;
- Nắm được phương pháp quay lui.