



HUTECH
Đại học Công nghệ Tp.HCM

KHOA CÔNG NGHỆ THÔNG TIN

Bài giảng:

KỸ THUẬT LẬP TRÌNH

Bài 5:

ĐỆ QUY



C Ngôn ngữ lập trình số 1 thế giới

Giảng viên: Th.S Dương Thành Phết

Email: phetcm@gmail.com

Website: <http://www.thayphet.net>

Mobile: 0918158670



MỤC TIÊU

- ✓ Trình bày được khái niệm về đệ quy, các kiểu đệ quy;
- ✓ Mô tả được Ưu điểm và nhược điểm khi cài đặt hàm bằng phương pháp đệ quy;
- ✓ Minh họa được cách khai báo và viết hàm theo kiểu đệ quy;
- ✓ Giải quyết được một số bài toán kinh điển bằng phương pháp đệ quy;
- ✓ Xử lý được các giải thuật trên mảng 1 chiều bằng phương pháp đệ quy.



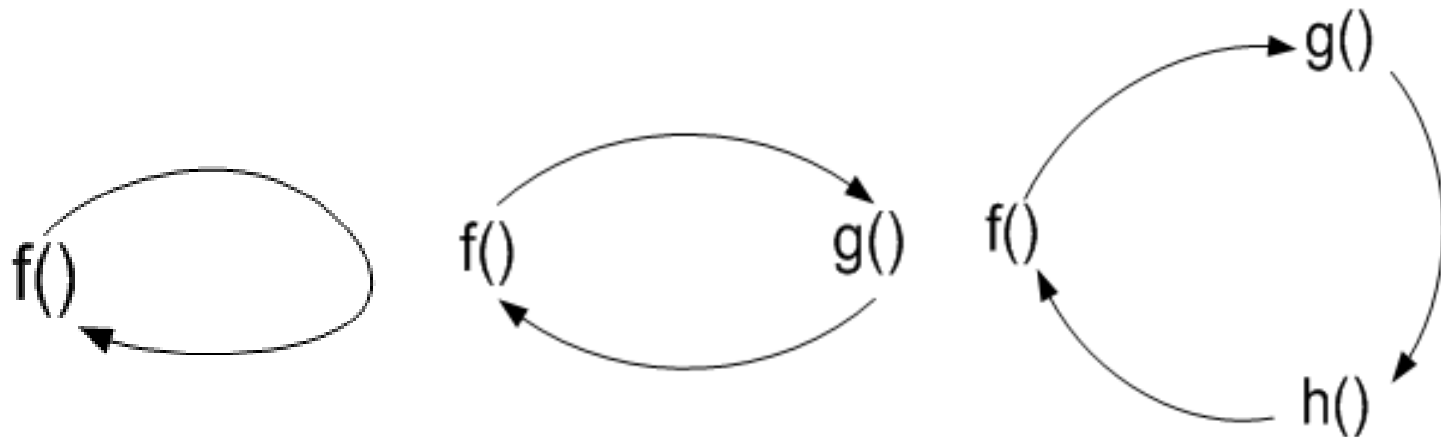
NỘI DUNG

1. Khái niệm về đệ quy.
2. Phân loại hàm đệ quy.
3. Kỹ thuật giải bài toán bằng đệ quy.
4. Nhận xét.
5. Cấu trúc lặp và đệ quy.
6. Khử đệ quy
7. Bài tập.

5.1. KHÁI NIỆM VỀ ĐỆ QUY

- ✓ Đệ quy là một thuật toán dùng để đơn giản hóa những bài toán phức tạp bằng cách phân nhỏ phép toán đó thành nhiều phần đồng dạng.
- ✓ Những lời giải sẽ được kết hợp lại để giải quyết bài toán lớn hơn.

“Một hàm được gọi là đệ quy nếu bên trong thân hàm có lời gọi đến chính nó một cách tường minh hoặc tiềm ẩn”



5.1. KHÁI NIỆM VỀ ĐỆ QUY

Ví dụ 1: Tính giai thừa của số nguyên dương n như sau:

$$n! = 1 * 2 * 3 * \dots * (n-1) * n = (n-1)! * n \quad (\text{với } 0! = 1)$$

Ta thấy

- ✓ Nếu $n=0$ thì $n!=1$
- ✓ Ngược lại thì $n!=n * (n-1)!$

Với định nghĩa trên thì hàm đệ quy tính $n!$ được viết:

```
long giaithua_khongdequy(int n)
{
    int i;
    long kq=1;
    for (i=2;i<=n;i++)
        kq=kq*i;
    return kq;
}
```

```
long giaithua_dequy(int n)
{
    if (n==0)
        return 1;
    else
        return n*giaithua_dequy(n-1);
}
```



5.1. KHÁI NIỆM VỀ ĐỆ QUY

```
void main()
{
    int n;
    printf("\n Nhap so n can tinh giai thua ");
    scanf("%d",&n);
    printf("\nGoi ham de quy:%d!=%ld",
           n, giaithua_dequy(n));
    printf("\nGoi ham khong de quy: %d != %ld ",
           n , giaithua_khongdequy(n));
    getch();
}
```

5.1. KHÁI NIỆM VỀ ĐỆ QUY

Ví dụ 2: Tính tổng $S(n) = 1 + 2 + 3 + \dots + n$.

Ta có : $S(n) = 1 + 2 + 3 + 4 + \dots + (n-1) + n = S(n-1) + n$;

```
long TinhtongLap (int n)
{
    long s = 0;
    for(int i=1; i <= n; i++)
        s = s + i;
    return s ;
}
```

```
long TinhtongDequy(int n)
{
    if (n == 0)
        return 0;
    else
        return n+TinhtongDequy(n-1);
}
```

```
void main()
{
    int n;
    printf("\n Nhap so n: ");
    scanf("%d",&n);
    printf("\nGoi ham de quy:%d!=%ld",n, TinhtongDequy(n));
    printf("\nGoi ham khong de quy: %d != %ld ",TinhtongLap(n));
    getch();
}
```



5.2. PHÂN LOẠI HÀM ĐỘ QUY

Tùy thuộc cách diễn đạt tác vụ độ quy mà có các loại độ quy sau:

- ✓ Độ quy tuyến tính.
- ✓ Độ quy nhị phân.
- ✓ Độ quy phi tuyến .
- ✓ Độ quy hỗn hợp .



5.2.1 ĐỆ QUY TUYẾN TÍNH

Trong thân hàm có **duy nhất một lời gọi hàm gọi lại chính nó** một cách tường minh.

```
<Kiểu dữ liệu> TenHam (<danh sách tham số>)
```

```
{
```

```
    if (điều kiện dừng)
```

```
    {
```

```
        //Trả về giá trị hay kết thúc công việc
```

```
    }
```

```
    else
```

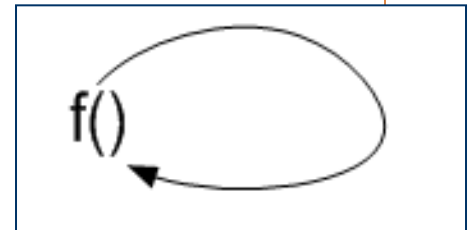
```
    {
```

```
        //Thực hiện một số công việc (nếu có)
```

```
        ...TenHam (<danh sách tham số>); //gọi đệ quy
```

```
    }
```

```
}
```



5.2.1 ĐỆ QUY TUYẾN TÍNH

Ví dụ 1: Tính tổng $S(n) = 2 + 4 + 6 + \dots + 2n$.

Ta có : $S(n) = 2 + 4 + 6 + \dots + 2(n-1) + 2n = S(n-1) + 2n$;

```
long Tongchan(int n)
{
    if(n==1)
        return 2;
    return 2*n + tongchan(n-1);
}
```

Ví dụ 2: Tính $S(n) = 1 * 2 * \dots * n$ (với $n > 0$)

- Điều kiện dừng: $S(0) = S(1) = 1$.
- Quy tắc (công thức) tính: $S(n) = n * S(n-1)$

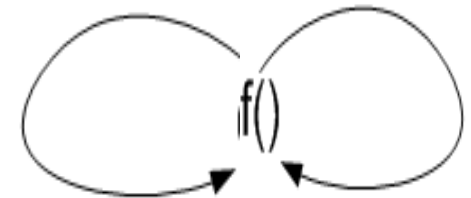
```
long Giaithua (int n){
    if(n==1)
        return 1;
    return n * Giaithua(n-1);
}
```



5.2.2 ĐỆ QUY NHỊ PHÂN

Trong thân của hàm có đúng 2 lời gọi hàm gọi lại chính nó một cách tường minh.

```
<Kiểu dữ liệu> TenHam (<danh sách tham số>)
{
    if (điều kiện dừng)
    {
        //Trả về giá trị hay kết thúc công việc
    }
    else
    {
        //Thực hiện một số công việc (nếu có)
        ....TenHam (<danh sách tham số>); //lần 1
        ....TenHam (<danh sách tham số>); //lần 2
        //Giải quyết vấn đề còn lại
    }
}
```





5.2.2 ĐỆ QUY NHỊ PHÂN

Ví dụ: Tính số hạng thứ n của dãy Fibonacci được định nghĩa như sau:

$$f_1 = f_0 = 1 ;$$

$$f_n = f_{n-1} + f_{n-2} ; \text{ (với } n > 1 \text{)}$$

Điều kiện dừng: $f(0) = f(1) = 1$

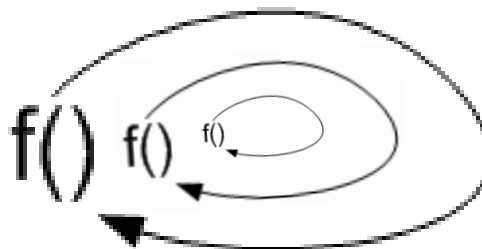
Tổng quát: $f(n) = f(n-1) + f(n-2)$

```
long Fibonacci (int n)
{
    if(n==0 || n==1)
        return 1;
    return Fibonacci(n-1) + Fibonacci(n-2);
}
```

5.2.3 ĐỆ QUY PHI TUYẾN

Trong thân của hàm có **lời gọi hàm gọi lại chính nó được đặt bên trong vòng lặp.**

```
<Kiểu dữ liệu> TenHam (<danh sách tham số>)
{
    for (int i = 1; i<=n; i++)
    {
        if (điều kiện dừng)
            //Trả về giá trị hay kết thúc công việc
        else
        {
            //Thực hiện một số công việc (nếu có)
            ...TenHam (<danh sách tham số>);
        }
    }
}
```





5.2.3 ĐỆ QUY PHI TUYẾN

Ví dụ: Tính số hạng thứ n của dãy $\{X_n\}$ được định nghĩa như sau:

$$X_0 = 1 ;$$

$$X_n = n^2 X_0 + (n-1)^2 X_1 + \dots + 1^2 X_{n-1} ; \quad (n \geq 1)$$

Điều kiện dừng: $X(0) = 1$.

```
long TinhXn (int n)
{
    if(n==0)
        return 1;
    long s = 0;
    for (int i=1; i<=n; i++)
        s = s + i * i * TinhXn(n-i);
    return s;
}
```

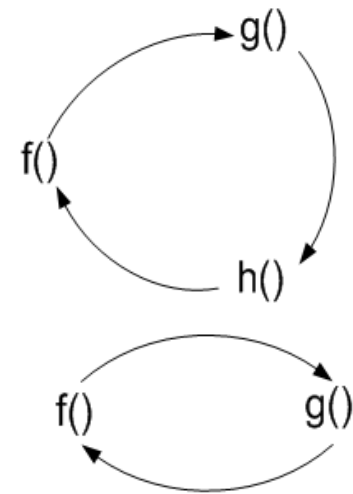
5.2.4 ĐỘ QUY TƯƠNG HỒ

Trong thân của hàm này có lời gọi hàm đến hàm kia và trong thân của hàm kia có lời gọi hàm tới hàm này.

```
<Kiểu dữ liệu> TenHam1 (<danh sách tham số>);
<Kiểu dữ liệu> TenHam2 (<danh sách tham số>);

<Kiểu dữ liệu> TenHam1 (<danh sách tham số>)
{
    //Thực hiện một số công việc (nếu có)
    ...TenHam2 (<danh sách tham số>);
    //Thực hiện một số công việc (nếu có)
}

<Kiểu dữ liệu> TenHam2 (<danh sách tham số>)
{
    //Thực hiện một số công việc (nếu có)
    ...TenHam1 (<danh sách tham số>);
    //Thực hiện một số công việc (nếu có)
}
```



5.2.4 ĐỆ QUY TƯƠNG HỒ

Ví dụ: Tính số hạng thứ n của hai dãy $\{X_n\}$, $\{Y_n\}$ được định nghĩa:

$$X_0 = Y_0 = 1 ;$$

$$X_n = X_{n-1} + Y_{n-1}; \quad (n > 0)$$

$$Y_n = n^2 X_{n-1} + Y_{n-1}; \quad (n > 0)$$

→ Điều kiện dừng: $X(0) = Y(0) = 1$.

```
long TinhXn (int n);
long TinhYn(int n);
long TinhXn (int n){
    if(n==0)
        return 1;
    return TinhXn(n-1) + TinhYn(n-1);
}
long TinhYn (int n){
    if(n==0)
        return 1;
    return n*n*TinhXn(n-1) + TinhYn(n-1);
}
```




5.3. KỸ THUẬT GIẢI QUYẾT BÀI TOÁN BẰNG ĐỘ QUY

- ✓ Thông số hóa bài toán.
- ✓ Tìm các điều kiện biên (chặn, dừng), tìm giải thuật cho các tình huống này.
- ✓ Tìm giải thuật tổng quát theo hướng độ quy lui dần về tình huống bị chặn.

5.3. KỸ THUẬT GIẢI QUYẾT BÀI TOÁN BẰNG ĐỆ QUY

Ví dụ Tính tổng 1 mảng a, n phần tử

- ✓ Thông số hóa: `int a [] , int n`
- ✓ Điều kiện biên: Mảng 0 phần tử thì tổng bằng 0.
- ✓ Giải thuật chung:

$$\text{Sum}(a,n) = \underbrace{a[0] + a[1] + \dots + a[n-3] + a[n-2] + a[n-1]}_{\text{Sum}(a,n-1)}$$

$$\text{Sum}(a,n) = \begin{cases} 0 & \text{khi } n = 0 \\ a[n-1] + \text{Sum}(a, n-1) & \text{khi } n > 0 \end{cases}$$



5.3. KỸ THUẬT GIẢI QUYẾT BÀI TOÁN BẰNG ĐỆ QUY

```
// Hàm cài đặt  
long Tongmang ( int a[ ], int n )  
{  
    if ( n==0 )  
        return 0;  
    else  
        return a[n-1 ] + tongmang ( a, n-1 ) ;  
}
```

Lưu ý : Với các thuật toán đệ quy trên mảng, ta nên giảm dần số phần tử của mảng.

5.3. KỸ THUẬT GIẢI QUYẾT BÀI TOÁN BẰNG ĐỆ QUY

5.3.1. Một số bài toán kinh điển dùng phương pháp đệ quy

Bài toán tháp Hà Nội:

Truyền thuyết kể rằng: Một nhà toán học Pháp sang Đông Dương đến một ngôi chùa cổ ở Hà Nội thấy các vị sư đang chuyển một chồng đĩa quý gồm 64 đĩa với kích thước khác nhau từ cột A sang cột C theo cách:

- ✓ Mỗi lần chỉ chuyển một đĩa
- ✓ Khi chuyển có thể dùng một cột trung gian B
- ✓ Trong suốt quá trình chuyển các chồng đĩa ở các cột luôn được xếp đúng (đĩa có kích thước bé được đặt trên đĩa lớn).
- ✓ Khi hỏi các vị sư cho biết khi nào chuyển xong chồng đĩa ?

Các vị trả lời : Đến ngày tận thế.

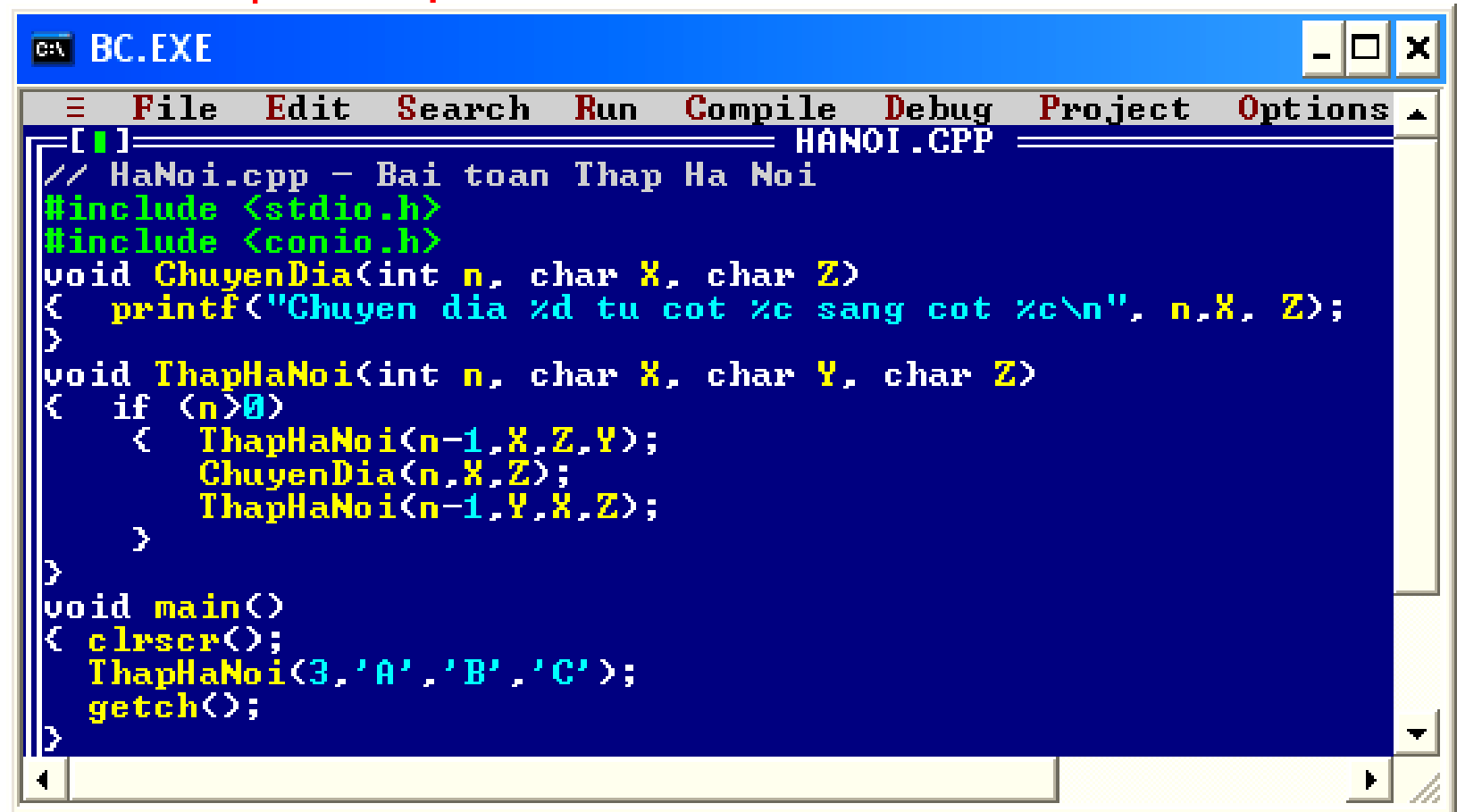
Vì với n đĩa cần $2^n - 1$ lần chuyển đĩa:

- Với $n=64 \rightarrow T=(2^{64} - 1) * t$.
- Giả sử $t=1/100s$ thì $T = 5.8$ tỷ năm.

5.3. KỸ THUẬT GIẢI QUYẾT BÀI TOÁN BẰNG ĐỆ QUY

5.3.1. Một số bài toán kinh điển dùng phương pháp đệ quy

Bài toán tháp Hà Nội:

A screenshot of a Turbo C++ IDE window titled 'BC.EXE'. The menu bar includes File, Edit, Search, Run, Compile, Debug, Project, and Options. The code editor shows a C++ program for the Hanoi Tower problem. The code defines two functions: 'ChuyenDia' for moving a disk and 'ThapHaNoi' for the recursive solution. The 'main' function initializes the screen and calls 'ThapHaNoi' with 3 disks and pegs A, B, and C.

```
[ ]===== HANOI.CPP =====
// HaNoi.cpp - Bai toan Thap Ha Noi
#include <stdio.h>
#include <conio.h>
void ChuyenDia(int n, char X, char Z)
{   printf("Chuyen dia %d tu cot %c sang cot %c\n", n, X, Z);
}
void ThapHaNoi(int n, char X, char Y, char Z)
{   if (n>0)
    {   ThapHaNoi(n-1, X, Z, Y);
        ChuyenDia(n, X, Z);
        ThapHaNoi(n-1, Y, X, Z);
    }
}
void main()
{   clrscr();
    ThapHaNoi(3, 'A', 'B', 'C');
    getch();
}
```



5.3. KỸ THUẬT GIẢI QUYẾT BÀI TOÁN BẰNG ĐỆ QUY

5.3.1. Một số bài toán kinh điển dùng phương pháp đệ quy

Phương pháp Chia để trị (Divide and Conquer):

- ✓ Giải thuật phân rã vấn đề thành những vấn đề con, giải những vấn đề con này và kết hợp những lời giải của những vấn đề con thành lời giải cho vấn đề nguyên thủy.
- ✓ Chiến lược gồm 3 bước sau đây ở mỗi cấp đệ quy:
 - *Phân chia* (divide) đầu vào thành các bài toán con
 - *Đệ quy* (recur): giải quyết các bài toán con bằng gọi đệ quy.
 - *Trị* (Conquer, combine): Kết hợp các giải pháp tìm được để giải quyết bài toán.

Chú ý: Độ phức tạp giải thuật thường là:

$$(\log n \times (\text{divide}(n) + \text{combine}(n))).$$

5.3. KỸ THUẬT GIẢI QUYẾT BÀI TOÁN BẰNG ĐỆ QUY

5.3.1. Một số bài toán kinh điển dùng phương pháp đệ quy

Phương pháp Chia để trị (Divide and Conquer):

Áp dụng giải bài toán sắp xếp trên mảng

78	53	62	59	89	11	29	50
----	----	----	----	----	----	----	----

78	53	62	59	89	11	29	50
----	----	----	----	----	----	----	----

78	53	62	59	89	11	29	50
----	----	----	----	----	----	----	----

78	53	62	59	89	11	29	50
----	----	----	----	----	----	----	----

53	78	59	62	11	89	50	29
----	----	----	----	----	----	----	----

53	59	62	78	11	29	50	89
----	----	----	----	----	----	----	----

11	29	50	53	59	62	78	89
----	----	----	----	----	----	----	----



5.3. KỸ THUẬT GIẢI QUYẾT BÀI TOÁN BẰNG ĐỆ QUY

5.3.1. Một số bài toán kinh điển dùng phương pháp đệ quy

Phương pháp Chia để trị (Divide and Conquer):

```
// Hàm cài đặt
void MergeSort (int a[], int Left, int Right)
{
    //Mảng có nhiều hơn 1 phần tử
    if(Left<Right)
    {
        int Mid = (Left+Right)/2;
        // Sắp xếp mảng bên trái
        MergeSort (a,Left,Mid);
        // Sắp xếp mảng bên phải
        MergeSort (a,Mid+1,Right);
        // Trộn 2 mảng lại với nhau
        Merge (a,Left,Mid,Right);
    }
}
```




5.4. NHẬN XÉT

Ưu điểm hàm Đệ quy:

- ✓ Hàm đệ quy là hàm: trong thân hàm lại gọi chính nó.
- ✓ Giải thuật đệ quy đẹp (gọn gàng, dễ viết code).
- ✓ Nhiều bài toán rất dễ mô tả với giải thuật đệ quy, hoặc bắt buộc phải sử dụng hàm đệ quy.

Khuyết điểm hàm Đệ quy:

- ✓ Vừa tốn bộ nhớ vừa chạy chậm
- ✓ Nhiều ngôn ngữ không hỗ trợ đệ quy (Fortran).
- ✓ Hàm đệ quy kém hiệu quả : tốn bộ nhớ và gọi hàm quá nhiều lần.

➔ Vì vậy tùy từng bài toán cụ thể mà người lập trình quyết định có nên dùng đệ quy hay không.



5.5. CẤU TRÚC LẶP VÀ ĐỀ QUY

- | | |
|--|--|
| <ul style="list-style-type: none"> ✓ Phương pháp lặp sử dụng cấu trúc lặp. ✓ Phương pháp lặp sử dụng vòng lặp tường minh ✓ Phương pháp lặp kết thúc khi điều kiện vòng lặp sai. ✓ Phương pháp lặp thay đổi biến đếm trong vòng lặp cho đến khi điều kiện lặp sai. ✓ Lặp sẽ không thoát khi điều kiện lặp không bao giờ sai. | <ul style="list-style-type: none"> ✓ Lập trình đệ quy sử dụng cấu trúc lựa chọn ✓ Phương pháp đệ quy lặp bằng cách gọi hàm. ✓ Phương pháp đệ quy kết thúc khi đến trường cơ sở. ✓ Phương pháp đệ quy làm cho các lời gọi hàm đơn giản dần đến trường cơ sở. ✓ Đệ quy không thoát khi các bước không hội tụ về trường cơ sở. |
|--|--|

“Đệ quy tồi hơn vì nó liên tục gọi hàm làm tốn thời gian của bộ vi xử lý và không gian nhớ”



5.6. KHỦ' ĐỆ QUY



5.7. BÀI TẬP DÙNG PHƯƠNG PHÁP ĐỆ QUY

Bài tập 1.

1. Hàm Nhập mảng 1 chiều các số thực gồm n phần tử ($0 < n < 100$)
2. Hàm Nhập mảng 1 chiều các số nguyên gồm n phần tử ($0 < n < 100$)
3. Viết hàm xuất mảng số nguyên n phần tử vừa nhập ở trên
4. Viết hàm xuất mảng số thực n phần tử vừa nhập ở trên
5. Tính tổng các phần tử có trong mảng
6. Tính tổng các phần tử chẵn có trong mảng
7. Tính tổng các phần tử lẻ có trong mảng
8. Tính tổng các phần tử nguyên tố có trong mảng
9. Tìm phần tử chẵn đầu tiên có trong mảng
10. Tìm phần tử lẻ đầu tiên có trong mảng



5.7. BÀI TẬP DÙNG PHƯƠNG PHÁP ĐỆ QUY

Bài tập 2.

11. Tìm phần tử nguyên tố đầu tiên có trong mảng
12. Tìm phần tử chẵn cuối cùng có trong mảng
13. Tìm phần tử chính phương cuối cùng có trong mảng
14. Tìm phần tử lớn nhất có trong mảng
15. Đếm số phần tử chẵn có trong mảng
16. Đếm số phần tử lớn nhất có trong mảng
17. In ra vị trí của phần tử lớn nhất đầu tiên có trong mảng
18. Sắp xếp mảng tăng dần
19. Tương tự các câu trên cho mảng các số thực



5.7. BÀI TẬP DÙNG PHƯƠNG PHÁP ĐỆ QUY

Bài tập thêm.

1. Tìm USCLN của 2 số nguyên dương a, b

2. Tính $S=1+2+3+....+n$ với $n>0$

3. Tính $S=1+1.2+1.2.3+.....+1.2.3...n$ với $n>0$

4. Tính $P(x,y) = x^y$

5. Tính
$$S(n) = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \dots + \frac{1}{1+2+3+\dots+n}$$



HUTECH
Đại học Công nghệ Tp.HCM

KHOA CÔNG NGHỆ THÔNG TIN

Bài giảng:

KỸ THUẬT LẬP TRÌNH

HẾT BÀI 5 ĐỆ QUY



C Ngôn ngữ lập trình số 1 thế giới