BÀI THỰC HÀNH SỐ 03

1. MỤC TIÊU

Hiểu và cài đặt được các hàm bằng kỹ thuật đệ quy tuyến tính.

2. LÝ THUYẾT CẦN GHI NHỚ

2. 1. Khái niệm

Một hàm được gọi là đệ quy nếu bên trong thân hàm có lệnh gọi đến chính nó

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.

Qua việc giải những bài toán được phân nhỏ này, những lời giải sẽ được kết hợp lại để giải quyết bài toán lớn hơn.

Ví dụ 1:giai thừa của một số nguyên dương n được định nghĩa như sau:

```
n!  = 1* 2 * 3 *…* (n-1) *n
    = (n-1)! *n (với 0!=1)
```

Như vậy, để tính n! ta thấy nếu n=0 thì n!=1 ngược lại thì n! = (n -1)! * n

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

```
#include <stdio.h>
#include <conio.h>
/*Hàm tính n! bằng đệ quy */
int giaithua_dequy(int n)
{
     if (n==0)
          return 1;
     else
          return n*giaithua_dequy(n-1);
}
int main()
{    int n;
     printf("\n Nhap so n can tinh giai thua ");
     scanf("%d",&n);
     printf("\nGoi ham de quy: %d != %d",n,giaithua_dequy(n));
     printf("\nGoi ham khong de quy: %d != %d ", n,
                                        giaithua_khongdequy(n));
     return 0;
}
```

2. 2. Phân loại hàm đệ quy

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

- Đệ quy tuyến tính.
- Đệ quy nhị phân.
- Đệ quy phi tuyến .
- Đệ quy hỗ tương .

2.2.1. *Đệ quy tuyến tính*

- Một hàm được gọi là đệ quy tuyến tính khi bên trong thân hàm có duy nhất một lời gọi hàm lại chính nó
- *Ví dụ 2* Tính tổng S= 2+4+6+…+ 2n .

  Ta có: `S(n) = 2 + 4 + 6 + .. +2(n-1)+ 2n = S(n-1) + 2n ;`

```
// Hàm cài đặt
long tongchan(int n)
{   if(n==1)
        return 2;
    return 2*n + tongchan(n-1);
}
```

2.2.2. *Đệ quy nhị phân*

- Một hàm được gọi là đệ quy nhị phân khi bên trong thân hàm có 2 lời gọi hàm gọi lại chính nó một cách tường minh.
- Chúng ta thường dùng để cài đặt thuật toán chia để trị hay duyệt cây nhị phân.
- *Ví dụ 3*: Viết chương trình tính số hạng thứ n của chuỗi Fibonaci. Chuỗi số Fibonacci có dạng: 1 1 2 3 5 8 13 ...

  Dãy Fibonaci được định nghĩa truy hồi như sau:

  - $F_0 = F_1 = 1$
  - $F_n = F_{n-1} + F_{n-2}$ nếu n>=2

```
// hàm cài đặt
long fibo (int n)
{
    if (n <=1)
        return 1;
    else
        return fibo(n-1)+ fibo(n-2);
}
```

2.2.3. *Đệ quy phi tuyến*

- Trong các chương trình đệ quy phi tuyến, việc gọi đệ quy sẽ được thực hiện bên trong vòng lặp.
- Ví dụ 4 Cho dãy Xn được xác định theo công thức truy hồi như sau:

$$\begin{cases} X_0 = 1 \\ X_n = 1^2x_0 + 2^2x_1 + \ldots + n^2x_{n-1} \end{cases}$$

```
long Xn (int n)
{   long tp= 0;
    if (n == 0)
        return 1;
    for (int i =0 ; i < n ; i++)
        tp = tp + (i+1) * (i+ 1) * Xn (i);
    return tp;
}
```

2.2.4. Đệ quy hỗ tương

- Hai hàm được gọi là đệ quy hỗ tương khi bên trong thân hàm này có lời gọi hàm tới hàm kia, và bên trong thân hàm kia có lời gọi hàm tới hàm này.
- Ví dụ 5 Viết hàm tính số hạng thứ n của 2 dãy số sau:

$$\begin{cases} x_0=1 \\ x_n=x_{n-1}+y_{n-1} \quad (n>=1) \end{cases}$$

$$\begin{cases} y_0=1 \\ y_n=2x_{n-1}+3y_{n-1} \quad (n>=1) \end{cases}$$

```
// hàm cài đặt
long tinhx (int n)
{
    if (n == 0)
        return 1;
    return tinhx (n-1) + tinhy (n-1) ;
}
long tinhy (int n)
{
    if (n == 0)
        return 1;
    return 2 * tinhx (n-1)+ 3 * tinhy (n-1);
}
```

2. 3. Các bước đề nghị khi xây dựng hàm đệ quy

Bước 1. Tham số cần truyền cho hàm là gì? Số lượng tham số là bao nhiêu?
Bước 2. Hàm có trả về giá trị hay không? *(void, int, float, ...)*
Bước 3. Điều kiện dừng là gì? ⇨ Khi đó giá trị trả về là bao nhiêu?
Bước 4. Phần không đệ quy là gì? *(n-1, n/10, n%10, A[n-1], ...)*
Bước 5. Phần đệ quy là gì? ⇨ Kích thước nhỏ hơn của n là gì *(n-1, n/10, n%10, ...)*?
Bước 6. Tinh chỉnh lại nội dung hàm (nếu có thể).

3. BÀI TẬP CÓ HƯỚNG DẪN

Bài 1. Yêu cầu viết hàm đệ quy tính giai thừa cho số nguyên dương n.

B1: Tham số cần truyền cho hàm là số nguyên n.
B2: Hàm trả về số nguyên là giai thừa của n.
B3: Điều kiện dừng: khi n=1 ⇒ 1!=1.

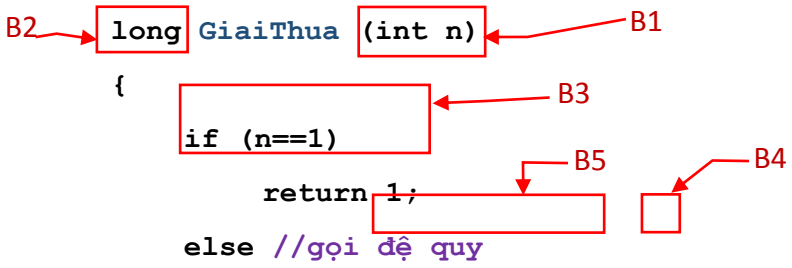
Do: (n)! = 1 * 2 * 3 * … * (n-2) * (n-1) * **n**

(n-1)! = 1 * 2 * 3 * … * (n-2) * **(n-1)**

(n-2)! = 1 * 2 * 3 * … * **(n-2)**

B4: ⇒ Phần không đệ quy: **n**
B5: ⇒ Phần đệ quy: **n-1**

Nội dung hàm cần xây dựng:



B6: Tinh chỉnh lại hàm ở B5:

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

Bài 2. Viết hàm đệ quy tính tổng các số có trong mảng 1 chiều các số nguyên A gồm n phần tử.

B1: Tham số cần truyền cho hàm là: mảng số nguyên A và số nguyên n (số lượng phần tử đang có trong mảng.

B2: Hàm trả về số nguyên là tổng các số có trong mảng A.

Do chỉ số của các phần tử trong mảng được tính từ 0 đến n-1, nên:

Tổng các số từ A[0] đến A[n-1]: Tổng các giá trị tử A[0] đến A[n-2] + **A[n-1]**

Tổng các số từ A[0] đến A[n-2]: Tổng các giá trị tử A[0] đến A[n-3] + **A[n-2]**

Tổng các số từ A[0] đến A[n-3]: Tổng các giá trị tử A[0] đến A[n-4] + **A[n-3]**

B3: Điều kiện dừng: khi n=1 ⇒ return A[0] hoặc khi n=0 return 0.

B4: ⇒ Phần không đệ quy: **A[n-1]**

B5: ⇒ Phần gọi đệ quy: mảng **A** và số lượng phần tử là **n-1**

Nội dung hàm cần xây dựng:

```
int Tong(int A[], int n)
{
     if (n == 0)
          return 0;
     return Tong(A, n - 1) + A[n - 1];
}
```

Bài 3. Viết hàm đệ quy tính tổng các số lẻ có trong mảng 1 chiều các số nguyên A gồm n phần tử.

Tương tự như bài 2, hàm này có tham số truyền cho hàm, kiểu dữ liệu trả về và điều kiện dừng là không thay đổi, chỉ khác ở bước 4 và bước 5

B1: Tham số cần truyền cho hàm là: mảng số nguyên A và số nguyên n (số lượng phần tử đang có trong mảng.

B2: Hàm trả về số nguyên là tổng các số có trong mảng A.

Do chỉ số của các phần tử trong mảng được tính từ 0 đến n-1, nên:

Tổng các số từ A[0] đến A[n-1]: Tổng các giá trị tử A[0] đến A[n-2] + **A[n-1]**

Tổng các số từ A[0] đến A[n-2]: Tổng các giá trị tử A[0] đến A[n-3] + **A[n-2]**

Tổng các số từ A[0] đến A[n-3]: Tổng các giá trị tử A[0] đến A[n-4] + **A[n-3]**

B3: Điều kiện dừng: khi n=1 ⇒ return A[0] hoặc khi n=0 return 0.

B4: Phần không đệ quy:

- Nếu A[n-1] là số lẻ: **+ A[n-1]**
- Nếu A[n-1] là số chẵn: **+ 0**.

B5: Phần gọi đệ quy: luôn mảng **A** và số lượng phần tử là **n-1**

Nội dung hàm cần xây dựng:

```
int TongLe(int A[], int n)
{
    if (n == 0)
        return 0;
    if (A[n - 1]%2==1)
        return TongLe(A, n-1) + A[n-1];
    else
        return TongLe(A, n-1);
}
```

B6: Tinh chỉnh lại nội dung của hàm ở bước 5:

```
int TongLe(int A[], int n)
{
    if (n == 0)
        return 0;
    return TongLe(A,n-1) + (A[n-1]%2 == 1 ? A[n-1] : 0);
}
```

4. BÀI TẬP DO SINH VIÊN TỰ THỰC HIỆN

Dùng phương pháp đệ quy giải các bài tập sau

4. 1. Thao tác trên 1 số nguyên

Bài 1. Đếm số lượng chữ số của số nguyên dương n. VD: n=29730=> 29730 gồm 5 chữ số

Bài 2. Tính tổng các chữ số của số nguyên dương n.

Bài 3. Tính tích các chữ số của số nguyên dương n.

Bài 4. Đếm số lượng chữ số lẻ của số nguyên dương n.

Bài 5. Tính tổng các chữ số chẵn của số nguyên dương n.

Bài 6. Tính tích các chữ số lẻ của số nguyên dương n.

Bài 7. Cho số nguyên dương n. Tìm chữ số đầu tiên của n.

Bài 8. Tìm chữ số đảo ngược của số nguyên dương n.

Bài 9. Tìm chữ số lớn nhất của số nguyên dương n.

Bài 10. Tìm chữ số nhỏ nhất của số nguyên dương n.

Bài 11. Kiểm tra số nguyên dương n có toàn chữ số lẻ hay không?

Bài 12. Tìm ước số lẻ lớn nhất của số nguyên dương n. Ví dụ n=100 ước lẻ lớn nhất của 100 là 25.

Bài 13. Kiểm tra số nguyên dương n có toàn chữ số chẵn hay không?

Bài 14. Tìm ước số chung lớn nhất của hai số nguyên dương a và b.

Bài 15. Tính $P(n) = 1\ x\ 3\ x\ 5\ x\ \ldots x\ (2n+1)$, với n>0

Bài 16. Tính $S(n) = 1-2+3-4+\cdots+(-1)^{n+1}n$, với n>0

Bài 17. Tính $S(n) = 1+1.2+1.2.3+\cdots+1.2.3\ldots n$, với n>0

Bài 18. Tính $S(n) = 1^2+2^2+3^2+\cdots+n^2$, với n>0

Bài 19. Tính $S(n) = 1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$, với n>0

Bài 20. Tính $S(n) = 1+\frac{1}{1+2}+\frac{1}{1+2+3}+\cdots+\frac{1}{1+2+3+\cdots+n}$, với n>0

Bài 21. Tính P(x,y)=xy

Bài 22. Tính $S(n) = 1+(1+2)+(1+2+3)+\cdots+(1+2+3+\cdots+n)$, với n>0

Bài 23. Tính $S(n) = 1+\frac{1}{2}+\frac{1}{4}+\frac{1}{6}\ldots.+\frac{1}{2n}$

Bài 24. Tính $S(n) = 1+\frac{1}{3}+\frac{1}{5}+\frac{1}{7}+\ldots.+\frac{1}{2n+1}$

Bài 25. Tính $S(n) = \frac{1}{1\times 2}+\frac{1}{2\times 3}+\frac{1}{3\times 4}+\ldots.+\frac{1}{n\times(n+1)}$

Bài 26. Tính $S(n) = \frac{1}{2}+\frac{3}{4}+\frac{5}{6}\ldots.+\frac{2n+1}{2n+2}$

Bài 27. Tính S(n) = x2 + x4 +…+x2n.

Bài 28. Tính $S(n) = 1+\frac{1}{1+2}+\frac{1}{1+2+3}+\frac{1}{1+2+3+4}+\ldots.+\frac{1}{1+2+3+\ldots+n}$

Bài 29. Tính $S(n) = x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\ldots.+\frac{x^n}{n!}$

4. 2. Vẽ hình

Bài 30. Viết chương trình cho người dùng nhập cạnh (n) của tam giác. Giả sử với n=5, chương sẽ lần lượt in ra các hình sau:

a

```
* * * * *
* * * *
* * *
* *
*
```

b

```
        *
      * *
    * * *
  * * * *
* * * * *
```

c

```
* * * * *
*     *
*   *
* *
*
```

d

```
        *
      * *
    *   *
  *     *
* * * * *
```

4. 3. Mảng một chiều

Bài 31. Viết hàm nhập mảng một chiều các số nguyên gồm n phần tử (0< n <= 100).

Bài 32. Viết hàm xuất mảng số nguyên n phần tử vừa nhập ở trên.

Bài 33. Tính tổng tất cả các phần tử trong mảng

Bài 34. Tính tổng các số lẻ có trong mảng.

✍ *Mở rộng:* cho trường hợp: số chẵn, số nguyên tố, số chính phương, …

Bài 35. Tìm phần tử chẵn đầu tiên có trong mảng.

✍ *Mở rộng:* cho trường hợp: số lẻ, số nguyên tố, số chính phương, …

Bài 36. Tìm phần tử chẵn cuối cùng có trong mảng.

✍ *Mở rộng*: cho trường hợp giá trị cuối cùng là số lẻ, số nguyên tố, giá trị cuối cùng là số chính phương, …

Bài 37. Tìm phần tử lớn nhất có trong mảng.

Bài 38. Đếm số phần tử chẵn có trong mảng.

Bài 39. Đếm số phần tử lớn nhất có trong mảng.

Bài 40. In ra vị trí của phần tử lớn nhất đầu tiên có trong mảng.

Bài 41. Sắp xếp mảng tăng dần.

Bài 42. Kiểm tra xem mảng có được sắp xếp tăng dần hay không?

Bài 43. Kiểm tra xem mảng có chứa số nguyên tố hay không?