

Nội dung

1

Vấn đề 4: Lãi suất kép

2

Vấn đề 5: Độ qui trên mảng

3

Vấn đề 6: Kỹ thuật đo thời gian

4

Thảo luận

Vấn đề 4: Lãi suất kép

- Số tiền gửi: y
 - Lãi suất cố định 1 năm: $x\%$
 - Thời gian gửi: n năm.
- ? Tính số tiền thu được (vốn + lãi).

❖ Yêu cầu:

1. Gọi $P(n)$ là số tiền thu được sau n năm gửi số tiền y .
Lập công thức đệ quy tính $P(n)$

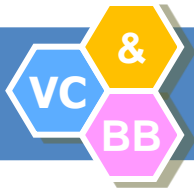
$$P(0) = y$$

$$\text{Sau 1 năm } P(1) = P(0) + x\% * P(0) = (1 + x\%) * P(0)$$

$$\text{Sau 2 năm } P(2) = P(1) + x\% * P(1) = (1 + x\%) * P(1)$$

....

$$\text{Sau } n \text{ năm } P(n) = P(n-1) + x\% * P(n-1) = (1 + x\%) * P(n-1)$$



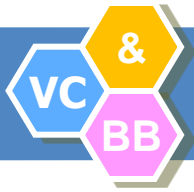
Vấn đề 4: Lãi suất kép

- Số tiền gửi: y
 - Lãi suất cố định 1 năm: $x\%$
 - Thời gian: n năm.
- ? Tính số tiền thu được (vốn + lãi).

❖ Yêu cầu:

1. Gọi $P(n)$ là số tiền thu được sau n năm gửi số tiền y vào A. Lập công thức đệ quy tính $P(n)$

$$P(n) = \begin{cases} y & \text{nếu } n = 0 \\ (1 + x\%) * P(n - 1) & \text{nếu } n > 0 \end{cases}$$



Vấn đề 4: Lãi suất kép

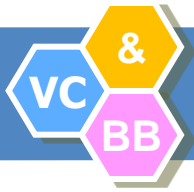
$$P(n) = \begin{cases} y & \text{nếu } n = 0 \\ (1 + x\%) * P(n - 1) & \text{nếu } n > 0 \end{cases}$$

❖ **Yêu cầu:** 2. Cài đặt các hàm đệ quy *trả về số tiền thu được sau n năm*

- Tên hàm: ?
- Kiểu giá trị trả về: ?
- Tham số: ?, kiểu dữ liệu của tham số: ?

=> Nguyên mẫu hàm:

```
float laiKep(int n, float y, float x ) ;
```



Vấn đề 4: Lãi suất kép

$$P(n) = \begin{cases} y & \text{nếu } n = 0 \\ (1 + x\%) * P(n - 1) & \text{nếu } n > 0 \end{cases}$$

❖ Yêu cầu:

2. Cài đặt các hàm đệ quy *trả về số tiền thu được sau n năm*

```
float laiKep(int n, float y, float x )  
{  
    if( n == 0) return y;  
    return (1 + x/100)*laiKep(n-1,y,x)  
}
```



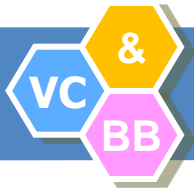
Vấn đề 4: Lãi suất kép

```
float laiKep(int n, float y, float x )
```

❖ Yêu cầu:

3. Cài đặt chương trình thực hiện vấn đề 4 bằng hàm chính:

```
int main()  
{  
    int n;  
    float y,x;  
    printf( "Nhap so nam n = " ); scanf( "%d", &n);  
    printf( "Nhap tien von y= " ); scanf( "%f", &y);  
    printf( "Nhap lai suất x= " ); scanf( "%f", &x);  
    printf( "Tien = %.4f", laiKep(n,y,x));  
}
```



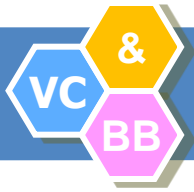
Vấn đề 5: Đệ quy trên mảng

- ❖ **Yêu cầu:** Cài đặt hàm để sinh ngẫu nhiên n phần tử ($1 \leq n \leq 10^6$) và lưu vào mảng $Q[]$.

```
void input(int Q[], int &n)
```

- Khai báo thư viện `#include <stdlib.h>`
- Trong thân hàm gọi thủ tục `srand` gọi trước khi gọi hàm `rand()`

```
srand (time(NULL));
```



Vấn đề 5: Đệ quy trên mảng

❖ **Yêu cầu:** Cài đặt hàm để sinh ngẫu nhiên n phần tử ($1 \leq n \leq 10^6$) và lưu vào mảng $Q[]$.

```
void input(int Q[], int &n)
```

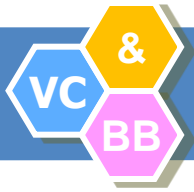
- Sử dụng hàm **rand()** để sinh số ngẫu nhiên:
 - Số nguyên từ 0 đến n : $\text{rand()} \% (n+1)$;
 - Số nguyên từ a đến b : $a + \text{rand()} \% (b-a+1)$
 - Số thực từ 0 đến n : $n * \text{rand()} / \text{RAND_MAX}$;
 - Số thực từ a đến b : $a + (b - a) * \text{rand()} / \text{RAND_MAX}$;



Vấn đề 5: Độ quy trên mảng

```
#include <stdlib.h>

void input(int Q[], int &n)
{
    printf("\n Nhap so phan tu n =:");
    scanf("%d", &n);
    while (n < 0)
    {
        printf("\n Nhap lai so phan tu :");
        scanf("%d", &n);
    }
    srand (time(NULL));
    for(int i=0; i < n; i++)
        Q[i]= 1 + rand()%100;
}
```

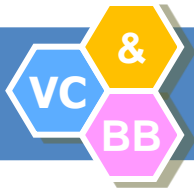


Vấn đề 5: Đệ quy trên mảng

- ❖ **Yêu cầu:** Cài đặt hàm để hiển thị các phần tử của **Q[]** lên màn hình.

```
void output(int Q[], int n)
```

```
void output(int Q[], int n)
{
    printf("\n In mang:");
    for(int i=0; i < n; i++)
        printf("%d\t", Q[i]);
}
```



Vấn đề 5: Độ quy trên mảng

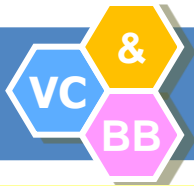
- ❖ **Yêu cầu:** xây dựng định nghĩa đệ quy để tính tổng n phần tử của dãy Q

```
long sum_rec(int Q[], int n)
```

Gợi ý:

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

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



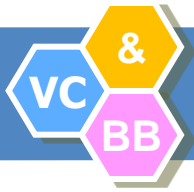
Vấn đề 5: Độ quy trên mảng

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

❖ Yêu cầu:

7. Cài đặt các hàm đệ quy *trả về tổng các phần tử của dãy số Q*.

```
long sum_rec(int Q[], int n)
{
    if( n == 0) return 0;
    return Q[n-1] + sum_rec(Q, n-1);
}
```



Vấn đề 5: Độ quy trên mảng

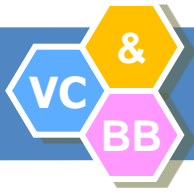
- ❖ **Yêu cầu: 9.** Xây dựng định nghĩa độ quy tìm phần tử lớn nhất của mảng Q

```
int max_rec(int Q[], int n)
```

Gợi ý:

$Max(q, n) = \underline{q[0], q[1], q[2], \dots, q[n-2], q[n-1]}$
 $Max(q, n-1)$

$$Max(q, n) = \begin{cases} q[0] & \text{nếu } n = 1 \\ q[n-1] > Max(q, n-1) ? q[n-1] : Max(q, n-1) & \text{nếu } n > 1 \end{cases}$$



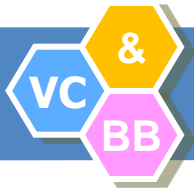
Vấn đề 5: Độ quy trên mảng

$$Max(q, n) = \begin{cases} q[0] & \text{nếu } n = 1 \\ q[n-1] > Max(q, n-1) ? q[n-1] : Max(q, n-1) & \text{nếu } n > 1 \end{cases}$$

❖ Yêu cầu:

9. xây dựng định nghĩa đệ quy tìm phần tử lớn nhất của mảng Q

```
int max_rec(int Q[], int n)
{
    if( n == 1) return Q[0];
    if(Q[n-1] > max_rec(Q, n-1))
        return Q[n-1];
    return max_rec(Q, n-1);
}
```



Vấn đề 5: Đệ quy trên mảng

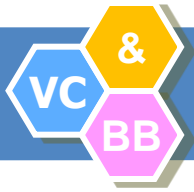
❖ **Yêu cầu: 10.** Xây dựng định nghĩa đệ quy *tìm phần tử nhỏ nhất* của mảng Q

```
int min_rec(int Q[], int n)
```

Gợi ý:

$\text{Min}(q, n) = \underline{q[0], q[1], q[2], \dots, q[n-2], q[n-1]}$
 $\text{Min}(q, n-1)$

$$\text{Min}(q, n) = \begin{cases} q[0] & \text{nếu } n = 1 \\ q[n-1] < \text{Min}(q, n-1) ? q[n-1] : \text{Min}(q, n-1) & \text{nếu } n > 1 \end{cases}$$



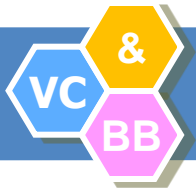
Vấn đề 5: Độ quy trên mảng

- ❖ **Yêu cầu:** 11.xây dựng hàm *không đệ quy* để tính tổng n phần tử của dãy Q

```
long sum (int Q[], int n)
```

Gợi ý: sử dụng vòng lặp

```
long sum (int Q[], int n)
{
    long s = 0;
    for(int i = 0; i < n; i++)
        s = s + Q[i];
    return s;
}
```

Vấn đề 6: Kỹ thuật đo thời gian

```
#include<time.h>
int main()
{
    int n = 10000;
    // Đo thời gian thực hiện hàm không đệ qui
    clock_t start= clock();
    sum(Q,n);
    clock_t end = clock();
    cout<<"Time = "<<(float) (end-start)/100<<" (s) "<<endl;
}
```