

Phần STRUCT kết hợp ARRAY

Câu 1: VCT tính tổng 1 dãy phân số, sắp xếp lại dãy theo thứ tự tăng & xuất kết quả.

Câu 2: Đề xuất kiểu dữ liệu thích hợp để biểu diễn 1 đa giác & viết hàm tính chu vi.

Câu 3: Viết hàm tìm đa thức có giá trị lớn nhất tại điểm x_0 trong 1 danh sách các đa thức.

Biết đa thức bậc N có dạng $\{ a_0 + a_1X + a_2X^2 + \dots + a_NX^N \}$ với a_i là các hệ số thực và $a_N \neq 0$

Câu 4: Viết hàm sắp xếp 1 danh sách các thời điểm, với mỗi thời điểm bao gồm 5 thông tin : ngày – tháng – năm – giờ - phút.

Câu 5: Bài toán cổ “100 trâu 100 cỏ, trâu đứng ăn 5, trâu nằm ăn 3, lụ khụ trâu già, 3 con ăn 1” là một bài toán có nhiều nghiệm. Hãy khai báo 1 struct chứa nghiệm và viết các hàm sau :

a/ Tìm nghiệm đầu tiên, sao cho có thể gọi lại hàm này tìm nghiệm kế tiếp, viết bằng 2 cách : không đệ quy và có đệ quy.

b/ Tìm tất cả các nghiệm và đưa vào 1 mảng, viết bằng 2 cách : không đệ quy và có đệ quy.

Bài Sửa

Câu 1: (VCT tính tổng 1 dãy phân số, sắp xếp lại dãy theo thứ tự tăng & xuất kết quả)

*** Project nên được chia thành 03 module : Main, Phân_Số, Mảng_Phân_Số.**

+ Module **Main** chứa CT chính gọi các hàm thực hiện các thao tác theo yêu cầu của đề bài :

- Nhập_Dãy_Phân_Số
- Tính_Tổng_Dãy_Phân_Số
- Xuất_Phân_Số_Kết_Quả
- Sắp_Xếp_Dãy_Phân_Số
- Xuất_Dãy_Phân_Số_Kết_Quả

+ Module **Phân_Số** chứa định nghĩa kiểu PhânSố và các hàm Nhập_1_Phân_Số, Xuất_1_Phân_Số, Tính_Tổng_2_Phân_Số.

Ngoài các hàm chính trên, để phân số được xuất ra ở dạng dễ nhìn cần rút gọn thành dạng tối giản trước khi xuất, tức nên có thêm hàm phụ Rút_Gọn_Phân_Số, và phát sinh thêm 1 hàm hỗ trợ nữa là Tìm_Uớc_Chung_Lớn_Nhất.

Module này sẽ được tổ chức thành 02 file : file <PhanSo.h> chứa khai báo định nghĩa kiểu PhânSố và Prototype của các hàm chính trong module đã liệt kê ; file <PhanSo.cpp> chứa các định nghĩa hàm chính và các hàm phụ cần thiết.

+ Module **Mảng_Phân_Số** chứa định nghĩa kiểu MảngPhânSố và các hàm thao tác trên dãy phân số : Nhập_Dãy_Phân_Số, Xuất_Dãy_Phân_Số, Sắp_Xếp_Dãy_Phân_Số.

Module này cũng sẽ được tổ chức thành 02 file : file <Mang.h> chứa khai báo định nghĩa kiểu MảngPhânSố và Prototype của các hàm chính; file <Mang.cpp> chứa các định nghĩa hàm.

*** Mã nguồn của các module có thể viết như sau:**

1/ **Module Main** (Nội dung file Main.cpp) :

```
#include "Mang.h"
void main(){
    MANG_PS P;
    cout << "Nhap day phan so @ ";
    Nhap(P);
    PHANSO sum = Tong(P);
    cout << "Tong tat ca phan so trong day: ";
    XuatPS(sum);
    SapXep(P);
    cout << "\n Day phan so sau khi sap xep: ";
    Xuat(P);
    cout << endl << "OK?";
}
```

2/ **Module Phân_Số**

Nội dung file PhanSo.h :

```
#include <iostream>
using namespace std;
typedef struct {
    int tu;
    int mau;
} PHANSO;

void NhapPS(PHANSO&);
void XuatPS(PHANSO);
PHANSO CongPS(PHANSO, PHANSO);
PHANSO TruPS(PHANSO, PHANSO);
```

Nội dung file PhanSo.cpp :

| | |
|--|--|
| <pre>#include "PhanSo.h" int UCLN(int a, int b) { int ucln = a < b ? a : b; while (a % ucln b % ucln) ucln--; return ucln; } void RutGonPS(PHANSO &a){ int v1 = a.tu, v2 = a.mau; if (v1 < 0) v1 = -v1;</pre> | <pre>if (v2 < 0) v2 = -v2; int ucln = UCLN(v1, v2); a.tu /= ucln; a.mau /= ucln; } void NhapPS(PHANSO &a) { cout << " # tu va mau: "; cin >> a.tu >> a.mau; } void XuatPS(PHANSO a) {</pre> |
|--|--|

```

        RutGonPS(a);
        cout <<a.tu <<"/"<< a.mau<<" ";
    }
    PHANSO CongPS(PHANSO a, PHANSO b){
        PHANSO c;
        c.mau = a.mau * b.mau;
        c.tu = a.tu * b.mau + b.tu * a.mau;
        return c;
    }
    PHANSO DoiDauPS(PHANSO a){
        PHANSO b;
        b.mau = a.mau;
        b.tu = -a.tu;
        return b;
    }
    PHANSO TruPS(PHANSO a, PHANSO b){
        PHANSO c = CongPS(a, DoiDauPS(b));
        return c;
    }

```

3/ Module **Mảng_Phân_Số**

Nội dung file Mang.h :

```

#include "PhanSo.h"
#define MAX 100
typedef struct {
    PHANSO a[MAX];
    int n; // so phan tu cua mang
}MANG_PS;

void Nhap(MANG_PS&);
void Xuat(MANG_PS);
PHANSO Tong(MANG_PS); // tinh tong tat ca phan so trong mang
void SapXep(MANG_PS&); // theo thu tu tang

```

// Trường hợp các bạn include tập tin .H dư thừa ở nhiều chỗ trong project và xài phải 1 compiler khó chịu không đồng ý thì có thể ngăn vụ duplicate bằng cách đặt các điều kiện tiền xử lý để phần nội dung của file .H không xuất hiện nhiều lần (dù cho có include khắp nơi) như sau :

```

#ifndef HEADER_H // ví dụ như PHANSO_H với file PhanSo.h
#define HEADER_H
/* contents of header file */
#endif

```

Nội dung file Mang.cpp :

```

#include "Mang.h"

void Nhap(MANG_PS& F) {
    cout << "Nhap so phan tu: ";
    cin >> F.n;
    for (int i = 0; i < F.n; i++) {
        cout << "Nhap phan so thu " << i + 1 << ": ";
        NhapPS(F.a[i]);
    }
}

void Xuat(MANG_PS F) {
    for (int i = 0; i < F.n; i++)
        XuatPS(F.a[i]);
}

```

```

}

PHANSO Tong(MANG_PS F) {
    PHANSO sum = F.a[0];
    for (int i = 1; i < F.n; i++)
        sum = CongPS(sum, F.a[i]);
    return sum;
}

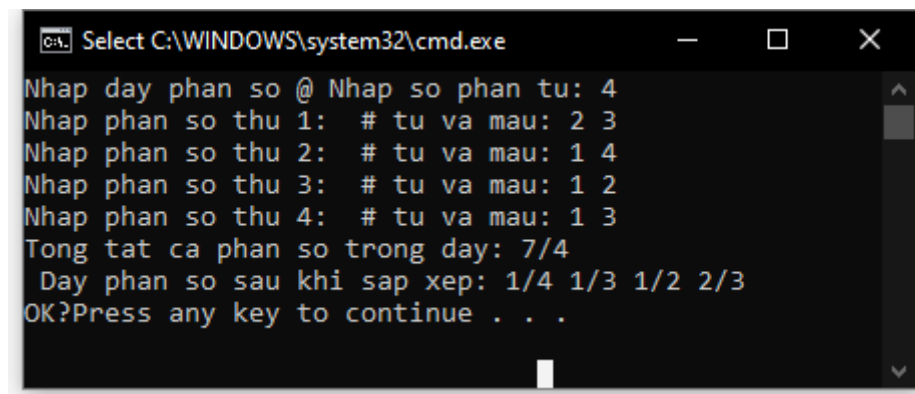
int max_index(const PHANSO a[], int n) {
    int maxIdx = 0;
    float max = (float)a[maxIdx].tu / a[maxIdx].mau;
    for (int i = 1; i < n; ++i)
        if ((float)a[i].tu / a[i].mau > max) {
            maxIdx = i;
            max = (float)a[maxIdx].tu / a[maxIdx].mau;
        }
    return maxIdx;
}

void SapXep(MANG_PS& F) {
    int k = F.n;
    while (--k){
        int j = max_index(F.a, k + 1);
        swap(F.a[j], F.a[k]);
    }
}

```

~~~~~

\* **Kết quả chạy của chương trình :**



```

C:\> Select C:\WINDOWS\system32\cmd.exe
Nhap day phan so @ Nhap so phan tu: 4
Nhap phan so thu 1: # tu va mau: 2 3
Nhap phan so thu 2: # tu va mau: 1 4
Nhap phan so thu 3: # tu va mau: 1 2
Nhap phan so thu 4: # tu va mau: 1 3
Tong tat ca phan so trong day: 7/4
Day phan so sau khi sap xep: 1/4 1/3 1/2 2/3
OK?Press any key to continue . . .

```

**Câu 2:** (Đề xuất kiểu dữ liệu thích hợp để biểu diễn 1 đa giác & viết hàm tính chu vi đa giác)

**\* Kiểu dữ liệu biểu diễn đa giác :**

+ Một đa giác (phẳng) được xác định bởi N đỉnh ( $N > 2$ ), trong đó mỗi đỉnh là một điểm trên mặt phẳng được xác định bởi tọa độ (x, y) cụ thể (hoành độ và tung độ). Do đó kiểu dữ liệu thích hợp để biểu diễn N giác (đa giác có N đỉnh) sẽ là kiểu mảng N phần tử, với mỗi phần tử là một điểm. **[1 điểm(10)]**

+ Như vậy kiểu Đa giác có thể được định nghĩa như sau :

```

#define MAX 100 // giả sử  $N < 100$ 
    
```

```

typedef struct {
    
```

```

        int N;           // số đỉnh của đa giác
    
```

```

        POINT a [MAX]; // tọa độ các đỉnh 0, 1, 2, ... N-1
    
```

```

} POLYGON; [25 điểm(thiếu comment trừ 1 điểm)]
    
```

Trong đó kiểu Điểm chỉ đơn giản là :

```

typedef struct {
    
```

```

        float x, y;
    
```

```

} POINT; [0.5 điểm]
    
```

+ Với định nghĩa trên thì đa giác P có thể khai báo :

```

POLYGON P; [0.5 điểm]
    
```

Khi này số đỉnh của đa giác P (cũng là số cạnh) sẽ là  $\langle P.N \rangle$  ;

Các đỉnh của đa giác P sẽ là  $\langle P.a[0] \rangle$  ,  $\langle P.a[1] \rangle$  , ...,  $\langle P.a[P.N-1] \rangle$  ;

Các cạnh của đa giác là các đoạn nối đỉnh 0 với đỉnh 1, đỉnh 1 với đỉnh 2, ..., đỉnh N-2 với đỉnh N-1, và đỉnh N-1 với đỉnh 0. **[0.5 điểm]**

Chiều dài cạnh nối 2 đỉnh kế nhau  $\langle P.a[i] \rangle$  và  $\langle P.a[i+1] \rangle$  sẽ là :

float **d** = **sqrt ( d1\*d1 + d2\*d2 )** , với  $d1 = P.a[i].x - P.a[i+1].x$  và  $d2 = P.a[i].y - P.a[i+1].y$

(vì công thức tính khoảng cách giữa 2 điểm  $A(x_A, y_A)$  và  $B(x_B, y_B)$  là  $\text{sqrt}((x_A - x_B)^2 + (y_A - y_B)^2)$

**[1 điểm]**

**\* Hàm tính chu vi đa giác :**

Trên cơ sở kiểu Đa giác đã đề xuất và cách tính Chiều dài cạnh nối 2 đỉnh kế nhau nêu trên, ta có thể viết hàm tính Chu vi của đa giác như sau :

```
float Perimeter (POLYGON P) {    // Hàm tính chu vi của đa giác P
    float p = distance (P.a[P.N-1], P.a[0]); // cạnh nối đỉnh cuối N-1 với đỉnh 0
    for (int i = 0; i < P.N - 1; ++i)
        p += distance (P.a[i], P.a[i+1]); // cộng dồn chiều dài các cạnh còn lại
    return p;
}
```

*[ 3 điểm (thiếu comment trừ 0.5 điểm) ]*

```
// Hàm tính chiều dài cạnh cũng là khoảng cách 2 điểm A, B
float distance (POINT A, POINT B) {
    float d1 = A.x - B.x, d2 = A.y - B.y;
    return sqrt(d1*d1 + d2*d2);
}
```

*[ 1 điểm ]*

*[ Phần Code nếu đặt các tên không tốt (tên hàm biến, hằng cấu trúc) và không thụt đầu dòng chính xác trừ 1 điểm ]*