ĐỀ KIỂM TRA GIỮA KỲ # HK1*2022-2023

Môn: **Nhập môn Lập trình** - Lớp: **22CLC5** - GV: Thái Hùng Văn

Thời gian làm bài: **60** phút - ***Không*** *dùng tài liệu, Phone, Laptop*

Bài 1*(3đ)*: Xét hàm:

```
void F (int N, double S, float& P) {
    S = P = 1;
    for (int j=2; j<=N; j++)
      if (j%2) S += j;
      else     P *= j;
}
```

a/ Cho biết hàm trên thực hiện việc gì?
b/ Chỉ ra các chỗ chưa ổn của hàm?
c/ Thay cấu trúc **for** trong thân hàm bằng **while** hoặc **do..while** sao cho kết quả chạy đoạn chương trình thân hàm không thay đổi.

Bài 2*(3đ)*: Viết các hàm

a/ Tính tổng $-1/2^2 + 2/3^2 - 3/4^2 + 4/5^2 + ... + (-1)^n.n/(n+1)^2$

b/ Tính thương hai số nguyên.

c/ So sánh hai phân số.

d/ Rút gọn phân số *(về dạng tối giản, ví dụ phân số 11/22 có dạng tối giản là 1/2)*.

Ghi chú: ***chỉ cần làm 3 trong 4 câu***, *mỗi câu viết đúng 01 hàm và cần viết sao cho có giá trị sử dụng cao*

Bài 3*(4đ)*: Viết thuật toán và chương trình tìm ước chung lớn nhì của ba số nguyên dương.
(*Ví dụ, nếu 3 số nguyên là 12, 18, 24 thì ước chung lớn nhì là 3)*

ĐÁP ÁN

Bài 1

a/(1đ) Nội dung thực hiện của hàm

- Đoạn mã thân hàm tính tổng S các số (nguyên) lẻ và tích P các số chẵn <=N *(trường hợp đặc biệt N<2 thì S=P=1)* // S = 1+3+5+...+N và P=2*4*6*..*(N-1) nếu N lẻ; S = 1+3+5+ ...+(N-1) và P=2*4*6*..*N nếu N chẵn.
- Hàm nhận tham số vào là số nguyên N và có kết quả ra ở tham biến P, còn biến S được truyền ở dạng tham trị *(do ko có dấu '&' phía trước trong khai báo hàm)* nên sau khi hàm thực hiện xong thì S không được cập nhật lại giá trị, do đó có thể nói là hàm chỉ tính được tích các số chẵn (và khi gọi hàm thì ở đối số thứ hai ko nhất thiết phải truyền vào 1 biến mà có thể truyền 1 hằng số /biểu thức cũng được).

b/(1đ) Các chỗ chưa ổn của hàm

- Tên hàm chỉ có mỗi một ký tự không đủ rõ để biết hàm làm gì.
- Không có các chú thích cần thiết *(hàm làm gì, các tham số mang ý nghĩa gì, kết quả trong tr.hợp đặc biệt, ...)*
- Tham số thứ hai S được thiết lập dạng tham trị nên sau khi tính xong tổng các số lẻ thì biến S sẽ được giải phóng không còn tồn tại nữa dẫn đến kết thúc hàm sẽ không nhận được kết quả tính S – việc tính tổng trở thành vô nghĩa.

- Tham số S chứa tổng và P chứa tích nên kiểu khai báo như vậy là tréo nghoe! *(tổng sẽ nhỏ hơn rất nhiều so với tích nên lẽ ra phải khai báo ngược lại: S kiểu float còn P phải kiểu double).*

c/(1đ) Thay cấu trúc **for** trong thân hàm bằng **while** *(kết quả chạy không thay đổi):*

```
S = P = 1;
int j=2;
while (j<=N) {
    if ( j%2)
            S += j;
    else    P *= j;
    j ++;
}
```

Hoặc thay cấu trúc while trên như sau thì hay hơn 1 chút:

```
while (j<=N) {
    P *= j++;
    S += j++;
}
if (j==N+2)   S -= N+1;
```

Bài 2

a/(1đ) Hàm tính tổng $-1/2^2 + 2/3^2 - 3/4^2 + 4/5^2 + ... + (-1)^n.n/(n+1)^2$

```
double TinhTong ( int n) { // hàm trả về tổng  -1/2^2 + 2/3^2 - 3/4^2 + 4/5^2 + ... + (-1)^n.n/(n+1)^2
        double S = 0;
        for ( int i = 1, float sign=-1; i<=n; i++, sign=-sign)
                S += sign*i / sqr(i+1);
        return S;
}
```

b/(1đ) Hàm tính thương 2 số nguyên

```
// tinh <thuong> = a div b và <du> = a mod b. Hàm trả về false nếu không có kết quả
bool TinhThuong ( int a, int b, int & thuong, int & du) {
        if ( b == 0 ) return false;
        thuong = a / b;
        du = a % b;
        return true;
}
```

c/(1đ) Hàm So sánh 2 phân số

```
// So sánh 2 phân số tu1/mau1 & tu2/mau2. Hàm trả về -1 nếu phân số đầu nhỏ hơn, +1 nếu lớn hơn, và 0 nếu cả 2 bằng nhau
int SoSanhPhanSo ( int tu1, int mau1, int tu2, int mau2)  {
        double giatri1 = tu1/mau1, giatri2 = tu2/mau2;
        if ( giatri1 == giatri2 ) return 0;
        if ( giatri1 < giatri2 ) return -1;
        else return +1;
}
```

d/(1đ) Hàm Rút gọn phân số

```
// Rút gọn phân số tu/mau thành tối giản (vd tu=4, mau=-8 thì sau khi rút gọn tu=-1, mau=2)
void RutgonPhanSo ( int & tu, int & mau)  {
        if ( tu == 0 ) return;
        int ucln = __gdc ( tu, mau); // tìm ước chung lớn nhất // để dùng hàm __gdc này cần  #include <algorithm>
        tu /= ucln;
```

```
        mau /= ucln;
        if ( mau < 0 ) {    // phân số rút gọn phải có mẫu là số dương
                mau = - mau;
                tu = - tu;
        }
}
```

Bài 3. Hàm tìm ước chung lớn nhì của ba số nguyên dương.

Cách 1:

*** Thuật toán tìm ước chung lớn nhì của 3 số tự nhiên A, B, C:**

B1: Tìm giá trị nhỏ nhất của 3 số A, B, C và đưa vào biến Min. *(ước chung (UC) lớn nhất chắc chắn <=Min)*

B2: Tìm ước chung gần Min nhất của A, B, C và đưa vào biến UCL1.

B3: U2 = UCL1/2; *(// có thể cho U2=UCL1 - 1 cũng được)*

B4: Tìm ước chung gần U2 nhất của A, B, C => Đó chính là ước chung lớn nhì.

* **Thuật toán tìm ước chung gần X nhất của 3 số A, B, C** *(// ước gần X nhất là ước lớn nhất trong các ước <= X)*

B0: Nếu X=0: Kết thúc (return 0)

B1: Uoc_ = X

B2: Kiểm tra Uoc_ có phải là ước của cả 3 số A, B, C hay không - nếu phải chuyển đến B4

B3: Giảm giá trị Uoc_ đi 1. Quay lại B2

B4: Kết thúc (giá trị của Uoc_ lúc này chính là kết quả cần tìm)

*** Code:**

```
int UCLNhi ( int a, int b, int c ) { // tìm UC lớn nhì của ba số nguyên dương a, b, c; hàm trả về 0 nếu không có
        int Min = Min3So (a, b, c);
        int UCL1 = UC_GanNhat ( a, b, c, Min ); // UC lớn nhất chính là UC gần Min nhất của ba số a, b, c
        return UC_GanNhat ( a, b, c, UCL1/2 );
}
int UC_GanNhat ( int a, int b, int c, int X ) {  // tìm UC gần với X nhất của ba số nguyên dương a, b, c
        if (X==0) return 0;
        while ( (a % X ) || (b % X) || (c % X) ) // X không phải là UC của cả 3 số
                X --;     // có thể dùng chính biến X thay cho biến Uoc_ trong thuật toán
        return X;
}
int Min3So ( int a, int b, int c ) { // tìm số nhỏ nhất trong 3 số a, b, c
        return Min2So ( Min2So(a, b), c );
}
int Min2So ( int a, int b) { // tìm số nhỏ nhất trong 2 số a, b
        if  ( a < b ) return a;
        return b;
}
```

Cách 2:

*** Thuật toán tìm ước chung lớn nhì của 3 số A, B, C:**

B1: Tìm ước chung lớn nhất của 2 số A, B và đưa vào biến UCL1_ab.

B2: Tìm ước chung lớn nhất của 2 số UCL1_ab, C và đưa vào biến UCL1. *(đây là UC lớn nhất của cả 3 số)*

B3: Kiểm tra nếu UCL1==1: trả về kết quả là 0 (không có UC lớn nhì) và kết thúc

B4: Gán UocNhoNhat=2

B5: Kiểm tra nếu UocNhoNhat có là ước của UCL1 thì chuyển đến B7

B6: Tăng giá trị UocNhoNhat thêm 1 và Quay lại B5

B7: Trả về kết quả (UCL1/ UocNhoNhat) và kết thúc. *(// vì UC lớn nhì là ước lớn nhất của UCL1, và bằng UCL1 chia cho ước nhỏ nhất của nó)*

* **Thuật toán tìm ước chung lớn nhất của 2 số tự nhiên A, B**: *// có thể dùng hàm __gdc có sẵn như câu 2d và bỏ qua vụ này!*

(có nhiều cách tìm khác nhau, cách giảm dần mỗi lần 1 đơn vị như dạng tìm ước 3 số ở trên sẽ chậm hơn hẳn cách của bổ đề tổ sư Euclid đưa ra từ những năm 300 tr.CN - https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_Euclid)

B1: Gán x = b và y = a

B2: Kiểm tra y có phải đã bằng 0 - nếu phải chuyển đến B7

B3: Gán r = x mod y

B4: Gán x =y

B5: Gán y = r

B6: Quay lại B2

B7: Kết thúc và Trả về kết quả là x (giá trị của x lúc này chính là ước chung lớn nhất của a và b)

Thuật toán trên có thể viết lại dưới dạng không dùng thêm 2 biến trung gian x, y:

B1: Kiểm tra a có phải đã bằng 0 - nếu phải chuyển đến B5

B2: Gán r = b mod a // chia lấy phần dư

B3: Gán b = a

B3: Gán a= r

B4: Quay lại B1

B5: Trả về kết quả là b (giá trị của b lúc này chính là UC lớn nhất của a và b ban đầu)

* Code:

```
int UCLNhi ( int a, int b, int c ) { // tìm ước chung lớn nhì của ba số nguyên dương a, b, c; hàm trả về 0 nếu không có
        int UCL1_ab = UCLNhat ( a, b ); // UCL1_ab =  ước chung lớn nhất của a, b
        int UCL1 = UCLNhat (UCL1_ab, c ); // UCL1 =  ước chung lớn nhất của a, b, c
        if  ( UCL1 == 1 )  return 0;
        int UocNhoNhat = 2;
        while (UCL1 % UocNhoNhat  != 0)
                UocNhoNhat  ++;
        return UCL1/UocNhoNhat; // vì UC lớn nhì là ước lớn nhất của UCL1, và bằng UCL1 chia cho ước nhỏ nhất của nó
}
int UCLNhat ( int a, int b) { // hàm tìm ước số chung lớn nhất của 2 số a, b
        int r;
        while (a) {
                r = b % a;
                b = a;
                a = r;
        }
        return b;
}
```