THAM SỐ DÒNG LỆNH

Câu 1:

- Tong2SoNguyen.exe: Nhập sai vì số tham số = 1 khác 3.
- Tong2SoNguyen.exe 1 : Nhập sai vì số tham số = 2 khác 3.
- Tong2SoNguyen.exe 1 2: Số tham số = 3. Thực hiện tính tổng.
- Tong2SoNguyen.exe 1 2 3: số tham số = 4 > 3 => nhập sai.

Câu 2:

```
void main(int iSoThSo, char* mangThSo[])
{
if (iSoThSo < 3) //Giải thích chỗ này? : vì tham số đầu tiên là tên chương trình và tham só thứ 2 là số phần tử có trong mảng. theo sau là các phần tử của mảng. nhưng iSoThSo<3 tức = 2. Là chỉ có tên chương trình và số phần tử của mảng.
{
printf("Nhap sai qui dinh");
return;
}
int n = atoi(mangThSo[1]);
if (n != iSoThSo - 2) //Giải thích??  =>  vì n là số phần tử có trong mảng tức từ tham số thứ 3 trở đi. Bỏ 2 tham số đầu tiên. Mà mangThSo[1] là số phần tử có trong mảng. nếu 2 số khác nhau thì số phần tử có trong mảng khác vs số phần tử mảng dc truyền vào.
{
printf("So phan tu khong khop");
return;
}
```

printf("Mang vua nhap vao la: ");

for (int i = 2; i < iSoThSo; i++) // Vì sao i = 2??? Vì 2 tham số đầu là tên chương trình và số phần tử của mảng. các phần tử mảng bắt đầu từ tham số thứ 3.

printf("%d ",atoi(mangThSo[i]));

}

- NhapXuatMang.exe 3 : nhap sai quy định vi so tham so < 3.
- NhapXuatMang.exe 3 7 5 : số phần tử không hợp lệ vì mangThSo[1]= 3 tức mảng có 2 phần tử nhưng chỉ có 1 phần tử theo sau.
- NhapXuatMang.exe 3 7 5 8 : Mang vua nhap vao la: 7 5 8
- NhapXuatMang.exe 3 7 5 8 1 : số phần tử không hợp lệ vì mangThSo[1]=3 tức mảng có 3 phần tử nhưng có 4 phần tử theo sau.

Lab 1_ Ôn Tập

Bài 1:

1. Biên dịch đoạn chương trình trên.

2. Nếu lệnh

int a = 1;

được đổi thành

int a = 10;

Cho biết giá trị của *pa.

->*pa=10;

3. Nếu dòng

int *pa = &a;

được sửa lại thành

int *pa;

Cho biết kết quả biên dịch chương trình? Chương trình có báo lỗi khi thực thi không? Nếu có, cho biết tại sao lỗi.

->biến cục bộ pa chưa được khởi tạo;

4. Nếu trước dòng

printf("Gia tri a: %d \n", *pa);

ta thêm dòng code

*pb = 2;

Cho biết kết quả của lệnh xuất

printf("Gia tri a: %d \n", *pa);

Giải thích tại sao có kết quả xuất như vậy.

->*pa=2; vì trước đó có dòng int *pb=pa;

Tức con trỏ *pb trỏ đến cùng vùng nhớ với pa;

Khi ta thay đổi giá trị *pb tức thay đổi giá trị vùng nhớ mà pa và pb cùng trỏ vào. Nên *pa = 2;

Bài 2:

1. Biên dịch đoạn chương trình trên.

2. Nhập vào một vài mảng số nguyên, nhận xét về kết quả của 2 lệnh xuất sau các lần chạy.

printf("a[0] = %d\n", a[0]);

printf("*a = %d\n", *a);

->sau các lần chạy thì kết quả giống nhau.

3. Giải thích tại sao có thể rút ra kết luận ở câu 2.

->vì a trỏ đến địa chỉ a[0], nên giá trị của a[0]=*a;

4. Sửa lại đoạn chương trình trên để nhập vào một mảng số nguyên và xuất ra tổng các số trong

mảng đó.

```
int n;
int s = 0;
printf("Nhap so luong phan tu: ");
scanf("%d", &n);
int*  a = new int[n];
for (int i = 0; i < n; i++)
{
        printf("Nhap a[%d]: ", i);
        scanf("%d", &a[i]);
}
printf("a[0] = %d\n", a[0]);
printf("*a = %d\n", *a);
for (int i = 0; i < n; i++)
{
        s += a[i];
}
cout << "tong s= " << s << endl;
```

5. Viết chương trình cho phép nhập vào một mảng 2 chiều các số nguyên dùng cấp phát động.

Gợi ý:

Mảng 2 chiều mxn các số nguyên được khai báo như sau

int** b = new int*[m];

trong đó mỗi b[i] (kiểu int*) là một mảng một chiều n số nguyên

b[i] = new int[n];

+>

```
int **b = new int*[m];
for(int i = 0; i<m; i++)
  b[i] = new int[n];
```