Bài thực hành 5

Mục tiêu

- Viết chương trình làm việc với mảng: khai báo, tham chiếu
- Truyền mảng dữ liệu vào hàm
- Tìm kiếm trên mảng
- Sắp xếp trên mảng
- Mảng nhiều chiều

Câu 1. [fix.cpp]

Hãy sửa lỗi cho hàm `sum` tính tổng giá trị các phần tử của mảng kiểu `double` trong chương trình dưới đây.

```
#include <iostream>
using namespace std;

// Ham tinh tong gia tri cac phan tu trong mang
// Tham so a: ten mang
// Tham so n: so phan tu cua mang
double sum(const double a[2], int n);

int main(){
    double m[] = {3, 4, 7};
    cout << "Tong mang m: " << sum(m, sizeof(m)/sizeof(double));
    cin.get();
    return 0;
}

double sum(const double a[2], int n){
    double sum = 0;
    for(int i = 0; i < n; i++) sum += i;
    return sum;
}
```

Câu 2. [product.cpp]

Câu này chưa yêu cầu xử lý số lớn.

a) Viết hàm `product` tính tích của 2 số kiểu int. Bổ sung lệnh dưới đây vào hàm main và quan sát kết quả

```
cout << product(2, 10.2);
```

b) Hãy chú thích hàm `product` ở phần a, rồi viết hàm `product` tính tích 2 số kiểu `double`. Quan sát kết quả của câu lệnh `cout` nói trên.
c) Hãy bỏ chú thích hàm `product` ở phần a rồi dịch và chạy lại chương trình. Thử giải thích vì sao xuất hiện lỗi. Hãy xóa đi một hàm `product` bạn thấy không cần thiết.
d) Viết một hàm `product` sử dụng đối số mặc định có thể tính tích của 2, 3 hoặc 4 số. Hãy xóa bỏ hàm ở phần trước nếu hai hàm này có thể xung đột.
e) Viết một hàm `product` tính tích của nhiều số. Hàm này nên có 2 tham số.

Câu 3. [int_array.cpp]
Lần lượt định nghĩa các hàm dưới đây và viết chương trình để chạy thử chúng.

a) Hàm đọc vào tối đa `maxsize` số nguyên từ bàn phím vào mảng `a`. Dừng đọc nếu người dùng nhập EOF (Control + Z). Số số nguyên đọc vào sẽ được lưu ở `n`.

Chú ý bổ sung lời gọi `cin.clear()` ở cuối hàm để không ảnh hưởng tới các hàm nhập khác trong chương trình. Cho trước khai báo sau:

```
void readArray(int a[], int& n, int maxsize);
```

b) Hàm in `n` số nguyên trong mảng `a` ra màn hình, tách nhau bởi dấu phẩy. Cho trước khai báo sau:

```
void printArray(int a[], int n);
```

c) Hàm `reverse` đảo chiều mảng `a`.

d) Hàm `erase` xóa đi phần tử có chỉ số `index` trong mảng `a` (tức đẩy tất cả các phần tử sau `index` tiến lên 1 vị trí), đồng thời cập nhật tham số `n` lưu số phần tử. Hãy dùng macro assert để kiểm tra điều kiện với `index` và `n`.

```
void erase(int a[], int& n, int index);
```

e) Hàm `max` trả về giá trị phần tử lớn nhất trong mảng `a` nếu mảng này không rỗng (nghĩa là số phần tử khác 0). Hãy dùng macro assert để kiểm tra điều kiện với số phần tử.

f) Hàm `ssort` sắp xếp tăng dần các phần tử trong a theo thuật toán sắp xếp lựa chọn (selection sort) nêu trong bài giảng 5.

g) Hàm `isSorted` kiểm tra xem mảng `a` có sắp xếp tăng dần hay không? Nếu có thì trả về `true`, nếu không thì trả về `false`.

h) Hàm `search` để tìm xem một số nguyên x có trong a hay không? Nếu có thì trả về chỉ số tương ứng, nếu không thì trả về -1.

Câu 4. [card_game.cpp]

Viết chương trình mô phỏng chia bài tiến lên cho 4 người.

Hãy cho biết mỗi người chơi nhận được tay bài gồm 13 lá nào. Có những tứ quý gì?

Dưới đây là gợi ý cấu trúc dữ liệu cho bài toán (Sinh viên có thể chọn một cấu trúc khác nếu có thể lý giải được là sẽ tối ưu hơn.):

52 lá bài được chia lần lượt bằng việc lặp lại lệnh `rand() % 52`.

Chương trình dùng một mảng `bool c[13][4]` để đánh dấu những quân bài đã được chia ra. Quy ước gắn từng lá bài với một ô trong mảng `c` như bảng bên dưới.

- Trước khi chia bài cả 52 ô của mảng `c` đều được gán false.
- Ta mô phỏng mỗi lá bài chia ra bằng một lần gọi `rand() % 52`. Nếu nó sinh ra số x, bạn sẽ đánh dấu ô ở hàng = x / 4, cột = x % 4 là `true` nếu ô này chưa được đánh dấu `true` trước đó. Nếu ô này đã được đánh dấu `true`, lặp lại tăng x lên 1 và kiểm tra tới khi nào gặp ô `false`. Nếu đã tăng đến cuối mảng mà vẫn gặp ô `true` thì quay về đầu mảng.

	cơ ↓	rô ↓	bích ↓	tép ↓
A→	`c[0][0]`	`c[0][1]`	`c[0][2]`	`c[0][3]`
2→	`c[1][0]`	`c[1][1]`	`c[1][2]`	`c[1][3]`
3→	`c[2][0]`	`c[2][1]`	`c[2][2]`	`c[2][3]`
4→	`c[3][0]`	`c[3][1]`	`c[3][2]`	`c[3][3]`
5→	`c[4][0]`	`c[4][1]`	`c[4][2]`	`c[4][3]`
6→	`c[5][0]`	`c[5][1]`	`c[5][2]`	`c[5][3]`
7→	`c[6][0]`	`c[6][1]`	`c[6][2]`	`c[6][3]`
8→	`c[7][0]`	`c[7][1]`	`c[7][2]`	`c[7][3]`
9→	`c[8][0]`	`c[8][1]`	`c[8][2]`	`c[8][3]`
10→	`c[9][0]`	`c[9][1]`	`c[9][2]`	`c[9][3]`
J→	`c[10][0]`	`c[10][1]`	`c[10][2]`	`c[10][3]`
Q→	`c[11][0]`	`c[11][1]`	`c[11][2]`	`c[11][3]`
K→	`c[12][0]`	`c[12][1]`	`c[12][2]`	`c[12][3]`