Kỹ Thuật Lập Trình

Lab 8 – Hàm

1 MỤC TIÊU CỦA BÀI THỰC HÀNH

- Xác định được phần mô tả hàm cho một công việc
- Viết được phần hiện thực hàm để làm một công việc
- Hiểu và sử dụng các kiểu truyền tham số.
- Hiểu và sử dụng mảng, con trỏ với hàm
- Sử dụng được kỹ thuật đệ quy để giải bài toán trong thực tế

2 BÀI TẬP BẮT BUỘC

Câu 1:

Cho chương trình

```
#include <iostream>

using namespace std;

int main(){

	int a = 9;

	cout << “before:” << “a =” << a << endl;

	thay_doi1(a);

	thay_doi2(&a);

	cout << “after:” << “a =” << a << endl;


	return 0;

}

void thay_doi1(int& x){

	x = 100;
```

```
}

void thay_doi2(int* x){

      *x = 100;

}
```

Khi biên dịch chương trình báo lỗi, vì bộ biên dịch không tìm thấy khai báo cho các hàm ở các lời gọi hàm sau: `thay_doi1(a);` và `thay_doi2(a);`

a) Không cần phải copy cả hai hàm trên lên phía trước hàm main(). Có cách nào khác?
b) Nếu bỏ dấu & trong phần định nghĩa hàm "thay_doi1" nói trên, thì kết quả của biến a trước và sau khi gọi hàm `thay_doi1(a);` có khác không vì sao?
c) Thay dòng: `int a = 9;` bằng phát biểu: `int* a = new int(9);` Cũng như thêm dòng: `delete a;` vào trước dòng có lệnh `return`.

Hãy thay 2 lời gọi hàm: `thay_doi1(a);` và `thay_doi2(&a);` bằng cách lời gọi khác cho phù hợp để thay đổi giá trị **mà a chỉ đến** (hiện tại là 9) thay cho chính địa chỉ được giữ trong a.

d) Nếu đổi tên để hai hàm đang có đều là "thay_doi" thay cho "thay_doi1" và "thay_doi2". Đổi luôn lời gọi hàm sang "thay_doi" tương ứng. Biên dịch có lỗi không, vì sao? Khái niệm tiếng Anh nào chỉ ra khả năng các hàm có thể cùng tên, nhưng khác chữ ký hàm?
e) Bổ sung luôn hàm "thay_doi", nhưng truyền thông số bằng trị. Biên dịch chương báo lỗi không? Lỗi ở khâu nào? Vì sao?

Câu 2:

a) Định nghĩa một hàm theo yêu cầu:
   - Tên: sinh viên tự đặt
   - Thông số vào (hàm chỉ đọc, không thay đổi): một mảng của các số kiểu double => sinh viên tự xác định có mấy tham số cho việc này, kiểu truyền là gì? Biết trước rằng hàm này sẽ chỉ đọc mảng truyền vào => cần dùng const để ép buộc bên trong hàm không được thay đổi giá trị mảng.

- Thông số ra (hàm sẽ ghi các giá trị vào các tham số này trước khi trả về): giá trị trung bình, phương sai, và một mảng chứa biểu đồ xác suất - được tính giống như Bài tập lớn 01.
    - Sinh viên nên thiết kế thêm các hàm tính: giá trị trung bình, phương sai, và biểu đồ, hàm này gọi lại các hàm thiết kế thêm này để hoàn thành giá trị.
    - Sinh viên cần xác định kiểu truyền là gì để có thể **trả về giá trị thông qua thông số**.
    - Thông số ra thứ 3 là một mảng. Chương trình này giả thiết rằng: hàm gọi (caller) phải chuẩn bị trước một mảng có đủ phần tử theo yêu cầu và truyền mảng này vào hàm đang nói ở đây. Hàm đang thiết kế chỉ việc ghi kết quả vào mảng đã chuẩn bị sẵn. Ví dụ, như BTL01, thì hàm gọi phải chuẩn bị trước mảng có đủ 10 phần tử (10 bins) và truyền mảng này cùng với số 10 vào hàm đang được thiết kế ở đây.
- Hàm trả về void.

b) Viết chương trình chính (main) làm các việc:

- Sinh dữ liệu ngẫu nhiên có phân phối chuẩn như LAB trước, hay xem: http://www.cplusplus.com/reference/random/normal_distribution/
- Gọi hàm được thiết kế ở câu a) để lấy các giá trị trung bình, phương sai, biểu đồ và in ra màn hình. Kể cả in ra biểu đồ cột như link nói trên.

)

Câu 3:

Giả sử chúng ta có kiểu dữ liệu Matrix (định nghĩa theo sau). Nó là ma trận **vuông** kích thước **N** (dùng define để định nghĩa macro này, ví dụ là 5). Các ma trận có hình dạng tổng quát sẽ được học trong LAB đến.

```
typedef struct{
        double data[N][N];
} Matrix;
```

Yêu cầu:

a) Hãy bổ sung những định nghĩa cho các toán tử: cộng (+), trừ(-), nhân(*) để người lập trình có thể thực hiện được các phép cộng (+), trừ(-), nhân(*) giữa các ma trận vuông với nhau hay giữa các ma trận vuông với các số vô hướng kiểu double. Trong trường hợp giữa ma trận vuông với số vô hướng, sinh viên **tự quy định** ý nghĩa của toán tử này. Ví dụ, nếu cộng một số vào ma trận nghĩa là cộng số đó với từng phần tử của ma trận va trả về ma trận kết quả. Tóm lại, việc bổ sung có thể cho phép người lập trình thực hiện phép toán giữa các ma trận và số vô hướng như ví dụ sau (sinh viên nên thay thế đoạn mã sau bởi đoạn mã của riêng mình để kiểm tra kết quả)

```
Matrix a = {/*khởi động danh sách giá trị*/} // kết hợp hàm sinh số ngẫu nhiên
Matrix b = {/*khởi động danh sách giá trị*/} // kết hợp hàm sinh số ngẫu nhiên
Matrix c, d;

c = a + b + 0.5;
cout << c << endl;
d = 1.5 + a*c;
cout << d << endl;
```

Câu 4: Tương tự như câu 3, bổ sung định nghĩa cho các toán tử +, -, * và / cho các số phức.

Câu 5: Dùng phương pháp đệ quy để thực hiện:

(nếu sinh viên chưa giải hết bằng phương pháp lặp thì cũng giải luôn)

a) Tính tổng và tích các phần tử trong mảng.
b) In các giá trị trong mảng theo thứ tự thuận hay nghịch bằng đệ quy.
c) Đánh giá biểu thức bậc N (nguyên dương), các hệ số của biểu thức cho trong mảng đưa vào hàm.
d) Đổi một số thập phân sang nhị phân bằng đệ quy.
e) Tính tổng các phần tử trong cây nhị phân.

Kiểm tra xem tại tất cả các cây con và cây chính có thoã mãn tính chất: tất cả các giá trị nằm trong cây bên trái nhỏ hơn giá trị tại gốc, và đều nhỏ hơn các giá trị tại cây bên phải.

Câu 6: Thiết kế hàm để giải các câu từ Câu 5 đên Câu 10 trong LAB trước (về con trỏ). Lưu ý, với mỗi câu không chỉ có 01 hàm. Đến LAB này sinh viên có thể sử dụng tất cả các kỹ thuật đã học để làm tốt nhất có thể.