BÀI TẬP CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Lưu hành nội bộ

AUGUST 1, 2018

FIT-UTC

Mục lục

I. Cơ bản về C++

Bài 1. Viết chương trình sử dụng toán tử cin, cout để nhập vào 3 số nguyên, kiểm tra xem nó có thể tạo thành 3 cạnh của tam giác vuông không?

Bài 2. Nhập vào xâu ký tự (kiểu string) bằng cin, chuyển đổi xâu ký tự đó thành xâu ký tự hoa, xâu ký tự thường, in ra màn hình xâu ký tự hoa, xâu ký tự tường.

Ví dụ:

- Nhập vào: Hello World
- In ra màn hình: HELLO WORLD và hello world

Bài 3. Viết hàm tìm giá trị lớn nhất (max) và giá trị nhỏ nhất (min) của 3 số nguyên theo nguyên mẫu dưới đây. Viết hàm main cho phép nhập vào 3 số nguyên tìm giá trị lớn nhất và nhỏ nhất bằng hàm đã xây dựng.

void maxmin(int a, int b, int c, int &max, int &min);

Bài 4. Viết hàm có đối mặc định để có thể vừa tìm *max* hoặc tìm *min* tùy theo đối *ismax* theo nguyên mẫu dưới đây, sau đó viết hàm main nhập vào 3 số nguyên khác nhau tìm số không phải *max, min* bằng cách tính tổng rồi trừ đi *max* và *min.*

int MaxMin(int a, int b, int c, bool ismax = true);

Ví dụ:

- Nhập vào 3 số: 10 5 15
- Số không phải max, min là: (10+5+15) – MaxMin(10,5,15) – MaxMin(10,5,15, false)

Bài 5. Viết hàm mẫu (template) tìm max của hai số, áp dụng tìm max của 2 số nguyên và hai số thực.

Bài 6. Viết hàm mẫu (template) tìm ước chung lớn nhất của 2 số.

Bài 7. Viết hàm mẫu (template) nhập 1 dãy số từ bàn phím, hàm mẫu (template) in một dãy số ra màn hình. Viết hàm main, sử dụng các hàm này nhập vào và in ra màn hình 1 dãy số thực.

II. Lập trình hướng đối tượng trong C++

Bài 1. Xây dựng lớp biểu diễn đối tượng thời gian gồm các thuộc tính: hour (giờ), mimute (phút), second (giây). Các phương thức nhập thời gian (giờ, phút, giây), in thời gian lên

màn hình theo định dạng **hour:minute:second** (*lưu ý: nếu giá trị hour, minute, second < 10 thì thêm số 0 vào trước, ví dụ:* ***01:12:03***), các phương thức thiết lập giờ, phút, giây; các phương thức lấy ra giờ, phút, giây.

Gợi ý bạn xây dựng lớp thời gian có cấu trúc như sau:

```
class Time
{
     private:
          int hour, minute, second;
     public:
          Time();
          input();
          display();
          void setHour(int h);
          void setMinute(int m);
          void setSecond(int s);
          int  getHour();
          int  getMinute();
          int  getSecond();
};
```

Bài 7. Xây dựng lớp điểm trong mặt phẳng gồm 2 thuộc tính x, y, có các phương thức: nhập, in; các phương thức thiết lập giá trị thuộc tính x, thuộc tính y; lấy giá trị thuộc tính x, lấy giá trị thuộc tính y; phương thức tính khoảng cách từ nó tới điểm khác.

Gợi ý bạn xây dựng lớp điểm có cấu trúc như sau:

```
class Point
{
     private:
          float x, y;
     public:
          Point();
          input();
          display();
          void setX(float x1);
          void setY(float y1);
          float getX();
          float getY();
          float distance(Point d);
};
```

Bài 8. Sử dụng lớp Point trong bài 7 xây dựng tam giác (Triangle) có thuộc tính là tọa độ của 3 đỉnh và các phương thức: nhập tọa độ đỉnh từ bàn phím, in tọa độ 3 đỉnh lên màn hình, tính diện tích, tính chu vi.

Viết hàm main, nhập vào 1 tam giác, in lên màn hình diện tích, chu vi của tam giác đó.

Bài 9. Bổ sung **hàm tạo có đối** cho các lớp Time, Point, Triangle ở các bài 6, 7, 8.

Bài 10. Cải tiến các lớp Time, Point, Traingle bằng cách thay các phương thức input, display bằng các toán tử nhập >>, toán tử xuất <<.

Bài 11. Xây dựng lớp biểu diễn các Vector mẫu (template) trong không gian *n* chiều có các phương thức toán tử: +, - hai vector, * tích vô hướng hai vector, - đổi dấu một vector, toán tử >>, <<.

Viết hàm main, nhập vào 2 vector cùng số chiều, in lên màn hình các vector tổng, hiệu của 2 vector đã nhập.

III. Phân tích độ phức tạp

Bài 1. Phân tích độ phức tạp tiêm cận của từng thuật toán dưới đây (Ex1,..,Ex5)

```
Algorithm Ex1(A):
     Input: An array A storing n ≥ 1 integers.
     Output: The sum of the elements in A.
     s←A[0]
     for i←1 to n−1 do
          s←s+A[i]
     return s
Algorithm Ex2(A):
     Input: An array A storing n ≥ 1 integers.
     Output: The sum of the elements at even cells in A.
     s←A[0]
     for i←2 to n−1 by increments of 2 do
          s←s+A[i]
     return s
Algorithm Ex3(A):
     Input: An array A storing n ≥ 1 integers.
     Output: The sum of the prefix sums in A.
     s←0
     for i←0 to n−1 do
          s←s+A[0]
          for j ←1 to i do
          s←s+A[j]
     return s
```

```
Algorithm Ex4(A):
     Input: An array A storing n ≥ 1 integers.
     Output: The sum of the prefix sums in A.
     s←A[0]
     t ←s
     for i←1 to n-1 do
          s←s+A[i]
          t ← t+s
     return t
Algorithm Ex5(A,B):
     Input: Arrays A and B each storing n ≥ 1 integers.
     Output: The number of elements in B equal to the sum of
     prefix sums in A.
     c←0
     for i←0 to n-1 do
          s←0
          for j ←0 to n-1 do
               s←s+A[0]
               for k←1 to j do
                    s←s+A[k]
          if B[i] = s then
               c←c+1
     return c
```

Bài 2. Cài đặt thuật toán **prefixAverages1** và **prefixAverages2** trong bài giảng, thực hiện phân tích thử nghiệm thời gian chạy của từng thuật toán với các bộ dữ liệu đầu vào có kích thước n = 100, 200, …., 2000. Vẽ biểu đồ log-log biểu thị quan hệ giữa thời gian và kích thước dữ liệu đầu vào của 2 thuật toán.

Bài 3. Hãy viết thuật toán giả mã (Pseudo-code) tính giá trị của 1 đa thức $p(x) = \sum_{i=0}^{n} a_i x^i$

a. Thuật toán có độ phức tạp $O(n^2)$
b. Thuật toán có độ phức tạp O(n)

Bài 4. Mô tả thuật toán giả mã (pseudo-code) nhân hai ma trận A, B trong đó A có kích thước n×m, B có kích thước m× p. Lưu ý C = A.B được định nghĩa như sau $C[i][j] = \sum_{k=1}^{m} A[i][k] * B[k][j]$. Phân tích thời gian tiệm cận của thuật toán?

IV. Cấu trúc dữ Vector và List

Bài 1. Cài đặt cấu trúc dữ liệu lớp Vector, lớp bộ lặp của lớp Vector theo lý thuyết đã học

Bài 2. Sử dụng lớp Vector và lớp bộ lặp của lớp vector xây dựng chương trình có các chức năng sau:

1. Chèn 1 phần tử vào vector
2. Xóa 1 phần tử của vector
3. Thay thế một phần tử của vector
4. Lấy giá trị của một phần tử của vector
5. In danh sách các phần tử hiện có trong vector

Các phần tử lưu vào vector là các số thực

Bài 3. Sử dụng lớp Vector và lớp bộ lặp của lớp vector xây dựng chương trình quản lý 1 danh sách các thí sinh có các chức năng sau:

1. Đọc danh sách thí sinh từ 1 file
2. Ghi danh sách thí sinh ra file
3. Bổ sung 1 thí sinh vào danh sách (bổ sung vào cuối)
4. Xóa thí sinh khi biết số báo danh
5. Cập nhật thông tin thí sinh khi biết số báo danh
6. Hiển thông tin của thí sinh khi biết số báo danh
7. In danh sách các thí sinh hiện có trong vector, mỗi thí sinh trên 1 dòng.

Biết mỗi thí sinh gồm các thông tin: Số báo danh, họ tên, năm sinh, giới tính, điểm.

Bài 4. Cài đặt lớp Node, lớp SingleList, lớp bộ lặp của lớp SingleList theo lý thuyết đã học.

Bài 5. Sử dụng lớp SingleList và lớp bộ lặp của lớp SingleList xây dựng chương trình quản lý 1 danh sách các sinh viên có các chức năng sau:

1. Đọc danh sách sinh viên từ 1 file
2. Ghi danh sách sinh viên ra file
3. Bổ sung 1 sinh viên vào danh sách (bổ sung vào cuối)
4. Xóa sinh viên khi biết mã sinh viên

5. Cập nhật thông tin của sinh viên khi biết mã
6. Hiển thông tin của sinh viên khi biết mã
7. In danh sách các sinh viên hiện có lên màn hình, mỗi thí sinh trên 1 dòng.

Biết mỗi sinh viên gồm các thông tin: Mã sinh viên, họ tên, năm sinh, giới tính, quê quán

Bài 6. Cài đặt lớp DblNode, lớp DoubleList, lớp bộ lặp của lớp DoubleList theo lý thuyết đã học.

V. Ngăn xếp và hàng đợi

Bài 1. Cài đặt lớp Stack bằng danh sách liên kết đơn (các phần tử của danh sách được lưu trữ trong 1 danh sách liên kết đơn)

Bài 2. Viết chương trình cho phép nhập vào một biểu thức dạng trung tố bất kỳ. Tính giá trị của biểu thức đó.

Bài 3. Cài đặt lớp Queue bằng danh sách liên kết đơn (các phần tử của danh sách được lưu trữ trong 1 danh sách liên kết đơn)

Bài 4. Cài đặt lớp ứng dụng sử dụng lớp Queue để tổ chức lưu trữ các đối tượng là các số nguyên. Lớp có các chức năng:

1. Thêm vào Queue 1 phần tử
2. Lấy phần tử ra khỏi queue và hiển thị lên màn hình
3. Cho biết số phần tử hiện có của Queue
4. Cho biết Queue rỗng hay đầy

VI. Cây

Bài 1. Cài đặt các lớp biểu diễn cấu trúc cây theo lý thuyết đã học.

Bài 2. Vẽ cây biểu diễn biểu thức sau đây

$$(((a+b)-c) *((45-a)/(x-y)))/(a-x)$$

Bài 3. Vẽ cây theo mô tả sau đây: Nút gốc là X, X có 4 con là A, B, C,D, nút con A có 2 con A_1, A_2, nút con B có 3 nút con M, N, K, nút K có 2 con K_1, K_2, nút D có 3 con D_1, D_2, D_3, nút D_3 có 2 con Z, Y.

VII. Thuật toán tìm sắp xếp

Bài 1. Tìm hiểu lý thuyết cách xây dựng hàm có đối là hàm và hoàn thành cài đặt mã trong ví dụ vào máy.

Lý thuyết

Như chúng ta đã biết, khi xây dựng các hàm thì đối của nó có thể là con trỏ/tham chiếu (cần truyền địa chỉ khi gọi hàm), tham trị (cần truyền giá trị khi gọi hàm). Vậy có khi nào chúng ta cần truyền một hàm vào trong hàm không? Câu trả lời là có, ví dụ bài toán sắp xếp. Giả sử bạn được yêu cầu: hãy xây dựng một hàm sắp xếp một dãy phần tử bất kỳ bằng một thuật toán sắp xếp nào đó chẳng hạn thuật toán sắp xếp Bubble sort. Thuật toán sắp xếp thì bạn đã biết, tuy nhiên để sắp xếp các thuật toán cần phải so sánh các phần tử với nhau. Với yêu cầu của bài toán là sắp xếp dãy các phần tử bất kỳ thì chúng ta không thể sử dụng các phép so sánh thông thường >, < để so sánh được. Ví dụ các phần tử cần sắp là các thí sinh, khi đó với hai thí sinh x, y ta không thể viết **x>y**. Chúng ta không thể so sánh 2 thí sinh với nhau, nhưng chúng ta có thể so sánh trên một thuộc tính nào đó chẳng như họ tên của thí sinh hoặc năm sinh, ….. Khi sắp xếp các thí sinh thì chúng chỉ có thể sắp xếp dựa trên một thuộc tính nào đó. Để xây dựng hàm sắp xếp các phần tử bất kỳ thì chúng ta cần truyền vào cho hàm một hàm so sánh của đối tượng dữ liệu cần sắp. Hàm này thực hiện so sánh trên 1 thuộc tính nào đó của tập dữ liệu mà ta cần sắp xếp.

Khai báo hàm có đối là hàm hay con trỏ hàm

```
Type1 Tên_hàm([DS_đối],Type2 (*Tên_đối_hàm) ([DS kiểu đối))
{
      Code
}
```

Ví dụ: Cài đặt các thuật toán sắp xếp nổi bọt (Bubble sort)

```
template <class T>
void Swap(T &a, T &b)
{
    T tg =a;
    a= b;
    b=tg;
```

```
}
template <class T>
void BubbleSort(T *a, int n, int (*comp)(T, T)){
      int i, j;
      for (i=0;i<n-1;i++)
           for(j=n-1;j>i;j--)
                if(comp(a[j],a[j-1]))
                     Swap(a[j],a[j-1]);

}
```

Trong hàm BubbleSort đối *int (*comp)(T, T)* là đối con trỏ hàm, hàm này sẽ trả lại giá trị 0 hoặc 1 khi so sánh hai giá trị có kiểu T. Hàm swap tráo đổi giá trị của 2 biến.

Áp dụng xây dựng một chương trình, nhập vào một danh sách sinh viên được lữu trữ trong mảng, sắp xếp sinh viên theo họ tên, in danh sách sinh viên lên màn hình. Mỗi sinh viên gồm các thông tin: mã, họ tên, giới tính.

Giải quyết yêu cầu trên ta thực hiện lập trình như sau:

1. Xây dựng lớp sinh viên

```
#ifndef STUDENT_CPP
#include"conio.h"
#include"iostream"
using namespace std;
class Student
{
private:
     int masv;
     char hoten[30];
     char gioi[4];
public:
    int getMaSV(){ return masv;}
    char* getHoten(){ return hoten;}
     friend istream & operator >>(istream &is, Student &s);
     friend ostream & operator <<(ostream &os, Student s);
};
istream & operator >>(istream &is, Student &s)
{
      cout<<"\nNhap ma sv:";
      is>>s.masv;
```

```
      cout<<"Nhap ho va ten:";
      is.ignore(1);
      is.get(s.hoten,30);
      cout<<"Nhap gioi tinh:";
      is.ignore(1);
      is.get(s.gioi,4);
      return is;
}
ostream & operator <<(ostream &os, Student s)
{
       os<<s.masv<<"\t"<<s.hoten<<"\t" <<s.gioi;
      return os;
}
#endif
```

2. Xây dựng hàm nhập, in các phần tử của mảng

```
#ifndef ARRAY_H
#define ARRAY_H 0
#include"iostream"
using namespace std;
template <class T>
void InputArr(T *a, int n, char *c){
     for(int i=0;i<n;i++){
       cout<<c<<"["<<i<<"]=";
       cin>>a[i];
     }
}
template <class T>
void PrintArr(T *a, int n, int xuongdong){
 //xuongdong=1 thi in ra theo cot, nguoc lai in ra theo hang
      for(int i=0;i<n;i++)
         if (xuongdong)
            cout<<a[i]<<"\n";
         else
            cout<<a[i]<<" ";
}
```

```
#endif
```

3. Xây dựng hàm so sánh, hàm main

```
#include"conio.h"
#include"stdio.h"
#include"string.h"
#include"iostream"
#include"sortnn.cpp"
#include"array.cpp"
#include"student.cpp"
using namespace std;
int compare_Name(Student x, Student y){
   if (strcmp(x.getHoten(),y.getHoten())<0)
      return 1;
   else
      return 0;
}
int main(){
     Student *a;
     int n;
     system("cls");
     cout<<"Nhap so sinh vien n=";
     cin>>n;
     a =  new Student[n];
     InputArr(a, n, "Nhap SV thu ");
     system("cls");
     cout<<"Danh sach sinh vien:\n";
     BubbleSort(a,n,compare_Name);
     cout<<"\nDanh sach sinh vien sau khi sap xep:\n";
     PrintArr(a,n,1);
     getch();
     return 0;
}
```

Bài 2. Cài đặt các hàm sắp xếp một mảng các phần tử bất kỳ bằng các thuật toán sắp xếp chọn (Selecton Sort), thuật toán sắp xếp chèn (Insertion Sort), sắp xếp nhanh (Quick Sort), sắp xếp trộn (Merge Sort), sắp xếp vun đống (Heap Sort).

Bài 3. Xác định số phép toán cần phải thực hiện tối đa của mỗi thuật toán sắp xếp trong bài 2.

Bài 4. Lần lượt áp dụng các thuật toán cài đặt trong bài 2 để xắp xếp 1 danh sách các thí sinh theo thứ tự tổng điểm, biết mỗi thí sinh có các thông tin: Số báo danh, họ tên, điểm toán, điểm lý, điểm hóa.

Bài 5. Bổ sung phương thức **Sort** vào cấu trúc **Vector** trong phần III, để thực hiện sắp xếp các phần tử có trong Vector bằng thuật toán sắp xếp heap sort.

Bài 6. Bổ sung phương thức **Sort** vào cấu trúc **SingleList** trong phần III, để thực hiện sắp xếp các phần tử có trong SingleList bằng thuật toán sắp xếp chọn.

VIII. Thuật toán tìm kiếm

Bài 1. Bổ sung phương thức tìm kiếm tuần tự vào cấu trúc danh sách liên kết đơn, liên kết kép

Bài 2. Bổ sung phương thức tìm kiếm nhị phân vào cấu trúc Vector nếu các phần tử của vector đã được sắp.

Bài 3. Cài đặt thuật toán tìm kiếm nhị phân trên mảng có kiểu dữ liệu bất kỳ, mảng đã được sắp theo hàm so sánh.

Bài 4. Xây dựng lớp cây tìm kiếm nhị phân, để lưu trữ các phần tử bất kỳ, việc so sánh giữa 2 phần tử được thực hiện bằng hàm so sánh.

Bài 5. Sử dụng lớp cây tìm kiếm nhị phân xây dựng một chương trình tra cứu từ điển có các chức năng sau:

1. Đọc dữ liệu từ điển nạp vào cây từ tệp
2. Bổ sung từ mới vào cây
3. Xóa bỏ một từ khỏi cây
4. Tìm kiếm từ
5. Lưu cây vào tệp