Thực Hành 09

1. Bài tập 1

Định nghĩa kiểu dữ liệu mới **ToaDo** sử dụng cấu trúc **struct** để biểu diễn tọa độ của một điểm trên mặt phẳng 2 chiều, với hoành độ và tung độ kiểu số thực (**double**).

```
struct ToaDo { double X, Y; };
```

Viết hàm thực hiện các công việc sau:

1. Nhập dữ liệu cho hoành độ và tung độ của 1 điểm kiểu **ToaDo**
   ```
   void nhapToaDo(ToaDo& td);
   ```
2. In ra hoành độ và tung độ theo định dạng "**(X,Y)**"
   ```
   void inToaDo(const ToaDo& td);
   ```
3. Kiểm tra 1 điểm kiểu **ToaDo** có phải là gốc tọa độ hay không
   ```
   bool laGocToaDo(const ToaDo& td);
   ```
4. Kiểm tra 1 điểm kiểu **ToaDo** có nằm trên trục hoành hoặc trục tung hay không
   ```
   bool laTrenTungHoanh(const ToaDo& td);
   ```
5. Kiểm tra 1 điểm kiểu **ToaDo** có nằm trên đường chéo chính hoặc phụ hay không
   ```
   bool laTrenDuongCheo(const ToaDo& td);
   ```
   Cách 1: sử dụng kết hợp các biểu thức so sánh
   Cách 2: sử dụng hàm **fabs**
6. Kiểm tra 2 điểm kiểu **ToaDo** có đối xứng với nhau qua điểm gốc tọa độ hay không
   ```
   bool laDoiXung(const ToaDo& td1, const ToaDo& td2);
   ```
7. Xác định 1 điểm kiểu **ToaDo** nằm trên góc phần tư nào của mặt phẳng tọa độ
   ```
   int trenGocPhanTu(const ToaDo& td);
   ```
   Gợi ý: trả về một trong bốn hằng số **GOC_MOT**, **GOC_HAI**, **GOC_BA**, **GOC_BON**,
8. Tính khoảng cách giữa 2 điểm kiểu **ToaDo**
   ```
   double tinhKhoangCach(const ToaDo& td1,
                         const ToaDo& td2);
   ```

Viết chương trình kiểm tra các hàm trên đã cài đặt đúng hay chưa.
Nhập vào một mảng các điểm kiểu **ToaDo**.
Đếm số lượng điểm có một trong những đặc điểm như sau: là gốc tọa độ, trên trục tung hoặc trục hoành, trên đường chéo chính hoặc đường chéo phụ, ở góc phần tư thứ nhất, ở góc phần tư thứ hai, ở góc phần tư thứ ba, ở góc phần tư thứ tư.
Sau đó in ra số lượng các điểm thuộc một trong 7 đặc điểm trên.
Đếm số lượng cặp điểm đối xứng với nhau qua gốc tọa độ.
Sắp xếp các điểm kiểu **ToaDo** trong mảng theo thứ tự khoảng cách đến gốc tọa độ.
Lưu ý: đối với mảng đếm, sử dụng hằng số làm chỉ số của mảng (xem slide 3–4, bg09).

2. Bài tập 2

Định nghĩa kiểu dữ liệu mới **Date** sử dụng cấu trúc **struct** để biểu diễn ngày dương lịch bao gồm ngày, tháng, năm kiểu số nguyên (**int**).

```
struct Date { int day, month, year; };
```

Viết hàm thực hiện các công việc sau:

1. Nhập dữ liệu cho ngày, tháng, năm của 1 ngày dương lịch kiểu **Date**
   ```
   void nhapDate(Date& d);
   ```
2. In ra ngày, tháng, năm theo định dạng "**dd/mm/yy**", ví dụ 09/10/18
   ```
   void inDate(const Date& d);
   ```
3. Kiểm tra 1 ngày dương lịch kiểu **Date** có hợp lệ không, ví dụ 30/02/2018 không
   ```
   bool laDateHopLe(const Date& td);
   ```
   Gợi ý: cần viết hàm tính số ngày trong một tháng. Hàm này trả về số ngày của một tháng và năm tương ứng:
   ```
   int tinhSoNgayCuaThangNam(int month, int year);
   ```
4. Quy chuẩn 1 ngày dương lịch kiểu **Date**, ngày này luôn là ngày có trên niên lịch, nếu không, đưa về ngày trên niên lịch gần nhất, ***ví dụ*** 35/04/18 đưa về 30/04/18, 30/13/18 đưa về 30/12/18, -03/04/18 đưa về 01/04/18, 03/-04/18 đưa về 03/01/18.
   ```
   void quyChuanDate(const Date& d, Date& d_hop_le);
   ```
   Sử dụng hàm này khi nhập dữ liệu. Nếu nhập sai thông tin ngày dương lịch, đưa ra gợi ý ngày hợp lệ (**d_hop_le**). Nếu người dùng đồng ý, thì đổi sang ngày hợp lệ, nếu không yêu cầu người dùng nhập lại.
5. Tính ngày dương lịch hôm qua kiểu **Date**, ngày dương lịch ngày mai kiểu **Date**
   ```
   void tinhDateHomQua(const Date& hom_nay, Date& hom_qua);
   void tinhDateNgayMai(const Date& hom_nay, Date& ngay_mai);
   ```
6. Cộng ngày dương lịch kiểu **Date** thêm X ngày, trừ ngày dương lịch đi X ngày
   ```
   void congDate(const Date& d1, int ngay, Date& d2);
   ```
   Ví dụ: **d1** là 26/10/18, **ngay** là **1**, cộng thì tính ra **d2** là 27/10/18
   ```
   void truDate(const Date& d1, int ngay, Date& d2);
   ```
   Ví dụ: **d1** là 26/10/18, **ngay** là **1**, trừ thì tính ra **d2** là 25/10/18
7. Kiểm tra bằng, nhỏ hơn, lớn hơn giữa 2 ngày dương lịch kiểu **Date**
   ```
   bool ktBangDate(const Date& d1, const Date& d2);
   bool ktNhoHonDate(const Date& d1, const Date& d2);
   bool ktLonHonDate(const Date& d1, const Date& d2);
   ```
8. Tính khoảng cách giữa 2 ngày dương lịch kiểu **Date**
   ```
   int tinhKhoangCach(const Date & d1, const Date & d2);
   ```

Viết chương trình kiểm tra các hàm trên đã cài đặt đúng hay chưa.

Nhập vào một mảng các ngày dương lịch kiểu **Date**.

Đếm số lượng ngày dương lịch là đầu tháng, là cuối tháng.

Sắp xếp các ngày dương lịch kiểu **Date** trong mảng vừa nhập theo thứ tự tăng dần.

Đếm số lượng cặp ngày liền kề trong mảng, ví dụ cặp ngày liền kề (30/04/18, 01/05/18)

Đếm số lượng ngày dương lịch là thứ 2, thứ 3, thứ 4, thứ 5, thứ 6, thứ 7, chủ nhật.

3. Bài tập 3

Định nghĩa kiểu dữ liệu mới **DaThuc** sử dụng cấu trúc **struct** để biểu diễn một đa thức dạng $a_0 + a_1x^1 + \cdots + a_nx^n$, trong đó:

- n là bậc của đa thức
- a_0 đến a_n là các hệ số của đa thức

```
struct DaThuc { int bac; double mang_he_so[100]; };
```

Viết hàm thực hiện các công việc sau:

1. Nhập dữ liệu cho 1 đa thức
   ```
   void nhapDaThuc(DaThuc& dt);
   ```
2. In ra dữ liệu của 1 đa thức
   ```
   void inDaThuc(const DaThuc& dt);
   ```
3. Nếu hệ số bậc cao nhất của đa thức là 0 ($a_n = 0$) thì chuyển đa thức về bậc $n - 1$
   ```
   bool quyChuanDaThuc(const DaThuc& dt);
   ```
4. Tính tổng 2 đa thức
   ```
   void tong(const DaThuc& dt1, const DaThuc& dt2, DaThuc& dtTong);
   ```
5. Tính hiệu 2 đa thức
   ```
   void hieu(const DaThuc& dt1, const DaThuc& dt2, DaThuc & dtHieu);
   ```
6. Tính tích 2 đa thức
   ```
   void tich(const DaThuc& dt1, const DaThuc& dt2, DaThuc& dtTich);
   ```
7. Tính giá trị của 1 đa thức
   ```
   double tinhGiaTriDaThuc(const DaThuc& dt, double X);
   ```

Viết chương trình kiểm tra các hàm trên đã cài đặt đúng hay chưa.

4. Bài tập 4 *

Đối với bài tập này, định nghĩa cấu trúc **struct SinhVien**, và chữ ký của các hàm chỉ là gợi ý. Bạn có thể phải sửa đổi để thực hiện các thao tác đơn giản và hiệu quả hơn.

```
struct SinhVien {
    int         mssv;
    string      ho_ten;
    Date        ngay_sinh;
    DiemMonHoc int1008;
};
```

Định nghĩa kiểu dữ liệu mới **SinhVien** sử dụng cấu trúc **struct** để biểu diễn thông tin một sinh viên, bao gồm mã số sinh viên, họ tên, ngày tháng năm sinh, điểm môn học INT1008 – Nhập Môn Lập Trình.
Trong đó, điểm môn học Tin Học Cơ Sở 4 là kiểu dữ liệu mới **DiemMonHoc**, bao gồm điểm thành phần, điểm thi cuối kì, điểm tổng kết môn học và hệ số của từng đầu điểm.
Sử dụng cấu trúc **struct** định nghĩa kiểu dữ liệu mới **DiemMonHoc**, và các hàm phụ trợ cần thiết cho kiểu mới này, tương tự các bài Bài tập 1 – 3.
Viết hàm thực hiện các công việc sau:

1. Nhập dữ liệu cho 1 sinh viên
   void nhapSinhVien(SinhVien& sv);
2. In ra dữ liệu của 1 sinh viên
   void inSinhVien(const SinhVien& sv);
3. Tính điểm tổng kết môn học của 1 sinh viên
   void dtkSinhVien(SinhVien& sv);
4. Xếp hạng điểm tổng kết môn học của 1 sinh viên (A, B, C, D, E, F)
   void xhSinhVien(SinhVien& sv);

Viết chương trình kiểm tra các hàm trên đã cài đặt đúng hay chưa.
Nhập vào một mảng các thông tin sinh viên kiểu **SinhVien**.
Tính điểm tổng kết môn học và xếp hạng điểm tổng kết môn học cho mỗi sinh viên.
Tính điểm trung bình môn học của lớp học.
Sắp xếp mảng sinh viên theo thứ tự điểm tổng kết từ cao đến thấp.
Đếm số lượng mỗi loại xếp hạng (A, B, C, D, E, F) của lớp học.
Liệt kê mã số sinh viên của các sinh viên trượt môn học INT1006.
*** Liệt kê mã số sinh viên của các sinh viên có thể được phép làm thêm bài kiểm tra phụ để được nâng xếp hạng điểm tổng kết !!!

5. **Bài tập 5 ***

Sử dụng kiểu dữ liệu **`DaThuc`** trong Bài Tập 3 để biểu diễn hệ số của nhị thức Newton.
Hệ số của nhị thức Newton có thể xây dựng dựa trên tam giác Pascal (như đã thảo luận trong Thực hành 06 – Bài tập 04).
Sử dụng một mảng kiểu dữ liệu **`DaThuc`** để biểu diễn tam giác Pascal.
Viết hàm xây dựng tam giác Pascal dựa trên dữ liệu kiểu **`DaThuc`**:

- Cách 1: sử dụng phương pháp xây dựng tam giác Pascal như đã thảo luận
- Cách 2: sử dụng hàm tính tổng 2 đa thức (như đã xây dựng trong Bài tập 3). Bạn có thể phải viết thêm hàm gán đa thức để sao chép dữ liệu đa thức, do việc sử dụng trực tiếp phép gán có thể gây lỗi:

  ```
  dt1 = dt2;
  ```

  tương đương

  ```
  dt1.bac = dt2.bac;
  dt1.mang_he_so = dt2.mang_he_so;
  ```

Tính giá trị của $(a + b)^n$, với a, b, n là dữ liệu nhập vào chương trình.
Có cách nào để xác định giá trị $a^i b^{n-i}$ một cách hiệu quả không?

6. **Bài tập 6 ***

Định nghĩa kiểu dữ liệu mới **`TamGiac`** sử dụng cấu trúc **`struct`** để biểu diễn một tam giác có 3 đỉnh là các điểm trên mặt phẳng tọa độ 2 chiều
Hãy xây dựng nhưng hàm phụ trợ cần thiết cho kiểu dữ liệu mới này.
Viết hàm kiểm tra gốc tọa độ nằm trong tam giác, nằm ngoài tam giác, hay năm trên cạnh của tam giác có kiểu dữ liệu như trên.
Gợi ý: viết hàm kiểm tra hai đoạn thẳng trên mặt phẳng tọa độ 2 chiều có giao nhau hay không (có thể cần xây dựng kiểu dữ liệu mới **`DoanThang`**). Sau đó sử dụng hàm này để kiểm tra vị trí của gốc tọa so với vị trí của tam giác.