

TRƯỜNG ĐẠI HỌC ĐÀ LẠT
KHOA CÔNG NGHỆ THÔNG TIN

岱電院



**GIÁO TRÌNH THỰC HÀNH
NGUYÊN LÝ LẬP TRÌNH CẤU TRÚC**

*Nguyễn Văn Phúc - Trần Tuấn Minh - Nguyễn Thị Lương
Đà Lạt 2021*

LỜI NÓI ĐẦU

Hệ thống bài lab đi kèm theo giáo trình "Nguyên lý lập trình cấu trúc" giúp sinh viên :

- Cài đặt cấu trúc dữ liệu cơ bản và các thuật toán tương ứng
- Cài đặt các thuật toán đệ quy đơn giản
- Tổ chức các chương trình có cấu trúc theo thư viện hàm và có/không có hệ thống tùy chọn menu.
- Rèn luyện kỹ năng xây dựng chương trình theo cách hoàn thiện từng bước chương trình.
- Tổ chức các project theo Win32 Console Application trong môi trường MS Visual Studio 2015,2013,..

Hệ thống bài lab bao gồm 11 bài :

Lab	Tên lab	Số tiết
1	Làm quen với C++ và MS Visual studio 2015,2013, . . .	4
2	Định nghĩa hàm và gọi hàm	4
3	Các câu lệnh điều khiển	6
4	Tổ chức thư viện hàm và menu	6
5	Các kỹ thuật xử lý mảng 1 chiều	6
6	Các kỹ thuật xử lý mảng 2 chiều (Ma trận)	6
7	Các kỹ thuật xử lý xâu ký tự (chuỗi)	6
8	Kiểu cấu trúc	6
9	Biến động và con trỏ	6
10	Thuật toán đệ quy	4
11	Lập trình xử lý tập tin văn bản	4
ÔN TẬP		2

Dự kiến lịch thực tập như sau:

Buổi (15)	B1	B2	B3	B4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	Số tiết
Lab (11)	Lab1	Lab2	Lab3	Lab3	Lab4	Lab5	Lab5	Lab6	Lab7	Lab7	Lab8	Lab9	Lab9	Lab10	Lab11	2 t
	Lab1	Lab2	Lab3	Lab4	Lab4	Lab5	Lab6	Lab6	Lab7	Lab8	Lab8	Lab9	Lab10	Lab11	Ôn tập	2 t

Vì trình độ người biên soạn có hạn nên tập giáo trình không tránh khỏi nhiều khiếm khuyết, Chúng tôi rất mong sự góp ý của các bạn đồng nghiệp và sinh viên.

Cuối cùng, Chúng tôi cảm ơn sự động viên, giúp đỡ nhiệt thành của các bạn đồng nghiệp trong khoa Công nghệ thông tin để tập giáo trình này được hoàn thành.

Đà Lạt tháng 9/2021

Các tác giả

LAB 1: LÀM QUEN VỚI C++ VÀ VISUAL STUDIO

THỜI LƯỢNG: 4 TIẾT

A. Mục tiêu

- Giúp sinh viên làm quen với môi trường phát triển của Microsoft Visual Studio 2015(2013,2010,..)
- Sau khi hoàn thành bài thực hành này, sinh viên:
 - Tạo được dự án dạng Win32 Console Application.
 - Nắm vững các thao tác nhập – xuất dữ liệu
 - Hiểu rõ cách khai báo và sử dụng biến, hằng : ký tự, nguyên, thực.
 - Biết sử dụng các toán tử.
 - Sử dụng được các hàm toán học xây dựng sẵn (trong <math.h>)

B. Yêu cầu

- Sinh viên tự đọc các mục 1, 2 phần C (Hướng dẫn thực hành) lab 1.
- Sinh viên tự làm bài (làm ngoài giờ thực tập):
 - Khối lượng và nội dung : Các mục từ 3 đến 6 trong phần C (Hướng dẫn thực hành) lab 1.
 - Yêu cầu lưu trữ và nạp bài:
 - Tạo thư mục MaSV_Lab01_Tuhoc
 - Mỗi bài (từ 3 đến 6) tạo project riêng, lưu trữ trong thư mục trên.
 - Mỗi project, xóa thư mục debug.
 - Nén thư mục MaSV_Lab01_Tuhoc.
 - Nạp bài trên hệ thống edmodo.
 - Thời gian nạp : Xem thông báo của giáo viên .

Buổi thực tập thứ nhất (4 tiết)

- 2 tiết đầu :
 - Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung :
 - Các mục từ 7 đến 11 trong phần C (Hướng dẫn thực hành) lab 1.
 - Tạo tập tin word, đặt tên LoiChuongTrinh.docx để ghi lại các lỗi xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab01 chứa trong ổ đĩa qui định.
 - ✓ Mỗi bài (từ 7 đến 11) tạo project riêng, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
- 2 tiết cuối :
 - Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Các bài trong phần D (bắt buộc) lab 1.
 - Yêu cầu lưu trữ và nạp bài :
 - ✓ Sử dụng tiếp thư mục MaSV_Lab01

- ✓ Tạo project riêng cho mỗi bài trong phần D, lưu trữ trong thư mục trên.
- ✓ Mỗi project, xóa thư mục debug.
- ✓ Bổ sung nội dung vào tập tin LoiChuongTrinh.docx
- ✓ Thu bài : Xem thông báo của giáo viên .

C. Hướng dẫn thực hành

1. Thiết lập một số tùy chọn trong Visual Studio

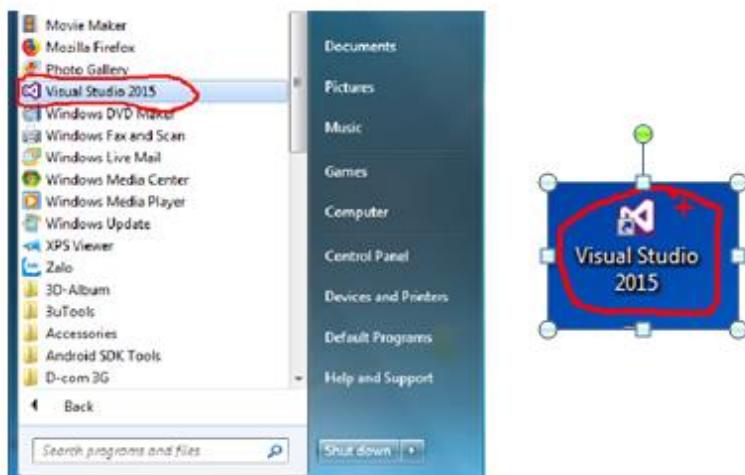
Trước khi sử dụng Visual Studio để viết chương trình, ta cần thiết lập một số tùy chọn. Việc này chỉ cần **thực hiện 1 lần, không cần phải thực hiện lại cho từng dự án.**

(Thiết lập mặc định hiển thị tên của toàn bộ dự án, hiển thị số thứ tự dòng lệnh, kích thước tab)

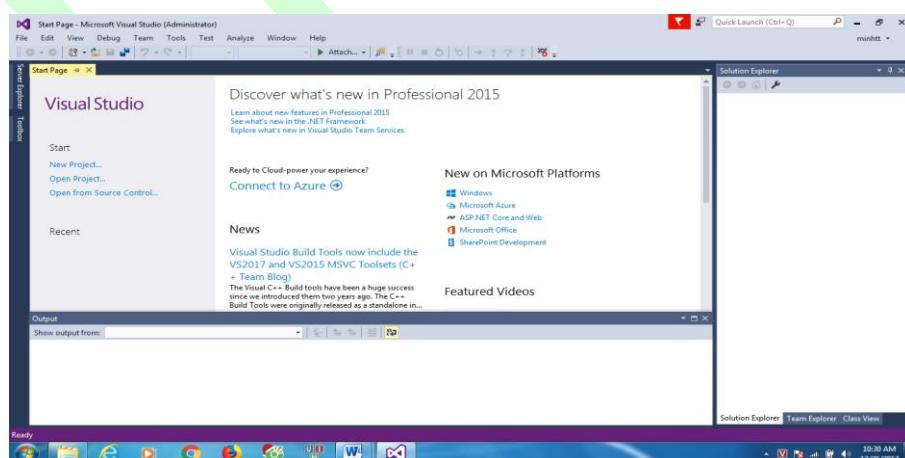
Bước 1.

- Cách 1: Mở Visual Studio bằng cách vào menu **Start > All Programs > Microsoft Visual Studio 2015 (2013,2010,..)** rồi chọn **Microsoft Visual Studio 2015 (2013,2010,..)** :
- Cách 2: Chọn icon **Visual Studio 2015 (2013,2010,..)** trên destop

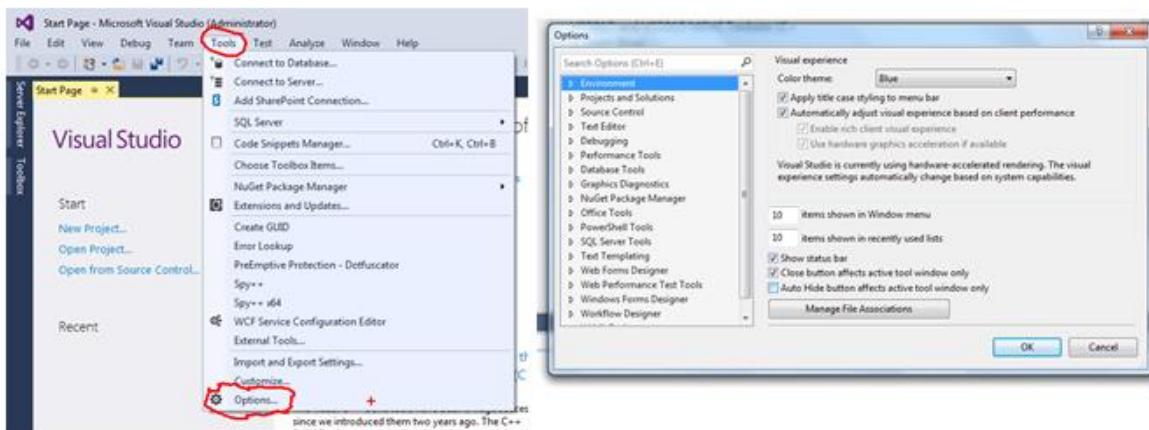
Hình sau:



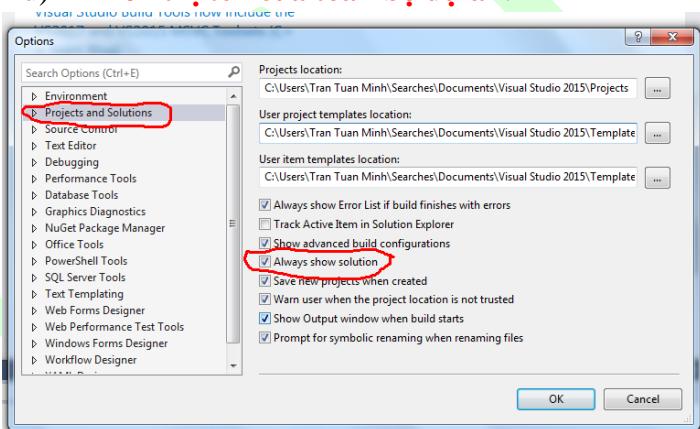
Sau khi mở **Visual Studio 2015**, ta có:



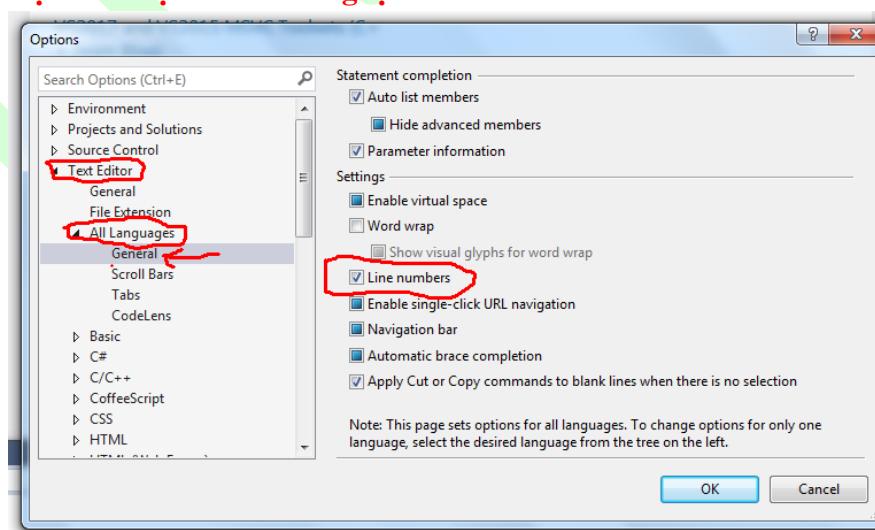
Bước 2. Chọn menu Tools > Options. Trên màn hình xuất hiện hộp thoại sau:



Bước 3. Trong cửa sổ Options, chọn **Projects and Solutions**, đánh dấu vào mục **Always show solution** (nếu chưa đánh dấu) để **hiển thị tên của toàn bộ dự án**.

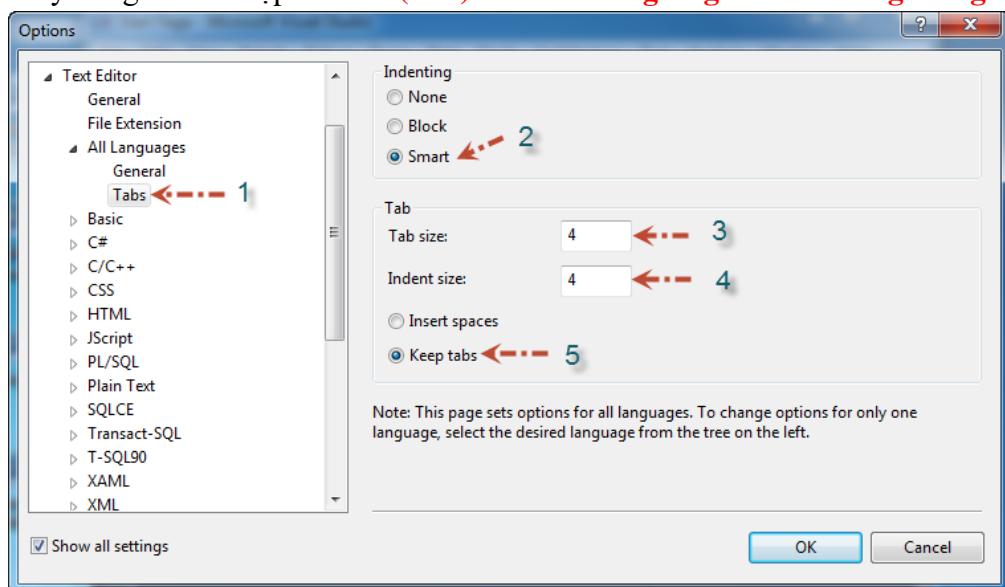


Bước 4. Tiếp theo, chọn mục **Text Editor > All Languages**, đánh dấu chọn mục **Line numbers** (nếu chưa) để **hiển thị số thứ tự của các dòng lệnh**.



Bước 5. Cũng trong cửa sổ trên, Chọn mục Tabs, sau đó, chọn và nhập các giá trị như hình dưới đây.

Bước này dùng để thiết lập **1 bước (dấu) Tab sẽ tương ứng với 4 khoảng trắng**.



Bước 6. Nhấn nút OK để kết thúc việc thiết lập các tùy chọn.

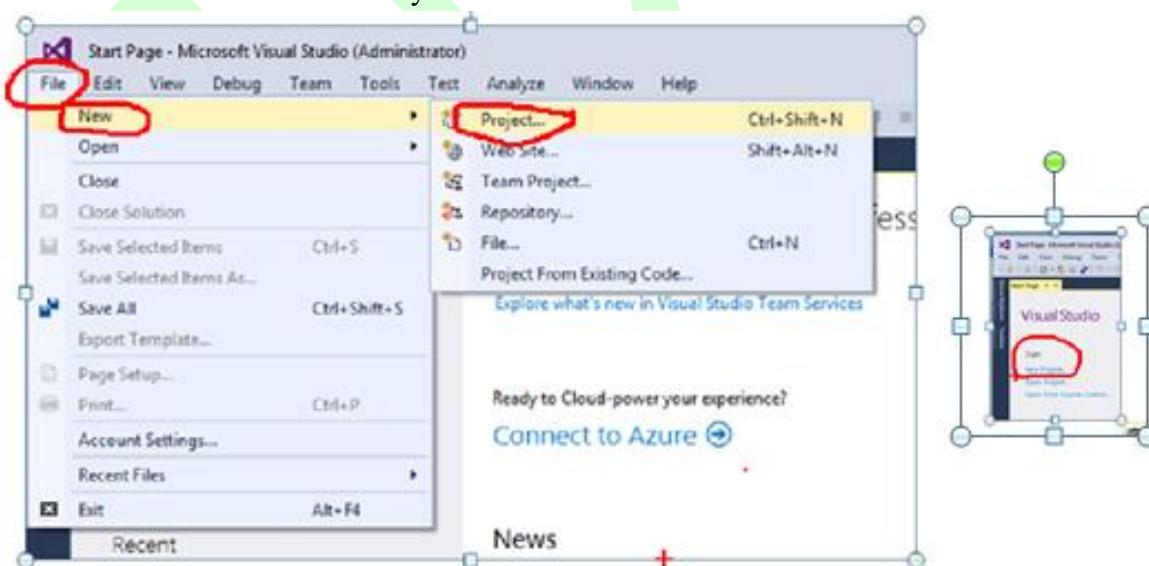
2. Tạo một dự án Win32 Console Application

Phần này hướng dẫn cách tạo một dự án sử dụng ngôn ngữ C++ theo dạng Win32 Console Application.

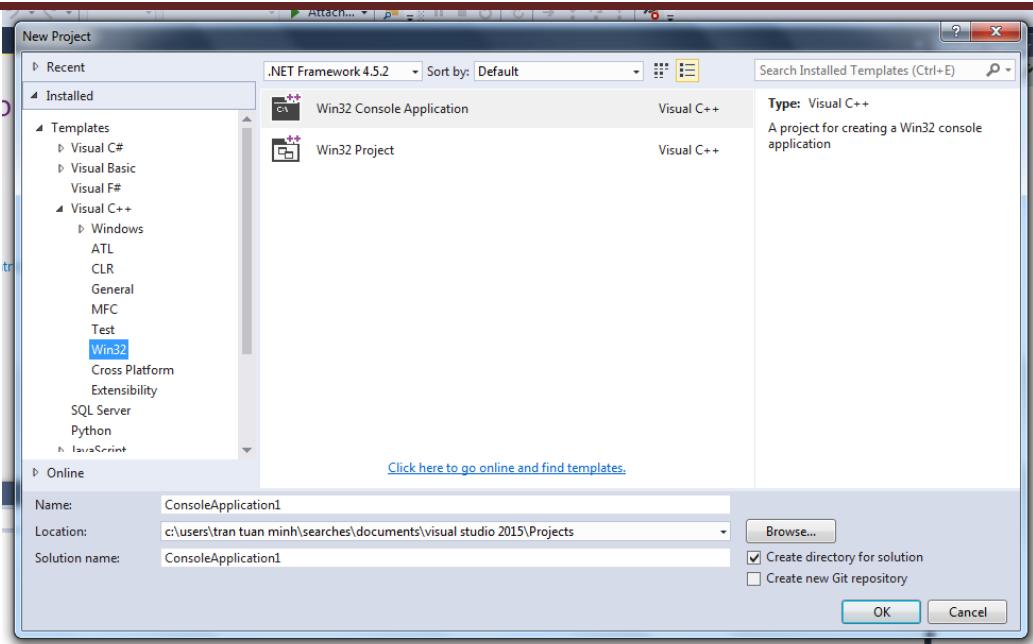
Bước 1. Mở chương trình Visual Studio như đã hướng dẫn trong phần 1, bước 1.

Bước 2. Trong cửa sổ Visual Studio, chọn menu **File > New->Project**, hoặc tổ hợp phím **Ctrl + Shift + N**, hoặc chọn **New Project (Start)** để mở cửa sổ tạo dự án.

Thao tác theo hình vẽ dưới đây :



Kết quả việc mở cửa sổ dự án ta có cửa sổ **New Project** như sau :



Bước 3. Trong cửa sổ mới, chọn **Kiểu** dự án, đặt **Tên** dự án, chọn **Nơi lưu trữ** dự án.

- Chọn kiểu dự án : Win32 Console Application

(Thao tác : Templates -> Visual C++ -> Win32 -> chọn Win32 Console Application)

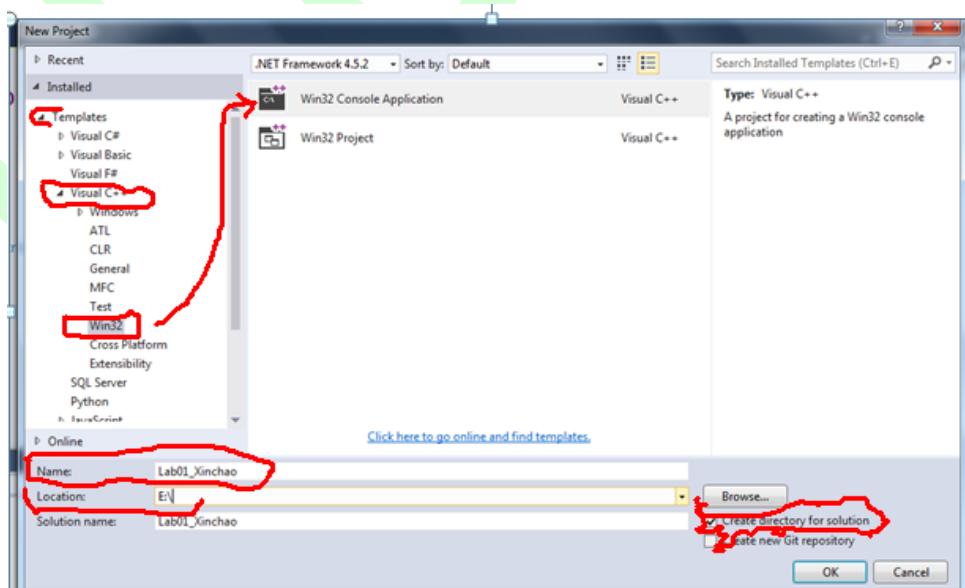
- Đặt tên dự án (mặc định là ConsoleApplicationX), đổi tên lại là : Lab01_Xinchao

(Thao tác : Trong mục **Name** : xóa tên mặc định, đặt lại tên mới, và đánh dấu ô Create directory fo solution).

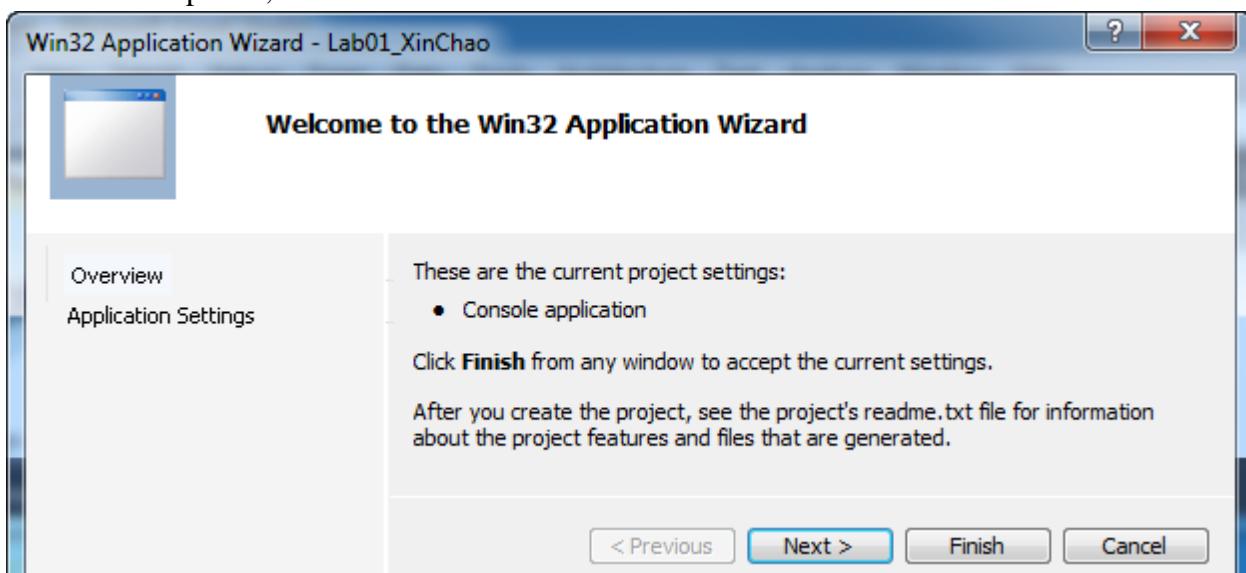
- Chọn nơi lưu trữ dự án (mặc định là thư mục : c:\users\tran tuan inh\searches\documents\visual studio 2015\Projects), đổi lại theo yêu cầu, chẳng hạn là thư mục gốc ổ đĩa E :

(Thao tác : trong mục **Location** sửa thư mục lưu trữ theo yêu cầu, gõ trực tiếp hay dùng browse).

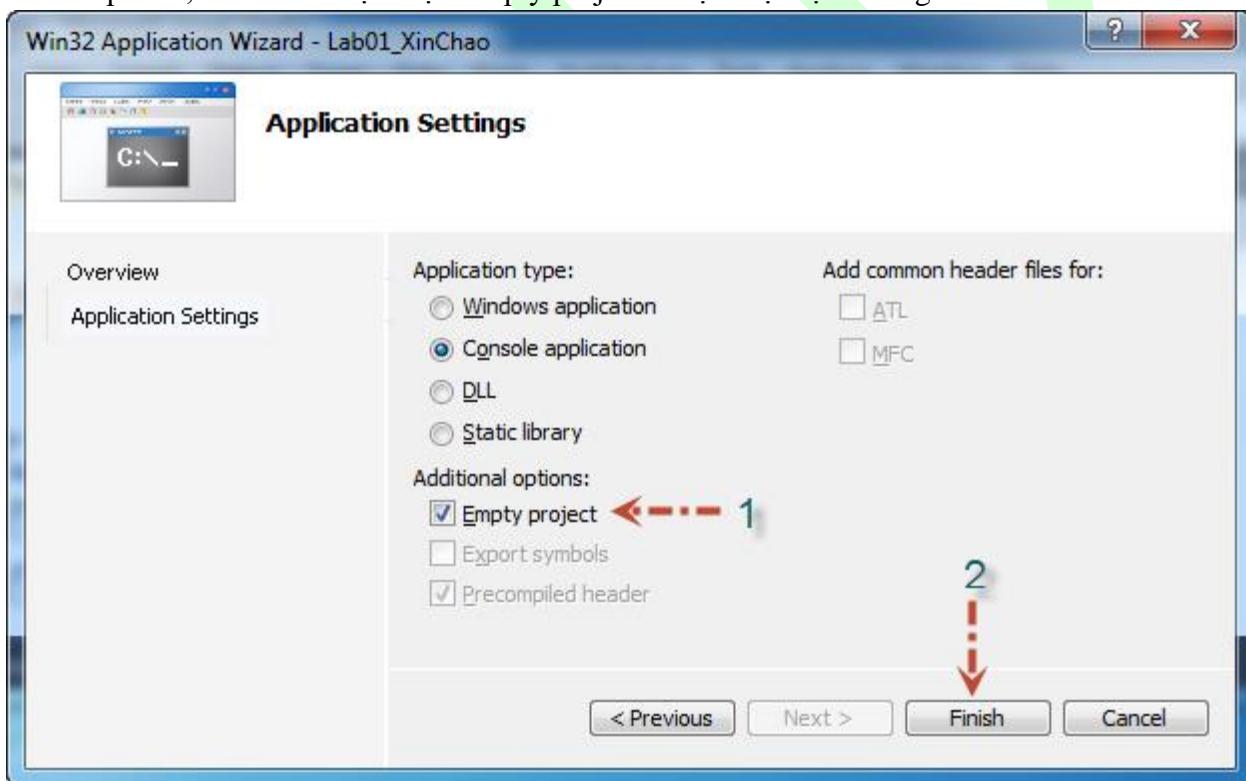
Thao tác dựa vào hình vẽ sau :



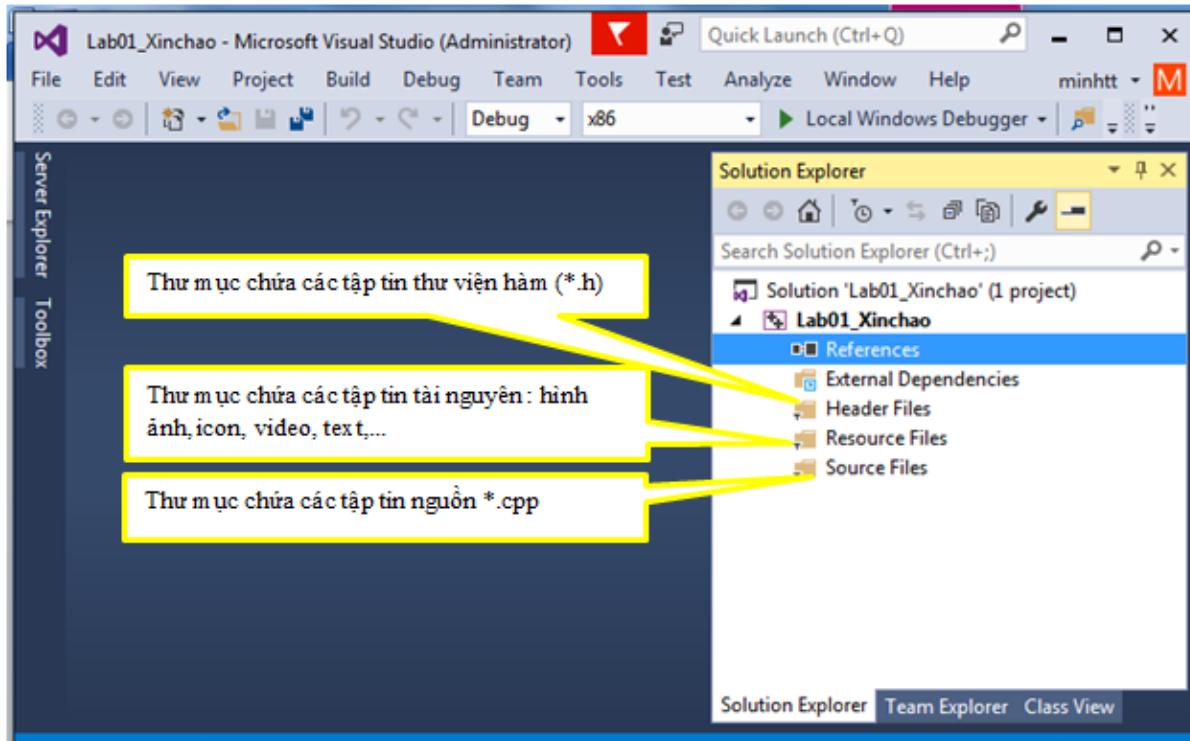
Bước 4. Nhấn OK (đã tạo được một dự án Win32 Console Application, có tên là Lab01_Xinchao). Trong cửa sổ tiếp theo, nhấn nút Next.



Bước 5. Tiếp theo, đánh dấu chọn mục Empty project để tạo một dự án rỗng. Nhấn nút Finish.



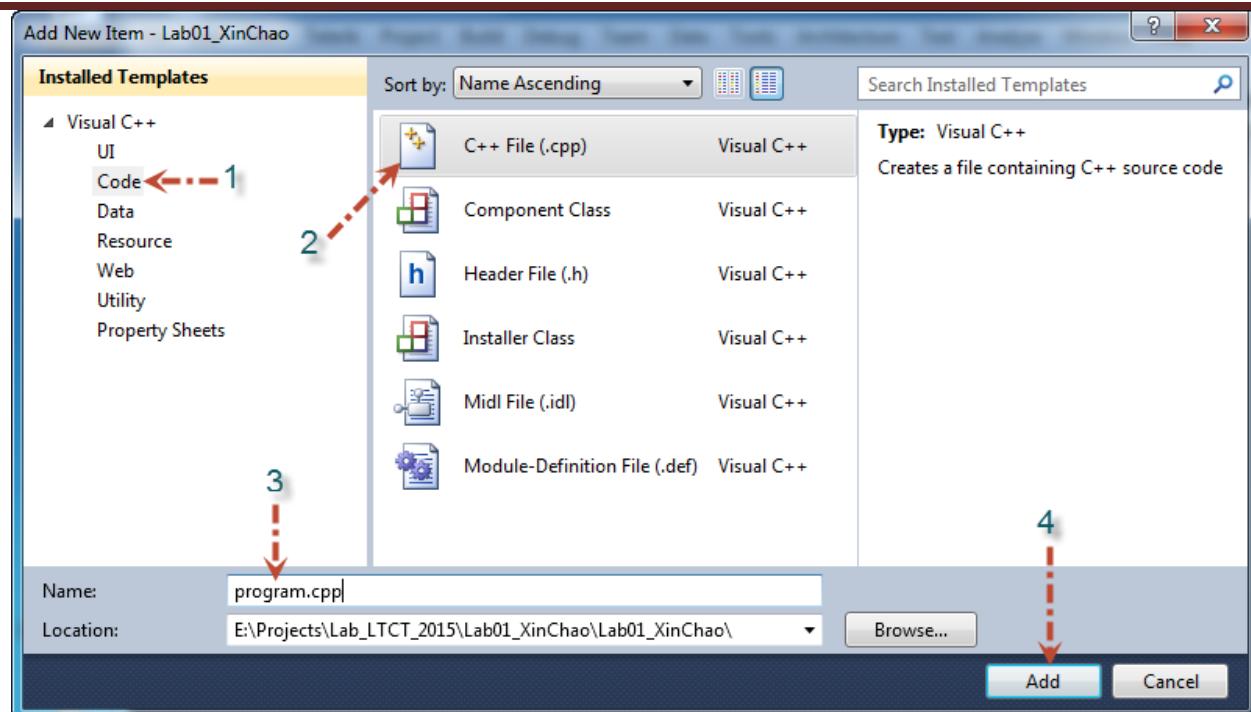
Bước 6. Nếu chương trình không hiển thị cửa sổ **Solution Explorer** như hình dưới đây, hãy chọn menu **View > Solution Explorer**. Cửa sổ này hiển thị cấu trúc của một dự án C++.



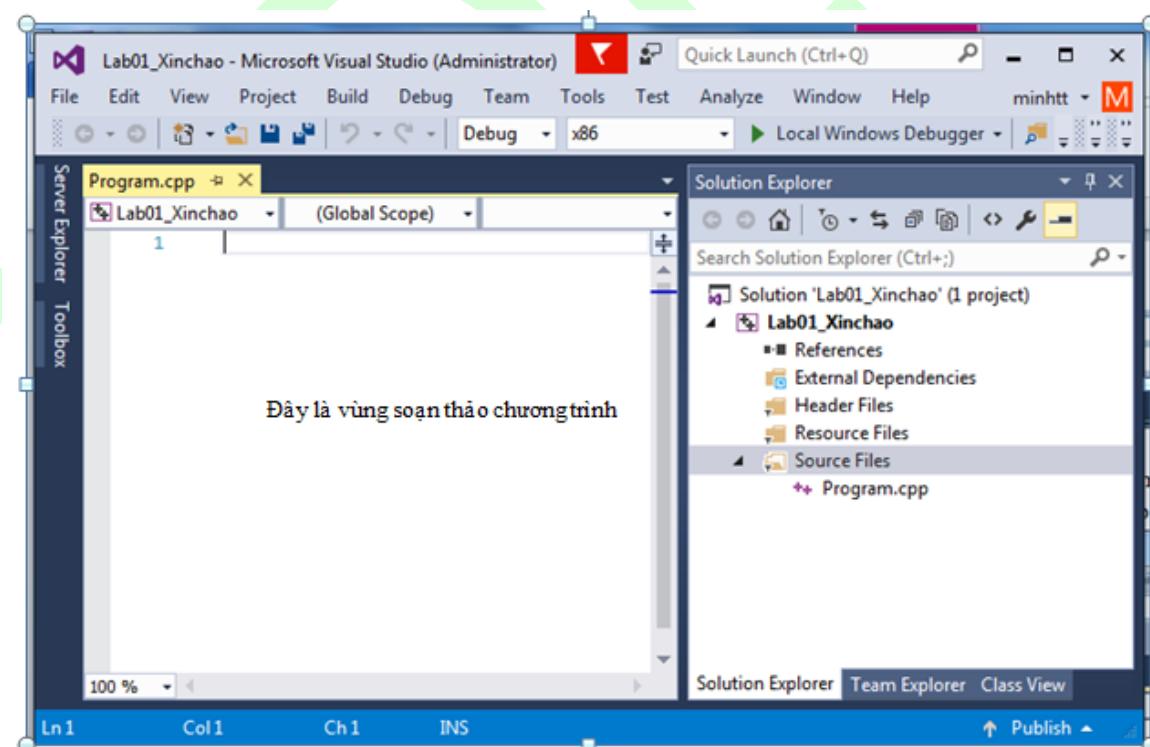
Bước 7. Tiếp theo, ta sẽ tạo một tập tin *.cpp để chứa mã nguồn của chương trình. Để tạo tập tin này, ta có thể thực hiện một trong các bước sau:

- Nhấp phải chuột vào thư mục **Source Files**, chọn **Add > New Items...**
- Chọn từ menu **Projects > Add New Items ...**
- Nhấn tổ hợp phím **Ctrl + Shift + A**

Bước 8. Trong cửa sổ Add New Item, chọn mục **Code > C++ File (.cpp)**, nhập tên tập tin là **program.cpp** vào mục **Name** rồi nhấn **Add**.



Bước 9. Trong thư mục Source Files xuất hiện một tập tin mới tên là program.cpp. Nhấp đôi chuột vào tập tin này để mở nó trong phần cửa sổ soạn thảo ở bên trái:



Bước 10. Đến đây, dự án đã được tạo xong. Ta có thể viết mã lệnh cho chương trình đầu tiên. Trong phần tiếp theo, ta sẽ viết mã lệnh để xuất ra màn hình một dòng thông báo có nội dung: “Chào mừng các bạn đến với C++”.

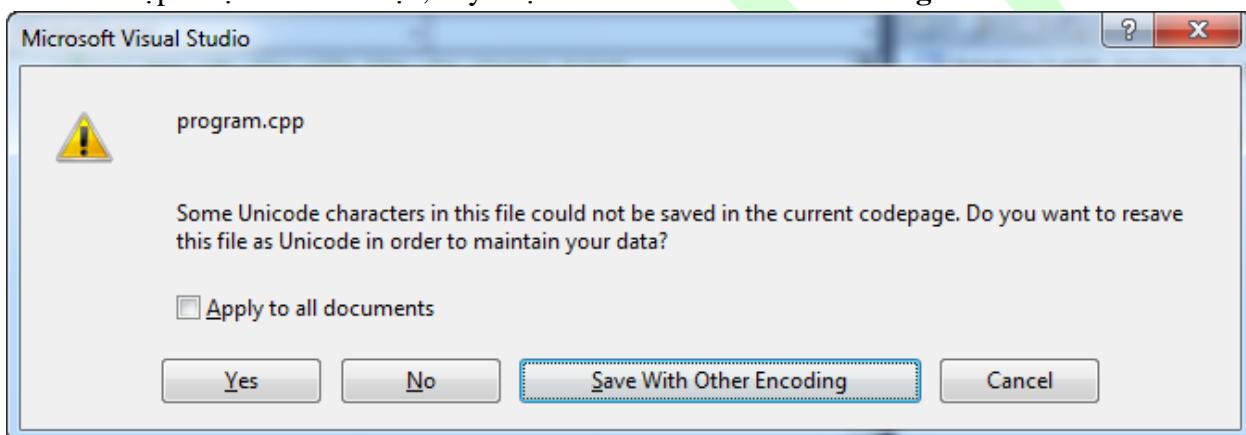
3. Chương trình C++ đầu tiên

Bước 1. Trong vùng soạn thảo, nhập đoạn mã sau (chú ý: không nhập các số ở lề bên trái)

```
1 // Nạp các thư viện hàm vào chương trình
2 #include<iostream>
3
4 using namespace std;
5
6 // Định nghĩa hàm main
7 int main()
8 {
9     // Xuất một câu chào ra màn hình
10    cout << "Chao mung cac ban den voi C++";
11
12    // Trả về giá trị 1
13    return 1;
14 }
```

Bước 2. Nhấn nút **Save** (hoặc **Save All**) hoặc nhấn tổ hợp phím **Ctrl + S** hoặc chọn menu **File > Save...** để lưu mã nguồn.

Bước 3. Nếu hộp thoại sau xuất hiện, hãy chọn **Save With Other Encoding ...**



Bước 4. Trong mục **Encoding** của cửa sổ **Advanced Save Options**, chọn **Unicode (UTF-8 with signature)** như hình sau.



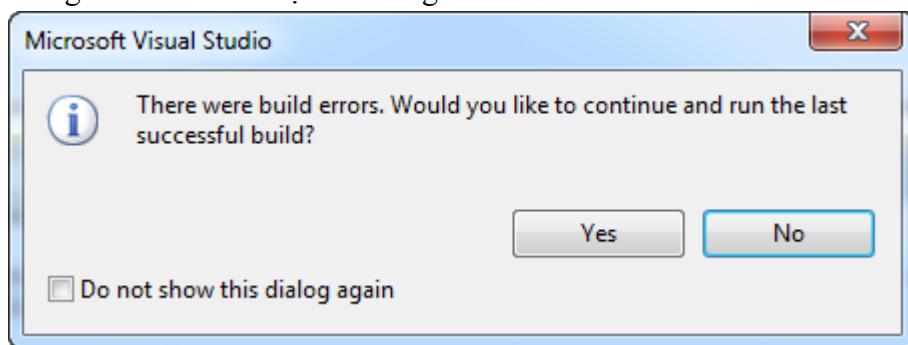
4. Biên dịch và chạy chương trình

Bước 5. Để biên dịch chương trình, chọn menu **Build > Build Solution** hoặc nhấn tổ hợp phím **Ctrl + Shift + B**.

Bước 6. Để chạy chương trình, thực hiện 1 trong 3 cách sau:

- Chọn menu **Debug > Start debugging**
- Nhấn tổ hợp phím **Ctrl+F5**
- Nhấn nút **Local Windows debugger** trên thanh công cụ.

Bước 7. Nếu xuất hiện hộp thông báo như hình sau thì có nghĩa chương trình bị lỗi, nhấn nút **No** và dò lại đoạn mã trong bước 1 để viết lại cho đúng.



Bước 8. Nếu không xuất hiện hộp thoại như trên, chương trình sẽ chạy và xuất hiện một cửa sổ màu đen, trong đó có dòng chữ “Chao mung cac ban den voi C++”. Tuy nhiên, cửa sổ này xuất hiện rồi tắt trong một khoảng thời gian rất ngắn. Lý do là sau khi xuất ra câu thông báo bởi câu lệnh **cout**, chương trình gặp câu lệnh **return** và kết thúc ngay.

5. Dùng chương trình để xem kết quả

Bước 9. Để cho phép người dùng có thể xem kết quả rồi nhấn một phím bất kỳ để kết thúc chương trình, ta cần sử dụng câu lệnh **_getch()**. Lệnh này nằm trong thư viện **conio.h** và có tác dụng là chờ người dùng nhấn một phím.

Bước 10. Bổ sung lệnh nạp thư viện hàm **conio.h** và lệnh **_getch()** để được kết quả như hình sau:

```
1 // Nạp các thư viện hàm vào chương trình
2 #include<iostream>
3 #include<conio.h>
4
5 using namespace std;
6
7 // Định nghĩa hàm main
8 int main()
9 {
10    // Xuất một câu chào ra màn hình
11    cout << "Chao mung cac ban den voi C++";
12
13    // Dừng chương trình và chờ nhấn 1 phím để kết thúc
14    _getch();
15
16    // Trả về giá trị 1
17    return 1;
18 }
```

Bước 11. Nhấn nút **Ctrl+F5** để chạy chương trình (Xem lại bước 6).

Bước 12. Chụp ảnh màn hình và ghi nhận kết quả vào file Word.

Bước 13. Thay đổi nội dung của câu chào trong dòng lệnh thứ 11 thành câu có nội dung sau: “C++ cũng không khó lắm”.

Bước 14. Chạy chương trình và ghi nhận kết quả.

Bước 15. Ở dòng thứ 12, bổ sung thêm câu lệnh sau:

```
cout << "Toi da hoan thanh chuong trinh dau tien";
```

Bước 16. Chạy chương trình và ghi nhận kết quả.

Bước 17. Tiếp tục dùng lệnh **cout** để xuất thêm 3 nội dung mà bạn muốn.

6. Gỡ rối chương trình

Phần này hướng dẫn cách giải quyết 1 số lỗi thường gặp khi viết chương trình với C++. Sinh viên cần học thuộc những thông báo lỗi phổ biến này.

Bước 1. Bỏ dấu ; sau câu lệnh `_getch()` và biên dịch lại chương trình. Chọn menu **View > Error List** hoặc nhấn tổ hợp phím **Ctrl + \ + E** để mở cửa sổ chứa thông báo lỗi (**Error List**).

Bước 2. Cho biết trong cửa sổ **Error List** xuất hiện bao nhiêu thông báo lỗi? Đó là những lỗi gì? Giải thích ý nghĩa của các thông báo lỗi.

Bước 3. Thêm dấu ; vào cuối dòng 14 và chạy lại chương trình.

Bước 4. Ghi nhớ thông báo lỗi và cách sửa lỗi **thiếu dấu chấm phẩy ;**

Bước 5. Tiếp theo, thay đổi tên hàm **main** thành **man** (hoặc **mani** hay 1 tên khác) và chạy lại chương trình.

Bước 6. Nhấn nút **No** khi hộp thoại thông báo lỗi xuất hiện và cho biết trong cửa sổ **Error List** xuất hiện bao nhiêu thông báo lỗi? Đó là những lỗi gì? Giải thích ý nghĩa của các thông báo lỗi.

Bước 7. Sửa lại tên hàm là main rồi chạy lại chương trình.

Bước 8. Ghi nhớ thông báo lỗi và cách sửa lỗi **sai tên hàm main hoặc thiếu hàm main.**

Bước 9. Xóa số 1 nằm sau câu lệnh **return** và chạy lại chương trình.

Bước 10. Nhấn nút **No** khi hộp thoại thông báo lỗi xuất hiện và cho biết trong cửa sổ **Error List** xuất hiện bao nhiêu thông báo lỗi? Đó là những lỗi gì? Giải thích ý nghĩa của các thông báo lỗi.

Bước 11. Sửa lại câu lệnh **return 1;** rồi chạy lại chương trình.

Bước 12. Ghi nhớ thông báo lỗi và cách sửa lỗi **thiếu giá trị trả về cho hàm main.**

Bước 13. Xóa dấu } nằm ở cuối hàm main và chạy lại chương trình.

Bước 14. Nhấn nút **No** khi hộp thoại thông báo lỗi xuất hiện và cho biết trong cửa sổ **Error List** xuất hiện bao nhiêu thông báo lỗi? Đó là những lỗi gì? Giải thích ý nghĩa của các thông báo lỗi.

Bước 15. Thêm dấu } vào cuối hàm main rồi chạy lại chương trình.

Bước 16. Ghi nhớ thông báo lỗi và cách sửa lỗi **không tìm thấy dấu } tương ứng với dấu {.**

Bước 17. Xóa dấu { nằm sau tên hàm main và chạy lại chương trình.

Bước 18. Nhấn nút **No** khi hộp thoại thông báo lỗi xuất hiện và cho biết trong cửa sổ **Error List** xuất hiện bao nhiêu thông báo lỗi? Đó là những lỗi gì? Giải thích ý nghĩa của từng thông báo lỗi.

Bước 19. Các thông báo lỗi đó có phản ánh đúng lỗi của chương trình hay không? Tại sao?

Bước 20. Sửa lại hàm main bằng cách thêm { vào sau tên hàm main rồi chạy lại chương trình.

Bước 21. Ghi nhớ thông báo lỗi và cách sửa lỗi **thiếu dấu {.**

Bước 22. Xóa dòng lệnh **using namespace std;** và chạy lại chương trình.

Bước 23. Nhấn nút **No** khi hộp thoại thông báo lỗi xuất hiện và cho biết trong cửa sổ **Error List** xuất hiện bao nhiêu thông báo lỗi? Đó là những lỗi gì? Giải thích ý nghĩa của các thông báo lỗi.

Bước 24. Bổ sung thêm lệnh **using namespace std;** như cũ rồi chạy lại chương trình.

Bước 25. Ghi nhớ thông báo lỗi và cách sửa lỗi **lệnh cout chưa được khai báo.**

Bước 26. Thực hiện lại bước 22 → 25 nhưng xóa dòng lệnh **#include <iostream>**

7. Xuất các giá trị rời rạc

Bước 1. Tạo một dự án C++ Win32 Console Application mới. Đặt tên là **Lab01_Bai02_XuatChuoi**. Xem lại phần 2 để biết cách tạo dự án (nếu cần).

Bước 2. Mở tập tin **program.cpp** và nhập đoạn mã sau vào phần soạn thảo.

```

1 // Nạp các thư viện hàm vào chương trình
2 #include<iostream>
3 #include<conio.h>
4
5 using namespace std;
6
7 // Định nghĩa hàm main
8 int main()
9 {
10    // Xuất 2 chuỗi nằm liền nhau
11    cout << "Xin chao" << " tat ca cac ban";
12
13    // Dừng chương trình và chờ nhấn 1 phím để kết thúc
14    _getch();
15
16    // Trả về giá trị 1
17    return 1;
18 }
```

Bước 3. Biên dịch và chạy chương trình. Nếu xuất hiện hộp thoại thông báo lỗi, nhấn nút **No** rồi dò và sửa lỗi trước khi tiếp tục.

Bước 4. Chụp ảnh màn hình và ghi nhận kết quả.

Bước 5. Sau lệnh cout, bổ sung thêm đoạn mã sau:

```
// Xuất một chuỗi và sau đó là một số
cout << "Chao nam moi " << 2015;
```

Bước 6. Biên dịch và chạy chương trình. Ghi nhận lại kết quả.

Bước 7. Để ý thấy câu “**Chao nam moi 2015**” nằm liền sau từ “**cac ban**”. Để nội dung “**Chao nam moi 2015**” nằm tách ra ở 1 dòng khác, ta phải bổ sung thêm ký tự xuống dòng **bằng 1 trong 2 cách** sau:

- Cách 1: Thêm ký tự **\n** hoặc **\r\n** vào sau từ “**cac ban**” ở lệnh **cout** thứ nhất.
`cout << "Xin chao" << " tat ca cac ban\n";`
- Cách 2: Thêm hằng số **endl** (end line = kết thúc 1 dòng) vào câu lệnh **cout** thứ hai
`cout << endl << "Chao nam moi " << 2015;`

Bước 8. Biên dịch và chạy chương trình. Ghi nhận lại kết quả.

Bước 9. Trước lệnh **_getch()**, bổ sung thêm đoạn mã sau (chú ý các khoảng trắng)

```
// Xuất nhiều giá trị trên 1 dòng
cout << endl << "Nam " << 2015
<< " thuoc the ky " << 21
<< " va co ten am lich la " << "At Mui";
```

Bước 10. Biên dịch và chạy chương trình. Ghi nhận lại kết quả.

Bước 11. Tiếp tục bổ sung thêm đoạn mã sau

```
// Dùng một lệnh cout để xuất trên nhiều dòng
cout << endl
    << endl << "Ho va ten : Phan Thanh Binh"
    << endl << "Gioi tinh : Nam"
    << endl << "Ngay sinh : 27/10/1995"
    << endl << "Diem TB : " << 7.75
    << endl << "SV Khoa : Cong nghe Thong tin";
```

Bước 12. Biên dịch và chạy chương trình. Ghi nhận lại kết quả.

8. Xuất nội dung có định dạng, canh lề

Trong phần này, ta sẽ sử dụng một số hàm xây dựng sẵn trong thư viện hàm **iomanip** để thực hiện việc định dạng số thực và canh lề (trái, phải) cho các giá trị được xuất ra màn hình. Chương trình dưới đây sẽ xuất ra một danh sách sinh viên. Mỗi hàng chứa thông tin về một sinh viên, bao gồm:

- mã số (tối đa 10 ký tự),
- họ tên (tối đa 25 ký tự),
- lớp (tối đa 10 ký tự),
- điểm trung bình (tối đa 10 ký tự) và
- điểm tích lũy (tối đa 10 ký tự).

Bước 1. Tạo một dự án Win32 Console Application mới. Đặt tên là **Lab01_Bai03_CanhLe**.

Bước 2. Mở tập tin **program.cpp** và nhập đoạn mã sau vào vùng soạn thảo.

```
1 // Nạp các thư viện hàm vào chương trình
2 #include<iostream>
3 #include<iomanip>
4 #include<conio.h>
5
6 using namespace std;
7
8 // Định nghĩa hàm main|
9 int main()
10 {
11     // Xuất dòng tiêu đề
12     cout << setiosflags(ios::left)           // Thiết lập canh lề trái
13         << setw(10) << "MSSV"                // MSSV chiếm 1 cột
14         << setw(25) << "Ho va ten"          // Họ tên chiếm 25 cột
15         << setw(10) << "Lop"                 // Lớp chiếm 10 cột
16         << setw(10) << "Diem TB"              // Điểm trung bình chiếm 10 cột
17         << setw(10) << "Diem TL"              // Điểm tích lũy chiếm 10 cột
18         << endl;                            // Xoảng dòng
19
20     // Dừng chương trình và chờ nhấn 1 phím để kết thúc
21     _getch();
22
23     // Trả về giá trị 1
24     return 1;
25 }
```

Bước 3. Biên dịch và chạy chương trình. Ghi nhận lại kết quả.

Bước 4. Trước lệnh `_getch()`, bổ sung thêm đoạn mã sau để xuất thông tin về 1 sinh viên

```
// Xuất thông tin sinh viên
cout << setiosflags(ios::left)
<< setw(10) << "1410234"
<< setw(25) << "Nguyen Quang Tam"
<< setw(10) << "CTK38CD"
<< setw(10) << "7.50"
<< setw(10) << setprecision(2) << 5.75
<< endl;
```

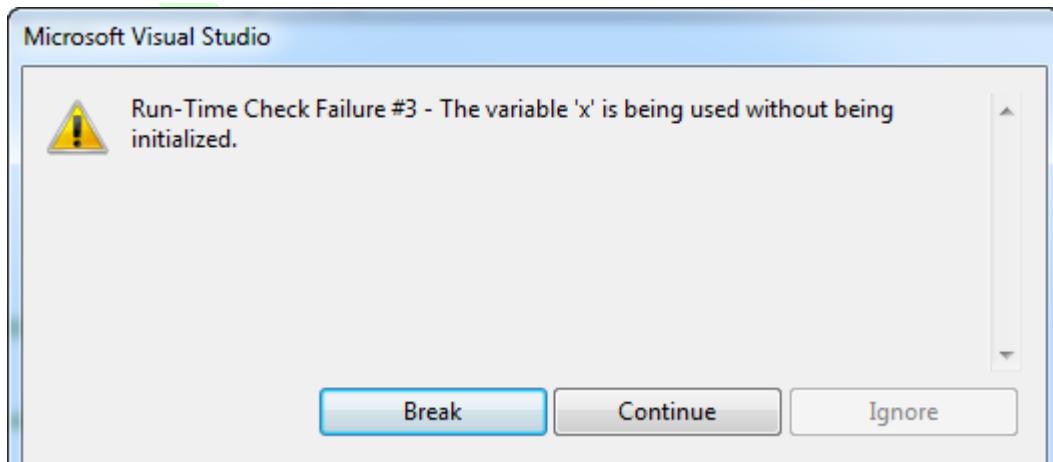
- Bước 5. Biên dịch và chạy chương trình. Ghi nhận lại kết quả.
Bước 6. Cho biết tác dụng của lệnh **setprecision(2)**. Có sự khác biệt gì giữa giá trị được xuất ra trên màn hình với giá trị trong đoạn mã trên?
Bước 7. Tương tự bước 4, viết mã để xuất thêm thông tin về 4 sinh viên khác.
Bước 8. Biên dịch và chạy chương trình. Ghi nhận lại kết quả.

9. Khai báo biến và nhập, xuất giá trị của các biến

- Bước 1. Tạo một dự án Win32 Console Application mới. Đặt tên là **Lab01_Bai04_NhapXuatBien**.

- Bước 2. Mở tập tin **program.cpp** và nhập đoạn mã sau vào vùng soạn thảo.

```
1 // Nạp các thư viện hàm vào chương trình
2 #include<iostream>
3 #include<conio.h>
4
5 using namespace std;
6
7 // Định nghĩa hàm main
8 int main()
9 {
10    // Khai báo một biến kiểu số nguyên, tên là x
11    int x;
12
13    // Xuất giá trị của biến x ra màn hình
14    cout << x;
15
16    // Dừng chương trình và chờ nhấn 1 phím để kết thúc
17    _getch();
18
19    // Trả về giá trị 1
20    return 1;
21 }
```



- Bước 3. Biên dịch và chạy chương trình. Ghi nhận lại kết quả. Hộp thoại thông báo nội dung gì? Đó có phải là một lỗi hay không? Nếu có, hãy giải thích lỗi đó.
- Bước 4. Nếu xuất hiện hộp thoại như hình trên, nhấn nút Break rồi nhấn tổ hợp phím **Shift + F5**.
- Bước 5. Sửa đổi dòng khai báo biến (dòng 11) như sau: **int x = 10;**
- Bước 6. Biên dịch và chạy chương trình. Ghi nhận kết quả. Chương trình còn xuất hiện hộp thoại như hình trên nữa hay không? Tại sao?
- Bước 7. Thay đổi lại lệnh **cout** ở dòng 14 như sau: **cout << "x = " << x;**
- Bước 8. Biên dịch và chạy chương trình. Ghi nhận kết quả. Cho biết 2 chữ **x** trong đoạn mã ở bước 7 có ý nghĩa giống nhau hay không? Giải thích.
- Bước 9. Hãy khai báo thêm 2 biến kiểu nguyên tên là **y, z** và gán giá trị tùy ý cho 2 biến đó.
- Bước 10. Xuất giá trị 2 biến **y, z** nằm trên 2 dòng khác nhau, tương tự như đã làm ở bước 7.
- Bước 11. Biên dịch và chạy chương trình, ghi nhận kết quả.
- Bước 12. Tiếp tục bổ sung đoạn mã sau vào chương trình

```
// Xuất tổng của 3 biến x, y, z  
cout << endl << "x + y + z = " << x + y + z;
```

- Bước 13. Biên dịch và chạy chương trình. Ghi nhận lại kết quả.
- Bước 14. Thay đổi đoạn mã ở bước 12 thành đoạn mã sau (có thể viết trên cùng 1 dòng, không cần xuống dòng)

```
// Xuất tổng của 3 biến x, y, z  
cout << endl  
    << x << " + "  
    << y << " + "  
    << z << " = "  
    << x + y + z;
```

- Bước 15. Biên dịch và chạy chương trình. Ghi nhận lại kết quả. Cho biết sự khác biệt so với kết quả ở bước 13. Giải thích tại sao có kết quả trên?

Lệnh **int x = 10;** trong bước 5 có ý nghĩa là khai báo một biến kiểu số nguyên tên là **x** và khởi gán giá trị cho nó là 10. Tuy nhiên, trên thực tế, giá trị này thường là không biết trước. Thay vào đó, ta phải nhập giá trị cho biến từ bàn phím hoặc gán kết quả trả về từ một biểu thức. Ví dụ sau sẽ hướng dẫn cách sử dụng biến để tính chu vi và diện tích hình chữ nhật khi biết chiều dài và chiều rộng.

- Bước 16. Trước lệnh **_getch()**, bổ sung thêm đoạn mã sau

```
// Khai báo 2 biến lưu kiểu nguyên để lưu chiều dài  
// và chiều rộng của hình chữ nhật  
int dai, rong;  
  
// Xuất thông báo cho người dùng để nhập chiều dài  
cout << endl << "Nhập chiều dài của HCN : ";  
  
// Chờ người dùng nhập giá trị rồi gán cho biến dai  
cin >> dai;  
  
// Xuất thông báo cho người dùng để nhập chiều rộng  
cout << endl << "Nhập chiều rộng của HCN : ";  
  
// Chờ người dùng nhập giá trị rồi gán cho biến rong
```

```
cin >> rong;

// Xuất giá trị của hai biến dài & rong
cout << endl
    << "Chieu dai hinh chu nhat la " << dai << ", "
    << "chieu rong hinh chu nhat la " << rong;
```

Bước 17. Biên dịch và chạy chương trình. Ghi nhận lại kết quả.

Bước 18. Tiếp theo, ta cần khai báo thêm 2 biến để lưu chu vi và diện tích của hình chữ nhật. Giá trị của 2 biến này sẽ được tính từ giá trị của 2 biến dài, rong như sau

```
// Khai báo hai biến kiểu nguyên mới để lưu
// chu vi và diện tích của hình chữ nhật
int chuVi, dienTich;

// Tính chu vi và diện tích của hình chữ nhật
chuVi = (dai + rong) * 2;
dienTich = dai * rong;
```

Bước 19. Bổ sung đoạn mã dưới đây để xuất chu vi và diện tích hình chữ nhật ra màn hình

```
// Xuất chu vi và diện tích của hình chữ nhật
cout << endl << "Chu vi cua HCN la : " << chuVi;
cout << endl << "Dien tich cua HCN : " << dienTich;
```

Bước 20. Biên dịch và chạy chương trình. Ghi nhận lại kết quả.

10. Định nghĩa hằng số sử dụng chỉ thị #define

Phần này hướng dẫn cách định nghĩa 4 hằng số MAX (số nguyên), KHOA (kiểu chuỗi), PI (kiểu số thực), TAB (kiểu ký tự) bằng cách sử dụng chỉ thị **#define**.

Bước 1. Tạo một dự án Win32 Console Application mới. Đặt tên là **Lab01_Bai05_HangSo**.

Bước 2. Mở tập tin **program.cpp** và nhập đoạn mã sau vào vùng soạn thảo.

```
1 // Nạp các thư viện hàm vào chương trình
2 #include<iostream>
3 #include<conio.h>
4
5 using namespace std;
6
7 // Định nghĩa các hằng số
8 #define MAX      100
9 #define KHOA     "Cong nghe Thong tin"
10 #define PI       3.1415926
11 #define TAB      '\t'
12
13 // Định nghĩa hàm main
14 int main()
15 {
16
17     // Dừng chương trình và chờ nhấn 1 phím để kết thúc
18     _getch();
19
20     // Trả về giá trị 1
21     return 1;
22 }
```

Bước 3. Biên dịch và chạy chương trình. Kết quả xuất ra trên màn hình là gì? Tại sao?

Bước 4. Tại dòng 16, nhập vào đoạn mã sau

```

16 // Xuất giá trị các hằng số
17 cout << endl << "Gia tri hang so MAX la : " << MAX;
18 cout << endl << "Gia tri hang so KHOA la : " << KHOA;
19 cout << endl << "Gia tri hang so PI la : " << PI;
20 cout << endl << TAB << "Dong nay duoc thut le 1 dau TAB";

```

Bước 5. Biên dịch và chạy chương trình. Ghi nhận kết quả. Giá trị của hằng số PI xuất ra trên màn hình có giống với giá trị đã định nghĩa trong bước 2 hay không? Tại sao?

Bước 6. Các bước tiếp theo minh họa cách sử dụng hằng kết hợp với biến để tính chu vi và diện tích hình tròn. Trước lệnh `_getch()`, nhập đoạn mã sau:

```

26 // Khai báo biến r kiểu số thực để lưu bán kính hình tròn
27 float r;
28
29 // Xuất thông báo yêu cầu người dùng nhập bán kính
30 cout << endl << "Nhập bán kính hình tròn : ";
31
32 // Chờ người dùng nhập giá trị rồi gán cho biến r
33 cin >> r;
34
35 // Khai báo 2 biến số thực để lưu chu vi và diện tích
36 float chuVi, dienTich;
37
38 // Tính chu vi và diện tích hình tròn
39 chuVi = 2 * PI * r;
40 dienTich = PI * r * r;
41
42 // Xuất chu vi và diện tích hình tròn ra màn hình
43 cout << endl << "Ban kinh : R = " << r
44     << ", Chu vi : C = " << chuVi
45     << ", Dien tich : S = " << dienTich;

```

Bước 7. Biên dịch và chạy chương trình. Ghi nhận kết quả.

Bước 8. Trong đoạn mã trên, chỉ ra dòng lệnh nào cho thấy sự kết hợp giữa hằng và biến?

11. Định nghĩa hằng số sử dụng từ khóa const

Bước 1. Xóa (hoặc comment) các lệnh từ dòng 7 đến dòng 11 trong bước 2 của phần 10.

Bước 2. Bổ sung đoạn mã sau vào trước các lệnh `cout` trong hàm `main`

```

// Định nghĩa các hằng số
const int MAX      = 100;
const char KHOA[] = "Cong nghe Thong tin";
const float PI     = 3.1415926;
const char TAB     = '\t';

```

Bước 3. Biên dịch và chạy lại chương trình. Ghi nhận kết quả. Có gì khác so với bước 7 phần 10?

Bước 4. Liệt kê những điểm khác nhau giữa hai cách định nghĩa hằng số.

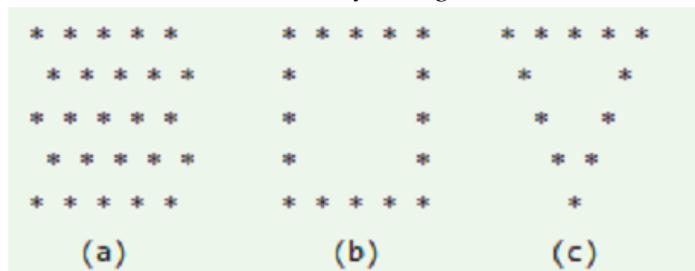
D. Bài tập bắt buộc

1. Biến

Viết chương trình khai báo 5 biến: **k** kiểu số nguyên (int), **x** kiểu số nguyên dài (long), **a** kiểu số thực (float), **h** kiểu số thực (double), **c** kiểu ký tự. Sau đó, nhập và xuất giá trị của các biến đó.

2. Vẽ hình

Viết chương trình vẽ các hình sau lên màn hình. Gợi ý: dùng lệnh cout để xuất dấu * và khoảng trắng.



3. Phép gán

Viết chương trình cho phép nhập vào 2 số nguyên x và y . Sau đó, tính giá trị của các biểu thức sau rồi xuất kết quả ra màn hình: $x + y$, $x - y$, $x * y$, x / y , $x \% y$.

Hướng dẫn:

- Khai báo 3 biến kiểu số nguyên: x , y và z (dùng để lưu kết quả).
- Nhập giá trị cho hai biến x , y từ bàn phím.
- Thực hiện phép gán: $z = x + y$.
- Xuất giá trị z ra màn hình.
- Thực hiện tương tự 2 bước trên cho các phép toán còn lại.

4. Hình tam giác

Viết chương trình cho phép nhập vào độ dài 3 cạnh của một tam giác. Sau đó, xuất ra màn hình chu vi và diện tích của tam giác đó. Diện tích tam giác khi biết độ dài 3 cạnh được tính bởi công thức:

$$S = \sqrt{p(p-a)(p-b)(p-c)} \text{ với } p = (a+b+c)/2$$

Hướng dẫn:

- Nạp thư viện hàm **math.h** để có thể sử dụng các hàm toán học.
- Khai báo 3 biến kiểu số nguyên: a , b và c .
- Khai báo 3 biến kiểu số thực: p , **chuVi** và **dienTich**.
- Nhập giá trị cho ba biến a , b , c từ bàn phím.
- Thực hiện phép gán: **chuVi** = $a + b + c$.
- Thực hiện phép gán: $p = \text{chuVi} / 2.0$
- Thực hiện phép gán: **dienTich** = $\text{sqrt}(p * (p-a) * (p-b) * (p-c))$
- Xuất chu vi và diện tích của hình tam giác.

5. Đổi giờ - phút - giây

Viết chương trình cho phép nhập vào số giây (ký hiệu là n) và sau đó quy đổi thời gian giây thành giờ, phút, giây. Xuất kết quả ra màn hình dưới dạng: giờ:phút:giây. Ví dụ: người dùng nhập $n = 3770$ thì xuất ra màn hình 1:2:50.

Hướng dẫn:

- Khai báo 4 biến kiểu số nguyên: n , **gio**, **phut** và **giay**.
- Định nghĩa hằng số MAX có giá trị bằng 3600 và SIXTY có giá trị bằng 60.
- Nhập giá trị cho biến n từ bàn phím.
- Thực hiện phép gán: **gio** = n / MAX .

- Thực hiện phép gán: $phut = (n \% MAX) / SIXTY$.
- Thực hiện phép gán: $giay = (n \% MAX) \% SIXTY$.
- Xuất giá trị của các biến $gio, phut, giay$ theo yêu cầu.

6. Phép toán

Viết chương trình trong đó khai báo hai biến kiểu số nguyên x và y . Thực hiện tuần tự các lệnh sau và xuất giá trị của các biến này **sau mỗi lệnh**.

```

x++
x--
++x
--x
x = x / y
y = x % y
x *= y

```

7. Số lớn

Viết chương trình cho phép người dùng nhập vào một số nguyên (kiểu int) có giá trị lớn (tối thiểu 10 chữ số). Sau đó, xuất ra màn hình số vừa nhập. Giải thích kết quả.

8. Lũy thừa

Viết chương trình cho phép nhập vào 2 số thực x, y và số nguyên n . Sau đó, xuất kết quả của biểu thức $x^n + y^n$ ra màn hình. *Gợi ý: sử dụng hàm pow(x, n) trong thư viện math.h để tính lũy thừa.*

9. Hình thang cân

Viết chương trình cho phép nhập vào độ dài đáy lớn (a), đáy bé (b) và chiều cao (h) của một hình thang cân. Sau đó, xuất ra màn hình chu vi và diện tích của hình thang đó. Chu vi và diện tích của hình thang được tính bởi công thức:

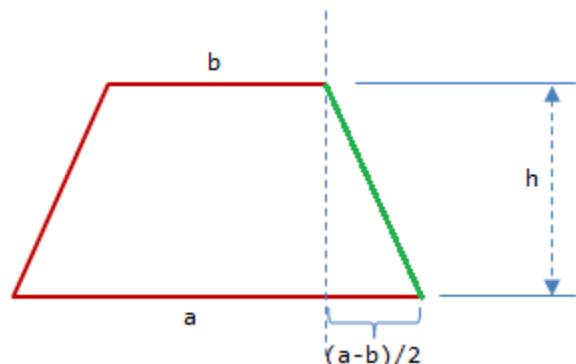
$$\text{cạnh bên} = \sqrt{\left(\frac{\text{đáy lớn} - \text{đáy bé}}{2}\right)^2 + (\text{chiều cao})^2}$$

$$\text{Chu vi} = \text{đáy lớn} + \text{đáy bé} + 2 * \text{cạnh bên}$$

$$\text{Diện tích} = \frac{(\text{đáy lớn} + \text{đáy bé}) * \text{chiều cao}}{2}$$

Hướng dẫn:

- Nạp thư viện hàm **math.h** để có thể sử dụng các hàm toán học.
- Khai báo 6 biến kiểu số thực: $a, b, h, p, canhBen, chuVi$ và $dienTich$.
- Nhập giá trị cho ba biến a, b, h từ bàn phím.
- Thực hiện phép gán: $p = (a - b) / 2, canhBen = sqrt(p * p + h * h)$.
- Tính chu vi và diện tích hình thang dựa vào các công thức đã cho ở trên.
- Xuất chu vi và diện tích của hình tam giác.



10. Danh sách sinh viên

Viết chương trình hiển thị một danh sách sinh viên theo mẫu sau:

DANH SACH SINH VIEN			
MSSV	Họ và Tên	Khoa	Điểm
1211520	Le Duy Tung	33	4.57
1210152	Tran Van Tien	34	VT
1211962	Pham Duc Anh	33	6.48
1211518	Nguyen Hong Phuc	36	VT
1211510	Pham Minh Duc	30	8.05
1211793	Nguyen Khac Uy	33	VT
1211519	Nguyen Tan Tai	34	2.89

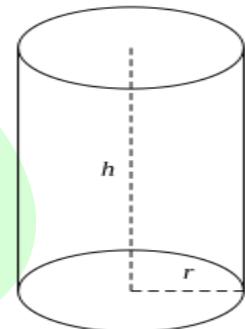
Gợi ý: xem mục 8 phần hướng dẫn thực hành để biết cách căn lề.

11. Hình trụ tròn

Viết chương trình cho phép người dùng nhập vào bán kính mặt đáy (R) và chiều cao (h) của một hình trụ tròn. Sau đó, xuất ra màn hình chu vi và diện tích của mặt đáy, diện tích xung quanh, diện tích toàn phần và thể tích của hình trụ tròn.

Hướng dẫn:

- Thể tích hình trụ tròn: $V = \pi R^2 h$
- Diện tích xung quanh: $S_{xq} = 2\pi Rh$
- Diện tích toàn phần: $S_{tp} = 2\pi Rh + 2\pi R^2$

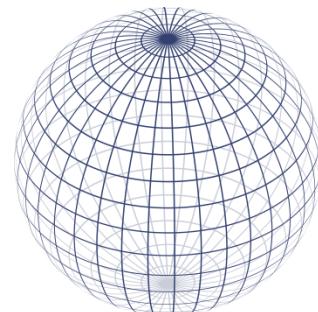


12. Hình cầu

Viết chương trình cho phép người dùng nhập vào bán kính hình cầu (R). Sau đó, xuất ra màn hình diện tích mặt cầu và thể tích khối cầu.

Hướng dẫn:

- Thể tích khối cầu: $V = \frac{4}{3}\pi R^3$
- Diện tích mặt cầu: $S = 4\pi R^2$



E. Bài tập làm thêm

1. Phép gán

Hãy cho biết ý nghĩa của (và cài đặt chương trình để kiểm chứng) hai đoạn mã sau đây:

```
int t = a;
b = t;
a = b;
```

```
int t = a;
a = b;
b = t;
```

2. Chỉ số khối cơ thể (Body Mass Index)

Viết chương trình cho phép người dùng nhập vào khối lượng cơ thể (w – tính theo đơn vị kg) và chiều cao của người đó (h – tính theo đơn vị mét). Sau đó, xuất ra màn hình chỉ số khối cơ thể (BMI) của người đó, biết rằng $BMI = w / h^2$.

3. Khoảng cách

Viết chương trình cho phép người dùng nhập vào tọa độ của 2 điểm **A**, **B** trong không gian Euclid. Sau đó, xuất ra màn hình khoảng cách giữa hai điểm đó. Công thức tính khoảng cách **d** giữa hai điểm **A**(x_a, y_a) và **B**(x_b, y_b) được cho như sau:

$$d = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$

4. Độ lạnh của gió (Wind chill)

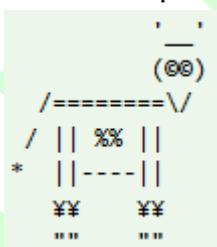
Cho **t** là nhiệt độ (đơn vị Fahrenheit) và **v** là vận tốc gió (đơn vị dặm/giờ), trung tâm khí tượng quốc gia định nghĩa nhiệt độ hiệu dụng (hay độ lạnh của gió) là đại lượng **w** được tính bởi:

$$w = 35.74 + 0.6215 t + (0.4275 t - 35.75) v^{0.16}$$

Viết chương trình cho phép người dùng nhập vào hai giá trị **t** và **v**, sau đó xuất ra màn hình độ lạnh của gió. Lưu ý khi nhập dữ liệu: $-50 \leq t \leq 50$ và $3 \leq v \leq 120$.

5. Chú chó

Viết chương trình xuất ra màn hình hình ảnh chú chó được kết hợp từ các ký tự như sau:



Gợi ý: Mã ký tự ¥ là 157 (009DH).

6. Thứ trong tuần

Viết chương trình cho phép nhập vào ngày (**d**), tháng (**m**) và năm (**y**) dương lịch. Sau đó, chương trình tự tính toán và xuất ra thứ trong tuần (**w**) ứng với ngày đó. Quy ước: 0 = chủ nhật, 1 = thứ hai, 2 = thứ ba, ...

Cách tìm thứ trong tuần (**w**) khi biết **d**, **m**, **y** được cho như sau:

$$t = y - (14 - m)/12$$

$$x = t + t/4 - t/100 + t/400$$

$$k = m + 12 * ((14 - m) / 12) - 2$$

$$w = (d + x + (31 * k) / 12) \% 7$$

Ví dụ, người dùng nhập ngày 26/12/2014 (**d** = 26, **m** = 12, **y** = 2014) thì **w** sẽ là 5 (ứng với thứ sáu).

$$t = 2014 - (14 - 12) / 12 = 2014 - 2 / 12 = 2014$$

$$x = t + t/4 - t/100 + t/400 = 2014 + 2014/4 - 2014/100 + 2014/400 = 2502$$

$$k = m + 12 * ((14 - m) / 12) - 2 = 12 + 12 * ((14 - 12) / 12) - 2 = 10$$

$$w = (d + x + (31 * k) / 12) \% 7 = (26 + 2502 + (31 * 10) / 12) \% 7 = 5$$

LAB 2. ĐỊNH NGHĨA HÀM VÀ GỌI HÀM

THỜI LƯỢNG: 4 TIẾT

A. Mục tiêu

- Giúp sinh viên làm quen với kỹ thuật lập trình hàm đơn giản, tổ chức project có một tập tin .cpp với nhiều hàm.
- Sau khi hoàn thành bài thực hành này, sinh viên:
 - Nắm vững cấu trúc của hàm, cách định nghĩa hàm, khai báo nguyên mẫu hàm và gọi hàm (bước đầu viết và sử dụng hàm)
 - Phân biệt được tham số hình thức và tham số thực
 - Xác định được biến cục bộ và tham số.
 - Hiểu rõ cách truyền giá trị cho tham số.
 - Ghi nhớ trình tự các bước cần thực hiện khi lập trình hàm.
- Tiếp tục tổ chức được các project có một tập tin .cpp với nhiều hàm.

B. Yêu cầu

- Sinh viên tự đọc kỹ phần C (ôn tập)

Buổi thực tập thứ hai (4 tiết)

• 2 tiết đầu :

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Các bài trong phần D (Hướng dẫn thực hành) lab 2.
 - Yêu cầu lưu trữ :
 - Tạo thư mục MaSV_Lab02 chứa trong ổ đĩa qui định.
 - Sử dụng lại tập tin word LoiChuongTrinh.docx từ lab1 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - Mỗi project, xóa thư mục debug.

• 2 tiết cuối :

- Sinh viên tiếp tục thực tập trong phòng lab :
 - Khối lượng và nội dung : Các bài trong phần E (bắt buộc) lab 2.
 - Yêu cầu lưu trữ và nộp bài :
 - Sử dụng tiếp thư mục MaSV_Lab02 để lưu trữ kết quả
 - Mỗi bài trong 5 bài (tùy chọn trong phần E) tạo project riêng, lưu trữ trong thư mục trên.
 - Mỗi project, xóa thư mục debug.
 - Bổ sung thêm các lỗi vào tập tin LoiChuongTrinh.docx và lưu trữ tập tin vào thư mục trên.
 - Thu bài : Xem thông báo của giáo viên .

C. Ôn tập lý thuyết

1. Cấu trúc của một hàm

```
[kdl] Tên_Hàm ( [ Danh_sách_tham_số ] ) // Đây là dòng tiêu đề của hàm
{
    // Các chỉ thị về kiểu
    // Các câu lệnh xử lý
    [ return [ Biểu_thức ]; ]
}
```

Trong đó:

- Những thành phần nằm trong dấu ngoặc vuông [] có thể có hoặc không.
- Nếu hàm không trả về giá trị thì KDL (kiểu dữ liệu) được thay thế bằng từ khóa **void** và không cần dùng câu lệnh **return**.
- Nếu không chỉ rõ kdl thì mặc định kdl là int. Vì vậy, bắt buộc phải có câu lệnh **return**.
- Tham số hình thức (hay là đối) là một tên, đại diện cho một giá trị được dùng để tính toán trong hàm.
- Mỗi tham số là một cặp **kdl tên_tham_số**. Nếu có nhiều tham số thì phải phân cách bởi dấu ,
- Kiểu dữ liệu của giá trị trả về phải tương thích (thấp hơn hoặc bằng) Kiểu dữ liệu của hàm.
- Tên hàm do người lập trình tự đặt và phải tuân theo qui tắc đặt tên : bắt đầu bằng chữ cái hoặc dấu gạch dưới _, chữ cái đầu tiên của mỗi từ viết bằng chữ in hoa, không được chứa khoảng trắng, không trùng với các tên định danh có sẵn trong ngôn ngữ C++.
- Tên hàm đặt nên dễ đọc, trực quan theo nghĩa liên tưởng được công việc của hàm.
Tên hàm có thể có nhiều từ, và từ đầu là động từ, ký tự đầu của mỗi từ nên viết HOA, có thể dùng dấu _ để phân tách từ nếu cần thiết.

2. Khai báo nguyên mẫu hàm

Dạng khai báo nguyên mẫu hàm:

```
[kdl] Tên_Hàm ( [ Danh_sách_tham_số ] ) ; // Chú ý: có dấu chấm phẩy ;
```

Chú ý:

- Khai báo nguyên mẫu hàm phải **đặt trước hàm main**.
- Để ý thấy dòng khai báo nguyên mẫu hàm chỉ khác dòng tiêu đề của hàm ở dấu chấm phẩy cuối dòng khai báo. Vì vậy, để tránh sai sót do dòng tiêu đề và dòng khai báo nguyên mẫu không khớp nhau, ta nên định nghĩa hàm trước. Sau đó, sao chép dòng tiêu đề hàm, đặt vào trước hàm **main** và thêm dấu chấm phẩy ; vào sau dòng khai báo nguyên mẫu.

3. Lời gọi hàm

Khi cần sử dụng một hàm, ta cần gọi tới hàm đó bằng cách viết lời gọi hàm như sau:

```
Tên_Hàm ( [ Danh_sách_đối_số ] )
```

Trong đó:

- Tên hàm trong lời gọi hàm phải giống y chang tên hàm đã định nghĩa.
- Đối số (hay tham số thực) là giá trị thực tế mà ta muốn truyền vào hàm để tính toán.
- Danh sách đối số (tham số thực) phải tương ứng với danh sách tham số hình thức trong phần định nghĩa hàm về cả số lượng tham số và kiểu dữ liệu của tham số.
- Khi viết danh sách đối số, chỉ cần viết tên, không cần viết kiểu dữ liệu của đối số.
- Trong trường hợp hàm trả về kiểu void, lời gọi hàm chỉ đơn giản như sau:

Tên_Hàm ([Danh_sách_đối_số]); // Có dấu chấm phẩy

- Trường hợp hàm có trả về giá trị khác kiểu void, lời gọi hàm có thể được dùng:
 - Làm về phải của phép gán. Ví dụ: int x = TinhTong(n);
 - Để in ra màn hình bởi lệnh cout. Ví dụ: cout << TinhTong(n);
 - Như một toán hạng trong biểu thức. Ví dụ: a = 2 * TinhTong(n) + 5;

4. Sử dụng hàm

Khi muốn sử dụng hàm, cần tuân thủ theo thứ tự sau:

- Định nghĩa hàm:** có thể đặt sau hoặc trước hàm **main**, hoặc trong tập tin thư viện (*.h).
- Khai báo nguyên mẫu hàm:** đặt trước hàm **main** hoặc trong tập tin thư viện.
- Viết lời gọi hàm:** trong hàm **main** hoặc trong các hàm khác.

D. Hướng dẫn thực hành

1. Tạo cấu trúc chung cho một chương trình có sử dụng hàm

Phần này hướng dẫn cách xây dựng cấu trúc cho một chương trình có sử dụng hàm. **Từ đây về sau, mỗi khi bắt đầu một dự án mới, cần phải xây dựng cấu trúc cho chương trình như trong hình ở bước 2 dưới đây.**

Bước 1. Tạo một dự án Win32 Console Application mới. Đặt tên là **Lab02_Bai01_HamDonGian**.

Bước 2. Mở tập tin **program.cpp** và nhập đoạn mã sau vào vùng soạn thảo.

```

1 // =====
2 // Nạp các thư viện hàm vào chương trình
3 // =====
4 #include<iostream>
5 #include<conio.h>
6
7 using namespace std;
8
9 // =====
10 // Định nghĩa hằng số và các kiểu dữ liệu mới
11 // =====
12 |
13
14 // =====
15 // Khai báo nguyên mẫu của các hàm
16 // =====
17
18
19 // =====
20 // Định nghĩa hàm MAIN
21 // =====
22 int main()
23 {
24
25     // Dừng chương trình và chờ nhấn 1 phím để kết thúc
26     _getch();
27
28     // Trả về giá trị 1
29     return 1;
30 }
31
32 // =====
33 // Định nghĩa các hàm xử lý
34 // =====

```

Dễ thấy, hình trên chia mã nguồn thành 5 phần:

- Phần 1 dùng để nạp các thư viện hàm vào chương trình. Nếu cần thêm các thư viện hàm khác như **math.h**, **iomanip**, **stdio.h**, ... thì chỉ cần viết thêm `#include` vào dòng 6 trở đi.
- Phần 2 dùng để định nghĩa các hằng số dùng chỉ thị `#define` hoặc định nghĩa các kiểu dữ liệu mới bằng chỉ thị **typedef**, **struct**, **enum**, **union**, ... Viết thêm mã từ dòng 12 trở đi.
- Phần 3 dùng để khai báo nguyên mẫu hàm do người lập trình định nghĩa. Viết mã từ dòng 17.
- Phần 4 dùng để định nghĩa nội dung hàm **main**. Chèn thêm mã lệnh vào dòng 24.
- Phần 5 dùng để viết các hàm do người dùng định nghĩa. Thêm mã lệnh từ dòng 35 trở đi.

2. Hàm không có tham số, không trả về giá trị

Bước 3. Tại dòng 36, nhập đoạn mã sau:

```

36 // Định nghĩa hàm xuất một thông báo ra màn hình
37 void ThongBao()
38 {
39     cout << endl << "Ban phai hoan thanh bai tap nay.";
40 }

```

Bước 4. Sao chép dòng tiêu đề, dán vào dòng sau phần khai báo nguyên mẫu hàm, thêm dấu ;

```
14 // =====  
15 // Khai báo nguyên mẫu của các hàm  
16 // =====  
17  
18 void ThongBao();
```

Bước 5. Trong hàm main, viết lời gọi hàm thông báo như sau:

```
20 // =====  
21 // Định nghĩa hàm MAIN  
22 // =====  
23 int main()  
24 {  
25     // Gọi hàm ThongBao  
26     ThongBao();  
27  
28     // Dừng chương trình và chờ nhấn 1 phím để kết thúc  
29     _getch();  
30  
31     // Trả về giá trị 1  
32     return 1;  
33 }
```

Bước 6. Chạy chương trình và ghi nhận kết quả.

3. Hàm có tham số, không trả về giá trị

Bước 7. Tiếp tục bổ sung đoạn mã sau vào cuối tập tin **program.cpp** để định nghĩa hàm XuatKyTu

```
59 // Định nghĩa hàm xuất ký tự có mã ASCII cho trước  
60 // Input:  
61 //     ma : Mã ASCII của ký tự  
62 // Output: Không có  
63 void XuatKyTu(short ma)  
64 {  
65     char kyTu = (char)ma;  
66     cout << endl << ma << " <=> " << kyTu;  
67 }
```

Bước 8. Sao chép dòng tiêu đề, dán vào dòng sau phần khai báo nguyên mẫu hàm, thêm dấu ;

```
14 // =====  
15 // Khai báo nguyên mẫu của các hàm  
16 // =====  
17  
18 void ThongBao();  
19 void XuatKyTu(short ma);
```

Bước 9. Sau lời gọi hàm ThongBao trong hàm **main**, viết thêm đoạn mã sau:

```
29 // Gọi hàm xuất ký tự  
30 XuatKyTu(157); // Dùng giá trị làm đối số
```

Bước 10. Chạy chương trình và ghi nhận kết quả.

Bước 11. Tiếp tục bổ sung đoạn mã sau vào sau lời gọi hàm **XuatKyTu**

```

32 // Khai báo biến kiểu số nguyên tên là dollar và ma
33 short dollar = 36, ma;
34 XuatKyTu(dollar);           // Dùng tên biến làm đối số

```

Bước 12. Chạy chương trình và ghi nhận kết quả.

Bước 13. Viết thêm đoạn mã dưới đây vào sau dòng 34.

```

36 // Nhập mã ASCII từ bàn phím rồi truyền vào hàm
37 cout << endl << "Nhập 1 số trong đoạn [30 .. 255] : ";
38 cin >> ma;
39
40 XuatKyTu(ma);           // Dùng tên biến làm đối số

```

Bước 14. Chạy chương trình và ghi nhận kết quả.

Bước 15. Có nhận xét gì về các lời gọi hàm **XuatKyTu**? Gợi ý: trả lời 5 câu hỏi sau để đưa ra nhận xét

- Số 157, biến dollar, ma gọi chung là gì?
- Tên của tham số hình thức được khai báo khi định nghĩa hàm là gì?
- Đối số và tham số hình thức có nhất thiết phải cùng tên hay không?
- Đối số và tham số hình thức có nhất thiết phải cùng kiểu dữ liệu hay không?
- Có thể viết lời gọi hàm **XuatKyTu(dollar, ma)** được không? Tại sao?

Bước 16. Tiếp tục định nghĩa thêm hàm sau

```

83 // Định nghĩa hàm xuất ra phương trình bậc nhất ax + b = 0
84 // Input:
85 //   a : Hệ số a
86 //   b : Hệ số b
87 // Output: Không có
88 void XuatPhuongTrinh(float a, float b)
89 {
90     cout << endl << a << " * x + " << b << " = 0";
91 }

```

Bước 17. Sao chép dòng tiêu đề, dán vào dòng sau phần khai báo nguyên mẫu hàm, thêm dấu ;

Bước 18. Trong hàm **main**, bổ sung tiếp đoạn mã sau để gọi hàm **XuatPhuongTrinh**.

```

43 // Gọi hàm xuất phương trình
44 XuatPhuongTrinh(2.3, 5);    // Dùng giá trị làm đối số
45
46 // Khai báo hai biến kiểu số thực
47 float p, q = 10;
48 XuatPhuongTrinh(7.5, q);    // Dùng cả giá trị và biến
49
50 // Nhập giá trị thực từ bàn phím, lưu vào biến p
51 cout << endl << "Nhập một số thực : ";
52 cin >> p;
53
54 XuatPhuongTrinh(p, q);      // Dùng tên biến làm đối số

```

Bước 19. Chạy chương trình và ghi nhận kết quả.

Bước 20. Có nhận xét gì về các lời gọi hàm **XuatPhuongTrinh**? Gợi ý: Tham khảo bước 15.

4. Hàm không có tham số, có trả về giá trị

Bước 1. Tạo dự án **Win32 Console Application** mới. Đặt tên là **Lab02_Bai02_HamTraVeGiaTri**.

Bước 2. Mở tập tin **program.cpp** và tạo cấu trúc cho chương trình như **bước 2 mục 1**.

Bước 3. Ở cuối chương trình, bổ sung phần định nghĩa hàm sau đây

```

38 // Định nghĩa hàm nhập một số nguyên từ bàn phím
39 // Input : Không có
40 // Output: Giá trị được nhập từ bàn phím
41 int NhapSoNguyen()
42 {
43     int so;      // Khai báo biến để lưu giá trị nhập vào
44
45     // Xuất thông báo để người dùng biết
46     cout << endl << "Nhập một số nguyên : ";
47
48     // Chờ người dùng nhập một số nguyên
49     cin >> so;
50
51     // Gán giá trị so cho hàm và trả về nơi gọi hàm
52     return so;
53 }
```

Bước 4. Sao chép dòng tiêu đề, dán vào dòng sau phần khai báo nguyên mẫu hàm, thêm dấu ;.

Bước 5. Bổ sung mã lệnh vào hàm main để được kết quả như hình sau:

```

20 =====
21 // Định nghĩa hàm MAIN
22 =====
23 int main()
24 {
25     // Khai báo biến để lưu số được nhập từ bàn phím
26     int x, y;
27
28     // Gọi hàm nhập số nguyên và lưu kết quả vào 2 biến
29     x = NhapSoNguyen();
30     y = NhapSoNguyen();
31
32     // Xuất tổng của 2 số vừa nhập
33     cout << x << " + " << y << " = " << x + y;
34
35     // Dừng chương trình và chờ nhấn 1 phím để kết thúc
36     _getch();
37
38     // Trả về giá trị 1
39     return 1;
40 }
```

Bước 6. Chạy chương trình và ghi nhận kết quả.

Bước 7. Khai báo thêm biến kiểu số nguyên tên là **z**.

Bước 8. Bổ sung đoạn mã sau vào trước lệnh **_getch()**:

```

35 // Sử dụng lời gọi hàm trong biểu thức
36 z = (x + y) * NhapSoNguyen();
37 cout << endl << "z = " << z;
```

Bước 9. Chạy chương trình và ghi nhận kết quả.

Bước 10. Có thể viết lệnh **cout << NhapSoNguyen();** trong hàm main được không? Kiểm chứng bằng chương trình.

5. Hàm có tham số, có trả về giá trị

Bước 11. Định nghĩa thêm hàm **TinhTong** (dùng để tính tổng các số từ 1 tới n) như sau:

```

67 // Định nghĩa hàm tính tổng các số từ 1 tới n
68 // Input :
69 //   n : Một số nguyên dương
70 // Output: Tổng các số từ 1 tới n
71 int TinhTong(unsigned int n)
72 {
73     // Khai báo biến để lưu kết quả
74     int sum = 0;
75
76     // Tính tổng 1 + 2 + ... + n
77     sum = n * (n+1) / 2;
78
79     // Gán giá trị sum cho hàm và trả về nơi gọi hàm
80     return sum;
81 }
```

Bước 12. Sao chép dòng tiêu đề, dán vào dòng sau phần khai báo nguyên mẫu hàm, thêm dấu ;.

Bước 13. Trong hàm main, viết thêm đoạn mã sau:

```

40 // Khai báo biến để lưu kết quả
41 int ketQua = 0;
42
43 // Gọi hàm tính tổng các số từ 1 tới 50
44 ketQua = TinhTong(50);
45
46 // Xuất kết quả ra màn hình
47 cout << endl << "1 + 2 + ... + 50 = " << ketQua;
```

Bước 14. Nhấn nút **F5** để chạy chương trình. Ghi nhận kết quả.

Bước 15. Tiếp tục viết thêm đoạn mã sau vào hàm main.

```

49 // Khai báo một biến số nguyên không âm
50 unsigned int m;
51
52 // Nhập giá trị cho biến m từ bàn phím
53 cout << endl << "Nhap mot so nguyen khong am : ";
54 cin >> m;
55
56 // Gọi hàm tính tổng các số từ 1 tới m
57 ketQua = TinhTong(m);
58
59 // Xuất kết quả ra màn hình
60 cout << endl << "1 + 2 + ... + "
61     << m << " = " << ketQua;
```

Bước 16. Chạy chương trình và ghi nhận kết quả.

6. Ví dụ: Tính diện tích hình chữ nhật và hình tam giác

Phần này hướng dẫn cách viết chương trình tính diện tích hình chữ nhật khi biết chiều dài, chiều rộng và diện tích hình tam giác khi biết độ dài 3 cạnh.

Bước 1. Tạo dự án **Win32 Console Application** mới. Đặt tên là **Lab02_Bai03_TinhDienTich**.

Bước 2. Mở tập tin **program.cpp** và tạo cấu trúc cho chương trình như **bước 2 mục 1**.

Bước 3. Nạp thư viện hàm **math.h** vào chương trình dùng chỉ thị **#include <math.h>**

Bước 4. Định nghĩa hàm tính diện tích hình chữ nhật và tam giác như sau:

```

45 // Định nghĩa hàm tính diện tích hình chữ nhật
46 // Input :
47 //   dai : chiều dài của hình chữ nhật
48 //   rong : chiều rộng của hình chữ nhật
49 // Output: Diện tích của hình chữ nhật
50 int TinhDienTichHCN(int dai, int rong)
51 {
52     int dt;      // Khai báo biến để lưu diện tích
53     // Tính diện tích, lưu vào biến dt
54     dt = dai * rong;
55
56     // Gán giá trị dt cho hàm và trả về nơi gọi hàm
57     return dt;
58 }
59
60
61 // Định nghĩa hàm tính diện tích hình tam giác
62 // Input :
63 // canhA : Độ dài cạnh a
64 // canhB : Độ dài cạnh b
65 // canhC : Độ dài cạnh c
66 // Output: Diện tích của hình tam giác
67 double TinhDienTichTamGiac(int canhA, int canhB, int canhC)
68 {
69     // Khai báo biến
70     double dt,          // để lưu diện tích
71         p;            // để lưu nửa chu vi
72
73     // Tính giá trị nửa chu vi, gán cho biến p
74     p = (canhA + canhB + canhC) / 2.0;
75
76     // Tính diện tích, lưu vào biến dt
77     dt = sqrt(p * (p - canhA) * (p - canhB) * (p - canhC));
78
79     // Gán giá trị dt cho hàm và trả về nơi gọi hàm
80     return dt;
81 }
```

Bước 5. Khai báo nguyên mẫu cho 2 hàm trên

```

15 -----
16 // Khai báo nguyên mẫu của các hàm
17 -----
18
19 int TinhDienTichHCN(int dai, int rong);
20 double TinhDienTichTamGiac(int canhA, int canhB, int canhC);
```

Bước 6. Trong hàm main, bổ sung phần khai báo biến như sau:

```

// Khai báo biến để lưu độ dài các cạnh
int a, b, c;

// Khai báo biến để lưu diện tích
double dienTich;
```

Bước 7. Tiếp tục viết mã lệnh để nhập giá trị cho 3 cạnh a, b, c

```
// Nhập độ dài cạnh a
cout << endl << "Nhập độ dài cạnh a : ";
cin >> a;

// Nhập độ dài cạnh b
cout << endl << "Nhập độ dài cạnh b : ";
cin >> b;

// Nhập độ dài cạnh c
cout << endl << "Nhập độ dài cạnh c : ";
cin >> c;
```

Bước 8. Thực hiện việc gọi hàm tính diện tích hình chữ nhật và xuất kết quả ra màn hình

```
// Gọi hàm tính diện tích hình chữ nhật khi biết
// chiều dài a và chiều rộng b
dienTich = TinhDienTichHCN(a, b);

// Xuất diện tích của hình chữ nhật
cout << endl << "Diện tích của hình chữ nhật có "
    << "chiều dài " << a << " và "
    << "chiều rộng " << b << " là "
    << dienTich;
```

Bước 9. Chạy chương trình và ghi nhận kết quả.

Bước 10. Giải thích tại sao hàm **TinhDienTichHCN** trả về số nguyên nhưng ta có thể gán nó cho một biến kiểu số thực?

Bước 11. Tiếp tục việc gọi hàm tính diện tích hình tam giác và xuất kết quả ra màn hình như sau:

```
// Gọi hàm tính diện tích hình tam giác khi biết
// chiều dài 3 cạnh : a, b, c
dienTich = TinhDienTichTamGiac(a, b, c);

// Xuất diện tích của hình tam giác
cout << endl << "Diện tích của hình tam giác có "
    << "3 cạnh : a = " << a << ", "
    << "b = " << b << ", c = " << c << " là "
    << dienTich;
```

Bước 12. Chạy chương trình và ghi nhận kết quả.

Bước 13. Có nhận xét gì về tên của các tham số hình thức và tên của đối số?

E. Bài tập bắt buộc

Tất cả các bài tập sau phải được viết bằng hàm, sau đó gọi chúng trong hàm main để xem kết quả.
Tên hàm, tên biến và tên tham số phải được đặt tên đúng theo quy ước.

1. Hình tròn

Viết chương trình tính chu vi và diện tích hình tròn khi biết bán kính R.

Hướng dẫn:

- Định nghĩa hằng số $PI = 3.1415926$ trong phần định nghĩa hằng và kiểu dữ liệu mới.
- Định nghĩa hai hàm để tính chu vi và diện tích hình tròn
 $\text{double TinhChuViHinhTron(double r)}$
 $\text{double TinhDienTichHinhTron(double r)}$
- Khai báo nguyên mẫu của 2 hàm trên.
- Trong hàm main, khai báo biến **r**, **chuVi**, **dienTich** và thực hiện nhập giá trị cho **r** từ bàn phím.

- Gọi hàm để tính chu vi, diện tích rồi gán cho 2 biến `chuVi`, `dienTich` tương ứng.
- Xuất giá trị của 2 biến `chuVi`, `dienTich` ra màn hình.

2. Đổi giờ - phút - giây

Viết chương trình cho phép nhập vào số giây (ký hiệu là `n`) và sau đó quy đổi thời gian giây thành giờ, phút, giây. Xuất kết quả ra màn hình dưới dạng: **giờ:phút:giây**. Ví dụ: người dùng nhập `n = 3770` thì xuất ra màn hình `1:2:50`.

Hướng dẫn:

- Định nghĩa hằng số `MAX` có giá trị bằng `3600` và `SIXTY` có giá trị bằng `60`.
- Định nghĩa hàm để đổi thời gian theo nguyên mẫu sau (xem bài tập 5 Lab1 để biết cách tính)
`void DoiThoiGian(unsigned int n)`
- Khai báo nguyên mẫu hàm.
- Trong hàm main, khai báo biến kiểu số nguyên: `soGiay` và nhập giá trị cho nó.
- Gọi hàm đổi thời gian và dùng biến `soGiay` làm đối số.

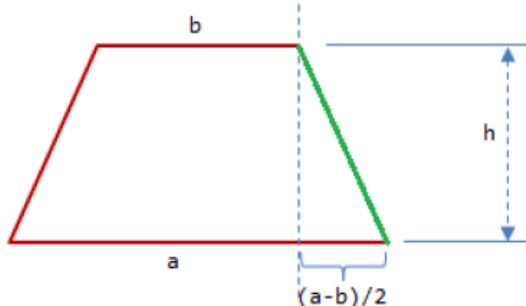
3. Hình thang cân

Viết chương trình cho phép nhập vào độ dài đáy lớn (`a`), đáy bé (`b`) và chiều cao (`h`) của một hình thang cân. Sau đó, xuất ra màn hình chu vi và diện tích của hình thang đó. Chu vi và diện tích của hình thang được tính bởi công thức:

$$\text{cạnh bên} = \sqrt{\left(\frac{\text{đáy lớn} - \text{đáy bé}}{2}\right)^2 + (\text{chiều cao})^2}$$

$$\text{Chu vi} = \text{đáy lớn} + \text{đáy bé} + 2 * \text{cạnh bên}$$

$$\text{Diện tích} = \frac{(\text{đáy lớn} + \text{đáy bé}) * \text{chiều cao}}{2}$$

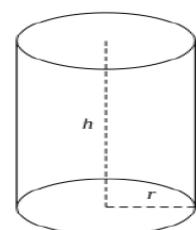


Hướng dẫn:

- Nạp thư viện hàm `math.h` để có thể sử dụng các hàm toán học.
- Định nghĩa các hàm sau đây
`double TinhCanhBen(int dayLon, int dayBe, int chieuCao)`
`double TinhChuVi(int dayLon, int dayBe, double canhBen)`
`double TinhDienTich(int dayLon, int dayBe, int chieuCao)`
- Khai báo nguyên mẫu của các hàm trên.
- Trong hàm main, khai báo 3 biến kiểu số nguyên: `a`, `b`, `h` và 3 biến kiểu số thực `canhBen`, `chuVi`, `dienTich`.
- Nhập giá trị cho ba biến `a`, `b`, `h` từ bàn phím.
- Gọi hàm để tính độ dài cạnh bên, gán cho biến `canhBen`
- Gọi hàm để tính chu vi và diện tích, gán cho biến `chuVi` và `dienTich` tương ứng.
- Xuất chu vi và diện tích của hình thang.

4. Hình trụ tròn

Viết chương trình cho phép người dùng nhập vào bán kính mặt đáy (`R`) và chiều cao (`h`) của một hình trụ tròn. Sau đó, xuất ra màn hình chu vi và diện tích của mặt đáy, diện tích xung quanh, diện tích toàn phần và thể tích của hình trụ tròn. Với mỗi yêu cầu, định nghĩa một hàm riêng.



Hướng dẫn:

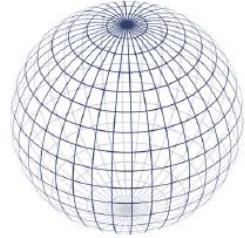
- Định nghĩa hằng số $PI = 3.1415926$ trước khi định nghĩa hàm
- Thể tích hình trụ tròn: $V = 4\pi R^2 h$
- Diện tích xung quanh: $S_{xq} = 2\pi Rh$
- Diện tích toàn phần: $Stp = 2\pi Rh + 2\pi R^2$

5. Hình cầu

Viết chương trình cho phép người dùng nhập vào bán kính hình cầu (R). Sau đó, xuất ra màn hình diện tích mặt cầu và thể tích khối cầu.

Hướng dẫn:

- Thể tích khối cầu: $V = \frac{4}{3}\pi R^3$
- Diện tích mặt cầu: $S = 4\pi R^2$



6. Chỉ số khối cơ thể (Body Mass Index)

Viết chương trình cho phép người dùng nhập vào khối lượng cơ thể (w – tính theo đơn vị kg) và chiều cao của người đó (h – tính theo đơn vị mét). Sau đó, xuất ra màn hình chỉ số khối cơ thể (**BMI**) của người đó, biết rằng $BMI = w / h^2$.

Hướng dẫn:

- Định nghĩa hàm với tiêu đề như sau
double TinhChiSoBMI(double kholuong, double chieucao)
- Khai báo nguyên mẫu hàm
- Trong hàm main, khai báo 2 biến w, h và nhập giá trị cho chúng.
- Gọi hàm **TinhChiSoBMI** với đối số là 2 biến w, h .
- Xuất giá trị trả về của hàm ra màn hình.

7. Khoảng cách

Viết chương trình cho phép người dùng nhập vào tọa độ của 2 điểm A, B trong không gian Euclid. Sau đó, xuất ra màn hình khoảng cách giữa hai điểm đó. Công thức tính khoảng cách d giữa hai điểm $A(xa, ya)$ và $B(xb, yb)$ được cho như sau:

$$d = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$

Hướng dẫn:

- Định nghĩa hàm với tiêu đề như sau
double TinhKhoangCach(double xa, double ya, double xb, double yb)
- Khai báo nguyên mẫu hàm
- Trong hàm main, khai báo 4 biến xa, ya, xb, yb và nhập giá trị cho chúng.
- Gọi hàm **TinhKhoangCach** với đối số là 4 biến xa, ya, xb, yb .
- Xuất giá trị trả về của hàm ra màn hình.

8. Độ lạnh của gió (Wind chill)

Cho t là nhiệt độ (đơn vị Fahrenheit) và v là vận tốc gió (đơn vị dặm/giờ), trung tâm khí tượng quốc gia định nghĩa nhiệt độ hiệu dụng (hay độ lạnh của gió) là đại lượng w được tính bởi:

$$w = 35.74 + 0.6215 t + (0.4275 t - 35.75) v^{0.16}$$

Viết chương trình cho phép người dùng nhập vào hai giá trị t và v , sau đó xuất ra màn hình độ lạnh của gió. Lưu ý khi nhập dữ liệu: $-50 \leq t \leq 50$ và $3 \leq v \leq 120$.

Hướng dẫn:

- Định nghĩa hàm với tiêu đề như sau
double WindChill(double t, double v)

9. So sánh

Viết hàm tìm số lớn nhất trong 4 số thực a, b, c, d được nhập từ bàn phím.

Hướng dẫn:

- Sử dụng toán tử tam phân ? : để so sánh 2 giá trị
- Định nghĩa hàm với tiêu đề như sau
double TimMax(double x, double y)
double TimMax4(double a, double b, double c, double d)
- Trong hàm TimMax4, gọi hàm TimMax trên cặp a, b sau đó là c, d. Cuối cùng, gọi hàm TimMax trên kết quả đó.

10. Chuyển đổi hệ tọa độ

Hệ tọa độ xích đạo được sử dụng rộng rãi bởi các nhà thiên văn học để chỉ ra vị trí của một ngôi sao trên bầu trời. Vị trí của các ngôi sao được xác định bởi xích vĩ độ (δ), góc của mũi giờ (H) và vĩ độ (ϕ) của chúng. Hệ tọa độ ngang (hay còn gọi là hệ tọa độ Alt/Az) rất hữu ích trong việc xác định sự bố trí / thời gian mọc lặn của các đối tượng trên bầu trời. Vị trí của chúng được xác định bởi độ cao (altitude – góc thẳng đứng từ đường chân trời) và góc phương vị (azimuth – góc ngang).

Cho vị trí của một ngôi sao trong hệ tọa độ xích đạo, hãy viết chương trình tìm vị trí của nó trong hệ tọa độ ngang bằng cách sử dụng công thức sau đây:

Độ cao: Altitude = asin ($\sin \phi \sin \delta + \cos \phi \cos \delta \cos H$)

Phương vị: Azimuth = $\arccos \left(\frac{\cos \phi \sin \delta - \sin \phi \cos \delta \cos H}{\sin \delta \sqrt{1 - \cos^2 \phi \sin^2 \delta}} \right)$

Hướng dẫn:

- Nạp thư viện **math.h** để sử dụng các hàm sin, cos, asin, acos.
- Định nghĩa hàm và khai báo nguyên mẫu hàm với tiêu đề như sau
double TinhDoCao(double phi, double delta, double h)
double TinhPhuongVi(double phi, double delta, double h, double altitude)
- Trong hàm main, cần khai báo các biến phi, delta, h và nhập giá trị cho chúng
- Gọi 2 hàm trên trong hàm main và xuất ra kết quả.

F. Bài tập làm thêm

1. Thứ trong tuần

Viết chương trình cho phép nhập vào ngày (**d**), tháng (**m**) và năm (**y**) dương lịch. Sau đó, chương trình tự tính toán và xuất ra thứ trong tuần (**w**) ứng với ngày đó.

Quy ước: 0 = chủ nhật, 1 = thứ hai, 2 = thứ ba, ...

Cách tìm thứ trong tuần (**w**) khi biết **d**, **m**, **y** được cho như sau:

$$t = y - (14 - m)/12$$

$$x = t + t/4 - t/100 + t/400$$

$$k = m + 12 * ((14 - m) / 12) - 2$$

$$w = (d + x + (31 * k) / 12) \% 7$$

Ví dụ, người dùng nhập ngày 26/12/2014 (**d** = 26, **m** = 12, **y** = 2014) thì **w** sẽ là 5 (ứng với thứ sáu).

$$t = 2014 - (14 - 12) / 12 = 2014 - 2 / 12 = 2014$$

$$x = t + t/4 - t/100 + t/400 = 2014 + 2014/4 - 2014/100 + 2014/400 = 2502$$

$$k = m + 12 * ((14 - m) / 12) - 2 = 12 + 12 * ((14 - 12) / 12) - 2 = 10$$

$$w = (d + x + (31 * k) / 12) \% 7 = (26 + 2502 + (31 * 10) / 12) \% 7 = 5$$

2. Chuyển đổi hệ màu

Có nhiều định dạng (hệ màu) khác nhau được sử dụng để biểu diễn các màu sắc. Chẳng hạn, định dạng màu RGB được dùng chủ yếu cho các màn hình tinh thể lỏng (LCD), các máy ảnh kỹ thuật số và các trang web. Trong hệ màu này, mỗi màu sắc được tổng hợp từ (hay biểu diễn bởi) ba màu cơ bản: Red (R – đỏ), Green (G – xanh lá) và Blue (B – xanh dương). Mỗi thành phần R, G, B có giá trị từ 0 đến 255. Một hệ màu khác là CMYK được dùng chủ yếu cho việc in ấn, xuất bản sách và tạp chí. Theo định dạng màu này thì mỗi màu sắc được tổng hợp từ 4 thành phần màu: Cyan (C – xanh lơ), Magenta (M – màu tím), Yellow (Y – màu vàng) và black (K – màu đen). Mỗi thành phần C, M, Y, K có giá trị từ 0.0 đến 1.0.

Hãy viết chương trình cho phép nhập vào màu sắc ở hệ RGB. Sau đó, đổi sang hệ CMYK và xuất ra màn hình giá trị tương ứng với từng thành phần màu C, M, Y, K. Lưu ý rằng, nếu mọi giá trị RGB đều là 0 thì các giá trị CMY là 0 và giá trị của K là 1.

Ví dụ: với $r = 75$, $g = 0$, $b = 130$, giá trị của các biến c , m , y , k là

$$c = 0.4230769230769229$$

$$m = 1.0$$

$$y = 0.0$$

$$b = 0.4901960784313726$$

3. Chuyển đổi thang đo nhiệt độ

Viết chương trình cho phép nhập vào giá trị nhiệt độ t oC (Celsius). Sau đó chuyển đổi giá trị này sang các thang đo nhiệt độ khác như Kelvin (K), Fahrenheit (F), Rankine (R).

Hướng dẫn:

- Công thức chuyển đổi nhiệt độ từ Celsius sang các thang đo khác

$$F = t * 9 / 5 + 32$$

$$K = t + 273.15$$

$$R = (t + 273.15) * 9 / 5$$

4. Thanh toán công nợ

Anh A quyết định vay ngân hàng một số tiền M trong vòng Y năm với lãi suất L %/tháng. Viết chương trình để tính số tiền mà anh A phải trả ngân hàng vào mỗi tháng.

5. Đa giác lồi

Viết chương trình tính tổng số đo các góc, tổng số đường chéo của một đa giác lồi có n đỉnh.

LAB 3. CÁC CÂU LỆNH ĐIỀU KHIỂN

THỜI LƯỢNG: 6 TIẾT

A. Mục tiêu

- Giúp sinh viên sử dụng thuận thục các câu lệnh điều khiển.
- Giúp sinh viên biết tổ chức chương trình theo nhiều hàm và cách hoàn thiện chương trình từng bước.
- Sau khi hoàn thành bài thực hành này, sinh viên phải:
 - Nắm vững cú pháp và cách sử dụng các câu lệnh điều khiển.
 - Tuân thủ đúng các quy ước khi viết mã lệnh.
 - Biết cách vận dụng các câu lệnh lặp thích hợp vào từng trường hợp cụ thể.
 - Biết cách sử dụng các cấu trúc điều khiển để giải một số bài toán đơn giản.
 - Tổ chức được các project có một tập tin .cpp gồm nhiều hàm theo cách hoàn thiện từng bước.

B. Yêu cầu

- Sinh viên tự đọc kỹ phần C (ôn tập lý thuyết)

Buổi thực tập thứ 3 (4 tiết)

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần D (Hướng dẫn thực hành) lab 3.
 - Yêu cầu lưu trữ :
 - Tạo thư mục MaSV_Lab03_HD chứa trong ổ đĩa qui định.
 - Mỗi bài trong phần D tạo project riêng, lưu trữ trong thư mục trên.
 - Mỗi project, xóa thư mục debug.
 - Sử dụng lại tập tin word LoiChuongTrinh.docx từ lab2 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - Thu bài : Xem thông báo của giáo viên .

Buổi thực tập thứ 4 (4 tiết)

- 2 tiết đầu :

- Sinh viên tiếp tục thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong các bài phần E (bắt buộc) lab 3.
(Sinh viên cần chuẩn bị bài trước)
 - Yêu cầu lưu trữ và nộp bài :
 - Tạo thư mục MaSV_Lab03_BB chứa trong ổ đĩa qui định
 - Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - Mỗi project, xóa thư mục debug.
 - Bổ sung thêm các lỗi vào tập tin LoiChuongTrinh.docx và lưu trữ tập tin vào thư mục trên.
 - Giáo viên sẽ thu bài qua hệ thống thu bài trực tuyến tại phòng lab (thu thư mục MaSV_Lab03_BB) trước khi giải lao.

C. Ôn tập lý thuyết

1. Câu lệnh rẽ nhánh (if)

Cú pháp

Dạng khuyết	Dạng đầy đủ
if (biểu_thức_logic) Khối_lệnh;	if (biểu_thức_logic) Khối_lệnh_1; else Khối_lệnh_2;
Ý nghĩa:	
Nếu biểu thức lôgic mang giá trị đúng (true), khối lệnh sẽ được thực hiện.	Nếu biểu thức logic mang giá trị đúng (true), khối lệnh 1 sẽ được thực hiện. Ngược lại (false), khối lệnh 2 sẽ được thực hiện.

Cách sử dụng:

- Khi cần thực hiện một thao tác nào đó, quyết định bởi một điều kiện đúng hay sai.
- Nếu khối lệnh có nhiều lệnh thì phải đặt chúng trong cặp dấu mốc nhọn { }
- Có thể lồng nhiều lệnh if – else với nhau.

2. Câu lệnh lựa chọn (switch)

Cú pháp	Ý nghĩa
switch (biểu_thức) { case Hàng_1: Khối_lệnh_1; [break;] case Hàng_2: Khối_lệnh_2; [break;] case Hàng_N: Khối_lệnh_N; [break;] [default: Khối_lệnh_N+1;] }	Nếu biểu thức có giá trị bằng Hàng_K thì khối lệnh thứ K sẽ được thực hiện. Sau đó, <ul style="list-style-type: none"> Nếu có thành phần break; chương trình sẽ thoát khỏi lệnh switch. Nếu không, các khối lệnh nằm sau khối lệnh K sẽ được thực hiện cho tới khi gặp lệnh break hoặc thoát ra khỏi switch. Nếu giá trị của biểu thức không trùng với hàng nào <ul style="list-style-type: none"> Nếu có thành phần default, khối lệnh N+1 sau default sẽ được thực hiện. Nếu không có thành phần default, chương trình sẽ thoát khỏi lệnh switch.

Cách sử dụng:

- Khi cần thực hiện thao tác nào đó dựa vào kết quả tính toán một biểu thức và giá trị kết quả đó phải nằm trong một tập hữu hạn cho trước.
- Không nên viết các lệnh khai báo biến trong các khối lệnh.
- Kiểu dữ liệu của biểu thức và các hàng phải là số nguyên hoặc ký tự.

3. Câu lệnh lặp for

Cú pháp	Cách sử dụng
<pre>for (biều_thúc_1; biều_thúc_2; biều_thúc_3) Khối_lệnh;</pre> <p>Có 2 dạng thường dùng:</p> <ul style="list-style-type: none"> • Dạng tiến <code>for (int i=0; i < n; i++)</code> Khối_lệnh; • Dạng lùi <code>for (int i = n-1; i >= 0; i--)</code> Khối_lệnh; 	<p>Thường dùng trong trường hợp thực hiện lặp đi lặp lại một thao tác nào đó với số lần lặp biết trước. Là dạng rút gọn của lệnh while.</p> <p>Có thể lồng nhiều lệnh for với nhau hoặc với các lệnh khác như if, while, switch...</p> <p>Biểu thức 1 thường dùng để khởi tạo biến điều khiển. Biểu thức 2 dùng để kiểm tra điều kiện lặp. Biểu thức 3 dùng để cập nhật lại giá trị biến điều khiển.</p> <p>Thứ tự thực hiện: biểu thức 1 → biểu thức 2 → khối lệnh → biểu thức 3 → biểu thức 2 → khối lệnh → biểu thức 3 → ...</p>

4. Câu lệnh lặp while và do .. while

Cú pháp lệnh while	Cú pháp lệnh do .. while
<pre>while (biều_thúc_logic) Khối_lệnh;</pre>	<pre>do { Khối_lệnh; } while (biều_thúc_logic)</pre>
Ý nghĩa	
Trong khi biểu thức còn đúng thì lặp lại việc thực hiện khối lệnh.	Thực hiện khối lệnh. Sau đó, kiểm tra giá trị biểu thức logic. Nếu biểu thức mang giá trị đúng (true), tiếp tục thực hiện khối lệnh. Nếu biểu thức sai (false), thoát khỏi lệnh do ... while.
Khối lệnh có thể không được thực hiện lần nào. Thường dùng khi số lần lặp không biết trước.	Khối lệnh được thực hiện ít nhất 1 lần. Thường dùng khi số lần lặp không biết trước.

Chuyển đổi từ lệnh for sang while

Cú pháp	Ví dụ
<pre>for (biều_thúc_1; biều_thúc_2; biều_thúc_3) Khối_lệnh;</pre>	<pre>for (int i=0; i < n; i++) Khối_lệnh;</pre>
<pre>Biểu_thúc_1; while (biểu_thúc_2) { Khối_lệnh; Biểu_thúc_3;</pre>	<pre>int i = 0; while (i < n) { Khối_lệnh; i++; }</pre>

{	}
---	---

5. Một số quy ước khi viết chương trình

Quy ước đặt tên:

- Tên (hàm, biến, tham số, hằng) phải bắt đầu bằng một chữ cái hoặc dấu gạch dưới _, không được chứa khoảng trắng, không được trùng tên với tên định danh có sẵn trong C++.
- Tên phải có ý nghĩa, dễ nhớ, phản ánh đúng tác dụng của nó.
- Tên hằng phải được viết IN HOA tất cả các chữ cái. **Ví dụ: MAX, SIXTY, PI, ...**
- Tên hàm phải viết In Hoa chữ cái đầu tiên trong mỗi từ. **Ví dụ: TinhTong, GiaiPhuongTrinh, ...**
- Tên biến và tham số viết in thường chữ đầu tiên, các từ còn lại viết hoa chữ cái đầu tiên. **Ví dụ: chuVi, dienTich, delta, phi, banKinh, donGiaNhap, ...**
- Tên kiểu dữ liệu mới viết In Hoa chữ cái đầu tiên trong mỗi từ. **Ví dụ: SinhVien, SanPham, ...**

Quy ước viết mã lệnh:

- Các lệnh cùng cấp (hay có vai trò tương đương nhau) nên bắt đầu từ 1 cột. Nghĩa là cạnh thẳng đều lề bên trái.
- Một câu lệnh (hay khối lệnh) phụ thuộc vào (hay con của) một lệnh khác thì phải viết thụt đầu dòng (dịch vào) một khoảng quy định trước, chẳng hạn như 1 bước Tab.
- Mỗi lệnh phải kết thúc bởi dấu chấm phẩy (;). Trừ các lệnh điều khiển if, for, while, switch.
- Nên giải thích rõ ý nghĩa của các lệnh bằng cách sử dụng chú thích (dấu // hoặc /* */)
- Khi định nghĩa hàm, cần chú thích rõ mục đích của hàm, đầu vào (input) và đầu ra (output) của hàm là gì, ý nghĩa của các tham số.
- Không được viết các hàm nằm lồng nhau.
- Không được viết nhiều lệnh nằm trên cùng 1 hàng.
- Khi viết các ký tự đi theo cặp như { }, (), " ", ' ', [], nên viết cả hai, sau đó chèn thêm nội dung khác ở giữa. Việc thiếu một trong 2 ký tự thường gây nên nhiều lỗi khó phát hiện.

D. Hướng dẫn thực hành

Bài 1:

Viết chương trình giải phương trình bậc 2: $ax^2 + bx + c = 0$ ($a \neq 0$)

Bước 7. Tạo 1 project Win32 Console Application mới, đặt tên là **Lab03_Bai01_GiaiPhuongTrinh**.

Lưu trong thư mục **MSSV_Lab03_D_HD**. Trong project này tạo tập tin **program.cpp**

Bước 8. Mở tập tin **program.cpp** và nhập đoạn mã sau vào vùng soạn thảo (phần code tối thiểu mà chương trình có thể chạy được).

```

1 // =====
2 // Nạp các thư viện hàm vào chương trình
3 // =====
4 #include<iostream>
5 #include<conio.h>
6
7 using namespace std;
8
9 // =====
10 // Định nghĩa hằng số và các kiểu dữ liệu mới
11 // =====
12 |
13
14 // =====
15 // Khai báo nguyên mẫu của các hàm
16 // =====
17
18
19 // =====
20 // Định nghĩa hàm MAIN
21 // =====
22 int main()
23 {
24
25     // Dừng chương trình và chờ nhấn 1 phím để kết thúc
26     _getch();
27
28     // Trả về giá trị 1
29     return 1;
30 }
31
32 // =====
33 // Định nghĩa các hàm xử lý
34 // =====

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 9. Nạp thư viện **iomanip**, **math.h**. Sau đó, từ dòng 36 nhập đoạn code sau (định nghĩa hàm nhập một số thực khác 0):

```

// Định nghĩa hàm nhập một số khác không từ bàn phím
// Input : Không có.
// Output: Một số thực khác 0.
float NhapMotSoKhacKhong()
{
    float so;           // Khai báo biến để lưu số thực

    // Thực hiện việc nhập và kiểm tra giá trị nhập
    // Nếu giá trị nhập = 0 thì yêu cầu nhập lại.
    do
    {
        cout << endl << "Nhập một số thực (khác 0) : ";
        cin >> so;
    } while (so == 0);

    // Gán giá trị so cho hàm và trả về nơi gọi hàm
    return so;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 10. Tiếp tục định nghĩa hàm giải phương trình bậc 2 một lần như dưới đây :

```
// Định nghĩa hàm giải phương trình bậc 2
// Input :
//   a : Hệ số a
//   b : Hệ số b
//   c : Hệ số c
// Output: Không có. Chỉ xuất ra nghiệm của phương trình.
void GiaiPhuongTrinhBacHai(float a, float b, float c)
{
    // Khai báo biến
    float delta,           // để lưu giá trị delta
          x;               // để lưu nghiệm của phương trình

    // Tính giá trị delta
    delta = b * b - 4 * a * c;

    // Kiểm tra phương trình có nghiệm hay không?
    if (delta < 0)
    {
        // Trường hợp vô nghiệm: Xuất thông báo PT vô nghiệm
        cout << endl << "Phương trình vô nghiệm";
    }
    else if (delta == 0.0)
    {
        // TH có nghiệm kép: Tìm nghiệm -> xuất kết quả
        x = -b / (2 * a);

        // Xuất thông báo có nghiệm kép
        cout << endl << "Phương trình có nghiệm kép x = " << x;
    }
    else // TH có 2 nghiệm phân biệt
    {
        // Xuất thông báo có 2 nghiệm phân biệt
        cout << endl << "Phương trình có 2 nghiệm phân biệt : ";

        // Tìm nghiệm thứ nhất
        x = (-b + sqrt(delta)) / (2 * a);

        // Xuất nghiệm thứ nhất
        cout << "x1 = " << setprecision(5) << x;

        // Tìm nghiệm thứ hai
        x = (-b - sqrt(delta)) / (2 * a);

        // Xuất nghiệm thứ hai
        cout << " và x2 = " << setprecision(5) << x;
    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 11. Sao chép dòng tiêu đề, dán vào dòng sau phần khai báo nguyên mẫu hàm, thêm dấu ;

```
// =====
// Khai báo nguyên mẫu của các hàm
// =====

float NhaphoSoKhacKhong();
void GiaiPhuongTrinhBacHai(float a, float b, float c);
```

Bước 12. Cập nhật lại hàm main để được kết quả như hình dưới đây:

```

int main()
{
    // Khai báo biến để lưu độ dài các cạnh
    float a, b, c;
    // Nhập hệ số a khác 0
    a = NhapMotSoKhacKhong();
    // Nhập hệ số b
    cout << endl << "Nhập hệ số b : ";
    cin >> b;

    // Nhập độ dài cạnh c
    cout << endl << "Nhập hệ số c : ";
    cin >> c;
    // Gọi hàm giải phương trình bậc 2 một ẩn
    GiaiPhuongTrinhBacHai(a, b, c);
    // Dừng chương trình và chờ nhấn 1 phím để kết thúc
    _getch();
    return 1;
}

```

Bước 13. Chạy chương trình và ghi nhận kết quả.

Bước 14. Hãy viết lại hàm **NhapMotSoKhacKhong** sử dụng vòng lặp **while** thay cho **do..while**.

Bài 2:

Viết chương trình tìm số ngày của một tháng khi biết tháng và năm.

Bước 1. Tạo project Win32 Console Application mới, đặt tên là **Lab03_Bai02_TimSoNgay**, lưu trong thư mục **MSSV_Lab03_D_HD**. Trong project này tạo tập tin **program.cpp**

Bước 2. Mở tập tin **program.cpp** và tạo cấu trúc chương trình như trong bước 2 mục 1 ở trên (tức là viết phần code tối thiểu mà chương trình chạy được)

Bước 3. Tại dòng 36, nhập đoạn mã sau để định nghĩa hàm nhập một số trong miền cho trước:

```

// Định nghĩa hàm nhập một số nằm trong đoạn [min..max]
// Input : min : số nhỏ nhất có thể nhập
//           max : số lớn nhất có thể nhập
// Output: Một số nguyên nằm trong đoạn [min..max].
int NhapSoTrongMien(int min, int max)
{
    int so; // Khai báo biến để lưu số được nhập

    // Thực hiện việc nhập và kiểm tra giá trị nhập. Nếu
    // giá trị nhập nằm ngoài phạm vi thì yêu cầu nhập lại.
    do
    {
        cout << endl << "Nhập một số trong đoạn ["
            << min << ".." << max << "] : ";
        cin >> so;
    } while (so < min || so > max);

    // Gán giá trị so cho hàm và trả về nơi gọi hàm
    return so;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 4. Định nghĩa tiếp hàm tìm số ngày của một tháng như hình dưới đây

// Định nghĩa hàm tìm số ngày của tháng và năm cho trước

// Input :

// thang : Tháng trong năm (1 -> 12)

// nam : Năm dương lịch (1 -> 3000)

// Output: Số ngày của tháng trong năm cho trước.

int TimSoNgay(int thang, int nam)

{

// Khai báo biến để lưu số ngày
int soNgay;

// Dùng lệnh switch để kiểm tra theo tháng
switch (thang)

{

// Nếu là tháng 1,3,5,7,8,10,12 thì số ngày là 31

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

soNgay = 31;
break;

// Nếu là tháng 4,6,9,11 thì số ngày là 30

case 4:

case 6:

case 9:

case 11:

soNgay = 30;
break;

// Trường hợp còn lại (tháng 2), số ngày = 28 hoặc // 29, phụ thuộc vào năm đó có nhuận hay không.

default:

// Kiểm tra có phải năm nhuận hay không?
if ((nam % 4 == 0 && nam % 100 != 0) || (nam % 400 == 0))
 soNgay = 29; // TH năm nhuận
else
 soNgay = 28; // TH không phải năm nhuận
break;

}

return soNgay;

}

Bước 5. Sao chép dòng tiêu đề của 2 hàm, dán vào sau phần khai báo nguyên mẫu hàm, thêm dấu ;

```
// =====
// Khai báo nguyên mẫu của các hàm
// =====

int NhapSoTrongMien(int min, int max);
int TimSoNgay(int thang, int nam);
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 6. Cập nhật lại hàm main (có gọi các hàm) để thực hiện chương trình:

int main()

{

// Khai báo biến để lưu độ dài các cạnh

```

int soNgay, thang, nam;

// Nhập tháng trong năm (1 -> 12)
cout << endl << "Nhập một tháng trong năm";
thang = NhapSoTrongMien(1, 12);

// Nhập năm (1 -> 3000)
cout << endl << "Nhập năm dương lịch";
nam = NhapSoTrongMien(1, 3000);

// Gọi hàm tìm số ngày của tháng và năm đã nhập
soNgay = TimSoNgay(thang, nam);

// Xuất kết quả
cout << endl << "Tháng " << thang
    << " năm " << nam
    << " có " << soNgay << " ngày. ";

// Dừng chương trình và chờ nhấn 1 phím để kết thúc
_getch();

return 1;
}

```

Bước 7. Chạy chương trình và ghi nhận kết quả.

Bước 8. Hãy viết lại hàm **TimSoNgay** sử dụng lệnh **if** thay cho lệnh **switch**.

Bài 3:

Bài toán phân loại tam giác: Ví dụ kết hợp if và switch

Yêu cầu bài toán: Viết chương trình cho phép nhập vào độ dài 3 cạnh của tam giác. Sau đó, xuất ra màn hình thông báo đó là tam giác gì (đều, cân, vuông, vuông cân, thường, không phải tam giác). Hàm phân loại tam giác được định nghĩa dưới đây:

$$PhanLoaiTamGiac(a, b, c) = \begin{cases} 1; & \text{nếu } a, b, c \text{ là 3 cạnh của tam giác đều} \\ 2; & \text{nếu } a, b, c \text{ là 3 cạnh của tam giác cân} \\ 3; & \text{nếu } a, b, c \text{ là 3 cạnh của tam giác vuông} \\ 4; & \text{nếu } a, b, c \text{ là 3 cạnh của tam giác vuông cân} \\ 5; & \text{nếu } a, b, c \text{ là 3 cạnh của tam giác thường} \\ 0; & \text{nếu } a, b, c \text{ không phải là 3 cạnh của tam giác} \end{cases}$$

Bước 1. Tạo project Win32 Console Application mới, đặt tên là **Lab03_Bai03_PhanLoaiTamGiac**, lưu trong thư mục **MSSV_Lab03_D_HD**. Trong project này tạo tập tin **program.cpp**

Bước 2. Mở tập tin **program.cpp** và tạo cấu trúc chương trình như trong bước 2 mục 1 ở trên.

Bước 3. Định nghĩa hàm phân loại tam giác như sau

// Định nghĩa hàm phân loại tam giác

// Input :

// a : Độ dài cạnh a
// b : Độ dài cạnh b
// c : Độ dài cạnh c

// Output:

// 1 : Tam giác đều
// 2 : Tam giác cân
// 3 : Tam giác vuông
// 4 : Tam giác vuông cân

```

// 5 : Tam giác thường
// 0 : Không phải tam giác
int PhanLoaiTamGiac(double a, double b, double c)
{
    int kq = 0; // Khai báo biến lưu kết quả phân loại

    // Nếu a, b, c là 3 cạnh của tam giác
    if (a + b > c && a + c > b && b + c > a)
    {
        // Kiểm tra nếu 3 cạnh bằng nhau thì đó là TG đều
        if (a == b && b == c)
            kq = 1;
        else // TH không phải TG đều
            // Kiểm tra nếu 2 cạnh bằng nhau thì là TG cân
            if (a == b || b == c || a == c)
            {
                // Kiểm tra có phải là TG vuông?
                if (a*a + b*b == c*c || a*a + c*c == b*b || b*b + c*c == a*a)
                    kq = 4;
                else
                    kq = 2;
            }
        else // TH không phải TG cân
            // Kiểm tra có phải tam giác vuông hay không?
            if (a*a + b*b == c*c || a*a + c*c == b*b || b*b + c*c == a*a)
                kq = 3;
            else
                kq = 5;
    }

    // Gán giá trị kq cho hàm và trả về nơi gọi hàm
    return kq;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 4. Tiếp tục định nghĩa hàm thông báo kết quả như hình dưới đây:

```

// Định nghĩa hàm thông báo kết quả phân loại tam giác
// Input :
// loaiTg: Kết quả phân loại tam giác
// a : Độ dài cạnh a
// b : Độ dài cạnh b
// c : Độ dài cạnh c
// Output: Không có. Chỉ xuất thông báo.
void ThongBao(int loaiTg, double a, double b, double c)
{
    // Dùng lệnh switch để kiểm tra theo tháng
    switch (loaiTg)
    {
        case 0: cout << endl << a << ", " << b << ", " << c
                  << " khong phai la 3 canh cua 1 tam giac";
                  break;
        case 1: cout << endl << a << ", " << b << ", " << c
                  << " la do dai 3 canh cua 1 tam giac deu";
                  break;
        case 2: cout << endl << a << ", " << b << ", " << c
                  << " la do dai 3 canh cua 1 tam giac can";
                  break;
        case 3: cout << endl << a << ", " << b << ", " << c
                  << " la do dai 3 canh cua 1 tam giac vuong";
                  break;
    }
}

```

```

        case 4: cout << endl << a << ", " << b << ", " << c
                  << " la do dai 3 canh cua 1 tam giac vuong can";
                  break;
        case 5: cout << endl << a << ", " << b << ", " << c
                  << " la do dai 3 canh cua 1 tam giac thuong";
                  break;
    }
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 5. Sao chép dòng tiêu đề của 2 hàm, dán vào sau phần khai báo nguyên mẫu hàm, thêm dấu ;

```

// =====
// Khai báo nguyên mẫu của các hàm
// =====

int PhanLoaiTamGiac(double a, double b, double c);
void ThongBao(int loaiTg, double a, double b, double c);

```

Bước 6. Cập nhật lại hàm **main** để hoàn chỉnh chương trình:

```

int main()
{
    // Khai báo biến để lưu độ dài các cạnh tam giác
    double a, b, c;

    // Nhập độ dài các cạnh của tam giác
    // Nhập năm (1 -> 3000)
    cout << endl << "Nhập độ dài cạnh a : ";
    cin >> a;

    cout << endl << "Nhập độ dài cạnh b : ";
    cin >> b;

    cout << endl << "Nhập độ dài cạnh c : ";
    cin >> c;

    // Gọi hàm phân loại tam giác
    int ketQua = PhanLoaiTamGiac(a, b, c);

    // Gọi hàm thông báo để xuất kết quả phân loại
    ThongBao(ketQua, a, b, c);

    // Dừng chương trình và chờ nhấn 1 phím để kết thúc
    _getch();

    return 1;
}

```

Bước 7. Chạy chương trình và ghi nhận kết quả.

Bước 8. Có thể dùng lệnh **switch** thay cho các lệnh **if** trong hàm **PhanLoaiTamGiac** được không? Tại sao? Ta chỉ có thể dùng lệnh **switch** thay cho lệnh **if** trong trường hợp nào?

Bài 4:

Viết chương trình tính N! (giai thừa của N) và tổng 1+2+...+N

Bước 1. Tạo project **Win32 Console Application** mới, đặt tên là **Lab03_Bai04_GiaiThua**, lưu trong thư mục **MSSV_Lab03_D_HD**. Trong project này tạo tập tin **program.cpp**

Bước 2. Mở tập tin **program.cpp** và tạo cấu trúc cho chương trình như **bước 2 mục 1**.

Bước 3. Ở cuối chương trình, bổ sung phần định nghĩa các hàm sau đây

```
// Định nghĩa hàm tính giai thừa của N
// Input :
//   n : Một số nguyên không âm
// Output:
//   Giai thừa của n (n!).
int TinhGiaiThua(unsigned int n)
{
    int kq;
    if (n < 2)
        return 1;
    else
    {
        kq = 1;           // Khai báo biến lưu kết quả

        // Duyệt qua các số từ 2 tới n
        for (int i=2; i<=n; i++)
            kq *= i;

        // Gán giá trị kq cho hàm và trả về nơi gọi hàm
        return kq;
    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

// Định nghĩa hàm tính tổng N số nguyên dương đầu tiên

```
// Input :
//   n : Một số nguyên không âm
// Output:
//   Tổng các số từ 1 tới n.
int TinhTong(unsigned int n)
{
    int sum = 0;           // Khai báo biến lưu kết quả

    // Duyệt qua các số từ 2 tới n
    for (int i=1; i<=n; i++)
        sum += i;

    // Gán giá trị kq cho hàm và trả về nơi gọi hàm
    return sum;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 4. Sao chép dòng tiêu đề của 2 hàm, dán vào sau phần khai báo nguyên mẫu hàm, thêm dấu ;

```
// =====
// Khai báo nguyên mẫu của các hàm
// =====

long TinhGiaiThua(unsigned int n);
long TinhTong(unsigned int n);
```

Bước 5. Cập nhật lại hàm **main** để hoàn chỉnh chương trình:

```
int main()
{
    // Khai báo biến
```

```

unsigned int n;

// Nhập giá trị cho biến n
cout << endl << "Nhập một số nguyên không âm : ";
cin >> n;

// Khai báo biến để lưu kết quả
long ketQua;

// Gọi hàm tính n!
ketQua = TinhGiaiThua(n);

// Xuất kết quả
cout << endl << n << " ! = " << ketQua;

// Gọi hàm tính tổng n số nguyên dương đầu tiên
ketQua = TinhTong(n);

// Xuất kết quả
cout << endl << "1 + 2 + ... + n = " << ketQua;

// Dừng chương trình và chờ nhấn 1 phím để kết thúc
_getch();

return 1;
}

```

Bước 6. Chạy chương trình và ghi nhận kết quả.

Bài 5: Bài toán thống kê điểm (Kết hợp for và switch)

Viết chương trình cho phép nhập điểm thi của N sinh viên. Sau đó, thống kê xem có bao nhiêu sinh viên đạt điểm $\geq i$ với $i = 0..10$.

Bước 1. Tạo project **Win32 Console Application** mới. Đặt tên là **Lab03_Bai05_ThongKeDiem**, lưu trong thư mục **MSSV_Lab03_D_HD**. Trong project này tạo tập tin **program.cpp**

Bước 2. Mở tập tin **program.cpp** và tạo cấu trúc cho chương trình như **bước 2 mục 1**.

Bước 3. Định nghĩa thêm hàm **NhapDiem** như sau:

```

// Định nghĩa hàm nhập điểm từ bàn phím
// Input :
// stt : Số thứ tự của sinh viên
// Output: Điểm được nhập từ bàn phím, thuộc đoạn [0..10]
unsigned short NhapDiem(int stt)
{
    unsigned short diem;
    do
    {
        cout << endl << "Nhập điểm của SV thứ " << stt << " : ";
        cin >> diem;
    }
    while (diem < 0 || diem > 10);

    return diem;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 4. Định nghĩa tiếp hàm thống kê điểm như hình dưới đây:

```

// Định nghĩa hàm thống kê điểm
// Input :
//   n : Số lượng sinh viên
// Output:
//   Không có. Chỉ xuất kết quả thống kê.
void ThongKe(unsigned int n)
{
    // Khai báo các biến đếm để thống kê theo điểm số.
    int d0 = 0, d1 = 0, d2 = 0, d3 = 0, d4 = 0, d5 = 0,
        d6 = 0, d7 = 0, d8 = 0, d9 = 0, d10 = 0;

    unsigned short diem;

    // Duyệt qua từng sinh viên
    for (int i=0; i<n; i++)
    {
        // Nhập điểm cho sinh viên thứ i
        diem = NhapDiem(i + 1);

        switch(diem)
        {
            case 10: d10++;;
            case 9: d9++;;
            case 8: d8++;;
            case 7: d7++;;
            case 6: d6++;;
            case 5: d5++;;
            case 4: d4++;;
            case 3: d3++;;
            case 2: d2++;;
            case 1: d1++;;
            case 0: d0++;;
        }
    }

    // Xuất kết quả thống kê
    cout << endl << "So SV co diem >= 0 :" << d0;
    cout << endl << "So SV co diem >= 1 :" << d1;
    cout << endl << "So SV co diem >= 2 :" << d2;
    cout << endl << "So SV co diem >= 3 :" << d3;
    cout << endl << "So SV co diem >= 4 :" << d4;
    cout << endl << "So SV co diem >= 5 :" << d5;
    cout << endl << "So SV co diem >= 6 :" << d6;
    cout << endl << "So SV co diem >= 7 :" << d7;
    cout << endl << "So SV co diem >= 8 :" << d8;
    cout << endl << "So SV co diem >= 9 :" << d9;
    cout << endl << "So SV co diem >= 10 :" << d10;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 5. Sao chép dòng tiêu đề của 2 hàm, dán vào sau phần khai báo nguyên mẫu hàm, thêm dấu ;

```

// =====
// Khai báo nguyên mẫu của các hàm
// =====

unsigned short NhapDiem(int stt);
void ThongKe(unsigned int n);

```

Bước 6. Cập nhật lại hàm main :

```

int main()
{
    // Khai báo biến lưu số lượng sinh viên

```

```

unsigned int n;

// Nhập giá trị cho biến n
cout << endl << "Nhập số luồng sinh viên : ";
cin >> n;

// Gọi hàm thống kê
ThongKe(n);

// Dừng chương trình và chờ nhấn 1 phím để kết thúc
_getch();

return 1;
}

```

Bước 7. Nhấn nút Ctrl+F5 để chạy chương trình. Ghi nhận kết quả.

Bước 8. Tại sao trong hàm thống kê, ta không sử dụng lệnh **break** sau mỗi mệnh đề **case**. Nếu đảo tật trụ các mệnh đề **case** (case d0 -> case d10) thì kết quả có còn đúng không? Tại sao?

Bài 6: Bài toán kiểm tra và liệt kê số nguyên tố (Ví dụ về lệnh while)

Viết chương trình tìm và xuất ra N số nguyên tố đầu tiên.

Tổ chức chương trình: Cần cài đặt 2 hàm sau:

- int KiemTra_NT(int n): để kiểm tra n có phải là số nguyên tố?
- void LietKeSoNT(unsigned int n): liệt kê n số nguyên tố đầu tiên.

Bước 27. Tạo project **Win32 Console Application** mới, đặt tên là **Lab03_Bai06_SoNguyenTo**, lưu trong thư mục **MSSV_Lab03_D_HD**. Trong project này tạo tập tin **program.cpp**

Bước 28. Mở tập tin **program.cpp** và tạo cấu trúc cho chương trình như **bước 2 mục 1**.

Bước 29. Nạp thư viện hàm **math.h** vào chương trình dùng chỉ thị **#include <math.h>**

Bước 30. Định nghĩa hằng số TAB như sau:

```

// =====
// Định nghĩa hằng số và các kiểu dữ liệu mới
// =====

#define TAB '\t'

```

Bước 31. Định nghĩa hàm kiểm tra số nguyên tố :

// Định nghĩa hàm kiểm tra số n có phải là số nguyên tố

// Input :

// n : Số cần kiểm tra có phải là số nguyên tố

// Output: kq

// kq = 1 : n là số nguyên tố

// kq = 0 : n không phải là số nguyên tố

int KiemTra_NT(int n)

{

 int kq, m, i;

 if (n < 2) // Những số nhỏ hơn 2 đều

 kq = 0; // không phải là số nguyên tố

 else

 {

 m = (int)sqrt((float)n); // Lưu căn bậc 2 của n

 i = 2;

 // Lưu chỉ số để kiểm tra

```

kq = 1; // Lưu kết quả kiểm tra

// Duyệt qua từng giá trị i từ 2 -> m để kiểm tra
// n có chia hết cho i không? Nếu có, gán kq = 0.
while (i <= m && kq)
{
    if (n % i == 0)
        kq = 0;
    i++;
}
return kq;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 32. Tiếp tục định nghĩa hàm tìm và liệt kê n số nguyên tố đầu tiên:

// Định nghĩa hàm tìm n số nguyên tố đầu tiên

// Input :

// n : Số lượng số nguyên tố

// Output:

// Không có. Chỉ xuất ra n số nguyên tố đầu tiên.

void LietKeSoNT(unsigned int n)

```
{
    int dem = 0, so = 2;
```

// Trong khi chưa tìm đủ n số nguyên tố thì

while (dem < n)

{ // Kiểm tra so có phải là số nguyên tố?

if (KiemTra_NT(so))

{

// Nếu đúng, xuất số đó và tăng biến đếm

cout << so << TAB;

dem++;

}

// Tăng so lên 1 đơn vị để kiểm tra số tiếp theo

so++;

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 33. Khai báo nguyên mẫu cho các hàm trên

```

// =====
// Khai báo nguyên mẫu của các hàm
// =====

```

```

int KiemTra_NT(int n);
void LietKeSoNT(unsigned int n);

```

Bước 34. Cập nhật lại hàm main :

```

int main()
{
    // Khai báo biến lưu số lượng số nguyên tố cần tìm
    unsigned int n;

    // Nhập giá trị cho biến n

```

```

cout << endl << "Nhập số lượng số nguyên tố : ";
cin >> n;

// Gọi hàm liệt kê số nguyên tố
LietKeSoNT(n);

// Dừng chương trình và chờ nhấn 1 phím để kết thúc
_getch();

return 1;
}

```

Bước 35. Nhấn Ctrl+ **F5** để chạy chương trình. Ghi nhận kết quả.

Bước 36. Hãy viết lại hàm KiemTra_NT bằng cách dùng lệnh **for**, **do..while** thay cho lệnh **while**.

Bước 37. Hãy viết lại hàm LietKeSoNT bằng cách dùng lệnh **do..while** thay cho lệnh **while**.

E. Bài tập bắt buộc

Tất cả các bài tập sau phải được viết bằng hàm, sau đó gọi chúng trong hàm main để xem kết quả. **Tên hàm, tên biến và tên tham số phải được đặt tên đúng theo quy ước. Chương trình phải được tổ chức theo đúng mẫu trong bước 2 mục 1 phần D.**

1. Phương trình bậc nhất 1 ẩn

Viết chương trình giải phương trình bậc nhất một ẩn có dạng $ax + b = 0$.

Hướng dẫn:

- Định nghĩa hàm **void GiaiPTBacNhat(double a, double b)**.
- Xét các trường hợp: TH1: $a=b=0$, TH2: $a=0, b \neq 0$ và TH3: $a \neq 0$.

2. Phép toán số học

Viết chương trình cho phép người dùng nhập vào hai số thực x, y khác 0 và ký tự k (thuộc 1 trong 4 ký tự: $+, -, *, /$). Tùy thuộc vào ký tự k , hãy xuất ra tổng, hiệu, tích, thương của x và y .

Hướng dẫn:

- Định nghĩa hàm **double TinhBieuThuc(double x, double y, char k)**.
- Dùng lệnh **switch** để kiểm tra ký tự k .
- Nếu $k='+'$ thì return $x + y$. Tương tự cho các trường hợp còn lại.
- Nếu $k \notin \{+, -, *, /\}$ thì return 0.

3. Đổi cơ số

Viết chương trình cho phép người dùng nhập vào hai số nguyên dương n và b ($2 \leq b \leq 16$). Sau đó, xuất ra màn hình số n ở hệ cơ số b . Trường hợp $10 < b \leq 16$, sử dụng các ký tự A đến F để biểu diễn các số từ 10 tới 15.

Hướng dẫn:

- Định nghĩa hàm **void Xuat(int so)** để xuất 1 chữ số (hoặc chữ cái nếu $so > 9$) ra màn hình.
- Định nghĩa hàm **int TimLuyThua(int b, int n)** để tìm số nguyên lớn nhất nhỏ hơn hoặc bằng n và là lũy thừa của b . ($= \text{Max}\{v \in N : v \leq n \text{ & } v = b^i, i \in N\}$)
- Định nghĩa hàm **void DoiCoSo(int n, int b)** để thực hiện việc đổi cơ số.
 - B1. Gán $v = \text{TimLuyThua}(b, n)$.
 - B2. Kiểm tra $n < v$ hay $n \geq v$.

- Nếu $n < v$, gọi hàm $Xuat(0)$;
- Nếu $n \geq v$, gán : $so = n / v$ rồi gọi hàm $Xuat(so)$. Cập nhật $n = n - so * v$.
- B3. Gán $v = v / b$. Nếu $v > 0$ thì quay lại B2. Ngược lại thì dừng thuật toán.

Cài đặt :

```
void DoiCoSo(int n, int b)
```

```
{
    int v, so;
    v = TimLuyThua(b, n);
    while (v > 0)
    {
        if (n < v)
            Xuat(0);
        else
        {
            so = n / v;
            Xuat(so);
            n = n - so * v;
        }
        v = v / b;
    }
}
```

Ví dụ: Cho $n = 12609$, $b = 8$, ta tính được $v = 4096 = 8^4$. Các bước đổi được tóm tắt trong bảng sau.

n	V	n >= v	so = n/v	Xuat	n mới = n - so * v	v mới = v / b
12609	4096	Đúng	3	3	$12609 - 3 * 4096 = 321$	$4096 / 8 = 512$
321	512	Sai		0		$512 / 8 = 64$
321	64	Đúng	5	5	$321 - 5 * 64 = 1$	$64 / 8 = 8$
1	8	Sai		0		$8 / 8 = 1$
1	1	Đúng	1	1		$1 / 8 = 0 \Rightarrow$ dừng

Kết quả: $12609_{10} = 30501_8$

4. Chỉ số khối cơ thể

Viết chương trình cho phép người dùng nhập vào khối lượng và chiều cao của họ. Sau đó, đưa ra lời khuyên cho người dùng dựa vào chỉ số khối cơ thể - **BMI** (Xem lại bài tập 6 Lab 2 để biết cách tính). Biết rằng, chỉ số **BMI** được phân loại như sau:

- Đói khát: $BMI < 15$
- Biếng ăn: $BMI < 17.5$
- Thiếu cân: $BMI < 18.5$
- Lý tưởng: $18.5 \leq BMI < 25$
- Thừa cân: $25 \leq BMI < 30$
- Béo phì: $30 \leq BMI < 40$
- Trẻ em bị béo phì: $BMI \geq 40$

5. Dãy số

Viết chương trình tính các tổng sau: (Yêu cầu sử dụng cả 3 dạng lặp **for**, **while** và **do..while**)

a. $H_N = \sum_{i=1}^n \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ (còn gọi là các số Harmonic)

b. $S_n = \sum_{i=1}^n \frac{i+1}{i^2} = 2 + \frac{3}{4} + \frac{4}{9} + \dots + \frac{n+1}{n^2}$

6. Tìm nghiệm

Viết chương trình tìm tất cả các nghiệm của phương trình:

$$3a - 2b + 4c + 5d - e = 30 \text{ biết } 0 \leq a, b, c, d, e \leq 10.$$

7. Căn bậc 2

Viết chương trình tính căn bậc 2 của một số thực dương a mà không dùng hàm **sqrt**.

Hướng dẫn:

Dãy

$$\begin{cases} x_0 = 1; \\ x_n = \left(\frac{a}{x_{n-1}} + x_{n-1} \right) / 2; n > 1 \end{cases}; \quad x_n \rightarrow \sqrt{a}$$

- $x_n = 1$
- **Lặp:**
 - $x_0 = x_n$
 - $x_n = (a/x_0 + x_0)/2$
- Trong khi ($|x_n - x_0| > Eps$); $Eps = 10^{-15}$
- **return** $x_n;$

```
double CanBacHai(double a)
{
    double xo, xn;//ket qua
    if (a == 0) xn = 0;
    else {
        xn = 1;
        do {
            xo = xn;
            xn = (a / xo + xo) / 2.0;
        } while (myabs(xn - xo) >= e);
    }
    return xn;
}
```

8. Ước chung – bội chung

Viết chương trình cho phép người dùng nhập vào hai số nguyên dương m và n . Sau đó:

- Xuất ra màn hình các ước chung của m và n . **Ví dụ: 6 và 8 có các ước chung 1, 2.**
- Tìm ước chung lớn nhất của m và n . **Ví dụ: 12 và 16 có ước chung lớn nhất là 4.**
- Tìm bội chung nhỏ nhất của m và n . **Ví dụ: 12 và 16 có bội chung nhỏ nhất là 48.**

Hướng dẫn:

- Định nghĩa hàm **int TimUCLN(int m, int n)** để tìm ước chung lớn nhất theo thuật toán Euclid
- Định nghĩa hàm **int TimBCNN(int m, int n)** để tìm bội chung nhỏ nhất của m và n . $BCNN(m, n) = m * n / UCLN(m, n)$.

9. Trò chơi đoán số

Máy tính “nghĩ ra” một số nguyên (bằng cách sinh ngẫu nhiên) và không hiển thị số đó. Người chơi đoán số đó bằng cách nhập từ bàn phím.

- Nếu số mà người chơi nhập nhỏ hơn số máy nghĩ ra, máy tính sẽ xuất một thông báo “Số cần đoán lớn hơn”.
- Nếu số mà người chơi nhập lớn hơn, máy tính sẽ xuất thông báo “Số cần đoán nhỏ hơn”.
- Nếu số mà người chơi nhập bằng với số máy sinh ra, thông báo người chơi thắng.

Người chơi chỉ có thể đoán tối đa k lần. Nếu sau cả k lần mà người chơi đoán không trúng thì máy tính thông báo người chơi thua cuộc.

Viết chương trình minh họa trò chơi trên.

Hướng dẫn:

- Định nghĩa hàm **int SinhSoNgauNhien()** để tạo số máy “nghĩ ra : soDe”.

- Sử dụng hàm `rand((unsigned int)time(NULL))` để gieo số ngẫu nhiên đầu tiên (trong `<time.h>`)
- Sử dụng hàm `rand()` để sinh số ngẫu nhiên (trong `<stdlib.h>`)
- Định nghĩa hàm `int XuLySoDoan(int soDe, int k)` để minh họa các bước đoán số, với `soDe`: là đè(số máy tính nghĩ ra), `k`: số lần đoán
(Hàm trả về 1 nếu đoán đúng, trả về 0 nếu cả `k` lần đều đoán sai. Mỗi lần số đoán sai, hàm thông báo có hướng dẫn theo yêu cầu bài toán)
- Viết hàm `void ThongBao_kqtc(int soDe, int kq)` để thông báo kết quả thắng, thua của người chơi.
- Có thể viết thêm hàm chọn lựa mìrc chơi (trả về `k`: số lần đoán)

F. Bài tập làm thêm

1. Số Fibonacci

Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1, các phần tử sau đó được thiết lập theo quy tắc mỗi phần tử luôn bằng tổng hai phần tử trước nó cộng lại. Công thức truy hồi của dãy Fibonacci là:

$$F(n) = \begin{cases} 0 & \text{khi } n = 0 \\ 1 & \text{khi } n = 1 \\ F(n - 1) + F(n - 2) & \text{khi } n \geq 2 \end{cases}$$

Viết chương trình cho phép người dùng nhập vào một số nguyên dương n . Sau đó:

- d. Tìm và xuất ra số Fibonacci thứ n .
- e. Liệt kê các số Fibonacci nhỏ hơn hoặc bằng n .
- f. Liệt kê n số Fibonacci đầu tiên.

2. Hàm lượng giác – hàm số mũ

Viết chương trình tính giá trị các hàm $\sin(x)$, $\cos(x)$, e^x với giá trị x được nhập từ bàn phím. Yêu cầu: không được sử dụng các hàm \sin , \cos trong thư viện `math.h`.

Hướng dẫn: Sử dụng các công thức khai triển Taylor sau với độ chính xác 10^{-5}

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots, -\infty < x < \infty$$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \dots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} + \dots$$

3. Trò chơi bốc kẹo

Có một chiếc hộp đựng 100 cái kẹo. Hai em bé A và B bày ra một trò chơi. Mỗi người lần lượt bốc một số viên kẹo. Mỗi lần bốc tối thiểu 1 viên và tối đa là 4 viên. Người nào bốc sau cùng sẽ là người thắng cuộc.

Hãy viết chương trình một chương trình minh họa trò chơi trên. Trong đó, máy tính đóng vai trò người chơi A và bạn là người B. Máy tính được bốc trước. Khi kẹo trong hộp hết, chương trình thông báo người thắng cuộc.

Cũng với trò chơi trên, bạn hãy viết một chương trình khác. Trong đó, máy tính đóng vai trò người chơi A và được đi sau. Bạn là người chơi B và được bốc trước. Hãy lập chiến lược để máy tính luôn thắng.

4. Trò chơi “Let’s Make a Deal” (1970)

Người chơi tham gia Let’s Make a Deal sẽ có cơ hội nhận một món quà rất có giá trị. Người chơi sẽ đứng trước 3 ô cửa bí mật. Đầu sau một trong số ô cửa đó là một giải thưởng có giá trị lớn và sau hai ô cửa kia là hai món quà bất ngờ. Sau khi người chơi quyết định chọn một ô cửa, người dẫn chương trình sẽ mở một trong 2 ô cửa còn lại (dù nhiên là không mở ô cửa chứa giải thưởng). Sau đó, người chơi sẽ có một cơ hội để thay đổi quyết định của mình: đổi sang ô cửa khác chưa mở. Bằng trực giác, có vẻ như là khả năng chừa giải thưởng trong 2 ô cửa là như nhau. Vì thế, người chơi thường không có động lực để thay đổi quyết định của mình.

Hãy viết một chương trình giả lập trò chơi và kiểm tra lại trực giác này. Chương trình cho phép nhập vào số N (là số lần chơi hay số người chơi). Sau đó in ra khả năng chiến thắng (số lần lấy được giải thưởng / N) của người chơi trong 2 trường hợp:

- Cả N lần chơi đều không thay đổi quyết định.
- Cả N lần chơi đều thay đổi quyết định.

5. Tài khoản tiết kiệm

Anh Bình có một số tiền N và muốn gửi vào ngân hàng ABC trong thời hạn T tháng với lãi suất hàng tháng là $L\%$. Tiền lãi được tích lũy dần vào tài khoản vào cuối mỗi tháng. Giả sử lãi suất không thay đổi và anh Bình không rút tiền trong suốt thời gian gửi. Hết thời hạn gửi, số tiền mà anh Bình nhận được là bao nhiêu?

LAB 4. TỔ CHỨC THƯ VIỆN HÀM VÀ MENU

THỜI LƯỢNG: 6 TIẾT

A. Mục tiêu

- Giúp sinh viên làm quen với việc tổ chức chương trình theo các thư viện hàm (*.h).
- Hướng dẫn sinh viên cách tạo menu để người dùng chọn chức năng của chương trình.
- Tiếp tục rèn luyện kỹ năng cài đặt chương trình theo cách từng bước bổ sung các chức năng vào chương trình và viết chương trình có cấu trúc hình thức.
- Sau khi hoàn thành bài thực hành này, sinh viên cần phải:
 - Tạo được dự án dạng Win32 Console Application có sử dụng các tập tin header (*.h).
 - Sử dụng thuận thực các cấu trúc điều khiển, định nghĩa kiểu dữ liệu, biến, hằng,...
 - Sử dụng được hàm.
 - Hiểu rõ truyền tham số bằng trị
 - Biết cách tổ chức chương trình theo thư viện hàm, hệ thống menu.

B. Yêu cầu

- Trong phần phần C (hướng dẫn thực hành), sinh viên tự đọc mục 1, mục 2 (hướng dẫn tổ chức chương trình).

Buổi thực tập thứ 4:

- 2 tiết cuối:

- Sinh viên thực tập trong phòng lab :

- Khối lượng và nội dung : Các bài 1,2,3 trong phần C (Hướng dẫn thực hành) lab 4.
 - Yêu cầu lưu trữ :
 - Tạo thư mục MaSV_Lab04_HD chứa trong ổ đĩa qui định.
 - Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - Mỗi project, xóa thư mục debug.
 - Sử dụng lại tập tin word LoiChuongTrinh.docx từ lab3 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - Giáo viên sẽ thu bài qua hệ thống thu bài trực tuyến tại phòng lab (thu thư mục MaSV_Lab04_HD) vào cuối buổi thực tập.

Buổi thực tập thứ 5 (4 tiết)

- Sinh viên thực tập trong phòng lab :

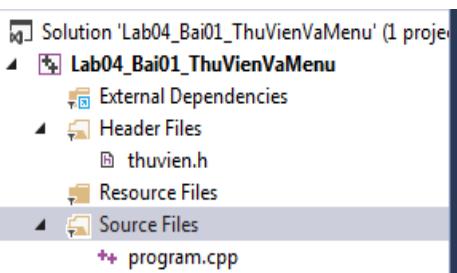
- Khối lượng và nội dung : Tất cả các bài trong phần D (Bắt buộc) lab 4.
 - Yêu cầu lưu trữ :
 - Tạo thư mục MaSV_Lab04_BB chứa trong ổ đĩa qui định.
 - Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - Mỗi project, xóa thư mục debug.
 - Sử dụng lại tập tin word LoiChuongTrinh.docx trong buổi thực tập thứ 4 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - Thu bài : Xem thông báo của giáo viên .

C. Hướng dẫn thực hành

1. Tạo cấu trúc cho chương trình có sử dụng thư viện hàm do người dùng định nghĩa

Phần này hướng dẫn cách tạo một dự án có sử dụng tập tin tiêu đề hay tập tin thư viện hàm (*.h). Các tập tin này được chứa trong thư mục Header Files của dự án. Hình bên cho thấy cấu trúc của dự án mà ta sẽ tạo.

- Tập tin **program.cpp** chứa hàm main, và hàm ChayChuongTrinh(). Hàm main() là nơi khởi đầu của chương trình, sẽ gọi hàm ChayChuongTrinh() để thực hiện chương trình, còn hàm ChayChuongtrinh() thực hiện các chức năng của chương trình.



- Tập tin **thuvien.h** sẽ chứa phần định nghĩa kiểu dữ liệu mới, hằng số và các hàm xử lý (tùy thuộc bài toán).

Từ đây trở đi, mỗi khi xây dựng dự án mới (có tổ chức thư viện, không có hệ thống menu tùy chọn), cần tạo cấu trúc cho chương trình như hình trên.

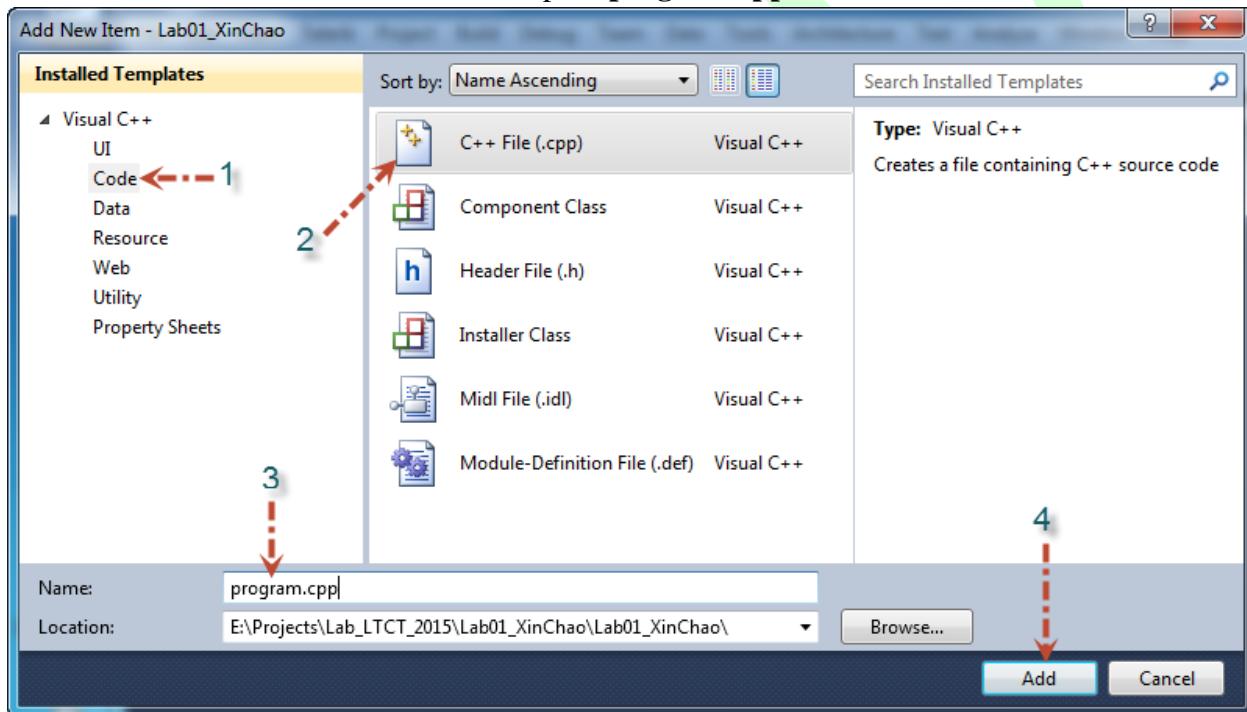
Theo cách tổ chức này, tập tin **program.cpp** và hàm main rất ít khi thay đổi mã lệnh.

Bước 1. Tạo dự án Win32 Console Application mới. Đặt tên là **Lab04_C_Muc1_ThuVien**.

Bước 2. Sử dụng 1 trong 3 cách sau để mở cửa sổ **Add New Item**

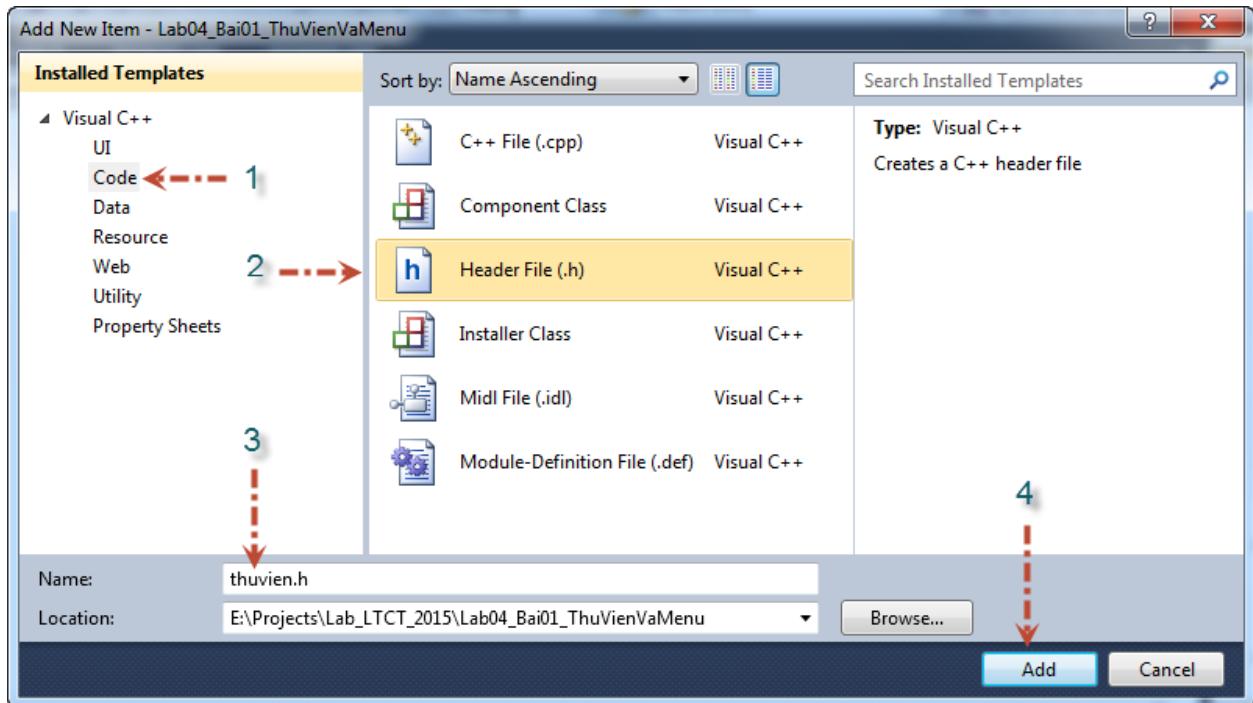
- Chọn thư mục **Source Files**, nhấn tổ hợp phím Ctrl + Shift + A
- Nhấp phải thư mục **Source Files** trong dự án, chọn Add > New Item ...
- Chọn thư mục **Source Files**, sau đó chọn menu Project > Add New Item...

Bước 3. Trong cửa sổ **Add New Item**, chọn mục **Code > C++ File (.cpp)**, đặt tên là **program.cpp** trong mục Name. Nhấn nút Add để thêm tập tin **program.cpp** vào dự án.



Bước 4. Thực hiện như bước 2 để mở cửa sổ Add New Item nhưng chọn thư mục **Header Files**.

Bước 5. Trong cửa sổ **Add New Item**, chọn mục **Code > Header File (.h)**, đặt tên là **thuvien.h** trong mục Name. Nhấn nút Add để thêm tập tin **thuvien.h** vào dự án.



Các bước tiếp theo sẽ soạn thảo nội dung cho các tập tin **Program.cpp**, **thuvien.h**; Lưu ý là khi soạn thảo nội dung chương trình, về mặt hình thức văn bản phải có cấu trúc : các câu lệnh không phụ thuộc nhau sẽ viết trên cùng một cột, nếu câu lệnh sau phụ thuộc câu lệnh trước thì dịch câu lệnh sau vào một tab,... Khi soạn nội dung cho tập tin, câu lệnh đầu tiên sẽ bắt đầu từ hàng đầu, cột đầu.

Bước 6. Trong tập tin **Program.cpp**, nhập đoạn code sau (phần tối thiểu để có thể chạy được chương trình)

```
//Chen cac tap tin thu vien co san
#include <iostream>
#include <conio.h>
using namespace std;
//Chen cac tap tin thu vien tu tao : luu y trinh tu chen.
#include "Thuvien.h"
//khai bao nguyen mau ham
void ChayChuongTrinh();
int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Sau đây, ta từng bước bổ sung chức năng để hoàn thiện chương trình, mỗi lần bổ sung xong ta chạy kiểm tra chương trình để sửa lỗi.

Bước 7. Trong tập tin **thuvien.h**, nhập nội dung như sau:

```
1 // =====
2 // Định nghĩa các hằng số và kiểu dữ liệu mới
3 // =====
4
5
6 // =====
7 // Khai báo nguyên mẫu các hàm xử lý
8 // =====
9
10 // =====
11 // Định nghĩa các hàm xử lý
12 // =====
13
14
15 ~
```

Nội dung tập tin thu viện.h sẽ bao gồm các phần : định nghĩa hằng, kiểu dữ liệu mới, định nghĩa các hàm chức năng của chương trình.

Mỗi lần bổ sung đoạn lệnh (định nghĩa hằng, kiểu dữ liệu mới, hàm xử lý,...) ta đều phải kiểm tra chương trình để sửa lỗi nếu có.

Bước 8. Trong tập tin **program.cpp**, ta sửa lại hàm **ChayChuongTrinh()** để thực hiện các công việc của chương trình.

Bước 9. Biên dịch và chạy chương trình để xem kết quả.

Hãy lưu ý việc viết chương trình theo cách hoàn thiện dần từng bước (tại mỗi bước đều có chạy kiểm tra chương trình để sửa lỗi nếu có).

Bài 1 : Trò chơi đoán số

Viết chương trình thực hiện trò chơi đoán số dưới dạng tổ chức thư viện.

Gợi ý tổ chức chương trình như sau :

- Viết hàm cho người dùng chọn mức số lần đoán.
- Viết hàm sinh số ngẫu nhiên (Đè : Số do máy tính nghĩ ra làm đè)
- Viết hàm xử lý trò chơi : Kết thúc trò chơi hàm trả về 1 (người chơi thắng) hoặc 0 (người chơi thua). Tại mỗi lần đoán sai, chương trình có thông báo hướng dẫn theo yêu cầu bài toán.
- Viết hàm thông báo kết quả thắng thua (lưu ý rằng cần phải xuất số đè ra màn hình).
- Viết hàm ChayChuongTrinh() điều khiển thực hiện trò chơi.
- Có thể điều khiển chương trình cho phép chơi nhiều lần cho đến khi người dùng không muốn chơi nữa thì dừng.

Bước 11. Tạo dự án Win32 Console Application mới. Đặt tên là :

Lab04_C_Bai01_TroChoiDoanSo.

Bước 12. Tạo cấu trúc chương trình như đã hướng dẫn trong **mục 1** (từ bước 1 đến bước 7).

Bước 13. Trong tập tin **thuvien.h**, từng bước bổ sung cài đặt các hàm xử lý :

//Định nghĩa hằng

```
#define MAX 1000 //giới hạn số dương ngẫu nhiên máy tính nghĩ ra < MAX.
```

```
//Định nghĩa kiểu dữ liệu mới – không có
```

```
//Khai báo nguyên mẫu các hàm xử lý
```

```
//... (bổ sung sau)
```

/Định nghĩa các hàm xử lý

3.1 Hàm chọn số lần đoán (Một lần chơi được phép đoán bao nhiêu lần)

```
//Input : không có
```

```
//Output : k = số lần đoán ( giả sử có 3 mức tương ứng với số lần đoán là : 3,5,7)
```

```
int ChonMucDoan()
```

```
{
```

```
    int k;
```

```
    do
```

```
    {
```

```
        cout << "\nChon so lan doan toi da ( 3,5,7 ) : k = ";  
        cin >> k;
```

```
    } while (k != 3 && k != 5 && k != 7);
```

```
    return k;
```

```
}
```

- Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Hàm sinh số ngẫu nhiên

```
//Input : không có
```

```
//Output : soDe = một số dương ngẫu nhiên < MAX
```

```
int SinhSoNgauNhien()
```

```
{
```

```
    int soDe;
```

```
//gio so ngau nhien
```

```
srand((unsigned int)time(0)); //khai bao <time.h>
```

```
//tao so ngau nhien
```

```
soDe = rand() % MAX; //khai bao <stdlib.h>
```

```
return soDe;
```

```
}
```

- Trong tập tin **Program.cpp**, khai báo bổ sung các thư viện :

```
#include <time.h>
```

```
#include <stdlib.h>
```

- Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3. Viết hàm xử lý trò chơi

```

//Input: k = số lần đoán
//      soDe = đề do máy tính nghĩ ra
//Output : 1 (người chơi thắng)
//          0 (người chơi thua)

int XuLyTroChoi(int k, int soDe)
{
    int i,
        soDoan,
        kq = 0;
    for (i = 1; i <= k; i++)
    {
        cout << "\nDoan lan " << i << ", so doan = ";
        cin >> soDoan;
        if (soDoan == soDe)
        {
            kq = 1;
            break;
        }
        else
        if (soDoan > soDe)
            cout << "\nSo doan lon hon";
        else
            cout << "\nSo doan nho hon";
    }
    return kq;
}

```

- Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Viết hàm thông báo kết quả trò chơi

```

//Input : kq,
//      soDe;
//Output : không có

```

```

void ThongBaoKetQua(int kq, int soDe)
{
    system("CLS");
    cout << "\nKET QUA TRO CHOI : ";
    if (kq)
        cout << "\nNguoi choi thang";
    else
        cout << "\nNguoi choi thua";
    cout << "\nDe cho so : " << soDe;
}

```

- Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.5. Bổ sung khai báo nguyên mẫu các hàm (tại vùng khai báo nguyên mẫu hàm):

```
int ChonMucDoan();
```

```
int SinhSoNgauNhien();
int XuLyTroChoi(int k, int soDe);
void ThongBaoKetQua(int kq, int soDe);
```

- Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 6: Trong tập tin **Program.cpp**, cập nhật lại hàm **ChayChuongTrinh()** như sau để hoàn thiện chương trình.

- Hàm **ChayChuongTrinh()** sẽ điều khiển lặp lại việc chơi trò chơi.

```
void ChayChuongTrinh()
{
    char kt;
    int kq,k, soDe;
    do
    {
        system("CLS");
        soDe = SinhSoNgauNhien();
        k = ChonMucDoan();
        kq = XuLyTroChoi(k, soDe);
        system("CLS");
        cout << "\nTRO CHOI DOAN SO VOI SO LAN DOAN : k = " << k << " :\n";
        ThongBaoKetQua(kq, soDe);
        _getch();
        system("CLS");
        cout << "\nChoi nua khong, nhan ESC neu khong!\n";
        kt = _getch();
    } while (kt != 27);
}
```

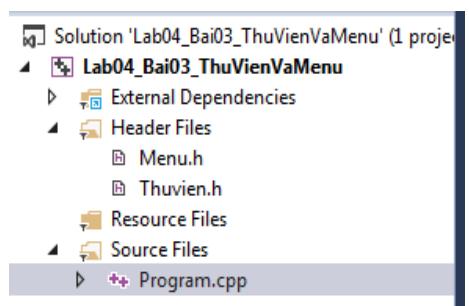
- Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

- Thực hiện trò chơi, kiểm tra kết quả có phù hợp hay không ?

2. Tạo cấu trúc cho chương trình có sử dụng thư viện hàm do người dùng định nghĩa, và có tổ chức hệ thống menu :

Phần này hướng dẫn cách tạo một dự án tổ chức theo thư viện hàm (*.h) và hệ thống menu tùy chọn. Các tập tin này được chứa trong thư mục Header Files của dự án. Hình bên cho thấy cấu trúc của dự án mà ta sẽ tạo.

- Tập tin **program.cpp** chứa hàm main, và hàm **ChayChuongTrinh()**. Hàm main() là nơi khởi đầu của chương trình, sẽ gọi hàm ChayChuongTrinh() để thực hiện chương trình, còn hàm ChayChuongTrinh() điều khiển và thực hiện các chức năng của chương trình.
- Tập tin **thuvien.h** sẽ chứa phần định nghĩa kiểu dữ liệu mới, hằng số và các hàm xử lý (tùy thuộc bài toán).



- Tập tin **menu.h** chứa các hàm để xử lý menu chức năng do người dùng chọn.

Tùy chỉnh trỏ di, mỗi khi xây dựng dự án mới tổ chức theo thư viện và có hệ thống tùy chọn menu, sẽ tạo cấu trúc cho chương trình như trên.

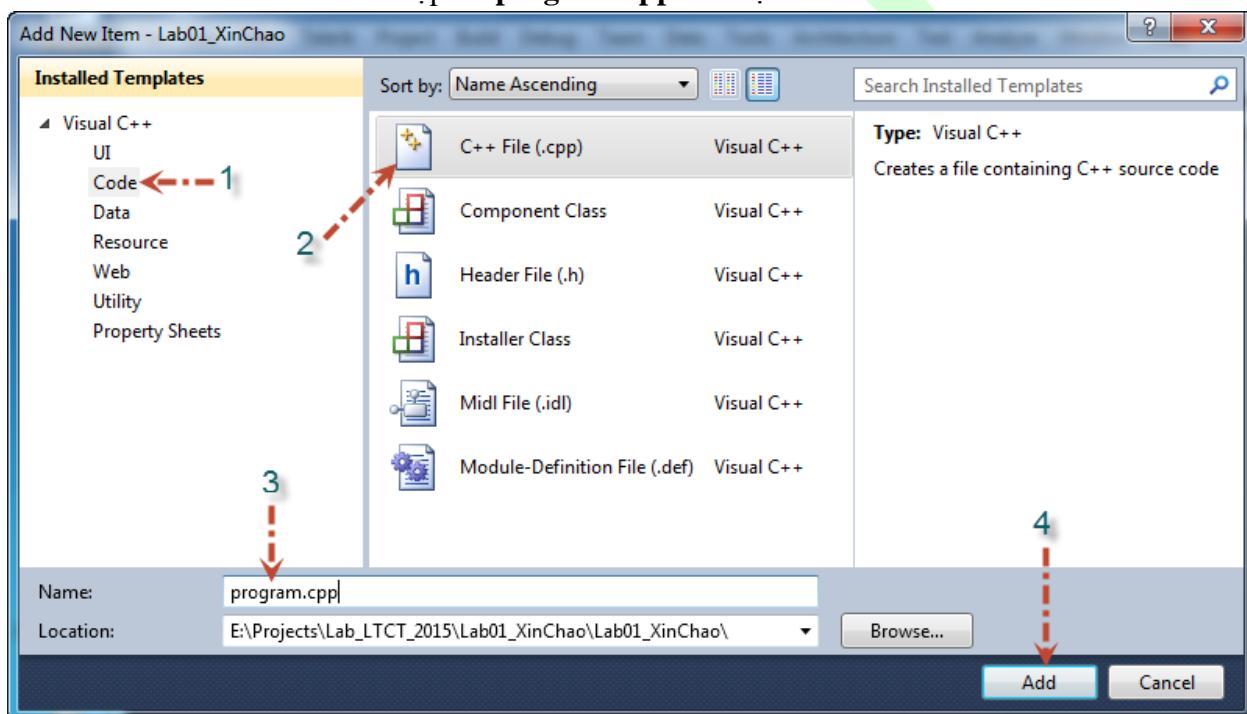
Theo cách tổ chức này, tập tin program.cpp và hàm main rất ít khi thay đổi mã lệnh.

Bước 1: Tạo dự án Win32 Console Application mới. Đặt tên là **Lab04_C_Muc2_ThuVienVaMenu**.

Bước 2: Sử dụng 1 trong 3 cách sau để mở cửa sổ **Add New Item**

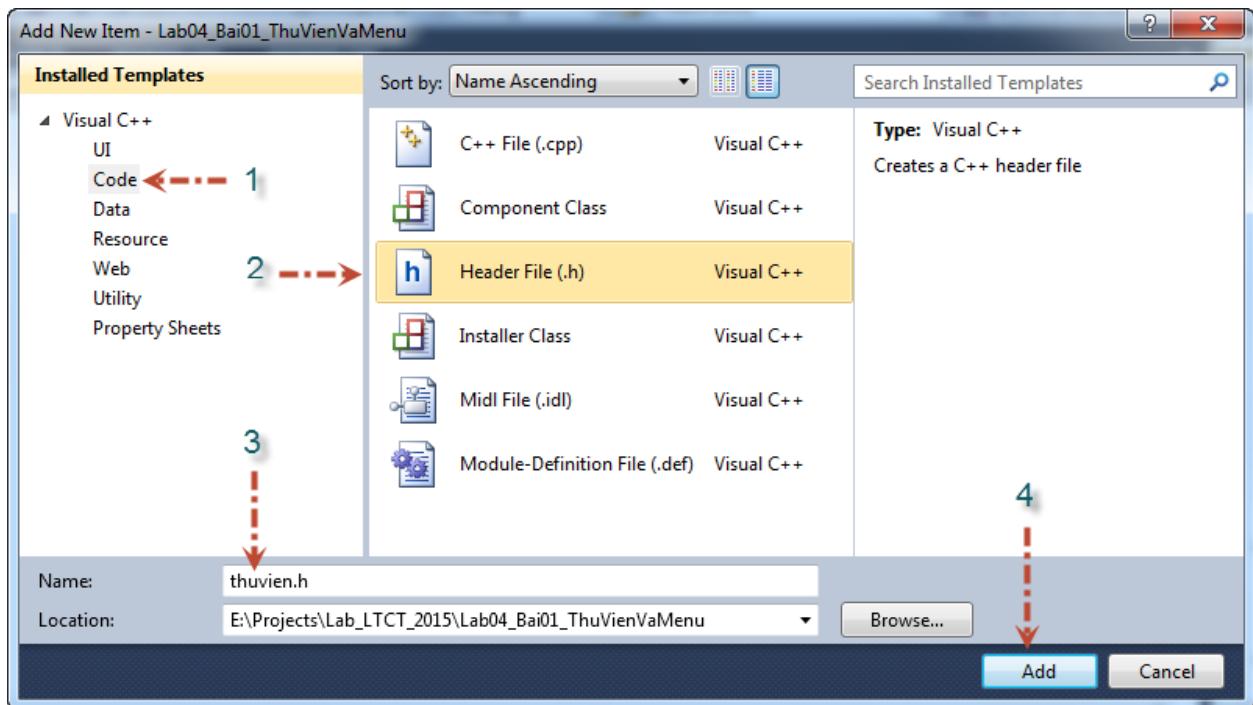
- Chọn thư mục **Source Files**, nhấn tổ hợp phím Ctrl + Shift + A
- Nhấp phải thư mục **Source Files** trong dự án, chọn Add > New Item ...
- Chọn thư mục **Source Files**, sau đó chọn menu Project > Add New Item...

Bước 3: Trong cửa sổ **Add New Item**, chọn mục **Code > C++ File (.cpp)**, đặt tên là **program.cpp** trong mục Name. Nhấn nút Add để thêm tập tin **program.cpp** vào dự án.



Bước 4: Thực hiện như bước 2 để mở cửa sổ Add New Item nhưng chọn thư mục **Header Files**.

Bước 5: Trong cửa sổ **Add New Item**, chọn mục **Code > Header File (.h)**, đặt tên là **thuvien.h** trong mục Name. Nhấn nút Add để thêm tập tin **thuvien.h** vào dự án.



Bước 6: Làm lại bước 4 và 5 ở trên nhưng đặt tên tập tin là **menu.h**.

Bước 7: Trong tập tin **Program.cpp**, nhập đoạn code sau (phần tối thiểu để có thể chạy được chương trình :

```
//Chen cac tap tin thu vien co san
#include <iostream>
#include <conio.h>
using namespace std;

//Chen cac tap tin thu vien tu tao : luu y trinh tu chen.
#include "Thuvien.h"
#include "Menu.h"

//khai bao nguyen mau ham
void ChayChuongTrinh();

int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Sau đây, ta bổ sung từng bước chức năng để hoàn thiện chương trình, mỗi lần bổ sung ta phải chạy kiểm tra chương trình để sửa lỗi.

Bước 8: Trong tập tin **thuvien.h**, nhập nội dung như sau:

```
1 // =====
2 // Định nghĩa các hằng số và kiểu dữ liệu mới
3 // =====
4
5
6 // =====
7 // Khai báo nguyên mẫu các hàm xử lý
8 // =====
9
10
11 // =====
12 // Định nghĩa các hàm xử lý
13 // =====
14
15 ~
```

Bước 9: Trong tập tin **menu.h**, nhập đoạn mã sau:

```
1 //=====
2 //Khai bao nguyen mau cac ham trong he thong menu
3 //=====
4 void XuatMenu();
5 int ChonMenu(int soMenu);
6 void XuLyMenu(int menu);
7
8 //=====
9 //Dinh nghia cac ham trong he thong menu
10 //=====
11 //Ham XuatMenu : Xuat danh sach cac chuc nang ra man hinh:
12 // Gia Su co 4 chuc nang
13 //Input : Khong co
14 //Output : Khong co
15 void XuatMenu()
16 {
17     cout << "\n===== HE THONG CHUC NANG =====";
18     cout << "\n0. THOAT KHOI CHUONG TRINH";
19     cout << "\n1. chuc nang 1";
20     cout << "\n2. chuc nang 2";
21     cout << "\n3. chuc nang 3";
22     cout << "\n4. chuc nang 4";
23     cout << "\n=====";
24 }
25
26 //Ham ChonMenu : Chon mot chuc nang trong danh sach
27 //Input : soMenu = So chuc nang
28 //Output : stt == thu tu chuc nang duoc chon
```

```
29 | int ChonMenu(int soMenu)
30 | {
31 |     int stt; //bien luu so thu tu chuc nang duoc chon
32 |     for (;;)
33 |     {
34 |         system("CLS");
35 |         XuatMenu();
36 |         cout << "\nNhap 1 so stt (1 <= stt <= " << soMenu
37 |             << ") de chon menu : stt = ";
38 |         cin >> stt;
39 |         if (0 <= stt && stt <= soMenu)
40 |             break;
41 |     }
42 |     return stt;
43 | }
44 |
45 | //Ham XuLyMenu : Xu ly chuc nang duoc chon
46 | //Input : menu = So thu tu menu da chon
47 | //Output : khong co
48 | void XuLyMenu(int menu)
49 | {
50 |     switch (menu)
51 |     {
52 |     case 0:
53 |         cout << "\n0. THOAT KHOI CHUONG TRINH.\n";
54 |         break;
55 |     case 1:
56 |         cout << "\n1. Ban da chon chuc nang 1";
57 |         //Xu ly chuc nang 1
58 |         break;
59 |     case 2:
60 |         cout << "\n2. Ban da chon chuc nang 2";
61 |         //Xu ly chuc nang 2
62 |         break;
63 |     case 3:
64 |         cout << "\n3. Ban da chon chuc nang 3";
65 |         //Xu ly chuc nang 3
66 |         break;
67 |     case 4:
68 |         cout << "\n4. Ban da chon chuc nang 4";
69 |         //Xu ly chuc nang 4
70 |         break;
71 |     }
72 |     _getch();
73 | }
```

Bước 10. Trong tập tin **program.cpp**, ta sửa lại hàm **ChayChuongTrinh()** :

```

1 //=====
2 //Nạp các tập tin thư viện hàm xây dựng sẵn
3 //=====
4 #include <iostream>
5 #include <conio.h>
6 using namespace std;
7 #include "Thuvien.h"
8 #include "Menu.h"
9 //Xây dựng thêm hàm Chạy chương trình
10 void ChayChuongTrinh();
11 int main()
12 {
13     ChayChuongTrinh();
14     return 1;
15 }
16 void ChayChuongTrinh()
17 {
18     int menu,
19         soMenu = 4;
20     do
21     {
22         menu = ChonMenu(soMenu);
23         XuLyMenu(menu);
24     } while (menu > 0);
25 }

```

Bước 11. Biên dịch và chạy chương trình để xem kết quả.

Lưu ý quan trọng

- Tùy thuộc vào chương trình, ta cần thay đổi các hàm main, XuatMenu, XuLyMenu.
- Để đổi tên chức năng hoặc thêm các chức năng, viết lại các lệnh cout trong hàm **XuatMenu**.
- Để thay đổi cách xử lý từng menu, viết lại mã lệnh trong hàm **XuLyMenu**.
- Khi số menu thay đổi, gán giá trị mới cho biến soMenu trong hàm **ChayChuongTrinh**.
- Nếu tất cả các chức năng đều sử dụng chung một đầu vào thì có thể bổ sung các tham số tương ứng cho hàm **XuLyMenu**.

Bài 2: Tính diện tích hình học

Bước 1: Tạo dự án Win32 Console Application mới. Đặt tên là **Lab04_C_Bai2_DienTichHinhHoc**.

Bước 2: Tạo cấu trúc chương trình như đã hướng dẫn trong **mục 2** (từ bước 1 đến bước 8).

Bước 3: Trong tập tin menu.h ta viết lại như sau (cấu trúc giống như bước 9 mục 2, chỉ thay đổi nội dung theo yêu cầu bài toán) :

```

// =====
// Khai báo nguyên mẫu các hàm xử lý menu
// =====
//bổ sung sau
// =====
// Định nghĩa các hàm xử lý menu
// =====

```

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình

```
// Input : Không có  
// Output: Không có  
void XuatMenu()  
{  
    cout << endl << "===== CHON CHUC NANG =====";  
    cout << endl << "0. Thoát khỏi chương trình";  
    cout << endl << "1. Tính diện tích hình vuông";  
    cout << endl << "2. Tính diện tích hình chữ nhật";  
    cout << endl << "3. Tính diện tích hình tam giác";  
    cout << endl << "4. Tính diện tích hình tròn";  
    cout << endl << "===== =====";  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách:

Hàm này điều khiển người dùng trong việc chọn chức năng. Người dùng chỉ được phép nhập số thứ tự menu trong khoảng [0,...,soMenu].

```
// Input : soMenu = Số lượng menu có thể chọn.  
// Output: Số thứ tự menu do người dùng nhập vào.
```

```
int ChonMenu(int soMenu)  
{  
    int stt;  
    for (;;) {  
        system("CLS");  
        XuatMenu();  
        cout << "\nNhập 1 số để chọn menu ( 0 <= stt <= " << soMenu << " ) : stt = ";  
        cin >> stt;  
        if (0 <= stt && stt <= soMenu)  
            break;  
    }  
    return stt;  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Định nghĩa hàm xử lý menu tùy thuộc vào chức năng:

```
// Input : menu = Số thứ tự menu do người chọn.  
// Output: Không có.
```

```
void XuLyMenu(int menu)  
{  
    switch (menu)  
    {  
        case 0:
```

```

        cout << endl << "0. Thoat khoi chuong trinh";
        break;
    case 1:
        cout << endl << "1. Tinh dien tich hinh vuong";
        //bo sung sau
        break;
    case 2:
        cout << endl << "2. Tinh dien tich hinh chu nhat";
        //bo sung sau
        break;
    case 3:
        cout << endl << "3. Tinh dien tich hinh tam giac";
        //bo sung sau
        break;
    case 4:
        cout << endl << "4. Tinh dien tich hinh tron";
        //bo sung sau
        break;
}
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```

void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu);

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Trong bước tiếp theo, ta soạn thảo từng phần (chức năng) trong thuvien.h, bổ sung xử lý các chức năng trong hàm *XuLyMenu* của *menu.h*, viết lại hàm *ChayChuongTrinh* trong *program.cpp* để điều khiển tùy chọn thực hiện menu chương trình (chọn 0 là dừng chương trình)

Bước 4 :

4.1 Trong tập tin program.cpp, ta cập nhật lại hàm ChayChuongTrinh :

```

// Định nghĩa hàm lặp lại việc chọn menu, xử lý menu
// cho tới khi người dùng chọn menu 0 thì thoát CT.
void ChayChuongTrinh()
{
    // Khai báo biến
    int menu,           // lưu số thứ tự menu được chọn
        soMenu = 4;    // lưu số lượng menu chức năng.

    // Lặp lại việc chọn và xử lý menu cho tới khi
    // người dùng chọn chức năng 0. Thoát khỏi CT.
    do
    {

```

```

        menu = ChonMenu(soMenu);
        XuLyMenu(menu);
    } while (menu > 0);
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra thực hiện chức năng chọn 0 thì dừng chương trình, chọn từ 1 đến 4 thì không làm gì cả.

Bước 5:

5.1 Trong **thuvien.h**, bổ sung:

- Trong phần định nghĩa nghĩa các hàm xử lý, ta định nghĩa hàm tính diện tích hình vuông.

// Định nghĩa hàm tính diện tích hình vuông

// Input : canh = Độ dài cạnh hình vuông.

// Output: Diện tích hình vuông

double DienTichHinhVuong(double canh)

{

return canh * canh;

}

- Trong phần khai báo nguyên mẫu hàm, bổ sung khai báo nguyên mẫu hàm tính diện tích hình vuông :

5.2 Trong tập tin **menu.h** : ta bổ sung xử lý thực hiện tính diện tích hình vuông trong case 1 của hàm XuLyMenu. Trong hàm này ta khai báo thêm các biến thực a (để lưu trữ giá trị cạnh), dienTich để lưu trữ giá trị diện tích hình vuông :

```

void XuLyMenu(int menu)
{
    // Khai báo các biến
    double a, dienTich;
    switch (menu)
    {
        case 0:
            cout << endl << "0. Thoat khoi chuong trinh";
            break;
        case 1:
            cout << endl << "1. Tinh dien tich hinh vuong";
            // Thông báo người dùng nhập độ dài cạnh
            cout << endl << "Nhap do dai canh hinh vuong : ";
            cin >> a;

            // Gọi hàm tính diện tích hình vuông
            dienTich = DienTichHinhVuong(a);

            // Xuất kết quả
            cout << endl << "Dien tich hinh vuong canh "
                << a << " la " << dienTich;
            break;
        case 2:
            cout << endl << "2. Tinh dien tich hinh chu nhat";
    }
}

```

```
//bổ sung sau  
break;  
case 3:  
    cout << endl << "3. Tinh dien tich hinh tam giac";  
    //bổ sung sau  
    break;  
case 4:  
    cout << endl << "4. Tinh dien tich hinh tron";  
    //bổ sung sau  
    break;  
}  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra chương trình việc thực hiện chức năng 1

Tiếp theo bổ sung các chức năng khác.

Bước 6 :

6..1 Trong **thuvien.h**, bổ sung :

- Trong phần định nghĩa nghĩa các hàm xử lý, ta định nghĩa hàm tính diện tích hình chữ nhật :

```
// Định nghĩa hàm tính diện tích hình chữ nhật  
// Input : dai = Kích thước chiều dài của HCN.  
//          rong = Kích thước chiều rộng của HCN  
// Output: Diện tích hình chữ nhật  
double DienTichHinhChuNhat(double dai, double rong)  
{  
    return dai * rong;  
}
```

- Tiếp tục bổ sung khai báo nguyên mẫu hàm tính diện tích hình chữ nhật :

```
double DienTichHinhChuNhat(double dai, double rong);
```

6.2 Trong tập tin **menu.h** :

ta bổ sung xử lý thực hiện tính diện tích hình chữ nhật trong case 2 của hàm XuLyMenu. Trong hàm này ta khai báo thêm một biến thực b (đã có a và dienTich) để lưu trữ chiều rộng.

```
void XuLyMenu(int menu)  
{  
    // Khai báo các biến  
    double a, b, dienTich;  
    switch (menu)  
    {  
        // như 5.2  
        case 2:  
            cout << endl << "2. Tinh dien tich hinh chu nhat";  
    }
```

```

// Thông báo người dùng nhập chiều dài & rộng
cout << endl << "Nhập chiều dài hình CN : ";
cin >> a;

cout << endl << "Nhập chiều rộng hình CN : ";
cin >> b;

// Gọi hàm tính diện tích hình chữ nhật
dienTich = DienTichHinhChuNhat(a, b);

// Xuất kết quả
cout << endl << "Diện tích hình chữ nhật "
    << " có chiều dài = " << a
    << " chiều rộng = " << b << " là " << dienTich;
break;
// như 5.2
}
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có và kiểm tra việc thực hiện chức năng 2.

Bước 7 :

7.1 Trong *thuvien.h*,

Tiếp tục bổ sung :

- Trong phần định nghĩa nghĩa các hàm xử lý, ta định nghĩa hàm tính diện tích hình tam giác :

// Định nghĩa hàm tính diện tích hình tam giác

// Input : a = Độ dài cạnh a

// b = Độ dài cạnh b

// c = Độ dài cạnh c

// Output: Diện tích hình tam giác

double DienTichTamGiac(double a, double b, double c)

{

 double p = (a + b + c)/2;

 return sqrt(p * (p - a) * (p - b) * (p - c));

}

- Tiếp tục bổ sung khai báo nguyên mẫu hàm tính diện tích hình tam giác :

double DienTichTamGiac(double a, double b, double c);

7.2 Trong tập tin *menu.h* :

Ta bổ sung xử lý thực hiện tính diện tích hình tam giác trong case 3 của hàm XuLyMenu. Trong hàm này ta khai báo thêm một biến thực c (đã có a,b và dienTich) . a,b,c lưu trữ giá trị 3 cạnh tam giác.

void XuLyMenu(int menu)

{

 // Khai báo các biến

 double a, b, c, dienTich;

 switch (menu)

{

```

//Như 6.2
case 3:
    cout << endl << "3. Tinh dien tich hinh tam giac";
    // Thông báo người dùng nhập chiều dài 3 cạnh
    cout << endl << "Nhap do dai canh a : ";
    cin >> a;

    cout << endl << "Nhap do dai canh b : ";
    cin >> b;

    cout << endl << "Nhap do dai canh c : ";
    cin >> c;

    // Gọi hàm tính diện tích hình tam giác
    dienTich = DienTichTamGiac(a, b, c);

    // Xuất kết quả
    cout << endl << "Dien tich hinh tam giac "
        << " co 3 canh a = " << a << ", b = " << b
        << ", c = " << c << " la " << dienTich;
    break;

//như 6.2
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có và kiểm tra việc thực hiện chức năng 3.

Bước 8 :

8.1 Trong **thuvien.h**,
Tiếp tục bổ sung :

- Trong phần định nghĩa hằng, ta bổ sung định nghĩa hằng PI :

```

// =====
// Định nghĩa các hằng số và kiểu dữ liệu mới
// =====
#define PI 3.1415926

```

- Trong phần định nghĩa các hàm xử lý, ta định nghĩa hàm tính diện tích hình tròn :

```

// Định nghĩa hàm tính diện tích hình tròn
// Input : banKinh = Độ dài bán kính hình tròn.
// Output: Diện tích hình tròn
double DienTichHinhTron(double banKinh)
{
    return PI * banKinh * banKinh;
}

```

- Tiếp tục bổ sung khai báo nguyên mẫu hàm tính diện tích hình tròn :

```
double DienTichHinhTron(double banKinh);
```

8.2 Trong tập tin **menu.h** : ta bổ sung xử lý thực hiện tính diện tích hình tròn trong case 4 của hàm XuLyMenu. Trong hàm này ta khai báo thêm biến thực r để lưu trữ bán kính hình tròn nhập từ bàn phím. Hàm XuLyMenu đây đủ như sau :

```
void XuLyMenu(int menu)
{
    // Khai báo các biến
    double a, b, c, r, dienTich;
    switch (menu)
    {
        case 0:
            cout << endl << "0. Thoat khoi chuong trinh";
            break;
        case 1:
            cout << endl << "1. Tinh dien tich hinh vuong";
            // Thông báo người dùng nhập độ dài cạnh
            cout << endl << "Nhap do dai canh hinh vuong : ";
            cin >> a;

            // Gọi hàm tính diện tích hình vuông
            dienTich = DienTichHinhVuong(a);

            // Xuất kết quả
            cout << endl << "Dien tich hinh vuong canh "
                << a << " la " << dienTich;
            break;
        case 2:
            cout << endl << "2. Tinh dien tich hinh chu nhat";
            // Thông báo người dùng nhập chiều dài & rộng
            cout << endl << "Nhap chieu dai hinh CN : ";
            cin >> a;

            cout << endl << "Nhap chieu rong hinh CN : ";
            cin >> b;

            // Gọi hàm tính diện tích hình chữ nhật
            dienTich = DienTichHinhChuNhat(a, b);

            // Xuất kết quả
            cout << endl << "Dien tich hinh chu nhat "
                << " co chieu dai = " << a
                << " chieu rong = " << b << " la " << dienTich;
            break;
        case 3:
            cout << endl << "3. Tinh dien tich hinh tam giac";
            // Thông báo người dùng nhập chiều dài 3 cạnh
            cout << endl << "Nhap do dai canh a : ";
    }
}
```

```

cin >> a;

cout << endl << "Nhập độ dài cạnh b : ";
cin >> b;

cout << endl << "Nhập độ dài cạnh c : ";
cin >> c;

// Gọi hàm tính diện tích hình tam giác
dienTich = DienTichTamGiac(a, b, c);

// Xuất kết quả
cout << endl << "Diện tích hình tam giác "
    << " có 3 cạnh a = " << a << ", b = " << b
    << ", c = " << c << " là " << dienTich;
break;

case 4:
    cout << endl << "4. Tính diện tích hình tròn";
    // Thông báo người dùng nhập độ dài bán kính
    cout << endl << "Nhập độ dài bán kính : ";
    cin >> a;

    // Gọi hàm tính diện tích hình tròn
    dienTich = DienTichHinhTron(a);

    // Xuất kết quả
    cout << endl << "Diện tích hình tròn ban kinh "
        << a << " là " << dienTich;
    break;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra việc thực hiện các chức năng.

Kết thúc chương trình.

Bài 3: Chương trình tính tổng các dãy số

Trong phần này, ta sẽ xây dựng chương trình để tính tổng các dãy số sau:

$$\bullet \quad R = \sum_{i=1}^n \frac{1}{i} \qquad \bullet \quad S = \sum_{i=1}^n \frac{i+1}{i^2} \qquad \bullet \quad T = \sum_{i=1}^n \frac{(-1)^i i}{i+1}$$

Bước 18. Tạo dự án Win32 Console Application mới. Đặt tên là **Lab04_C_Bai3_TongDaySo**.

Bước 19. Tạo cấu trúc cho chương trình như đã hướng dẫn trong **mục 2** (từ 1- 8 để có cấu trúc nội dung tối thiểu chạy được chương trình).

Bước 3: Trong tập tin **menu.h** ta viết lại như sau (cấu trúc giống như bước 9 mục 2, chỉ thay đổi nội dung theo yêu cầu bài toán) :

```
// ======  
// Khai báo nguyên mẫu các hàm xử lý menu  
// ======  
//bổ sung sau  
// ======  
  
// ======  
// Định nghĩa các hàm xử lý menu  
// ======
```

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình

```
// Input : Không có  
// Output: Không có  
void XuatMenu()  
{  
    cout << endl << "===== HE THONG CHUC NANG =====";  
    cout << endl << "0. Thoat khoi chuong trinh";  
    cout << endl << "1. Tinh tong R";  
    cout << endl << "2. Tinh tong S";  
    cout << endl << "3. Tinh tong T";  
    cout << endl << "=====";  
}
```

3.2 Định nghĩa hàm chọn một menu trong danh sách

```
// Input : soMenu = Số lượng menu có thể chọn.  
// Output: Số thứ tự menu do người dùng nhập vào.  
int ChonMenu(int soMenu)  
{  
    int stt;  
    for (;;) {  
        system("CLS");  
        XuatMenu();  
        cout << "\nNhap 1 so khong khoang [0,..," << soMenu << "] de chon chuc nang, stt = ";  
        cin >> stt;  
        if (0 <= stt && stt <= soMenu)  
            break;  
    }  
    return stt;  
}
```

- Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Định nghĩa hàm xử lý menu :

Chú ý rằng cả 3 tổng này đều có đầu vào chung là số nguyên dương n (*kích thước khi dùng của dãy số*). Vì vậy ta bổ sung n làm đối của hàm **XuLyMenu** (ngoài tham số menu)

```
// Input : menu = Số thứ tự menu do người chọn.  
//           n   = Một số nguyên dương  
// Output: Không có.
```

```

void XuLyMenu(int menu, unsigned int n)
{
    switch (menu)
    {
        case 0:
            cout << endl << "0. Thoat khoi chuong trinh";
            break;
        case 1:
            cout << endl << "1. Tinh tong R";
            //bổ sung sau
            break;
        case 2:
            cout << endl << "2. Tinh tong S";
            //bổ sung sau
            break;
        case 3:
            cout << endl << "3. Tinh tong T";
            //bổ sung sau
            break;
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```

void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, unsigned int n);

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Trong bước tiếp theo, ta soạn thảo từng phần (chức năng) trong tập tin **thuvien.h**, bổ sung xử lý chức năng trong hàm **XuLyMenu** của **menu.h**, viết lại hàm **ChayChuongTrinh** trong **program.cpp** để điều khiển tùy chọn thực hiện menu chương trình (chọn 0 là dừng chương trình).

Bước 4 :

4.1 Trong tập tin **program.cpp**, ta cập nhật lại hàm **ChayChuongTrinh** :

```

// Định nghĩa hàm lặp lại việc chọn menu, xử lý menu
// cho tới khi người dùng chọn menu 0 thì thoát CT.
void ChayChuongTrinh()
{
    // Khai báo biến
    int menu,           // lưu số thứ tự menu được chọn
        soMenu = 3;   // lưu số lượng menu chức năng.
    // Khai báo biến n
    unsigned int n;

    // Nhập giá trị cho biến n
    cout << endl << "Nhập mot so nguyen duong : ";
    cin >> n;
    // Lặp lại việc chọn và xử lý menu cho tới khi

```

```
// người dùng chọn chức năng 0. Thoát khỏi CT.
do
{
    menu = ChonMenu(soMenu);
    XuLyMenu(menu,n);
} while (menu > 0);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra thực hiện chức năng chọn 0 thì dừng chương trình, chọn từ 1 đến 3 thì không làm gì cả.

Bước 5:

5.1 Trong **thuvien.h**,

Bổ sung:

- Trong phần định nghĩa nghĩa các hàm xử lý, ta định nghĩa hàm tính TongR :

```
// Định nghĩa hàm tính tổng 1 + 1/2 + 1/3 + ... + 1/n
// Input : n = một số nguyên dương.
// Output: Tổng 1 + 1/2 + 1/3 + ... + 1/n
double TinhTongR(unsigned int n)
{
    double sum = 0; // Khai báo biến lưu kết quả
    unsigned int i;
    // Duyệt qua từng số i từ 1 đến n
    for ( i=1; i<=n; i++)
        sum += 1.0 / i; // Cộng dồn 1/i vào kết quả

    return sum;
}
```

- Trong phần khai báo nguyên mẫu hàm, bổ sung khai báo nguyên mẫu hàm tính TongR :

double TinhTongR(unsigned int n);

5.2 Trong tập tin **menu.h** :

Ta bổ sung thực hiện xử lý tính TinhTongR .

Trong hàm này, ta khai báo biến thực sum để lưu trữ giá trị tổng cần tính của dãy số.

```
void XuLyMenu(int menu, unsigned int n)
{
    double sum;
    switch (menu)
    {
        case 0:
            cout << endl << "0. Thoát khỏi chương trình";
            break;
        case 1:
            cout << endl << "1. Tính tong R";
            // Gọi hàm tính tổng R
            sum = TinhTongR(n);
```

```

        // Xuất kết quả
        cout << endl << "R = " << sum;
        break;
    case 2:
        cout << endl << "2. Tính tong S";
        //bổ sung sau
        break;
    case 3:
        cout << endl << "3. Tính tong T";
        //bổ sung sau
        break;
}
 getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra việc thực hiện chức năng 1

Tiếp theo bổ sung các chức năng khác.

Bước 6 :

5.1 Trong **thuvien.h**,

Ta bổ sung:

- Trong phần định nghĩa nghĩa các hàm xử lý, ta định nghĩa hàm tính TinhTongS :

```

// Định nghĩa hàm tính tổng 2/1 + 3/4 + 4/9 + ...
// Input : n = một số nguyên dương.
// Output: Tổng 2/1 + 3/4 + 4/9 + ... + (n+1)/(n^2)
double TinhTongS(unsigned int n)
{

```

double sum = 0; // Khai báo biến lưu kết quả
unsigned int i;

// Duyệt qua từng số i từ 1 đến n
// và cộng dồn (i+1)/(i^2) vào kết quả
for (i=1; i<=n; i++)
 sum += (i + 1.0) / (i * i);

return sum;

- Trong phần khai báo nguyên mẫu hàm, bổ sung khai báo nguyên mẫu hàm tính TinhTongS :

```
double TinhTongS(unsigned int n);
```

6.2 Trong tập tin **menu.h** : ta bổ sung thực hiện xử lý tính TongS (trong case 2:)

```
void XuLyMenu(int menu, unsigned int n)
```

```
{
    double sum = 0;

    switch (menu)
    {
        case 0:

```

```

cout << endl << "0. Thoát khỏi chương trình";
break;
case 1:
    cout << endl << "1. Tính tong R";
    // Gọi hàm tính tổng R
    sum = TinhTongR(n);
    // Xuất kết quả
    cout << endl << "R = " << sum;
    break;
case 2:
    cout << endl << "2. Tính tong S";
    // Gọi hàm tính tổng S
    sum = TinhTongS(n);
    // Xuất kết quả
    cout << endl << "S = " << sum;
    break;
case 3:
    cout << endl << "3. Tính tong T";
    //bổ sung sau
    break;
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra việc thực hiện chức năng 2

Bổ sung các chức năng còn lại.

Bước 7 :

7.1 Trong **thuvien.h**, ta bổ sung:

- Trong phần định nghĩa nghĩa các hàm xử lý, ta định nghĩa hàm tính TinhTongT :

/ Định nghĩa hàm tính tổng $-1/2 + 2/3 - 3/4 + \dots$

// Input : n = một số nguyên dương.

// Output: Tổng $-1/2 + 2/3 - 3/4 + \dots + n * (-1)^n / (n+1)$

double TinhTongT(unsigned int n)

{

 double sum = 0; // Khai báo biến lưu kết quả

 unsigned int i;

 // Duyệt qua từng số i từ 1 đến n và cộng dồn

 // hoặc trừ bớt $i * (-1)^i / (i+1)$ khỏi kết quả

 for (i=1; i<=n; i++)

 if (i % 2 == 0)

 sum += i / (i + 1.0);

 else

 sum -= i / (i + 1.0);

 return sum;

}

- Trong phần khai báo nguyên mẫu hàm, bổ sung khai báo nguyên mẫu hàm tính TinhTongT :

double TinhTongT(unsigned int n);

7.2 Trong tập tin **menu.h** : ta bổ sung thực hiện xử lý tính TinhTongT (trong case 3:)

```

void XuLyMenu(int menu, unsigned int n)
{
    switch (menu)
    {
        case 0:
            cout << endl << "0. Thoat khoi chuong trinh";
            break;
        case 1:
            cout << endl << "1. Tinh tong R";
            // Gọi hàm tính tổng R
            sum = TinhTongR(n);
            // Xuất kết quả
            cout << endl << "R = " << sum;
            break;
        case 2:
            cout << endl << "2. Tinh tong S";
            // Gọi hàm tính tổng S
            sum = TinhTongS(n);
            // Xuất kết quả
            cout << endl << "S = " << sum;
            break;
        case 3:
            cout << endl << "3. Tinh tong T";
            // Gọi hàm tính tổng T
            sum = TinhTongT(n);
            // Xuất kết quả
            cout << endl << "T = " << sum;
            break;
    }
    getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra việc thực hiện chức năng.

Kết thúc chương trình.

E. Bài tập bắt buộc

Các bài tập dưới đây yêu cầu phải định nghĩa hàm, tổ chức menu và tạo cấu trúc chương trình như đã chỉ dẫn trong mục 1 của phần C. Tên hàm, hằng và biến phải được đặt tên theo đúng quy ước như trong Lab 3.

1. Số nguyên

Viết chương trình cho phép người dùng nhập vào một số nguyên dương **n**. Sau đó:

- Xuất các số từ 1 tới **n**, các số cách nhau 1 dấu Tab và mỗi dòng chứa 10 số.
- Đếm số lượng các số chia hết cho 3 nhưng không chia hết cho 4 trong đoạn [1..**n**].

- c. Đếm số lượng chữ số của n . *Ví dụ: n = 12345 thì số lượng chữ số là 5.*
- d. Đảo ngược số n . *Ví dụ: n = 12345 thì sau khi đảo ngược, n = 54321.*
- e. Tính tổng các chữ số trong n . *Ví dụ: n = 12345 thì tổng các chữ số là 1+2+3+4+5 = 15.*
- f. Cho biết chữ số đầu tiên trong n . *Ví dụ: n = 12345 thì chữ số đầu tiên là 1.*
- g. Tìm số nguyên m lớn nhất sao cho tổng $1+2+\dots+m \leq n$.

2. Đổi cơ số

Viết chương trình cho phép người dùng nhập vào số nguyên dương n . Sau đó, xuất ra màn hình số n ở hệ cơ số b ($2 \leq b \leq 16$) do người dùng chọn. Trường hợp $10 \leq b \leq 16$, sử dụng các ký tự A đến F để biểu diễn các số từ 10 tới 16. Chương trình phải hiển thị các menu sau:

- a. Đổi sang hệ nhị phân ($b=2$)
- b. Đổi sang hệ bát phân ($b=8$)
- c. Đổi sang hệ thập lục phân ($b=16$)
- d. Đổi sang hệ cơ số 7 ($b=7$)

3. Số nguyên tố

Viết chương trình cho phép người dùng nhập vào một số nguyên dương n . Sau đó:

- a. Kiểm tra n có phải là số nguyên tố (sử dụng vòng lặp for).
- b. Xuất ra màn hình các số nguyên tố trong phạm vi từ 1 tới n , các số cách nhau 1 dấu Tab. Mỗi dòng chứa 5 số.
- c. Đếm số lượng số nguyên tố trong phạm vi từ 1 tới n .
- d. Tính tổng các ước số nguyên tố của n . *Ví dụ: n = 30 có các ước 1, 2, 3, 5, 6, 10, 15, 30. Tổng các ước số nguyên tố là 2 + 3 + 5 = 10.*
- e. Phân tích n thành tích các thừa số nguyên tố. *Ví dụ: n = 12 = 2.2.3, n = 30 = 2.3.5.*

4. Ước số

Viết chương trình cho phép người dùng nhập vào một số nguyên dương n . Sau đó:

- a. Xuất tất cả các ước số của n . *Ví dụ: Các ước số của n = 12 là 1, 2, 3, 4, 6, 12.*
- b. Đếm số lượng các ước số của n . *Ví dụ: Số lượng ước số của n = 12 là 6.*
- c. Tính tổng các ước số của n . *Ví dụ: Tổng các ước số của n = 12 là 28.*
- d. Tìm số lớn nhất nhỏ hơn hoặc bằng n mà là lũy thừa của 2. *Ví dụ: n = 1234 thì xuất 1024.*
- e. Kiểm tra số n có phải là số hoàn hảo (còn gọi là số hoàn chỉnh, số hoàn thiện)? Biết rằng, số hoàn hảo là số có tổng các ước bằng hai lần chính nó. *Ví dụ: 6, 28 là số hoàn chỉnh vì 6 = (1+2+3+6)/2, 28 = (1+2+4+7+14+28)/2.*

5. Khai triển Taylor

Viết chương trình tính giá trị các hàm $\sin(x)$, $\cos(x)$, e^x với giá trị x được nhập từ bàn phím. Yêu cầu: không được sử dụng các hàm \sin , \cos trong thư viện $math.h$.

Hướng dẫn: Sử dụng các công thức khai triển Taylor sau với độ chính xác 10^{-5}

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots, -\infty < x < \infty$$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \dots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} + \dots$$

6. Số Fibonacci

Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1, các phần tử sau đó được thiết lập theo quy tắc mỗi phần tử luôn bằng tổng hai phần tử trước nó cộng lại. Công thức truy hồi của dãy Fibonacci là:

$$F(n) = \begin{cases} 0 & \text{khi } n = 0 \\ 1 & \text{khi } n = 1 \\ F(n-1) + F(n-2) & \text{khi } n \geq 2 \end{cases}$$

Viết chương trình cho phép người dùng nhập vào một số nguyên dương n . Sau đó:

- g. Tìm và xuất ra số Fibonacci thứ n .
- h. Liệt kê các số Fibonacci nhỏ hơn hoặc bằng n .
- i. Liệt kê n số Fibonacci đầu tiên.

F. Bài tập làm thêm

1. Giải phương trình

Viết chương trình giải các phương trình và hệ phương trình sau:

- Phương trình bậc nhất 1 ẩn
- Phương trình bậc hai 1 ẩn
- Phương trình bậc ba 1 ẩn
- Phương trình trùng phương
- Hệ phương trình bậc nhất 1 ẩn
- Hệ phương trình bậc nhất 2 ẩn

2. Dãy số

Viết chương trình tính các tổng sau:

- c. $H_n = \sum_{i=1}^n \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ (còn gọi là các số Harmonic)
- d. $S_n = \sum_{i=1}^n \frac{i+1}{i^2} = 2 + \frac{3}{4} + \frac{4}{9} + \dots + \frac{n+1}{n^2}$
- e. $T_n = \sum_{i=1}^n \frac{(-1)^i i}{i+1} = -\frac{1}{2} + \frac{2}{3} - \frac{3}{4} + \dots + \frac{(-1)^n n}{n+1}$
- f. $U_n = \sum_{i=1}^n \frac{1}{i(i+1)} = \frac{1}{1.2} + \frac{1}{2.3} + \frac{1}{3.4} + \dots + \frac{1}{n(n+1)}$
- g. $F_n = \sum_{i=1}^n i! = 1! + 2! + 3! + \dots + n!$

3. Hình học 3 chiều

Viết chương trình tính thể tích của các hình sau:

- Hình lập phương
- Hình hộp chữ nhật
- Hình cầu
- Hình nón
- Hình trụ tròn
- Hình chóp

Tham khảo cách tính thể tích các hình 3D tại: <http://www.mathvn.com/2014/09/cong-thuc-tinh-tich-khoi-chop-lang-tru.html>

LAB 5. CÁC KỸ THUẬT XỬ LÝ MẢNG MỘT CHIỀU

THỜI LUỢNG: 6 TIẾT

A. Mục tiêu

- Giúp sinh viên hiểu rõ và thực hiện thuần thục các kỹ thuật xử lý trên mảng một chiều.
- Tiếp tục rèn luyện kỹ năng phát triển chương trình từng bước hoàn thiện chức năng.
- Sau khi hoàn thành bài thực hành, sinh viên :
 - Nắm vững các khái niệm và thao tác nhập, xuất trên mảng một chiều.
 - Nắm vững các kỹ thuật xử lý cơ bản trên mảng một chiều.
 - Biết cách định nghĩa và sử dụng kiểu dữ liệu mới bằng từ khóa **typedef**.
 - Hiểu rõ cơ chế gọi hàm và truyền tham số: truyền tham trị và truyền tham biến.
 - Thực hiện thuần thục cách tổ chức chương trình bằng thư viện hàm và menu.

B. Yêu cầu

- Trong phần C (hướng dẫn thực hành), sinh viên tự đọc.
- Sinh viên cần chuẩn bị bài trước để thực hành đủ khối lượng yêu cầu.

Buổi thực tập thứ 6 (4 tiết) :

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần D (Hướng dẫn thực hành) của lab 5.
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab05_HD chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Sử dụng lại tập tin word LoiChuongTrinh.docx từ lab4 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - ✓ Thu bài : Xem thông báo của giáo viên .

Buổi thực tập thứ 7 (4 tiết) :

- 2 tiết đầu:
 - Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần E (Bắt buộc) của lab 5.
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab05_BB chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Sử dụng lại tập tin word LoiChuongTrinh.docx trong lab5 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - ✓ Giáo viên sẽ thu bài qua hệ thống thu bài trực tuyến tại phòng lab (thu thư mục MaSV_Lab05_BB) trước khi giải lao.

C. Ôn tập lý thuyết

1. Cú pháp khai báo (định nghĩa) mảng một chiều

Cú pháp: **KDL Tên_bien_mảng [Kích_thước];**

Trong đó:

- **KDL**: là kiểu dữ liệu của các phần tử chứa trong mảng.
- **Tên_bien_mảng**: là tên của mảng, do người lập trình tự đặt và phải tuân theo quy tắc đặt tên.
- **Kích_thước**: là một số nguyên dương, cho biết số phần tử tối đa có thể chứa trong mảng.

2. Cú pháp định nghĩa kiểu dữ liệu mảng 1 chiều

Cú pháp: **typedef KDL Tên_kiểu_mảng [Kích_thước];**

Trong đó:

- **KDL**: là kiểu dữ liệu của các phần tử chứa trong mảng.
- **Tên_kiểu_mảng**: là tên của kiểu dữ liệu (mới) mảng một chiều.

3. Các thao tác nhập - xuất mảng một chiều

a. Trường hợp không sử dụng kiểu mảng một chiều

```
21 // Định nghĩa hàm nhập giá trị các phần tử của mảng
22 // bằng cách nhập lần lượt từ bàn phím.
23 // Input : a = mảng một chiều chứa tối đa MAX phần tử.
24 //          n = số phần tử thực sự được lưu trong mảng.
25 // Output: Không có.
26 void NhapMang(int a[MAX], int n)
27 {
28     // Duyệt qua từng phần tử từ vị trí 0 tới n-1
29     for (int i=0; i<n; i++)
30     {
31         // Xuất thông báo yêu cầu người dùng nhập
32         cout << "a[" << i << "] = ";
33
34         // Chờ người dùng nhập phần tử thứ i
35         cin >> a[i];
36     }
37 }
38 // Định nghĩa hàm nhập giá trị cho các phần tử của
39 // mảng bằng cách sinh các số ngẫu nhiên
40 // Input : a = mảng một chiều chứa tối đa MAX phần tử.
41 //          n = số phần tử thực sự được lưu trong mảng.
42 // Output: Không có.
43 void NhapTuDong(int a[MAX], int n)
44 {
45     // Gieo số ngẫu nhiên đầu tiên
46     srand(time_t(NULL));
47
48     // Duyệt qua từng phần tử từ vị trí 0 tới n-1
49     for (int i=0; i<n; i++)
50     {
51         // Sinh một số ngẫu nhiên trong phạm vi
52         // [0..MAX) rồi gán cho phần tử thứ i
53         a[i] = rand() % MAX;
54     }
55 }
```

```

58 // Định nghĩa hàm xuất các phần tử của mảng ra màn hình
59 // Input : a = mảng một chiều chứa tối đa MAX phần tử.
60 //          n = số phần tử thực sự được lưu trong mảng.
61 // Output: Không có. Chỉ xuất ra màn hình.
62 void XuatMang(int a[MAX], int n)
63 {
64     cout << endl << "Cac phan tu cua mang : " << endl;
65
66     for (int i=0; i<n; i++)
67         cout << a[i] << TAB;
68
69     cout << endl << endl;
70 }

```

Lưu ý quan trọng:

- Đối với mảng 1 chiều, có 2 giá trị thường đi kèm: **MAX** và **n**.
 - MAX**: là kích thước khai báo, là số phần tử tối đa mảng có thể chứa. Giá trị này phải xác định trước và thường được định nghĩa là một hằng số.
 - n**: là số phần tử thực sự chứa trong mảng ($n < MAX$) và giá trị này thay đổi trong mỗi lần chạy chương trình.
- Truyền tham số:

Tham số hình thức (tên khai báo lúc định nghĩa hàm)	Đối số (giá trị truyền vào lúc gọi hàm)
Tên mảng một chiều, viết 1 trong 2 dạng sau: int a[MAX] hoặc DaySo a trong đó: DaySo là kiểu dữ liệu mảng 1 chiều.	Tên của mảng một chiều có cùng kích thước, cùng kiểu với tham số hình thức.

b. Trường hợp sử dụng kiểu dữ liệu mảng một chiều

Trước hết, cần định nghĩa kiểu dữ liệu mảng một chiều. Đặt tên kiểu dữ liệu mới là **DaySo**.

```

9 // Định nghĩa kiểu dữ liệu mảng 1 chiều
10 typedef int DaySo[MAX];

```

Sau đó, thay thế các tham số **int a[MAX]** trong ba hàm nhập-xuất ở trên bởi **DaySo a**. Phần nội dung bên trong hàm không thay đổi.

```

16 void NhapMang(DaySo a, int n);
17 void NhapTuDong(DaySo a, int n);
18 void XuatMang(DaySo a, int n);

```

Với cách này, nếu cần thay đổi kiểu dữ liệu của các phần tử trong mảng hoặc kích thước mảng, ta chỉ cần sửa đổi mã lệnh ở dòng định nghĩa kiểu dữ liệu mảng (lệnh **typedef**).

4. Các kỹ thuật xử lý mảng một chiều

a. Kỹ thuật thử - sai

Áp dụng khi cần xác định Kq (kết quả) và biết $Kq \in \{ a_0, a_1, \dots, a_{n-1} \}$. Cách thực hiện như sau:

- Giả sử $Kq = a_0$
- Duyệt các phần tử còn lại để thử và xác định chính xác giá trị của Kq .

b. Kỹ thuật duyệt

- Duyệt toàn cục: duyệt hết tất cả các phần tử của mảng.
- Duyệt cục bộ: chỉ xét một phần của mảng.

c. Kỹ thuật kiểm tra tính đúng - sai

- Bài toán AND:
 - Dạng:
 - Đúng: nếu với mọi i , a_i đều thỏa mãn điều kiện.
 - Sai: nếu tồn tại i sao cho a_i không thỏa mãn điều kiện.
 - Cách thực hiện:
 - Gán $Kq = 1$; // Giả sử kết quả là đúng
 - Duyệt để tìm điều kiện sai. Nếu có gán $Kq = 0$ và dừng.
- Bài toán OR
 - Dạng:
 - Đúng: nếu tồn tại i sao cho a_i thỏa mãn điều kiện.
 - Sai: nếu với mọi i , a_i đều không thỏa mãn điều kiện.
 - Cách thực hiện:
 - Gán $Kq = 0$; // Giả sử kết quả là sai
 - Duyệt để tìm điều kiện đúng. Nếu có gán $Kq = 1$ và dừng.

D. Hướng dẫn thực hành

Phần này xây dựng chương trình thực hiện các thao tác trên cấu trúc dữ liệu mảng 1 chiều, minh họa các kỹ thuật xử lý mảng 1 chiều trong các bài toán tìm kiếm, tìm Max, Min, đếm, tính tổng, tích, sắp xếp, ...
Chương trình tổ chức theo thư viện và có/không có hệ thống menu (theo mục 1 hay 2 phần C- hướng dẫn - lab 4). Ngoài ra, khi soạn thảo chương trình tiếp tục rèn luyện cách viết có cấu trúc về mặt hình thức.

Bài 1: Chương trình minh họa bài toán AND, OR, kỹ thuật thử và sai.

Viết chương trình thực hiện các thao tác trên dãy a gồm n số nguyên. Chương trình cho phép người dùng chọn các chức năng từ menu sau:

1. Kiểm tra phần tử x có trong dãy a không? Nếu có, trả về 1. Nếu không, trả về 0.
2. Tìm vị trí xuất hiện đầu tiên của phần tử x trong mảng a . Nếu a không chứa x , trả về -1.
3. Kiểm tra mảng a có thứ tự tăng?
4. Tìm phần tử có giá trị lớn nhất
5. Tìm vị trí đầu tiên của phần tử lớn nhất
6. Kiểm tra phát biểu : nếu a chứa x thì a cũng chứa $-x$.

Bước 20. Tạo dự án Win32 Console Application mới. Đặt tên là **Lab05_D_Bai1**

Bước 21. Tạo cấu trúc cho chương trình như đã hướng dẫn trong **mục 2 lab 4** (từ 1- 8 để có cấu trúc nội dung tối thiểu chạy được chương trình).

Bước 3:

- Trong tập tin **thuvien.h** :

Ta bổ sung định nghĩa hằng, kiểu dữ liệu mới :

Vì chương trình thực hiện các thao tác trên mảng 1 chiều, nên ta cần định nghĩa một hằng là giá trị kích thước khai báo của mảng. Ngoài ra, ta có thể định nghĩa một kiểu dữ liệu mảng 1 chiều (lưu ý rằng có thể không cần thực hiện định nghĩa này vì ta có thể làm trực tiếp trên biến mảng)

//Dinh nghia hang

#define MAX 100 //kích thước khai báo mảng 1chieu

#define TAB '\t'

//Dinh nghia kieu du lieu moi:

```
typedef int DaySo[MAX];
```

```
//Khai bao nguyen mau cac ham xu ly, nhap xuat  
//bổ sung sau
```

```
//Dinh nghia cac ham xu ly, nhap xuat
```

```
//bổ sung sau
```

- Trong tập tin **menu.h** :

ta viết lại như sau (cấu trúc giống như bước 9 mục 2 lab 4, chỉ thay đổi nội dung theo yêu cầu bài toán) :

```
// Khai báo nguyên mẫu các hàm xử lý menu
```

```
//bổ sung sau
```

```
// Định nghĩa các hàm xử lý menu
```

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình

// Định nghĩa hàm xuất danh sách chức năng ra màn hình

// Ngoài các chức năng của bài toán, ta thêm chức năng xem dữ liệu số

// Input : Không có

// Output: Không có

```
void XuatMenu()
```

```
{
```

```
    cout << endl << "===== CHON CHUC NANG =====";
```

```
    cout << endl << "0. Thoát khỏi chương trình";
```

```
    cout << endl << "1. Kiểm tra x nam trong mang a";
```

```
    cout << endl << "2. Tìm vị trí đầu tiên x xuất hiện trong a";
```

```
    cout << endl << "3. Kiểm tra mang a là dãy tăng";
```

```
    cout << endl << "4. Tìm phần tử lớn nhất";
```

```
    cout << endl << "5. Tìm vị trí cuối cùng giá trị lớn nhất xuất hiện";
```

```
    cout << endl << "6. Nếu a chưa x thì cung chua -x";
```

```
    cout << endl << "7. Xem dữ liệu dãy số";
```

```
    cout << endl << "=====";
```

```
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách

// Input : soMenu = Số lượng menu có thể chọn.

// Output: Số thứ tự menu do người dùng nhập vào.

```
int ChonMenu(int soMenu)
```

```
{
```

```
    int stt;
```

```
    for (;;)
```

```
    {
```

```
        system("CLS");
```

```
        XuatMenu();
```

```
        cout << "\nNhập 1 số không khoang [0,..," << soMenu << "] để chọn chức năng, stt = ";
```

```
        cin >> stt;
```

```
        if (0 <= stt && stt <= soMenu)
```

```
            break;
```

```
}
```

```
return stt;
```

```
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Định nghĩa hàm xử lý menu :

Các thao tác đều thực hiện trên cùng một đầu vào là mảng 1 chiều kiểu DaySo, nên ta bổ sung thêm biến mảng a kiểu DaySo với kích thước mảng thực dùng trong mỗi lần thực hiện chương trình là số nguyên dương n làm đối của hàm **XuLyMenu**, ngoài tham số đã có là tham số menu.

```
// Input : menu = Số thứ tự menu do người chọn,
//          Dãy số a
//          số nguyên dương n
// Output: Không có.
void XuLyMenu(int menu, DaySo a, int n)
{
    // Khai báo biến

    switch (menu)
    {

        case 0:
            system("CLS");
            cout << endl << "\n0. Thoát khỏi chương trình\n";
            break;
        case 1:
            system("CLS");
            cout << endl << "1. Kiểm tra x nam trong mang a";
            //Bo sung sau
            break;
        case 2:
            system("CLS");
            cout << endl << "2. Tìm vị trí đầu tiên x xuất hiện trong a";
            //Bo sung sau
            break;
        case 3:
            system("CLS");
            cout << endl << "3. Kiểm tra mang a là dãy tăng";
            //Bo sung sau
            break;
        case 4:
            system("CLS");
            cout << endl << "4. Tìm phần tử lớn nhất";
            //Bo sung sau
            break;
        case 5:
            system("CLS");
            cout << endl << "5. Tìm vị trí phần tử lớn nhất";
            //Bo sung sau
            break;
        case 6:
            system("CLS");
            cout << endl << "6. Nếu a chưa x thì cung chưa -x";
            //Bo sung sau
            break;
    }
}
```

```

        case 7:
            system("CLS");
            cout << endl << "7. Xem du lieu day so";
            //Bo sung sau
            break;
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```

void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, DaySo a, int n);

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

- Trong tập tin **program.cpp** :

Hàm **ChayChuongTrinh** ta cập nhật lại như sau :

```

void ChayChuongTrinh()
{
    int soMenu = 7, //lưu số các chức năng
        menu, // lưu số thứ tự chức năng người dùng chọn
        n=0; //kích thước mảng và giá trị khởi tạo
    DaySo a;
    do
    {
        menu = ChonMenu(soMenu);
        XuLyMenu(menu,a,n);
    } while (menu > 0);
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra chức năng thoát khỏi chương trình (chọn 0)

Bước 4 :

Trong bước 4 này, ta làm công việc sau :

- Trong tập tin **thuvien.h**, soạn thảo các hàm nhập, xuất dãy số
- Trong tập tin **menu.h** bổ sung xử lý chức năng xem dữ liệu trong hàm **XuLyMenu**.
- Trong tập tin **program.cpp** cập nhật lại nội dung hàm **ChayChuongTrinh** : Nhập dữ liệu cho dãy số, điều khiển tùy chọn thực hiện menu chương trình (kiểm tra chức năng 7 - chọn 7 để xem dữ liệu, chọn từ 1 đến 6 thì chưa làm gì cả).

- Trong tập tin **thuvien.h** :

Bổ sung các hàm nhập xuất :

4.1 Hàm nhập dữ liệu mảng 1 chiều từ bàn phím

// Input : a = mảng một chiều chứa tối đa MAX phần tử.

// n = số phần tử thực sự được lưu trong mảng.

// Output: Không có.

void NhapMang(DaySo a, int n)

{

```

int i;
// Duyệt qua từng phần tử từ vị trí 0 tới n-1
for (i=0; i<n; i++)
{
    // Xuất thông báo yêu cầu người dùng nhập
    cout << "a[" << i << "] = ";
    // Chờ người dùng nhập phần tử thứ i
    cin >> a[i];
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.2 Định nghĩa hàm xuất các phần tử của mảng ra màn hình
 //Input : a = mảng một chiều chứa tối đa MAX phần tử.
 // n = số phần tử thực sự được lưu trong mảng.
 // Output: Không có. Chỉ xuất ra màn hình.

```

void XuatMang(DaySo a, int n)
{
    int i;
    for (i=0; i<n; i++)
        cout << a[i] << TAB; //hai giá trị cách nhau 1 tab
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.3 Bổ sung nguyên mẫu các hàm :

```

void NhapMang(DaySo a, int n);
void XuatMang(DaySo a, int n);

```

- Trong tập tin *menu.h* :

Bổ sung xử lý chức năng xem dữ liệu của dãy số trong case 7 (Các case từ 0 đến 6 giữ nguyên, bổ sung xử lý trong case 7)

```

void XuLyMenu(int menu, DaySo a, int n)
{
    // Khai báo biến
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << endl << "\n0. Thoát khỏi chương trình\n";
            break;
        //...
        case 7:
            system("CLS");
            cout << endl << "7. Xem dữ liệu dãy số";
            cout << "\nDãy số hiện hành:\n";
            XuatMang(a, n);
            break;
    }
    getch();
}

```

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

- Trong tập tin **program.cpp** :

4.4 Cập nhật lại nội dung hàm **ChayChuongTrinh()** :

Nhập dữ liệu cho mảng để chương trình xử lý theo menu

```
void ChayChuongTrinh()
{
    // Khai báo biến
    int menu,
        soMenu = 7,
        n = 0;
    DaySo a;
    cout << endl << "Nhập một số nguyên dương : ";
    cin >> n;
    //Nhập dữ liệu cho mảng a
    NhapMang(a, n);
    // Lặp lại việc chọn và xử lý menu cho tới khi
    //người dùng chọn chức năng 0. Thoát khỏi CT.
    do
    {
        menu = ChonMenu(soMenu);
        XuLyMenu(menu,a,n);
    } while (menu > 0); //menu =0 thi dung chuong trinh
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra thực hiện việc chọn chức năng 7 để xem dữ liệu.

Trong các bước tiếp theo, ta cập nhật chương trình bằng cách bổ sung và hoàn thiện từng chức năng vào chương trình :

- Lần lượt soạn thảo từng hàm chức năng trong tập tin **thuvien.h**,
- Lần lượt bổ sung xử lý chức năng trong hàm **XuLyMenu** của **menu.h**,

Bước 5: Bổ sung chức năng 1 (kiểm tra a có chứa x) chương trình..

- Trong **thuvien.h**:

5.1 Định nghĩa hàm kiểm tra mảng a có chứa phần tử x?

// Input : a - mảng một chiều chứa tối đa MAX phần tử.

// n - số phần tử thực sự được lưu trong mảng.

// x - phần tử cần kiểm tra

// Output:

// 1 : nếu mảng a chứa phần tử x

// 0 : nếu mảng a không chứa phần tử x

```
int ChuaX(DaySo a, int n, int x)
{
    int i, kq;
    kq = 0; // Ban đầu, giả sử mảng a không chứa x
    // Duyệt qua các phần tử để kiểm tra
    for (i=0; i<n; i++)
        if (a[i] == x)           // Nếu tìm thấy phần tử x
    {
```

```

        kq = 1;          // thì cập nhật kết quả và
        break;           // dừng, không cần tìm nữa
    }
    return kq;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.2 - Khai báo nguyên mẫu hàm :

```
int ChuaX(DaySo a, int n, int x);
```

- Trong **menu.h** :

Nội dung hàm XuLyMenu bổ sung khai báo biến x kiểu int (để lưu trữ giá trị cần tìm được nhập từ bàn phím), biến kq kiểu int để lưu trữ kết quả tìm kiếm, bổ sung việc thực hiện chức năng 1 trong case 1, các case khác giữ nguyên.

```

void XuLyMenu(int menu, DaySo a, int n)
{
    // Khai báo biến
    int x, kq;
    switch (menu)
    {
        //...
        case 1:
            cout << endl << "1. Kiem tra x nam trong mang a";
            cout << endl << "Nhap gia tri x : ";
            cin >> x;
            kq = ChuaX(a, n, x);
            // Xuất thông báo
            system("CLS");
            cout << "\nMang hien hanh:\n";
            XuatMang(a, n);
            if (kq)
                cout << endl << "Mang co chua " << x;
            else
                cout << endl << "Mang khong chua " << x;
            break;
        //...
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra thực hiện chức năng 1 (kiểm tra mảng có chứa x hay không ?).

Bước 6: Bổ sung chức năng 2 (tìm vị trí đầu tiên x xuất hiện trong a) vào chương trình.

- Trong **thuvien.h** :

6.1 Định nghĩa hàm tìm vị trí đầu tiên x xuất hiện trong a.?

```
// Input : a, n,x,
```

```
//Output:
```

-1 : nếu a không chứa phần tử x
 i : a[i] đầu tiên trùng x

```

int Tim_VTDT_X(DaySo a, int n, int x)
{
    int i,
    kq = -1; //Ban đầu, giả sử mảng a không chứa x

    // Duyệt qua các phần tử để kiểm tra
    for (i=0; i<n; i++)
        if (a[i] == x)           // Nếu tìm thấy phần tử x
        {
            kq = i;             // thì cập nhật kết quả là vị trí đầu tiên là I và
            break;               // dừng, không cần tìm nữa
        }
    return kq;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

6.2 Khai báo nguyên mẫu hàm :

```
int Tim_VTDT_X(DaySo a, int n, int x);
```

- Trong **menu.h** :

Bổ sung việc thực hiện chức năng 2 trong case 2, các case khác giữ nguyên.

```

void XuLyMenu(int menu, DaySo a, int n)
{
    // Khai báo biến
    int x, kq;
    switch (menu)
    {
        //...
        case 2:
            cout << endl << "2. Tim vi tri dau tien x xuat hien trong a";
            cout << endl << "Nhap gia tri x : ";
            cin >> x;
            kq = Tim_VTDT_X(a, n, x);
            // Xuất thông báo
            system("CLS");
            cout << "\nMang hien hanh:\n";
            XuatMang(a, n);
            if (kq == -1)
                cout << endl << "Mang khong chua " << x;
            else
                cout << "\nVi tri dau tien " << x << " xuat hien trong a la : " << kq;
            break;
        //...
    }
    getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có..

Kiểm tra kết quả thực hiện chức năng 2.

Bước 7: Bổ sung chức năng 3(kiểm tra mảng a có tăng) vào chương trình.

- Trong **thuvien.h**,

7.1 Định nghĩa hàm kiểm tra mang a có tăng ?

```
// Input : a , n
// Output:
//      1 : nếu mảng a có thứ tự tăng
//      0 : nếu mảng a không có thứ tự tăng
```

```
int KiemTraMangTang(DaySo a, int n)
{
    int i,
    kq = 1; // Ban đầu, giả sử mảng a có thứ tự tăng

    // Duyệt qua các phần tử để kiểm tra
    for (i=0; i<n-1; i++)
        if (a[i] > a[i+1]) // Nếu có cặp phần tử mà số
            {                   // đứng trước > số đứng sau
                kq = 0; // thì cập nhật kết quả và
                break; // dừng, không cần ktra nữa
            }
    return kq;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

7.2 Khai báo nguyên mẫu hàm :

```
int KiemTraMangTang(DaySo a, int n);
```

- Trong **menu.h** :

Bổ sung xử lý chức năng 3 trong case 3, các case khác giữ nguyên.

```
void XuLyMenu(int menu, DaySo a, int n)
{
    // Khai báo biến
    int x, kq;
    switch (menu)
    {
        //...
        case 3:
            cout << endl << "3. Kiểm tra mang a tang";
            kq = KiemTraMangTang(a,n);
            // Xuất thông báo
            system("CLS");
            cout << "\nMang hien hanh:\n";
            XuatMang(a, n);
            if (kq)
                cout << endl << "a La Mang tang.";
            else
                cout << endl << "a không phai Mang tang.";
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có..
Kiểm tra kết quả thực hiện chức năng 3.

Bước 8: Bổ sung chức năng 4(tìm giá trị lớn nhất) vào chương trình.

- Trong **thuvien.h**,

8.1 Đinh nghĩa hàm tính Max

// Input : a, n

// Output: Giá trị lớn nhất của a

```
int TinhMax(DaySo a, int n)
{
    int i,
        max; //lưu trữ giá trị lớn nhất của mảng
    max = a[0]; // Ban đầu, giả sử phần tử đầu tiên là lớn nhất
    // Duyệt qua các phần tử để kiểm tra giả thuyết
    for (i=1; i<n; i++)
        if (a[i] > max)          // Nếu có phần tử lớn hơn
            max = a[i];           // pt giả thuyết, cập nhật
    return max;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

8.2 Khai báo nguyên mẫu hàm :

int TinhMax(DaySo a, int n);

- Trong **menu.h** :

Bổ sung xử lý chức năng 4 trong case 4, các case khác giữ nguyên.

```
void XuLyMenu(int menu, DaySo a, int n)
{
    // Khai báo biến
    int x, kq;
    switch (menu)
    {
        //...
        case 4:
            system("CLS");
            cout << endl << "4. Tìm phần tử lớn nhất";
            kq = TinhMax(a, n);
            // Xuất thông báo
            system("CLS");
            cout << "\nMang hiện hành:\n";
            XuatMang(a, n);
            cout << "\nMax[0...," << n << "] = " << kq;
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có..
Kiểm tra kết quả thực hiện chức năng 4.

Bước 9: Bổ sung chức năng 5 (tìm vị trí xuất hiện cuối cùng của giá trị max) vào chương trình.

- Trong **thuvien.h**,

9.1 Đinh nghĩa hàm tìm vị trí xuất hiện cuối cùng của giá trị max
 //Ham tim vi tri cuoi cung max xuat hien, khong dung ham TinhMax
 // Input : a, n
 // Output: Vị trí cuối cùng tìm thấy giá trị lớn nhất

```
int TimViTriMax_CuoiCung(DaySo a, int n)
{
    // Ban đầu, giả sử phần tử đầu tiên là lớn nhất
    int vt = 0,
        max = a[vt];
    int i;
    // Duyệt qua các phần tử để kiểm tra giả thuyết
    for (i = 1; i<n; i++)
    {
        if (a[i] >= max)          // Nếu có phần tử không nhỏ hơn
            {                      // giá trị max giả định
                vt = i;             // thì cập nhật lại vị trí
                max = a[vt];         // và phần tử lớn nhất
            }
    }
    return vt;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

9.2 Khai báo nguyên mẫu hàm :

```
int TimViTriMax_CuoiCung(DaySo a, int n);
```

- Trong **menu.h** :

Bổ sung xử lý chức năng 5 trong case 5, các case khác giữ nguyên.

```
void XuLyMenu(int menu, DaySo a, int n)
{
    // Khai báo biến
    int x, kq;
    switch (menu)
    {
        //...
        case 5:
            system("CLS");
            cout << endl << "5. Tim vi tri cuoi cung gia tri lon nhat";
            kq = TimViTriMax_CuoiCung(a, n);

            // Xuất thông báo
            system("CLS");
            cout << "\nMang hien hanh:\n";
            XuatMang(a, n);
            cout << endl << "Vi tri xuat hien cuoi cung cua gia tri lon nhat la "
                << kq;
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có..
 Kiểm tra kết quả thực hiện chức năng 5.

Bước 10: Bổ sung chức năng 6 (nếu a chứa x thì cũng chứa -x) vào chương trình..

- Trong **thuvien.h**,

10.1 Định nghĩa hàm kiểm tra phát biểu Nếu a chứa x thì cũng chứa -x

//Chi cần xét các phần tử trong mảng .

// Input : a , n

// Output: 1; nếu đúng; // $\forall i : a[i] \in a$
 0, nếu sai. // $\exists i: -a[i] \notin a$

//Chi quan tam toi cac phan tu trong a

int ChuaXChuaTruX(DaySo a, int n)

{

 int i;//duyet cac phan cu a mang
 kq, // Luu ket qua kiem tra phat bieu
 x, //luu gia tri a[i]
 kqTam; //luu ket qua kiem tra a co chua -x

 kq = 1; //dau tien xem ket qua phat bieu la dung

 // Duyệt qua các phần tử để kiểm tra

 for (i = 0; i < n ; i++)

 {

 x = -a[i];

 kqTam = ChuaX(a, n, x);

 if (kqTam == 0) //neu co mot x (!=0) ma a khong cua -x

 {

 kq = 0; //phat bieu sai

 break;

 }

 }

 return kq;

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

10.2 Khai báo nguyên mẫu hàm :

int ChuaXChuaTruX(DaySo a, int n);

- Trong **menu.h** :

Bổ sung việc chức năng 6 trong case 6, các case khác giữ nguyên.

void XuLyMenu(int menu, DaySo a, int n)

{

 // Khai báo biến

 int x, kq;

 switch (menu)

 {

 //...

 case 6:

 system("CLS");

 cout << endl << "6. Neu a chua x thi cung chua -x";

 kq = ChuaXChuaTruX(a, n);

 // Xuất thông báo

```

        system("CLS");
        cout << "\nMang hien hanh:\n";
        XuatMang(a, n);

        if (kq)
            cout << endl << "phat bieu dung ";
        else
            cout << endl << "phat bieu sai ";
        break;
        //...
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có..

Kiểm tra kết quả thực hiện chức năng 6.

Kiểm tra lại tất cả các chức năng – Kết thúc chương trình.

//

Ghi chú 1:

Trong chương trình trên, dữ liệu được nhập trong hàm ChayChuongTrinh, truyền đến hàm XuLyMenu qua đối số a và n của hàm. Các chức năng thực hiện qua bộ dữ liệu này.

Khi đang thực hiện chương trình, các chức năng muốn thực hiện trên một dữ liệu khác thì không được. Ta chỉ có thể chạy lại chương trình và nhập liệu lại.

Để linh hoạt hơn, ta bổ sung thêm một chức năng mới, chức năng thứ 8, chọn lại bộ dữ liệu mới.

Chương trình bổ sung như sau :

- Trong **thuvien.h** ta đã có hàm nhập dữ liệu cho mảng .

- Trong **menu.h** :

+ Hàm XuatMenu, bổ sung thêm tên chức năng thứ 8 :

```

void XuatMenu()
{
    cout << endl << "=====";
    //tên các chức năng từ 0 đến 7 như cũ
    cout << endl << "8. Chon lai bo du lieu moi cho day so";
    cout << endl << "=====";
}

```

+ Hàm XuLyMenu bổ sung xử lý chức năng 8 trong case 8.

```

void XuLyMenu(int menu, DaySo a, int n)
{
    // Khai báo biến
    int x;//gia tri tim kiem
    kq;//ket qua tim kiem
    switch (menu)
    {

        //...
        case 8:
            system("CLS");
            cout << endl << "8. Chon bo du lieu khac";
            cout << "\nNhap lai kich thuoc n : ";
            cin >> n;
            //goi ham nhap du lieu
    }
}

```

```

        NhapMang(a, n);
        system("CLS");
        cout << "\nDay so moi nhap:\n";
        XuatMang(a, n);
        break;

    }
    _getch();
}

```

- Trong tập tin **program.cpp**, Hàm ChayChuongTrinh sửa lại giá trị soMenu, soMenu = 8.

Nhấn Ctrl + F5 để chạy chương trình thực hiện chức 8 để cập lại bộ dữ liệu mới nhập cho chương trình,
Sử dụng bộ dữ liệu mới này cho các chức năng khác? Không có được kết quả mong muốn!

Đó là vì khi thực hiện xong case 8, bộ dữ liệu mới nhập không lưu lại được nên không thay thế được bộ dữ liệu cũ (nói chung, với cách viết này, nếu dữ liệu của a có thay đổi trong các case của hàm xử XuLyMenu thì không lưu giữ lại được để dùng cho các case khác).

Để khắc phục điều này, trong đối n của hàm XuLyMenu ta sẽ dùng tham chiếu, tức là khi đó n xem như đối ra, cách truyền tham số là truyền bằng biến.

Dòng tiêu đề của hàm XuLyMenu viết lại :

void XuLyMenu(int menu, DaySo a, int &n)

Nhấn Ctrl + F5 để chạy chương trình, kiểm tra kết quả.

Từ đây về sau, nếu các chức năng của chương trình đều thực hiện trên mảng a kích thước n, và làm thay đổi dữ liệu của a, thì đối n trong hàm XuLyMenu sẽ được viết dưới dạng tham chiếu.

Ghi chú 2:

Ta có thể nhập dữ liệu trong hàm XuLyMenu (như một chức năng chương trình, đưa vào xử lý trong câu lệnh switch).

Ghi chú 3:

Để tiết kiệm thời gian, ta không nhập mảng từ bàn phím nữa, mà cho nhập tự động thông qua hàm nhập liệu tự động trong **thuvien.h** như sau :

```

void NhapTuDong(DaySo a, int n)
{
    int i;
    // Gieo số ngẫu nhiên đầu tiên
    srand((unsigned) time(0));
    // srand((unsigned) time(NULL));
    for (i = 0; i < n; i++)
    {
        // Sinh một số ngẫu nhiên trong phạm vi
        // [0..MAX) rồi gán cho phần tử thứ i
        a[i] = rand() % MAX;
    }
}

```

Khi đó :

- Thay thế hàm NhapMang bằng hàm NhapTuDong trong chương trình.
- Trong tập tin **program.cpp** ta khai báo bổ sung các thư viện **<time.h>**, **<stdlib.h>**

Ghi chú 4:

Trong chương trình nếu không định nghĩa kiểu Dayso, ta có thể định nghĩa trực tiếp biến mảng :

```
int a[MAX];
```

```
//////////////////////////////
```

Bài 2: Minh họa các bài tính toán trên dãy số

Viết chương trình thực hiện các thao tác trên dãy a gồm n số nguyên. Chương trình yêu cầu nhập dữ liệu cho a và cho phép người dùng chọn các chức năng trong menu:

- Đếm số lần xuất hiện của giá trị x trong a.
- Đếm và xuất các số nguyên tố trong a.
- Tính tổng giá trị các phần tử trong a
- Tính tổng các giá trị chỉ xuất hiện một lần trong dãy.
- Tính tổng các giá trị phân biệt trong dãy.

Trong bài này, ta không nhập dữ liệu từ hàm ChayChuongTrinh để truyền đến hàm XuLyMenu qua các đối a và n, mà ta sẽ nhập trong hàm XuLyMenu, xem như là một chức năng của chương trình. Vấn đề là lần đầu tiên thực hiện chức năng của chương trình, ta cần chọn trước chức năng nhập dữ liệu cho mảng a, sau đó mới chọn thực hiện các chức năng khác, khi đó các chức năng sẽ thao tác trên bộ dữ liệu đã nhập của a. Nếu chọn lại nhập dữ liệu, tức là ta muốn thực hiện các thao tác trên bộ dữ liệu mới.

Bước 1. Tạo dự án Win32 Console Application mới. Đặt tên là **Lab05_D_Bai2**

Bước 2. Tạo cấu trúc cho chương trình như đã hướng dẫn trong **mục 2 lab 4** (từ 1- 8 để có cấu trúc nội dung tối thiểu chạy được chương trình).

Bước 3:

- Trong tập tin **thuvien.h**, ta bổ sung định nghĩa hằng, kiểu dữ liệu mới :

```
//Dinh nghia hang, kieu du lieu moi
```

```
//Dinh nghia hang
```

```
#define MAX 100 //kich thuoc khai bao mang 1chieu
```

```
#define TAB '\t'
```

```
//Dinh nghia kieu du lieu moi:
```

```
typedef int DaySo[MAX];
```

```
//khai bao nguyen mau cac ham xu ly, nhap xuat
```

```
//Dinh ngia cac ham xu ly, nhap xuat
```

- Trong tập tin **menu.h** ta viết lại như sau (cấu trúc giống như bước 9 mục 2 lab 4, chỉ thay đổi nội dung theo yêu cầu bài toán) :

```
// Khai báo nguyên mẫu các hàm xử lý menu
```

```
//bổ sung sau
```

```
// Định nghĩa các hàm xử lý menu
```

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình

```
void XuatMenu()
```

```
{
```

```
    cout << endl << "===== CHON CHUC NANG =====";
```

```
    cout << endl << "0. Thoat khoi chuong trinh";
```

```
    cout << endl << "1. Nhap tu dong mang a";
```

```

cout << endl << "2. Xem du lieu mang a";
cout << endl << "3. Dem so lan xuat hien cua x trong a";
cout << endl << "4. Dem va xuat cac so nguyen to trong a";
cout << endl << "5. Tinh tong cac gia tri trong mang";
cout << endl << "6. Tính tong cac gia tri chi xuat hien mot lan trong mang";
cout << endl << "7. Tinh tong cac gia tri phan biet trong mang";
cout << endl << "=====";
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách

```

// Input : soMenu = Số lượng menu có thể chọn.
// Output: Số thứ tự menu do người dùng nhập vào.
int ChonMenu(int soMenu)
{
    int stt;
    for (;;)
    {
        system("CLS");
        XuatMenu();
        cout << "\nNhập 1 số khong khoang [0,..," << soMenu << "] để chọn chức năng, stt = ";
        cin >> stt;
        if (0 <= stt && stt <= soMenu)
            break;
    }
    return stt;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Định nghĩa hàm xử lý menu :

Chú ý rằng các thao tác đều thực hiện trên cùng một đầu vào là mảng kiểu DaySo, nên ta bổ sung thêm biến mảng a kiểu DaySo với kích thước mảng là số nguyên dương n làm đối của hàm **XuLyMenu** (ngoài tham số menu). Để lưu giữ sự thay đổi dữ liệu trong a, n sẽ được viết dạng tham chiếu.

```

// Input : menu = Số thứ tự menu do người chọn,
//          Dãy số a
//          số nguyên dương n
// Output: Không có.
void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến

    switch (menu)
    {

        case 0:
            system("CLS");
            cout << endl << "\n0. Thoát khỏi chương trình";
            break;
        case 1:
            system("CLS");
            cout << endl << "1. Nhập từ dòng mảng a";
            //Bo sung sau
    }
}

```

```

        break;
    case 2:
        system("CLS");
        cout << endl << "2. Xem du lieu mang a";
        //Bo sung sau
        break;
    case 3:
        system("CLS");
        cout << endl << "3. Dem so lan xuat hien cua x trong a";
        //Bo sung sau
        break;
    case 4:
        system("CLS");
        cout << endl << "4. Dem va xuat cac so nguyen to trong a";
        //Bo sung sau
        break;

    case 5:
        system("CLS");
        cout << endl << "5. Tinh tong cac gia tri trong mang";
        //Bo sung sau
        break;
    case 6:
        system("CLS");
        cout << endl << "6. Tinh tong cac gia tri chi xuat hien mot lan trong mang";
        //Bo sung sau
        break;
    case 7:
        system("CLS");
        cout << endl << "7. Tinh tong cac gia tri phan biet trong mang";
        //Bo sung sau
        break;
    }
    getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```

void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, DaySo a, int &n);

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có .

- Trong tập tin program.cpp cập nhật lại hàm ChayChuongTrinh để điều khiển chọn và thực hiện menu.

```

void ChayChuongTrinh()
{
    // Khai bao bien
    int menu,          // luu so thu tu menu duoc chon
        soMenu = 7;   // luu so luong chuc nang
    int n = 0; //kich thuoc khi dung cua mang va gia tri khoi tao
    DaySo a;

```

```
//khong nhap mang a trong ham nhu bai 1
do
{
    menu = ChonMenu(soMenu);
    XuLyMenu(menu, a, n);
} while (menu > 0); //menu == 0 thi dung chuong trinh
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra chức năng 0 - Thoát khỏi chương trình..

Bước 4 :

Trong bước 4 này, ta làm công việc sau :

- Trong **program.cpp**, khai báo bổ sung các thư viện cần thiết.
- Trong tập tin **thuvien.h**, soạn thảo các hàm nhập, xuất dãy số
- Trong tập tin **menu.h** bổ sung xử lý chức năng nhập, xuất dữ liệu trong hàm **XuLyMenu**.

- Trong tập tin **program.cpp** khai báo bổ sung các thư viện sau :

```
#include <time.h>
#include <stdlib.h>
```

- Trong tập tin **thuvien.h** bổ sung các hàm nhập xuất :

4.1 Hàm nhập tự động dữ liệu mảng 1 chiều.
// Input : a = mảng một chiều chứa tối đa MAX phần tử.
// n = số phần tử thực sự được lưu trong mảng.
// Output: Không có.

```
void NhapTuDong(DaySo a, int n)
{
    srand((unsigned) time(NULL));
    for (int i = 0; i < n; i++)
    {
        a[i] = (MAX / 2 - rand() % MAX) / 2; //ngẫu nhiên trong khoảng [-8,8]
    }
}
```

4.2 Định nghĩa hàm xuất các phần tử của mảng ra màn hình
void XuatMang(DaySo a, int n)

```
{
    int i;
    for (i=0; i<n; i++)
        cout << a[i] << TAB;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.3 Bổ sung nguyên mẫu các hàm :

```
void NhapTuDong(DaySo a, int n);
void XuatMang(DaySo a, int n);
```

- Trong tập tin **menu.h** ta bổ sung xử lý chức năng nhập,xem dữ liệu của mảng.

(Các case từ 3 đến 7 giữ nguyên, bổ sung xử lý trong case 1, case 2)

```
void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
```

```

switch (menu)
{
    case 0:
        system("CLS");
        cout << endl << "\n0. Thoat khoi chuong trinh\n";
        break;
    case 1:
        system("CLS");
        cout << endl << "1. Nhap tu dong mang a";
        cout << "\nNhap kich thuoc n : ";
        cin >> n;
        //goi ham nhap du lieu
        NhapTuDong(a, n);

        system("CLS");
        cout << endl << "\nDay so moi nhap:\n";
        XuatMang(a, n);
        break;

    case 2:
        cout << endl << "7. Xem du lieu day so";
        cout << endl << "\nDay so hien hanh:\n";
        XuatMang(a, n);
        break;
    //...
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra thực hiện việc chọn chức năng 1 (nhập dữ liệu), chức năng 2 (xem dữ liệu).

Trong các bước tiếp theo, ta soạn thảo từng hàm chức năng trong tập tin **thuvien.h**, bổ sung xử lý chức năng trong hàm **XuLyMenu** của **menu.h**.

Bước 5: Bổ sung chức năng 3 (đếm số lần xuất hiện của x trong a) vào chương trình.

- Trong **thuvien.h** :

5.1 Định nghĩa hàm đếm số lần phần tử x xuất hiện trong a?

```

int Dem_X(DaySo a, int n, int x)
{
    int i, dem = 0;
    for (i = 0; i < n; i++)
        if (a[i] == x)
            dem++;
    return dem;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.2 - Khai báo nguyên mẫu hàm :

```
int Dem_X(DaySo a, int n, int x);
```

- Trong **menu.h** :

Trong hàm **XuLyMenu** bổ sung khai báo biến x kiểu int (để lưu trữ giá trị cần xét nhập vào từ bàn phím), biến kq kiểu int để lưu trữ kết quả tính toán, bổ sung việc thực hiện chức năng đếm x trong case 3, các case khác giữ nguyên.

```
void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
    int x, kq;
    switch (menu)
    {
        //...
        case 3:
            system("CLS");
            cout << endl << "3. Dem so lan xuat hien cua x trong a";
            //Bo sung sau
            cout << "\nNhap gia tri can xet: x = ";
            cin >> x;
            kq = Dem_X(a, n, x);
            system("CLS");
            cout << "\nSo lan " << x << " xuat hien trong a: kq = " << kq;
            cout << "\nXem lai mang hien hanh de kiem tra :\n";
            XuatMang(a, n);
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 3.

Bước 6: Bổ sung chức năng 4 (đếm và xuất các số nguyên tố) vào chương trình

- Trong **program.cpp**:
Bổ sung thư viện **<math.h>**

- Trong **thuvien.h** :

6.1 Định nghĩa hàm một số nguyên có phải là số nguyên tố

Input : x; //số nguyên

Output :

1; nếu x nguyên tố
0; ngược lại

```
int KiemTra_NT(int x)
{
    int i, m,
        kq;
    if (x < 2)
        kq = 0;
    else
    {
        m = (int)sqrt((double)x);
        kq = 1;
        for (i = 2; i <= m; i++)
        if (x % i == 0)
        {
```

```

        kq = 0;
        break;
    }
}
return kq;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

6.2 Định nghĩa hàm đếm và xuất các số nguyên tố trong a

Input a://day so

N; //so nguyen

Output : dem //so cac so nguyen to trong a

```

int Dem_NT(DaySo a, int n)
{
    int i, dem = 0;
    cout << "\nCac so nguyen to trong a:\n";
    for (i = 0; i < n;i++)
        if (KiemTra_NT(a[i]))
    {
        dem++;
        cout << a[i] << TAB;
    }
    return dem;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

6.3 Khai báo nguyên mẫu hàm :

```

int Dem_NT(DaySo a, int n);
int KiemTra_NT(int x);

```

- Trong **menu.h** :

Bổ sung việc thực hiện chức năng 4 trong case 4, các case khác giữ nguyên.

Hàm **XuLyMenu** cập nhật lại như sau :

```

void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
    int x, kq;
    switch (menu)
    {
        //...
    case 4:
        system("CLS");
        cout << endl << "4. Dem va xuat cac so nguyen to trong a";
        kq = Dem_NT(a, n);
        if (kq)
            cout << "\nSo luong cac so nguyen to trong a : kq = " << kq;
        else
            cout << "\nKhong co so nguyen to nao trong a.";
        cout << "\nXem lai mang hien hanh de kiem tra :\n";
        XuatMang(a, n);
        break;
    //...
}

```

```

        }
        _getch();
    }

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 4.

Bước 7: Bổ sung chức năng 5 (tính tổng mảng) vào chương trình.

- Trong **thuvien.h** :

7.1 Định nghĩa hàm tính tổng các phần tử chỉ xuất hiện 1 lần:

Input: a,n

Output: sum = tổng giá trị các phần tử trong mảng

```

int TinhTong(DaySo a, int n)
{
    int i,
        sum = 0;
    for (i = 0; i < n; i++)
        sum += a[i];
    return sum;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

7.2 Khai báo nguyên mẫu hàm :

```
int TinhTong(DaySo a, int n);
```

- Trong **menu.h** :

Bổ sung xử lý chức năng 5 trong case 5, các case khác giữ nguyên.

```

void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
    int x, kq;
    switch (menu)
    {
        //...
        case 5:
            system("CLS");
            cout << endl << "5. Tính tong cac phan tu trong mang";
            cout << "\nTong cac phan tu trong mang: sum = "
                << TinhTong(a, n);
            cout << "\nXem lai mang hien hanh de kiem tra :\n";
            XuatMang(a, n);
            break;
        //...
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra thực hiện việc chọn chức năng 5.

Bước 8: Bổ sung việc thực hiện chức năng 6 (tính tổng duy nhất) vào chương trình

- Trong **thuvien.h** :

8.1 Định nghĩa hàm tính tổng các giá trị chỉ xuất hiện 1 lần:

Input: a,n

Output: sum = tổng các giá trị chỉ xuất hiện 1 lần

```
int TinhTongDuyNhat(DaySo a, int n)
{
    int i,
        sum = 0;
    for (i = 0; i < n; i++)
        if (Dem_X(a, n, a[i]) == 1)
            sum += a[i];
    return sum;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

8.2 Khai báo nguyên mẫu hàm :

```
int TinhTongDuyNhat(DaySo a, int n);
```

- Trong **menu.h** :

Bổ sung xử lý chức năng 6 trong case 6, các case khác giữ nguyên.

```
void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
    int x, kq;
    switch (menu)
    {
        //...
        case 6:
            system("CLS");
            cout << endl << "6. Tính tong cac phan tu chi xuat hien mot lan trong mang";
            cout << "\nTong cac phan tu trong mang chi xuat hien 1 lan: sum = "
                << TinhTongDuyNhat(a, n);
            cout << "\nXem lai mang hien hanh de kiem tra :\n";
            XuatMang(a, n);
            break;
        //...
    }
    getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 6.

```
//=====
```

Bước 9: Bổ sung việc thực hiện chức năng 7 vào chương trình

- Trong **thuvien.h** :

9.1 Định nghĩa hàm tính tổng giá trị phân biệt

Input: a,n

Output: sum = tổng các giá trị phân biệt

```
int TinhTong_PhanBiet(DaySo a, int n)
{
```

```

DaySo b; //b luu tru cac gia tri phan biet cua a
int i;//duyet a
    m, //kich thuoc cua b
    j;//duyet b
    dau;//dsanh dau de nhan dang a[i] da xuat hien trong b
    sum = 0;
m = 0;
for (i = 0; i < n; i++)
{
    dau = 1;//a[i] chua co trong b
    for (j = 0; j < m && dau; j++)
        dau = dau && (a[i] != b[j]);
    if (dau) //a[i] chua co trong b
    {
        b[m++] = a[i];
        sum += a[i];
    }
}
return sum;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

9.2 Khai báo nguyên mẫu hàm :

```
int TinhTong_PhanBiet(ĐaySo a, int n);
```

- Trong **menu.h** :

Bổ sung xử lý chức năng 7 trong case 7, các case khác giữ nguyên.

```

void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
    int x, kq;
    switch (menu)
    {
        //...
        case 7:
            system("CLS");
            cout << endl << "7. Tinh tong cac phan tu phan biet trong mang";
            kq = TinhTong_PhanBiet(a, n);
            cout << "\nTong cac phan tu phan biet trong a: sum = " << kq;
            cout << "\nXem lai mang hien hanh de kiem tra :\n";
            XuatMang(a, n);
            break;
        //...
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra thực hiện việc chọn chức năng 7.

Kiểm tra các chức năng chương trình. Kết thúc.



Bài 3: Minh họa các bài chèn, xóa, thay thế, sắp xếp

Viết chương trình thực hiện các thao tác trên dãy a gồm n số nguyên. Chương trình yêu cầu nhập dữ liệu cho a và cho phép người dùng chọn các chức năng trong menu:

- Nhập tự động dãy số.
- Xem dữ liệu dãy số
- Chèn giá trị x vào đầu dãy số
- Xóa phần tử cuối dãy
- Cắt phần tử đầu dãy rồi chèn vào cuối dãy
- Thay thế giá trị x trong dãy số bằng giá trị y
- Sắp dãy tăng dần
- Sắp dãy theo yêu cầu:
 - Đầu dãy là các số dương tăng dần
 - Tiếp theo là các số âm giảm dần
 - Cuối cùng là các số 0.

Bước 1. Tạo dự án Win32 Console Application mới. Đặt tên là **Lab05_D_Bai3**

Bước 2. Tạo cấu trúc cho chương trình như đã hướng dẫn trong **mục 2 lab 4** (từ 1- 8 để có cấu trúc nội dung tối thiểu chạy được chương trình).

Bước 3:

- Trong tập tin **thuvien.h**, ta bổ sung định nghĩa hằng, kiểu dữ liệu mới :

```
//Dinh nghia hang, kieu du lieu moi  
//Dinh nghia hang  
#define MAX 100 //kich thuoc khai bao mang 1chieu  
#define TAB '\t'
```

```
//Dinh nghia kieu du lieu moi:  
typedef int DaySo[MAX];
```

```
//khai bao nguyen mau cac ham xu ly, nhap xuat
```

```
//Dinh ngia cac ham xu ly, nhap xuat
```

- Trong tập tin **menu.h** ta viết lại như sau (cấu trúc giống như bước 9 mục 2 lab 4, chỉ thay đổi nội dung theo yêu cầu bài toán) :

```
// Khai báo nguyên mẫu các hàm xử lý menu  
//bổ sung sau  
// Định nghĩa các hàm xử lý menu
```

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình

```
void XuatMenu()  
{  
    cout << endl << "===== CHON CHUC NANG =====";  
    cout << endl << "0. Thoat khoi chuong trinh";  
    cout << endl << "1. Nhap tu dong day a";  
    cout << endl << "2. Xem du lieu day a";  
    cout << endl << "3. Chen x vao dau day";  
    cout << endl << "4. Xoa phan tu dau day";  
    cout << endl << "5. Cat phan tu dau day roi chen vao cuoi day";
```

```
cout << endl << "6. Thay the cac gia tri x trong a bang gia tri y";  
cout << endl << "7. Sap day tang dan";  
cout << endl << "8. Sap day theo yeu cau :Duong Tang – Am Giam - Khong";  
cout << endl << "=====";  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách

Viết hàm ChonMenu như 3.2 bài 2.

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Định nghĩa hàm xử lý menu :

Chú ý rằng các thao tác đều thực hiện trên cùng một đầu vào là mảng kiểu DaySo, nên ta bổ sung thêm biến mảng a kiểu DaySo với kích thước mảng là số nguyên dương n làm đối của hàm **XuLyMenu** (ngoài tham số menu). Để lưu giữ sự thay đổi dữ liệu trong a, n sẽ được viết dạng tham chiếu.

```
// Input : menu = Số thứ tự menu do người chọn,  
          Dãy số a  
//          số nguyên dương n  
// Output: Không có.  
void XuLyMenu(int menu, DaySo a, int &n)  
{  
    // Khai báo biến  
  
    switch (menu)  
    {  
        case 0:  
            system("CLS");  
            cout << endl << "\n0. Thoat khoi chuong trinh.\n";  
            break;  
        case 1:  
            system("CLS");  
            cout << endl << "1. Nhap tu dong day a";  
            //Bo sung sau  
            break;  
        case 2:  
            system("CLS");  
            cout << endl << "2. Xem du lieu day a";  
            //Bo sung sau  
            break;  
        case 3:  
            system("CLS");  
            cout << endl << "3. Chen x vao dau day";  
            //Bo sung sau  
            break;  
        case 4:  
            system("CLS");  
            cout << endl << "4. Xoa phan tu dau day";  
            //Bo sung sau  
            break;  
        case 5:
```

```
        system("CLS");
        cout << endl << "5. Cat phan tu dau day roi chen vao cuoi day";
        //Bo sung sau
        break;
    case 6:
        system("CLS");
        cout << endl << "6. Thay the cac gia tri x trong a bang gia tri y";
        //Bo sung sau
        break;
    case 7:
        system("CLS");
        cout << endl << "7. Sap day tang dan";
        //Bo sung sau
        break;
    case 8:
        system("CLS");
        cout << endl << "8. Sap day theo yeu cau :Duong Tang – Am Giam - Khong";
        //Bo sung sau
        break;
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```
void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, DaySo a, int &n);
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

- Trong tập tin program.cpp cập nhật lại hàm ChayChuongTrinh để điều khiển chọn và thực hiện menu.

```
void ChayChuongTrinh()
{
    // Khai bao bien
    int menu,           // luu so thu tu menu duoc chon
        soMenu = 8;    // luu so luong chuc nang
    int n = 0; //kich thuoc khi dung cua mang va gia tri khoi tao
    DaySo a;
    do
    {
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, a, n);
    } while (menu > 0); //menu =0 thi dung chuong trinh
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có .

Kiểm tra kết quả thực hiện chức năng 0 (thoát khỏi chương trình).

Bước 4 :

Trong bước 4 này, ta làm công việc sau :

- Trong **program.cpp**, khai báo bổ sung các thư viện cần thiết.

- Trong tập tin **thuvien.h**, soạn thảo các hàm nhập, xuất dãy số
- Trong tập tin **menu.h** bổ sung xử lý chức năng nhập, xem dữ liệu trong hàm **XuLyMenu**.

- Trong tập tin *program.cpp* khai báo bổ sung các thư viện sau :

```
#include <time.h>
#include <stdlib.h>
```

- Trong tập tin *thuvien.h* bổ sung các hàm nhập xuất :

4.1 Hàm nhập tự động dữ liệu mảng 1 chiều.

Ta viết khác với bước 4 bài 2 một chút, đó là nhập kích thước mảng trong hàm, khi đó đối số n viết dưới dạng tham chiếu.

```
void NhapTuDong(DaySo a, int &n)
```

```
{
    int i;
    cout << "\nNhập kích thước n : ";
    cin >> n;
    srand((unsigned) time(NULL));
    for (i = 0; i < n; i++)
        a[i] = (MAX / 2 - rand() % MAX) / 6;
}
```

4.2 Định nghĩa hàm xuất các phần tử của mảng ra màn hình

```
void XuatMang(DaySo a, int n)
```

```
{
    int i;
    for (i = 0; i < n; i++)
        cout << a[i] << TAB;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

4.3 Bổ sung nguyên mẫu các hàm :

```
void NhapTuDong(DaySo a, int n);
```

```
void XuatMang(DaySo a, int n);
```

- Trong tập tin *menu.h* ta bổ sung xử lý chức năng nhập,xem dữ liệu của mảng.

(Các case từ 3 đến 7 giữ nguyên, bổ sung xử lý nhập, xuất trong case 1, case 2)

```
void XuLyMenu(int menu, DaySo a, int &n)
```

```
{
    // Khai báo biến
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << endl << "\n0. Thoát khỏi chương trình\n";
            break;
        case 1:
            system("CLS");
            cout << endl << "1. Nhập tự động mảng a";
            //goi ham nhap du lieu
            NhapTuDong(a, n);

            system("CLS");
    }
}
```

```

cout << "\nDay so moi nhap:\n";
XuatMang(a, n);
break;

case 2:
    cout << endl << "7. Xem du lieu day so";
    cout << "\nDay so hien hanh:\n";
    XuatMang(a, n);
    break;
//...
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra thực hiện việc chọn chức năng 1 (nhập dữ liệu), chức năng 2 (xem dữ liệu).

Trong các bước tiếp theo, ta soạn thảo từng hàm chức năng trong tập tin **thuvien.h**, bổ sung xử lý chức năng vào hàm **XuLyMenu** trong **menu.h**

Bước 5: Bổ sung việc thực hiện chức năng 3 (chèn vào đầu dãy) vào chương trình.

- Trong **thuvien.h** :

5.1 Định nghĩa hàm chèn x vào đầu dãy a?

//Input : Day a, kich thuoc n, gia tri x can chen

//Output : Day a them x o dau

```

void ChenDauDay(DaySo a, int &n, int x)
{
    int i;
    for (i = n - 1; i >= 0; i--)
        a[i + 1] = a[i];//doi ra sau 1 vi tri, bat dau tu cuoi mang
    a[0] = x;//gan x tai vi tri dau mang
    n++;//kich thuoc mang tang len 1
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.2 - Khai báo nguyên mẫu hàm :

```
void ChenDauDay(DaySo a, int &n, int x);
```

- Trong **menu.h** :

Trong hàm XuLyMenu ta bổ sung khai báo biến x kiểu int (để lưu trữ giá trị cần chèn nhập vào từ bàn phím), bổ sung xử lý chức năng chèn x vào đầu dãy trong case 3, các case khác giữ nguyên.

```

void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
    int x;
    switch (menu)
    {
    //...
    case 3:
        system("CLS");
        cout << endl << "3. Chen x vao dau day";

```

```

cout << "\nNhap gia tri can chen: x = ";
cin >> x;
cout << "\nKich thuoc mang hien hanh : n = " << n;
cout << "\nDay so hien hanh:\n";
XuatMang(a, n);
ChenDauDay(a, n, x);
cout << "\n\nKich thuoc mang ket qua : n = " << n;
cout << "\nDay so ket qua sau khi chen "<<x<<" vao dau day :\n";
XuatMang(a, n);
break;
//...
}
 getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 3.

Bước 6: Bổ sung chức năng 4 (xóa giá trị đầu dãy) vào chương trình.

- Trong *thuvien.h* :

6.1 Định nghĩa hàm

```
//Ham xoa gia tri dau day
//Input : Day a, kich thuoc n,
//Output : Day a(bot vi tri dau day)
```

```
void XoaDauDay(DaySo a, int &n)
{
    int i;
    for (i = 1; i < n; i++)
        a[i - 1] = a[i];//doi ve truoc 1 vi tri, bat dau tu vi tri 1
    n--;//kich thuoc mang giam bot 1
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

6.2 - Khai báo nguyên mẫu hàm :

```
void XoaDauDay(DaySo a, int &n);
```

- Trong *menu.h* :

Nội dung hàm XuLyMenu bổ sung việc thực hiện chức năng 4 vào case 4, các case khác giữ nguyên.

```
void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
    int x;
    switch (menu)
    {
    //...
    case 4:
        system("CLS");
        cout << endl << "4. Xoa phan tu cuoi day";
        cout << "\nKich thuoc mang hien hanh : n = " << n;
        cout << "\nDay so hien hanh:\n";
        XuatMang(a, n);
        XoaDauDay(a, n);
    }
```

```

cout << "\n\nKich thuoc mang ket qua : n = " << n;
cout << "\nDay so ket qua sau khi xoa gia tri dau :\n";
XuatMang(a, n);
break;
//...
}
 getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra thực hiện việc chọn chức năng 4.

Bước 7: Bổ sung việc thực hiện chức năng 5 (Cắt phần tử đầu rồi chèn vào cuối dãy) vào chương trình

- Trong **thuvien.h** :

7.1 Định nghĩa hàm cắt phần tử đầu rồi chèn cuối dãy

//Ham cat dau chen cuoi

```

void CatDau_ChenCuoi(DaySo a, int &n)
{
    int i,
        x; //luu phan tu dau
    x = a[0];
    for (i = 1; i < n; i++)//Xoa dau
        a[i - 1] = a[i];//doi ve truoc 1 vi tri, bat dau tu vi tri 1
    a[n-1] = x;//gan x tai vi tri cuoi mang
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

7.2 - Khai báo nguyên mẫu hàm :

void CatDau_ChenCuoi(DaySo a, int &n);

- Trong **menu.h** :

Trong hàm XuLyMenu bổ sung xử lý chức năng 5 vào case 5, các case khác giữ nguyên.

```

void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
    int x;
    switch (menu)
    {
    //...
    case 5:
        system("CLS");
        cout << endl << "5. Cat phan tu dau day roi chen vao cuoi day";
        cout << "\nKich thuoc mang hien hanh : n = " << n;
        cout << "\nDay so hien hanh:\n";
        XuatMang(a, n);
        CatDau_ChenCuoi(a, n);
        cout << "\n\nKich thuoc mang ket qua : n = " << n;
        cout << "\nDay so ket qua sau khi cat gia tri dau roi chen vao cuoi day:\n";
        XuatMang(a, n);
    //...
    }
    getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra thực hiện việc thay thế.

Bước 8: Bổ sung việc thực hiện chức năng 6 (thay thế x bằng y) vào chương trình

- Trong **thuvien.h** :

8.1 Định nghĩa hàm thay thế x trong day bang y

//Input : Day a, kích thước n, giá trị x cần thay thế, giá trị thay thế y

//Output : Day a(x thay thế y)

```
void Thay_X_Bang_Y(DaySo a, int &n, int x, int y)
{
    int i;
    for (i = 0; i < n; i++)
        if (a[i] == x)
            a[i] = y;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

8.2 - Khai báo nguyên mẫu hàm :

```
void Thay_X_Bang_Y(DaySo a, int &n, int x, int y);
```

- Trong **menu.h** :

Nội dung hàm XuLyMenu bổ sung thêm khai báo biến y kiểu int để lưu trữ giá trị thay thế, biến x đã có sẵn lưu trữ giá trị cần thay thế, bổ sung nội dung xử lý chức năng 6 vào case 6, các case khác giữ nguyên.

```
void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
    int x, y;
    switch (menu)
    {
    //...
    case 6:
        system("CLS");
        cout << endl << "6. Thay thế giá trị x trong a bằng giá trị y";
        cout << "\nNhập giá trị cần thay thế: x = ";
        cin >> x;
        cout << "\nNhập giá trị thay thế: y = ";
        cin >> y;

        cout << "\nKích thước mảng hiện hành : n = " << n;
        cout << "\nDay so hiện hành:\n";
        XuatMang(a, n);
        Thay_X_Bang_Y(a, n, x, y);
        cout << "\n\nKích thước mảng kết quả : n = " << n;
        cout << "\nDay so kết quả sau khi thay " << x << " bằng giá trị " << y << ":" << endl;
        XuatMang(a, n);
        break;

    //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra thực hiện việc chúc năng 6.

Bước 9: Bổ sung chức năng 7 (sắp tăng dãy) vào chương trình

- Trong **thuvien.h** :

9.1 Định nghĩa hàm sắp tăng dãy:

```
//Input : Day a, kích thước n  
//Output : Day a(sắp tăng)  
void SapTang(DaySo a, int n)  
{  
    int i, j;  
    for (i = 0; i < n - 1; i++)  
        for (j = i + 1; j < n; j++)  
            if (a[i] > a[j])  
                HoanVi(a[i], a[j]);  
}
```

9.2 Định nghĩa hàm hoán vị:

```
void HoanVi(int &x, int &y)  
{  
    int tam;  
    tam = x;  
    x = y;  
    y = tam;  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

9.3 - Khai báo nguyên mẫu hàm :

```
void SapTang(DaySo a, int n);
```

```
void HoanVi(int &x, int &y);
```

- Trong **menu.h** :

Nội dung hàm XuLyMenu bổ sung nội dung thực hiện chức năng 7 vào case 7, các case khác giữ nguyên.

```
void XuLyMenu(int menu, DaySo a, int &n)  
{  
    // Khai báo biến  
    int x, y;  
    switch (menu)  
    {  
        ...  
        case 7:  
            system("CLS");  
            cout << endl << "7. Sap day tang dan";  
            cout << "\nDay so hien hanh:\n";  
            XuatMang(a, n);  
            SapTang(a, n);  
            cout << "\nDay so sau khi sap tang:\n";  
            XuatMang(a, n);  
            break;  
  
        ...  
    }  
    _getch();
```

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra thực hiện việc chức năng 7.

Bước 10: Bổ sung chức năng 8 (sắp dãy theo yêu cầu) vào chương trình.

- Trong **thuvien.h** :

10.1 Định nghĩa hàm sắp dãy theo yêu cầu : Dương tăng-Âm giảm-Không

//Input : Day a, kích thước n

//Output : Day a(da theo yêu cầu)

```
void Sap_DuongTang_AmGiam_Khong(DaySo a, int n)
{
    int i, j, mc;
    for (i = 0; i < n - 1; i++)
        for (j = i + 1; j < n; j++)
    {
        mc = (a[i] < 0 && a[j] < 0 && a[i] < a[j]) ||
            (a[i] < 0 && a[j] > 0) ||
            (a[i] == 0 && a[j] != 0) ||
            (a[i] > 0 && a[j] > 0 && a[i] > a[j]);
        if (mc)
            HoanVi(a[i], a[j]);
    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

10.2 - Khai báo nguyên mẫu hàm :

```
void Sap_DuongTang_AmGiam_Khong(DaySo a, int n);
```

- Trong **menu.h** :

Trong hàm XuLyMenu, bổ sung xử lý chức năng 8 vào case 8, các case khác giữ nguyên.

```
void XuLyMenu(int menu, DaySo a, int &n)
{
    // Khai báo biến
    int x, y;
    switch (menu)
    {
        ...
        case 8:
            system("CLS");
            cout << endl << "8. Sap day theo yeu cau :Duong Tang - Am Giam - Khong";
            cout << "\nDay so hien hanh:\n";
            XuatMang(a, n);
            Sap_DuongTang_AmGiam_Khong(a, n);
            cout << "\nDay so sau khi sap theo yeu cau:\n";
            XuatMang(a, n);
            break;
        ...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra chức năng 8.

Kiểm tra tất cả các chức năng. Kết thúc chương trình.

Bài 4:

Viết chương trình nhập một dãy n số nguyên, xuất ra các giá trị phân biệt của dãy và số lần xuất hiện của nó trong dãy.

Chương trình này tổ chức như mục 1, phần C, lab 4 : chỉ có 2 tập tin : program.cpp và thuvien.h, không có tập tin menu.h vì bài toán không yêu cầu tổ chức tùy chọn menu

Bước 1. Tạo dự án Win32 Console Application mới. Đặt tên là **Lab05_D_Bai4**

Bước 2. Tạo cấu trúc chương trình như đã hướng dẫn trong **mục 1 phần C lab 4** (từ bước 1 đến bước 7).

Bước 3.

- Trong **program.cpp** ta khai báo bổ sung :

```
#include <time.h>
#include <stdlib.h>
```

- Trong tập tin **thuvien.h**, từng bước bổ sung định nghĩa hằng, cài đặt các hàm xử lý, . . . :

//Định nghĩa hằng

```
#define MAX 100
#define TAB '\t'
```

//Khai báo nguyên mẫu các hàm xử lý

//. . . (bổ sung sau)

//Định nghĩa các hàm xử lý

3.1 Hàm nhập tự động mảng n số nguyên

```
void NhapTuDong(int a[MAX], int &n)
{
    int i;
    cout << "\nNhập kích thước n : ";
    cin >> n;
    srand((unsigned)time(NULL));
    for (i = 0; i < n; i++)
        a[i] = (MAX / 2 - rand() % MAX) / 6; //Trong khoảng [-8,+8]
}
```

Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

3.2 Hàm xuất mảng n số nguyên

```
void XuatMang(int a[MAX], int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout << a[i] << TAB;
}
```

Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

3.3 Hàm tìm các giá trị phân biệt của a và số lần xuất hiện của nó.

```
//Input : a, n
//Output: b, ( cac giá trị phân biệt của a lưu trong mang b)
//          c, (so lần xuất hiện của b[j] lưu trong c[j])
//          m ( kích thước của b,c )
//b,c,m làm doi ra của ham
```

```

void Tim_Day_GiaTri_PhanBiet(int a[MAX], int n, int b[MAX], int c[MAX], int &m)
{
    int i; //duyet theo n
    int j; //duyet b,c
    int dau; //danh dau a[i] co thuoc b
    for (i = 0; i < n; i++)
        c[i] = 1; //khoi tao c : so lan xuat hiem cua moi gia tri phan bat b[i] bang 1
    m = 0; //khoi tao kich thuoc b, c
    for (i = 0; i < n; i++)
    {
        dau = 0; //a[i] khong thuoc b
        for (j = 0; j < m; j++)
            if (a[i] == b[j])
            {
                dau = 1; //a[i] thuoc b
                c[j]++; //b[j] tang them 1 lan xuat hien
                break;
            }
        if (!dau) //khong thuoc
        {
            b[m] = a[i]; // chen a[i] vao cuoi b
            m++;
        }
    }
}

```

(Xem thêm tại trang 24 cách viết hàm : `int TinhTong_PhanBiet(DaySo a, int n)`)

Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

3.4 Khai báo nguyên mẫu các hàm :

```

void XuatMang(int a[MAX], int n);
void NhapTuDong(int a[MAX], int &n);
void Tim_Day_GiaTri_PhanBiet(int a[MAX], int n, int b[MAX], int c[MAX], int &m);

```

Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

Bước 4: Tích hợp các chức năng để hoàn chỉnh chương trình

Trong tập tin ***program.cpp*** :

- Khai báo bổ sung thư viện :

```
#include <iomanip>
```

- Bổ sung hàm **ChayChuongTrinh** với các nội dung :

Nhập tự động dữ liệu cho mảng, gọi hàm tìm giá trị phân biệt của a và số lần xuất hiện, xuất kết quả. Có thể điều khiển lặp việc chương trình cho đến khi người dùng dừng.

```

void ChayChuongTrinh()
{
    Char kt;
    int a[MAX], b[MAX], c[MAX];
    int i, n = 0, m = 0;
    do
    {

```

```
system("CLS");
NhapTuDong(a, n);
Tim_Day_GiaTri_PhanBiet(a, n, b, c, m);
cout << "\nDay dang xet:\n";
XuatMang(a, n);
cout << setiosflags(ios::left);
cout << endl << setw(20) << "Gia tri Phan Biet"
    << setw(20) << "So lan xuat hien";
for (i = 0; i < m; i++)
{
    cout << endl << setw(20) << b[i]
        << setw(20) << c[i];
}
cout << "\nNua khong, nhan ESC neu khong!\n";
kt = _getch();
} while (kt != 27);
}
```

Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

Thực hiện chương trình – kết thúc chương trình.

E. Bài tập bắt buộc

1. Tìm kiếm

- Tìm vị trí của số nguyên tố cuối cùng trong mảng *a*. Nếu *a* không chứa số nguyên tố, trả về -1.
- Tìm phần tử xuất hiện nhiều nhất và số lần xuất hiện của nó.
- Tìm phần tử có giá trị nhỏ nhất trong mảng và vị trí xuất hiện đầu tiên của nó.
- Tìm số âm lớn nhất và vị trí của nó.
- Tìm số dương nhỏ nhất và vị trí của nó.

2. Đếm

- Đếm số lượng số có 3 chữ số.
- Đếm các số nằm ngoài phạm vi [*min .. max*] cho trước.
- Đếm số lượng số chính phương (số chính phương là số bằng bình phương 1 số khác. Ví dụ: 4, 9, 16, 25, ...)
- Đếm số lần xuất hiện của phần tử *x* kể từ vị trí *vt* cho trước.
- Đếm số lượng các đường chạy trong dãy. Biết rằng, đường chạy là dãy con có thứ tự (tăng hoặc giảm) dài nhất gồm những phần tử nằm kế tiếp nhau.

3. Sắp xếp

- Sắp các số dương tăng dần, các số khác giữa nguyên thứ tự.
- Sắp các phần tử sao cho số 0 nằm ở cuối mảng, các số khác ở đầu mảng và tăng dần.
- Sắp các phần tử sao cho số 0 ở đầu mảng, số âm ở giữa và giảm dần, số dương ở cuối và tăng
- Sắp các số lẻ nằm đầu mảng và tăng dần, các số chẵn nằm cuối mảng và giảm dần.
- Sắp các số nguyên tố nằm đầu mảng và tăng, các số còn lại nằm ở cuối và giảm dần.

4. Chèn và thay thế

- Chèn phần tử *x* vào mảng *a* tại vị trí *vt* cho trước.

- Chèn phần tử x vào sau phần tử lớn nhất (đầu tiên tìm được) trong mảng.
- Chèn phần tử x vào trước số nguyên tố đầu tiên trong mảng, nếu không có số nguyên tố nào thì chèn đầu.
- Chèn phần tử x vào sau mỗi phần tử y cho trước. Nếu mảng không chứa y thì chèn tại vị trí 0.
- Thay thế giá trị nhỏ nhất bằng giá trị x cho trước.

5. Xóa

- Xóa phần tử nằm tại vị trí vt cho trước khỏi mảng a .
- Xóa phần tử x đầu tiên tìm được trong mảng a .
- Xóa mọi phần tử x trong mảng a .
- Xóa tất cả các phần tử trùng nhau, chỉ giữ lại một phần tử trong số các phần tử trùng đó.
- Xóa các phần tử nằm ngoài đoạn $[min .. max]$ cho trước.

6. Tính toán

- Tính trung bình cộng của các phần tử trong mảng
- Tính tổng bình phương của các phần tử trong mảng
- Tính độ lệch lớn nhất giữa 2 phần tử nằm liên tiếp nhau
- Tính tổng các số nguyên tố có 2 chữ số

7. Kiểm tra đúng sai của các phát biểu sau

- Mảng a không chứa phần tử 0.
- Mảng a có chứa 3 phần tử nằm liên tiếp có giá trị liên tiếp nhau.
- Mảng a chứa cả phần tử 0 lẫn 1.
- Mảng a chứa phần tử có giá trị bằng trung bình cộng của các phần tử.
- Mảng a không chứa giá trị âm.

F. Bài tập làm thêm

1. Thông kê

Cho mảng a chứa các số nguyên trong đoạn $[0..10000]$. Viết chương trình xuất ra màn hình các phần tử phân biệt của mảng a và số lần xuất hiện của chúng.

2. Bài cào

Bài cào là một kiểu chơi bài bằng bộ bài tây 52 lá. Bài được chia cho từng người, mỗi người 3 lá. Điểm của người chơi trong mỗi ván là số lá của tổng điểm 3 lá bài. Ví dụ, nếu tổng điểm 3 lá bài là 27 thì người đó được 7 điểm, nếu tổng là 10 điểm thì được 0 điểm. Cách tính điểm của các lá bài như sau:

- Các lá 2, 3, ..., 10 mỗi lá có điểm tương ứng với con số đó, bất kể lá bài màu gì.
- Lá A có điểm là 1, các lá J, Q, K đều được tính là 10 điểm.

Sau khi tính điểm và trình bài, ai có số điểm cao nhất là thắng ván đó. Trường hợp đặc biệt, ai sở hữu được cả 3 lá bài đều là bài tây (J, Q hoặc K) hoặc cả 3 lá đều cùng điểm số thì thắng ngay ván đó, không cần tính điểm. Nếu có từ 2 người trở lên có cùng điểm số cao nhất thì tiền cược được chia đều.

Viết chương trình minh họa trò chơi theo mô tả trên. Trong đó, máy tính đóng vai trò người chia bài. Sau mỗi ván, máy phải xáo bài trước khi chia. Có tất cả 10 người chơi, mỗi người được cấp một số tiền M . Chương trình sẽ dừng khi chỉ còn 1 người đủ tiền đặt cược hoặc khi người dùng chọn chức năng thoát chương trình. Tiền cược quy định cho mỗi ván là C với $C \leq M/10$. Những người chơi có số tiền bé hơn C không được phép tham gia tiếp.

Chương trình phải có các chức năng sau: thiết lập mức cược C , chia bài, tính điểm và thoát chương trình.

3. Xếp hạng

Cho mảng a chứa tối đa 10000 số nguyên phân biệt. Hãy viết chương trình xuất ra màn hình lần lượt từng phần tử của a và thứ tự của nó trong dãy a sau khi đã sắp xếp.

4. Tìm bia

Một nhóm bạn tham dự một bữa tiệc, mỗi người được phép uống 1 lon bia. Trong lúc đang ăn uống thì một sự cố xảy ra gây tắt đèn và chuông báo động cháy vang lên. Mọi người bình tĩnh đặt lon bia xuống bàn rồi thoát ra khỏi tòa nhà. Sau khi báo động tắt, sự cố được giải quyết, họ trở lại bữa tiệc và cố tìm lại lon bia của mình. Tuy nhiên, đa số họ quên mất trước đó mình ở vị trí nào. Do đó, họ cứ lấy ngẫu nhiên 1 lon bia. Tính xác suất để có ít nhất một người lấy đúng lon bia của mình trước đó.

Gợi ý:

- *Viết một chương trình cho phép nhập vào số lượng sinh viên (N) tham dự bữa tiệc và giả lập M lần ($M \geq 1000$) sự kiện được mô tả trong đề bài.*
- *Giả sử ban đầu các lon bia được đánh số từ 1 tới N ứng với số thứ tự từng sinh viên.*
- *Sử dụng phương pháp sinh ngẫu nhiên N số phân biệt từ 1 tới N hoặc xáo trộn ngẫu nhiên N số ban đầu để giả lập sự kiện lấy lại bia.*
- *Tính số thí nghiệm (gọi là F) xảy ra trường hợp có ít nhất 1 sinh viên lấy đúng bia của mình.*
- *In ra màn hình giá trị F/M .*
- *Thay đổi giá trị của N và M để đưa ra nhận xét.*

5. Phát hiện trùng lặp

Cho mảng a chứa N số nguyên có giá trị trong đoạn $[1..N]$. Hãy viết chương trình kiểm tra mảng có chứa ít nhất 2 phần tử trùng nhau hay không? Yêu cầu: chỉ được duyệt qua các phần tử của mảng một lần và không được dùng thêm mảng khác.

6. Nối dãy

Cho mảng a, b chứa tối đa MAX số thực. Hãy viết chương trình:

- Chèn các phần tử của mảng b vào cuối mảng a.
- Giả sử dãy a đã được sắp xếp tăng. Hãy chèn các phần tử của b vào a sao cho vẫn được thứ tự tăng.

LAB 6. CÁC KỸ THUẬT XỬ LÝ MẢNG HAI CHIỀU (MA TRẠN)

THỜI LUỢNG: 6 TIẾT

A. Mục tiêu

- Giúp sinh viên hiểu rõ và thực hiện thuần thục các kỹ thuật xử lý trên mảng hai chiều.
- Tiếp tục rèn luyện kỹ năng phát triển chương trình theo chức năng, chương trình tổ chức theo thư viện hàm và hệ thống menu.
- Sau khi hoàn thành bài thực hành này, sinh viên :
 - Nắm vững các khái niệm và thao tác nhập, xuất trên mảng hai chiều.
 - Nắm vững các kỹ thuật xử lý cơ bản trên mảng hai chiều, đặc biệt là truy cập được các phần tử của nó.
 - Biết cách định nghĩa và sử dụng kiểu dữ liệu ma trận bằng từ khóa **typedef**.
 - Truyền tham số thực, truyền tham biến.

B. Yêu cầu

- Trong phần C (Ôn tập lý thuyết), sinh viên tự đọc.
- Sinh viên cần chuẩn bị bài trước để thực hành đủ khối lượng yêu cầu.

Buổi thực tập thứ 7 (4 tiết) :

- 2 tiết cuối:
 - Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần D (Hướng dẫn thực hành) của lab 6.
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab06_HD chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Sử dụng lại tập tin word LoiChuongTrinh.docx từ lab5 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - ✓ Giáo viên sẽ thu bài qua hệ thống thu bài trực tuyến tại phòng lab (thư mục MaSV_Lab06_HD) vào cuối buổi thực tập.

Buổi thực tập thứ 8 (4 tiết) :

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần E (Bắt buộc) của lab 6.
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab06_BB chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Sử dụng lại tập tin word LoiChuongTrinh.docx trong lab6 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - ✓ Thu bài : Xem thông báo của giáo viên .

C. Ôn tập lý thuyết

1. Cú pháp khai báo biến mảng hai chiều (ma trận)

Cú pháp: **KDL Tên_bien_mảng [Số_dòng_tối_da] [Số_cột_tối_da];**

Trong đó:

- **KDL**: là kiểu dữ liệu của các phần tử chứa trong ma trận.
- **Tên_bien_mảng**: là tên của ma trận, do người lập trình đặt và phải tuân theo quy tắc đặt tên.
- **Số_dòng_tối_da** và **Số_cột_tối_da** là hai nguyên dương, cho biết số dòng và số cột tối đa của ma trận.

2. Cú pháp định nghĩa kiểu dữ liệu mảng 2 chiều

Cú pháp: **typedef KDL Tên_kiểu_mảng [Số_dòng_tối_da] [Số_cột_tối_da];**

Trong đó:

- **KDL**: là kiểu dữ liệu của các phần tử chứa trong ma trận.
- **Tên_kiểu_mảng**: là tên của kiểu dữ liệu (mới) ma trận.
- **Số_dòng_tối_da** và **Số_cột_tối_da** cho biết số dòng và số cột tối đa của ma trận.

3. Các thao tác nhập - xuất mảng hai chiều

c. Trường hợp không sử dụng kiểu mảng hai chiều

```
25 // Định nghĩa hàm nhập giá trị các phần tử của ma trận
26 // bằng cách nhập lần lượt từ bàn phím.
27 // Input : a = ma trận chứa tối đa SIZE dòng, SIZE cột.
28 //          m = số dòng thực sự của ma trận.
29 //          n = số cột thực sự của ma trận.
30 // Output: Không có.
31 void NhapMang(int a[SIZE][SIZE], int m, int n)
32 {
33     // Duyệt qua từng dòng từ dòng 0 tới m-1
34     for (int i=0; i<m; i++)
35     {
36         cout << endl << "Dòng thu : " << i << endl;
37
38         // Duyệt qua từng cột từ cột 0 tới n-1
39         for (int j=0; j<n; j++)
40         {
41             // Xuất thông báo yêu cầu người dùng nhập
42             cout << "a[" << i << " , " << j << "] = ";
43
44             // Chờ người dùng nhập giá trị cho ô [i,j]
45             cin >> a[i][j];
46         }
47     }
48 }
```

```
50 // Định nghĩa hàm nhập giá trị cho các phần tử của
51 // ma trận bằng cách sinh các số ngẫu nhiên
52 // Input : a = ma trận chứa tối đa SIZE dòng, SIZE cột.
53 //          m = số dòng thực sự của ma trận.
54 //          n = số cột thực sự của ma trận.
55 // Output: Không có.
56 void NhapTuDong(int a[SIZE][SIZE], int m, int n)
57 {
58     // Gieo số ngẫu nhiên đầu tiên
59     srand(time(NULL));
60
61     // Duyệt qua từng dòng từ dòng 0 tới m-1
62     for (int i=0; i<m; i++)
63     {
64         // Duyệt qua từng cột từ cột 0 tới n-1
65         for (int j=0; j<n; j++)
66         {
67             // Sinh một số ngẫu nhiên trong phạm vi
68             // [-MAX/2..MAX/2) rồi gán vào ô [i,j]
69             a[i][j] = rand() % MAX - MAX / 2;
70         }
71     }
72 }
73
74 // Định nghĩa hàm xuất các phần tử của mảng ra màn hình
75 // Input : a = ma trận chứa tối đa SIZE dòng, SIZE cột.
76 //          m = số dòng thực sự của ma trận.
77 //          n = số cột thực sự của ma trận.
78 // Output: Không có. Chỉ xuất ra màn hình.
79 void XuatMang(int a[SIZE][SIZE], int m, int n)
80 {
81     cout << endl << "Cac phan tu cua ma tran : " << endl;
82
83     // Duyệt qua từng dòng từ dòng 0 tới m-1
84     for (int i=0; i<m; i++)
85     {
86         // Duyệt qua từng cột từ cột 0 tới n-1
87         for (int j=0; j<n; j++)
88         {
89             // Xuất phần tử ở ô [i,j]
90             cout << a[i][j] << TAB;
91         }
92
93         // Kết thúc 1 dòng -> xuống dòng
94         cout << endl;
95     }
96     cout << endl << endl;
97 }
```

Lưu ý quan trọng:

- Đối với ma trận, có các giá trị sau thường đi theo:
 - SIZE** là kích thước khai báo, cho biết số dòng và số cột tối đa của ma trận. Giá trị này thường được định nghĩa trước như một hằng số.
 - m** (số dòng), **n** (số cột): là kích thước thực sự của ma trận trong mỗi lần chạy chương trình, $0 < m, n < SIZE$.
- Nếu là ma trận vuông, ta có $m = n$. Khi đó, ta chỉ cần một tham số **n**. Nguyên mẫu của các hàm nhập xuất trở thành:
 - NhapMang(int a[SIZE][SIZE], int n)
 - NhapTuDong(int a[SIZE][SIZE], int n)
 - XuatMang(int a[SIZE][SIZE], int n)
- Truyền tham số

Tham số	Đối số
Tên biến ma trận	Tên biến ma trận (có cùng kiểu, cùng kích thước)

d. Trường hợp định nghĩa và sử dụng kiểu dữ liệu ma trận (MaTran)

Trước hết, cần định nghĩa kiểu dữ liệu ma trận. Đặt tên kiểu dữ liệu mới là MaTran.

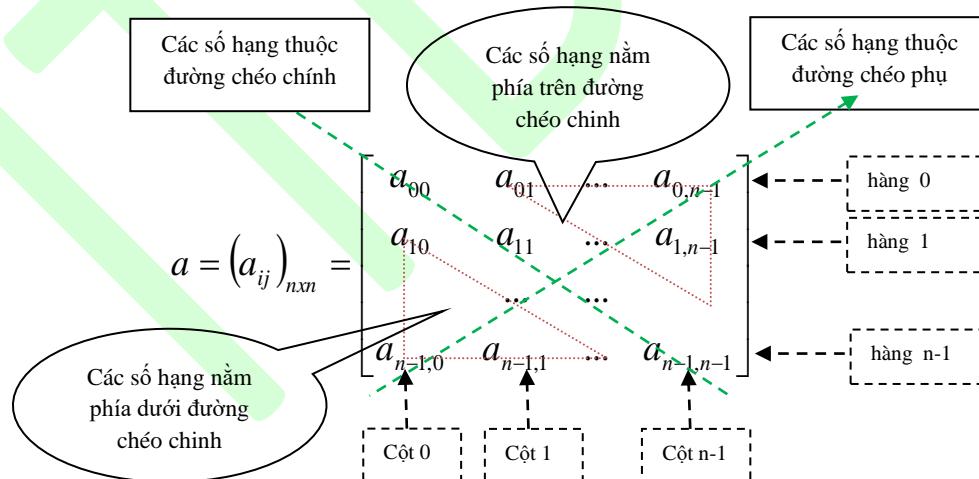
```
10 // Định nghĩa kiểu dữ liệu mảng 1 chiều
11 typedef int MaTran[SIZE][SIZE];
```

Sau đó, thay thế các tham số **int a[SIZE][SIZE]** trong ba hàm nhập, xuất ở trên bởi **MaTran a**. Phần nội dung bên trong hàm không thay đổi.

```
17 void NhapMang(MaTran a, int m, int n);
18 void NhapTuDong(MaTran a, int m, int n);
19 void XuatMang(MaTran a, int m, int n);
```

Với cách này, nếu cần thay đổi kiểu dữ liệu của các phần tử trong mảng hoặc kích thước mảng, ta chỉ cần sửa đổi mã lệnh ở dòng định nghĩa kiểu dữ liệu mảng (lệnh **typedef**).

10. Kỹ thuật truy cập các phần tử của ma trận vuông



Các tính chất

STT	Đặc trưng	Truy cập chỉ số	Giá trị biến điều khiển
1	Các phần tử thuộc đường chéo chính	$a_{ij}; i = j$	a_{ii}
2	Các phần tử thuộc đường chéo phụ	$a_{ij}; i + j = n - 1$	$j = n - 1 - i$
3	Các phần tử nằm dưới đường chéo chính	$a_{ij}; i > j$	$i = 1 \rightarrow n - 1, j = 0 \rightarrow i - 1$
4	Các phần tử nằm trên đường chéo chính	$a_{ij}; i < j$	$i = 0 \rightarrow n - 2, j = i + 1 \rightarrow n - 1$
5	Các phần tử nằm trên hàng i	$a_{ij}; j = 0 \rightarrow n - 1$	$a_{i0}, a_{i1}, a_{i2}, \dots, a_{i,n-1}$
6	Các phần tử nằm trên cột j	$a_{ij}; i = 0 \rightarrow n - 1$	$a_{0j}, a_{1j}, a_{2j}, \dots, a_{n-1,j}$

D. Hướng dẫn thực hành

Phần này xây dựng chương trình thực hiện các thao tác trên cấu trúc dữ liệu mảng 2 chiều với các chức năng cơ bản.

Tiếp tục tổ chức chương trình theo thư viện và có/không có hệ thống menu như lab 4, lab 5.

Bài 1: Truy cập vào các phần tử của ma trận vuông

Viết chương trình thực hiện các thao tác trên các ma trận vuông cấp n . Yêu cầu của chương trình:

- Tính tổng các phần tử của ma trận
- Xuất các phần tử thuộc đường chéo chính
- Xuất các phần tử thuộc đường chéo phụ.
- Tính tổng các phần tử nằm phía trên đường chéo chính
- Tính tích các phần tử nằm phía dưới đường chéo phụ
- Xuất các phần tử thuộc các đường chéo song song với đường chéo chính. Mỗi đường chéo xuất trên 1 dòng.

Bước 1. Tạo dự án Win32 Console Application mới. Đặt tên là **Lab06_D_Bai1**

Bước 2. Tạo cấu trúc cho chương trình như đã hướng dẫn trong **mục 2 lab 4** (từ 1- 8 để có cấu trúc nội dung tối thiểu chạy được chương trình).

Tức là ta có kết quả chương trình ở bước này như sau :

- Tập tin **thuvien.h** : Rỗng
- Tập tin **menu.h** : Rỗng
- Tập tin **program.cpp** có nội dung như sau :

```
#include <iostream>
#include <conio.h>
using namespace std;

#include "thuvien.h"
#include "menu.h"

void ChayChuongTrinh();

int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    _getch();
}
```

Nhấn Ctrl + F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra hoạt động của chương trình.

Bước 3:

Bước này ta định nghĩa kiểu dữ liệu mới, tổ chức và vận hành hệ thống menu.

- Trong tập tin ***thuvien.h*** :

Ta bổ sung định nghĩa hằng, kiểu dữ liệu mới :

Vì chương trình thực hiện các thao tác trên ma trận vuông, nên ta cần định nghĩa một hằng là giá trị kích thước khai báo của mảng. Ngoài ra, ta có thể định nghĩa một kiểu dữ liệu ma trận vuông (lưu ý rằng có thể không cần thực hiện định nghĩa này vì ta có thể làm trực tiếp trên biến mảng 2 chiều)

//Định nghĩa hằng

```
#define SIZE 5//kích thước khai báo mảng 2 chiều
```

```
#define TAB '\t'
```

//Định nghĩa kiểu dữ liệu mới:

```
typedef int MaTranVuong[SIZE][SIZE];
```

//Khai báo nguyên mẫu các hàm xử lý, nhập xuất

```
//bổ sung sau
```

//Định nghĩa các hàm xử lý, nhập xuất

```
//bổ sung sau
```

- Trong tập tin ***menu.h*** :

// Khai báo nguyên mẫu các hàm xử lý menu

```
//bổ sung sau
```

//Định nghĩa các hàm xử lý menu

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình

// Định nghĩa hàm xuất danh sách chức năng ra màn hình

// Ngoài các chức năng của bài toán, ta thêm chức năng xem dữ liệu

// Input : Không có

// Output: Không có

void XuatMenu()

```
{
```

```
    cout << endl << "===== CHON CHUC NANG =====";  
    cout << endl << "0. Thoát khỏi chương trình";  
    cout << endl << "1. Nhập ma trận vuông";  
    cout << endl << "2. Xem ma trận vuông";  
    cout << endl << "3. Tính tổng các phần tử của ma trận";  
    cout << endl << "4. Xuất các phần tử thuộc đường chéo chính";  
    cout << endl << "5. Xuất các phần tử thuộc đường chéo phụ";  
    cout << endl << "6. Tính tổng các phần tử nằm phía trên đường chéo chính";  
    cout << endl << "7. Tính tích các phần tử nằm phía dưới đường chéo phụ";  
    cout << endl << "8. Xuất các đường chéo // đường chéo chính";  
    cout << endl << "=====
```

```
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách

// Input : soMenu = Số lượng menu có thể chọn.

// Output: Số thứ tự menu do người dùng nhập vào.

int ChonMenu(int soMenu)

```
{
```

```
    int stt;  
    for (;;) {
```

```

    {
        system("CLS");
        XuatMenu();
        cout<<"\nNhập 1 số khong khoang [0,..," << soMenu << "] để chọn chức năng, stt = ";
        cin >> stt;
        if (0 <= stt && stt <= soMenu)
            break;
    }
    return stt;
}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```

3.3 Định nghĩa hàm xử lý menu :

Các thao tác đều thực hiện trên cùng một đàu vào là ma trận vuông, nên ta bổ sung thêm biến a kiểu MaTranVuong với kích thước mảng thực dùng trong mỗi lần thực hiện chương trình là số nguyên dương n làm đối của hàm **XuLyMenu**, ngoài tham số đã có là tham số menu. Vì ma trận vuông được nhập trong case 1 trong hàm, nên đổi n trong hàm XuLyMenu ta dùng tham chiếu.

```

// Input : menu = Số thứ tự menu do người chọn,
//          ma trận vuông a
//          số nguyên dương n
// Output: Không có.
void XuLyMenu(int menu, MaTranVuong a, int &n)
{
    // Khai báo biến

    switch (menu)
    {
        case 0:
            system("CLS");
            cout << endl << "0. Thoát khỏi chương trình\n";
            break;
        case 1:
            system("CLS");
            cout << endl << "1. Nhập ma trận vuông";
            //Bo sung sau
            break;
        case 2:
            system("CLS");
            cout << endl << "2. Xem ma trận vuông";
            //Bo sung sau
            break;
        case 3:
            system("CLS");
            cout << endl << "3. Tính tổng các phần tử của ma trận ";
            //Bo sung sau
            break;
        case 4:
            system("CLS");
            cout << endl << "4. Xuất các phần tử thuộc đường chéo chính";
            //Bo sung sau
            break;
    }
}

```

```

case 5:
    system("CLS");
    cout << endl << "5. Xuat cac phan tu thuoc duong cheo phu";
    //Bo sung sau
    break;
case 6:
    system("CLS");
    cout << endl << "6. Tinh tong cac phan tu nam phia tren duong cheo chinh";
    //Bo sung sau
    break;
case 7:
    system("CLS");
    cout << endl << "7. Tinh tích cac phan tu nam phia duoi duong cheo phu ";
    //Bo sung sau
    break;
case 8:
    system("CLS");
    cout << endl << "7. Tinh tích cac phan tu nam phia duoi duong cheo phu ";
    cout << endl << "8.Xuat cac duong cheo // duong cheo chinh ";
    //Bo sung sau
    break;
}
 getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```

void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, MaTranVuong a, int & n);

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

- Trong tập tin **program.cpp** :
Hàm **ChayChuongTrinh** ta cập nhật lại như sau :

```

void ChayChuongTrinh()
{
    int soMenu = 8, //lưu số các chức năng
        menu, // lưu số thứ tự chức năng người dùng chọn
        n=0; //kích thước mảng và giá trị khởi tạo
    MaTranVuong a;
    do
    {
        menu = ChonMenu(soMenu);
        XuLyMenu(menu,a,n);
    } while (menu > 0);
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra sự vận hành của hệ thống menu, đặc biệt chức năng 0 - thoát khỏi chương trình.

Bước 4 :

- Trong bước 4 này, ta làm công việc sau :
- Trong tập tin **thuvien.h**, soạn thảo các hàm nhập, xuất ma trận vuông
- Trong tập tin **menu.h** bổ sung xử lý chức năng xem dữ liệu trong hàm **XuLyMenu**.

- Trong tập tin **thuvien.h** :

Bổ sung các hàm nhập xuất :

4.1 Hàm nhập dữ liệu ma trận vuông cấp n từ bàn phím

void NhapMaTran(MaTranVuong a, int n)

```
{
    int i, j;
    for (i = 0; i < n; i++) //hang i
        for (j = 0; j < n; j++) //cot j
    {
        cout << "\na[" << i << "]" << j << "] = ";
        cin >> a[i][j];
    }
}
```

4.2 Hàm xuất dữ liệu ma trận vuông ra màn hình

void XuatMaTran(MaTranVuong a, int n)

```
{
    int i, j;
    for (i = 0; i < n; i++)
    {
        cout << '\n';
        for (j = 0; j < n; j++)
            cout << a[i][j] << TAB;
    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.3 **Bổ sung nguyên mẫu các hàm :**

void NhapMaTran(MaTranVuong a, int n);

void XuatMaTran(MaTranVuong a, int n);

- Trong tập tin **menu.h** :

Bổ sung xử lý chức năng nhập dữ liệu vào case 1, xem dữ liệu vào case 2 (Các case từ 3 đến 8 giữ nguyên)

```
void XuLyMenu(int menu, MaTranVuong a, int &n)
{
    // Khai báo biến
    switch (menu)
    {

        case 0:
            system("CLS");
            cout << endl << "\n0. Thoát khỏi chương trình\n";
            break;
        case 1:
            system("CLS");
            cout << endl << "1. Nhập ma trận vuông";
            cout << "\nNhập cấp ma trận vuông : n = ";
```

```

        cin >> n;
        NhapMaTran(a, n);
        cout << "\nMa tran vuong vua nhap:\n";
        XuatMaTran(a, n);
        break;
    case 2:
        system("CLS");
        cout << endl << "2. Xem ma tran vuong";
        cout << "\nMa tran vuong hien hanh:\n";
        XuatMaTran(a, n);
        break;
    // bo sung sau
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 1, 2.

Trong các bước tiếp theo, ta lần lượt bổ sung các chức năng khác vào chương .

- Lần lượt soạn thảo từng hàm chức năng trong tập tin **thuvien.h**,
- Lần lượt bổ sung xử lý chức năng trong hàm **XuLyMenu** của **menu.h**,

Bước 5: Bổ sung chức năng 3 (tính tổng các phần tử của ma trận) chương trình..

- Trong **thuvien.h**:

5.1 Định nghĩa hàm kiểm tra mảng a có chứa phần tử x?

```

// Input : ma trận vuông a, kích thước n
// Output: sum = tổng giá trị các phần tử của a.
int TinhTong_MaTran(MaTranVuong a, int n)
{
    int i, j, sum = 0;
    for (i = 0; i < n; i++) // hang i
        for (j = 0; j < n; j++) //cot j
            sum += a[i][j];
    return sum;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.2 - Khai báo nguyên mẫu hàm :

```
int TinhTong_MaTran(MaTranVuong a, int n);
```

- Trong **menu.h** :

Nội dung hàm **XuLyMenu** bổ sung khai báo biến sum kiểu int (để lưu tổng các phần tử ma trận) bổ sung chức năng 3 vào case 3, các case khác giữ nguyên.

```

void XuLyMenu(int menu, MaTranVuong a, int &n)
{
    // Khai báo biến
    int sum;
    switch (menu)
    {
        //...
        case 3:
            system("CLS");
            cout << endl << "3. Tinh tong cac phan tu cua ma tran ";

```

```

        cout << "\nMa tran hien hanh :\n";
        XuatMaTran(a, n);
        sum = TinhTong_MaTran(a, n);
        cout << "\nTong cac phan tu cua ma tran : sum = " << sum;
        break;
    //...
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 3.

Bước 6: Bổ sung chức năng 4 (xuất các phần tử thuộc đường chéo chính) vào chương trình..

- Trong **thuvien.h**:

6.1 Định nghĩa hàm xuất các phần tử thuộc đường chéo chính

// Input : ma trận vuông a, kích thước n

// Output: không có

```

void XuatDuongCheoChinh(MaTranVuong a, int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout << a[i][i] << TAB;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

6.2 - Khai báo nguyên mẫu hàm :

```
void XuatDuongCheoChinh(MaTranVuong a, int n);
```

- Trong **menu.h** :

Trong hàm XuLyMenu bổ sung chức năng 4 vào case 4, các case khác giữ nguyên.

```

void XuLyMenu(int menu, MaTranVuong a, int &n)
{
    // Khai báo biến
    int sum;
    switch (menu)
    {
        //...
        case 4:
            system("CLS");
            cout << endl << "4. Xuat cac phan tu thuoc duong cheo chinh";
            cout << "\nMa tran hien hanh :\n";
            XuatMaTran(a, n);
            cout << "\nCac phan tu thuoc duong cheo chinh cua a:\n";
            XuatDuongCheoChinh( a, n);
            break;
        //...
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 4.

Bước 7: Bổ sung chức năng 5 (xuất các phần tử thuộc đường chéo phụ) vào chương trình..

- Trong **thuvien.h**:

7.1 Định nghĩa hàm xuất các phần tử thuộc đường chéo chính

// Input : ma trận vuông a, kích thước n

// Output: không có

void XuatDuongCheoPhu(MaTranVuong a, int n)

```
{
    int i;
    for (i = 0; i < n; i++)
        cout << a[i][n-i-1] << TAB;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

7.2 - Khai báo nguyên mẫu hàm :

- Trong **menu.h** :

Trong hàm XuLyMenu bổ sung chức năng 5 vào case 5, các case khác giữ nguyên.

void XuLyMenu(int menu, MaTranVuong a, int &n)

```
{
    // Khai báo biến
    int sum;
    switch (menu)
    {
        //...
        case 5:
            system("CLS");
            cout << endl << "5. Xuat cac phan tu thuoc duong cheo phu";
            cout << "\nMa tran hien hanh :\n";
            XuatMaTran(a, n);
            cout << "\nCac phan tu thuoc duong cheo chinh cua a:\n";
            XuatDuongCheoPhu(a, n);
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 5.

Bước 8: Bổ sung chức năng 6 (tính tổng các phần tử phía trên đường chéo chính) vào chương trình..

- Trong **thuvien.h**:

8.1 Định nghĩa hàm tính tổng các phần tử phía trên đường chéo chính

// Input : ma trận vuông a, kích thước n

// Output: sum = tổng các phần tử phía trên đường chéo chính

int int TinhTong_Tren_CheoChinh(MaTranVuong a, int n)

```
{
    int i, j, sum = 0;
    for (i = 0; i < n-1; i++) // hang i
        for (j = i+1; j < n; j++) //cot j
            sum += a[i][j];
    return sum;
}
```

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

8.2 - Khai báo nguyên mẫu hàm :

`int TinhTong_Tren_CheoChinh (MaTranVuong a, int n);`

- Trong **menu.h** :

Trong hàm XuLyMenu bổ sung chức năng 6 vào case 6, các case khác giữ nguyên.

`void XuLyMenu(int menu, MaTranVuong a, int &n)`

```
{
    // Khai báo biến
    int sum;
    switch (menu)
    {
        //...
        case 6:
            system("CLS");
            cout << endl << "6. Tinh tong cac phan tu nam phia tren duong cheo chinh";
            cout << "\nMa tran hien hanh :\n";
            XuatMaTran(a, n);
            sum = TinhTong_Tren_CheoChinh (a, n);
            cout << "\nTong cac phan tu phia tren duong cheo chinh: sum = " << sum;
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 6.

Bước 9: Bổ sung chức năng 7 (tính tích các phần tử phía dưới đường chéo phụ) vào chương trình..

- Trong **thuvien.h**:

9.1 Định nghĩa hàm tính tích các phần tử phía dưới đường chéo phụ

// Input : ma trận vuông a, kích thước n

// Output: p = tích các phần tử phía dưới đường chéo phụ

//Tinh tich cac phan tu nam phia duoi duong cheo phu cua ma tran

`int TinhTich_duoi_CheoPhu(MaTranVuong a, int n)`

```
{
    int i, j, p = 1;
    for (i = 1; i < n ; i++) // hang i
        for (j = n-i; j < n; j++) //cot j
            p *= a[i][j];
    return p;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

9.2 - Khai báo nguyên mẫu hàm :

`int TinhTich_duoi_CheoPhu(MaTranVuong a, int n);`

- Trong **menu.h** :

Trong hàm XuLyMenu, khai báo thêm biến p kiểu int để lưu trữ kết quả tích các phần tử phía dưới đường chéo phụ, bổ sung chức năng 7 vào case 7, các case khác giữ nguyên.

`void XuLyMenu(int menu, MaTranVuong a, int &n)`

```
{
    // Khai báo biến
    int sum, p;
    switch (menu)
```

```

{
    //...
    case 7:
        system("CLS");
        cout << endl << "7. Tinh tích cac phan tu nam phia duoi duong cheo phu";
        cout << "\nMa tran hien hanh :\n";
        XuatMaTran(a, n);
        p = TinhTich_duoi_CheoPhu(a, n);
        cout << "\nTich cac phan tu nam phia duoi duong cheo phu: p = " << p;
        break;
    //...
}
 getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 7.

Bước 10: Bổ sung chức năng 8 (xuat cac duong cheo song song duong cheo chinh) vào chương trình..

- Trong *thuvien.h*:

10.1 Định nghĩa hàm xuất các đường chéo song song đường chéo chính

// Input : ma trận vuông a, kích thước n

// Output: không có

```

void Xuat_DuongCheo_SS_DCChinh(MaTranVuong a, int n)
{
    int i, //hang
        j;//cot
        k;//cac duong cheo
    cout << "\n\nCac duong cheo phia tren duong cheo chinh:\n";
    for (k = n - 1; k >= 1; k--)
    {
        cout << "|duong cheo thu " << k << ":\t";
        for (i = 0; i < n; i++)
            for (j = i + 1; j < n; j++)
                if (j - i == k)
                    cout << a[i][j] << '\t';
    }

    cout << "\n\nCac duong cheo phia duoi duong cheo chinh:\n";
    for (k = n - 1; k >= 1; k--)
    {
        cout << "|duong cheo thu " << k << ":\t";
        for (i = 1; i < n; i++)
            for (j = 0; j < i; j++)
                if (i - j == k)
                    cout << a[i][j] << '\t';
    }
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

10.2 - Khai báo nguyên mẫu hàm :

```
void Xuat_DuongCheo_SS_DCChinh(MaTranVuong a, int n);
```

- Trong **menu.h** :

Trong hàm XuLyMenu bổ sung chức năng 8 vào case 8, các case khác giữ nguyên.

```
void XuLyMenu(int menu, MaTranVuong a, int &n)
```

```
{
    // Khai báo biến
    int sum, p;
    switch (menu)
    {
        //...
        case 8:
            system("CLS");
            cout << endl << "8.Xuat cac duong cheo // duong cheo chinh ";
            cout << "\nMa tran hien hanh :\n";
            XuatMaTran(a, n);
            Xuat_DuongCheo_SS_DCChinh(a, n);
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 8

Bước 11: Bổ sung chức năng 9 (xuat cac duong cheo song song duong cheo phu) vào chương trình..

- Trong **thuvien.h**:

11.1 Định nghĩa hàm xuất các đường chéo song song đường chéo phu

// Input : ma trận vuông a, kích thước n

// Output: không có

```
void Xuat_DuongCheo_SS_DCPhu(MaTranVuong a, int n)
{
    int i, //hang
        j, //cot
        k; //cac duong cheo
    cout << "\n\nCac duong cheo nam ben trai duong cheo phu:\n";
    for (k = 0; k < n - 1; k++)
    {
        cout << "\nduong cheo thu " << k + 1 << ":";
        cout << endl;
        for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - 1 - i; j++)
        if (j + i == k)
            cout << a[i][j] << '\t';
    }

    cout << "\n\nCac duong cheo nam ben phai duong cheo phu:\n";
    for (k = n; k < 2 * n - 1; k++)
    {
        cout << "\nduong cheo thu " << k - n + 1 << ":";
        cout << endl;
        for (i = 1; i < n; i++)
        for (j = n - i; j < n; j++)
        if (j + i == k)
            cout << a[i][j] << '\t';
    }
}
```

```

    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

11.2 - Khai báo nguyên mẫu hàm :

```
void Xuat_DuongCheo_SS_DCPHU(MaTranVuong a, int n);
```

- Trong **menu.h** :

Trong hàm XuLyMenu bổ sung chức năng 9 vào case 9, các case khác giữ nguyên.

```
void XuLyMenu(int menu, MaTranVuong a, int &n)
```

```
{
    // Khai báo biến
    int sum, p;
    switch (menu)
    {
        //...
        case 9:
            system("CLS");
            cout << endl << "8.Xuat cac duong cheo // duong cheo chinh ";
            cout << "\nMa tran hien hanh :\n";
            XuatMaTran(a, n);
            Xuat_DuongCheo_SS_DCPHU(a, n);
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 9.

Kiểm tra tất cả các chức năng chương trình - Kết thúc chương trình

Lưu ý:

- Mỗi lần chạy chương trình, cần chọn chức năng 1 đầu tiên để nhập dữ liệu, khi đó mới có dữ liệu để các chức năng khác thực hiện. Trong quá trình lặp tùy chọn thực hiện các chức năng, nếu cần thay đổi bộ dữ liệu khác, chỉ cần chọn lại chức năng 1 để nhập lại dữ liệu.
- Trong trường hợp lần đầu người dùng không chọn chức năng 1 để tạo dữ liệu trước mà chọn các chức năng khác, ta nên xử lý như thế nào?

Bài 2: Tính toán trên ma trận

Viết chương trình thực hiện các thao tác trên ma trận cấp $m \times n$ với các phần tử là số nguyên. Chương trình cho phép người dùng chọn chức năng từ các menu sau:

- Tính giá trị lớn nhất của ma trận
- Tính giá trị lớn nhất hàng i
- Tính Tổng các phần tử thuộc hàng i .
- Tính giá trị nhỏ nhất cột j
- Tính Tích các phần tử thuộc cột j .
- Xuất ra màn hình các phần tử a_{ij} thỏa mãn: a_{ij} là phần tử lớn nhất hàng i và nhỏ nhất cột j .

Bước 1: Tạo dự án Win32 Console Application mới. Đặt tên là **Lab06_D_Bai1**

Bước 2: Tạo cấu trúc cho chương trình như bài trên (bài 1, từ 1- 8 để có cấu trúc nội dung tối thiểu chạy được chương trình).

Bước 3: Tổ chức hệ thống menu và kiểm tra sự vận hành của nó.

- Trong tập tin **thuvien.h** :

Vì thực hiện trên mảng 2 chiều nên ta bổ sung định nghĩa hàng là kích thước khai báo của mảng.

Ngoài ra, bài này ta sử dụng trực tiếp biến ma trận mà không định nghĩa một kiểu ma trận.

//Định nghĩa hàng

```
#define SIZE 5//kích thước khai báo mảng 2 chiều
```

```
#define TAB '\t'
```

//Định nghĩa kiểu dữ liệu mới:

//Khai báo nguyên mẫu các hàm xử lý, nhập xuất

```
//bổ sung sau
```

//Định nghĩa các hàm xử lý, nhập xuất

```
//bổ sung sau
```

- Trong tập tin **menu.h** :

ta viết lại như sau (cấu trúc giống như bước 9 mục 2 lab 4, chỉ thay đổi nội dung theo yêu cầu bài toán) :

// Khai báo nguyên mẫu các hàm xử lý menu

```
//bổ sung sau
```

// Định nghĩa các hàm xử lý menu

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình

```
// Định nghĩa hàm xuất danh sách chức năng ra màn hình
```

```
// Ngoài các chức năng của bài toán, ta thêm chức năng xem dữ liệu số
```

```
// Input : Không có
```

```
// Output: Không có
```

```
void XuatMenu()
```

```
{
```

```
    cout << endl << "===== CHON CHUC NANG =====";
```

```
    cout << endl << "0. Thoát khỏi chương trình";
```

```
    cout << endl << "1. Nhập ma trận chu nhát m x n";
```

```
    cout << endl << "2. Xem ma trận chu nhát";
```

```
    cout << endl << "3. Tính giá trị lớn nhất của ma trận";
```

```
    cout << endl << "4. Tính giá trị lớn nhất hàng i";
```

```
    cout << endl << "5. Tính tổng các phần tử hàng i";
```

```
    cout << endl << "6. Tính giá trị nhỏ nhất cột j";
```

```
    cout << endl << "7. Tính tích các phần tử cột j";
```

```
    cout << endl << "8. Xuất aij : lớn nhất hàng I và nhỏ nhất cột j";
```

```
    cout << endl << "=====
```

```
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách

```
// Input : soMenu = Số lượng menu có thể chọn.
```

```
// Output: Số thứ tự menu do người dùng nhập vào.
```

```
int ChonMenu(int soMenu)
```

```

{
    int stt;
    for (;;)
    {
        system("CLS");
        XuatMenu();
        cout << "\nNhap 1 so khong khoang [0,..." << soMenu << "] de chon chuc nang, stt = ";
        cin >> stt;
        if (0 <= stt && stt <= soMenu)
            break;
    }
    return stt;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Định nghĩa hàm xử lý menu :

Các thao tác đều thực hiện trên cùng một đầu vào là ma trận mxn, nên ta bổ sung thêm biến ma trận chu nhặt a với kích thước khai báo SIZE x SIZE, m hàng và n cột làm đối của hàm **XuLyMenu**, ngoài tham số đã có là tham số menu. Vì ma trận được nhập trong case 1 của hàm, nên đối m, n trong hàm XuLyMenu ta dùng tham chiếu.

```

// Input : menu = Số thứ tự menu do người chọn,
//          ma trận chu nhặt a
//          số nguyên dương m, n
// Output: Không có.
void XuLyMenu(int menu, int a[SIZE][SIZE], int &m, int &n)
{
    // Khai báo biến

    switch (menu)
    {
        case 0:
            system("CLS");
            cout << endl << "0. Thoát khỏi chương trình\n";
            break;
        case 1:
            system("CLS");
            cout << endl << "1. Nhập ma trận chu nhặt";
            //Bo sung sau
            break;
        case 2:
            system("CLS");
            cout << endl << "2. Xem ma trận chu nhặt";
            //Bo sung sau
            break;
        case 3:
            system("CLS");
            cout << endl << "3. Tính giá trị lớn nhất của ma trận";
            //Bo sung sau
            break;
        case 4:
            system("CLS");

```

```

        cout << endl << "4. Tinh gia tri lon nhat hang i ";
        //Bo sung sau
        break;
    case 5:
        system("CLS");
        cout << endl << "5. Tinh tong cac phan tu hang i ";
        //Bo sung sau
        break;

    case 6:
        system("CLS");
        cout << endl << "6. Tinh gia tri nho nhat cot j ";
        //Bo sung sau
        break;
    case 7:
        system("CLS");
        cout << endl << "7. Tinh tích cac phan tu cot j ";
        //Bo sung sau
        break;
    case 8:
        system("CLS");
        cout << endl << "8. Xuat aij : lon nhat hang i va nho nhat cot j ";
        //Bo sung sau
        break;
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```

void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, int a[SIZE][SIZE], int &m, int &n);

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

- Trong tập tin **program.cpp** :
- Hàm **ChayChuongTrinh** ta cập nhật lại như sau :

```

void ChayChuongTrinh()
{
    int soMenu = 8, //lưu số các chức năng
        menu, // lưu số thứ tự chức năng người dùng chọn
        m = 0, // kích thước hang và giá trị khởi tạo
        n=0; //kích thước cot và giá trị khởi tạo
    int a[SIZE][SIZE];
    do
    {
        menu = ChonMenu(soMenu);
        XuLyMenu(menu,a,m,n);
    } while (menu > 0);
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra sự vận hành của hệ thống menu.

Kiểm tra chức năng thoát khỏi chương trình (chọn 0)

Bước 4 : (tổ chức nhập xuất dữ liệu)

Trong bước 4, ta làm công việc sau :

- Trong **program.cpp** ta bổ sung thêm các thư viện cần thiết.
- Trong **thuvien.h**, soạn thảo các hàm nhập, xuất ma trận
- Trong **menu.h** bổ sung xử lý chức năng nhập ma trận, xem ma trận trong hàm **XuLyMenu**.

- Trong tập tin **program.cpp** bổ sung thêm các thư viện chứa các hàm xử lý số ngẫu nhiên :

```
#include <time.h>
#include <stdlib.h>
```

- Trong tập tin **thuvien.h** :

Bổ sung các hàm nhập xuất :

4.1 Hàm nhập dữ liệu ma trận m x n tự động:

```
void NhapTuDong_MaTran(int a[SIZE][SIZE], int m, int n)
{
    int i, j;
    srand((unsigned)time(NULL));
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            a[i][j] = (rand() % (m*n)) - (m*n) / 2;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.2 Hàm xuất dữ liệu ma trận m x n ra màn hình

```
void XuatMaTran(int a[SIZE][SIZE], int m, int n)
{
    int i, j;
    for (i = 0; i < m; i++)
    {
        cout << endl;
        for (j = 0; j < n; j++)
            cout << a[i][j] << TAB;
    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.3 Bổ sung nguyên mẫu các hàm :

```
void NhapTuDong_MaTran(int a[SIZE][SIZE], int m, int n);
void XuatMaTran(int a[SIZE][SIZE], int m, int n);
```

- Trong tập tin **menu.h** :

Bổ sung xử lý chức năng nhập dữ liệu vào case 1, xem dữ liệu vào case 2 (Các case từ 3 đến 7 giữ nguyên)

```
void XuLyMenu(int menu, int a[SIZE][SIZE], int &m, int &n)
{
    // Khai báo biến

    switch (menu)
    {
```

```

//...
case 1:
    system("CLS");
    cout << endl << "1. Nhập ma trận chu nhát";
    cout << "\nNhập số hàng : m = ";
    cin >> m;
    cout << "\nNhập số cột : n = ";
    cin >> n;
    NhapTuDong_MaTran(a, m, n);
    cout << "\nMa trận vừa nhập:\n";
    XuatMaTran(a, m, n);
    break;
case 2:
    system("CLS");
    cout << endl << "2. Xem ma trận chu nhát";
    cout << "\nMa trận hiện hành:\n";
    XuatMaTran(a, m, n);
    break;
//...
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 1, 2.

Trong các bước tiếp theo, ta bổ sung từng chức năng vào chương trình :

- Lần lượt soạn thảo từng hàm chức năng trong tập tin **thuvien.h**,
- Lần lượt bổ sung xử lý chức năng trong hàm **XuLyMenu** của **menu.h**,

Bước 5: Bổ sung chức năng 3 (tính giá trị lớn nhất của ma trận) chương trình..

- Trong tập tin **thuvien.h** :

5.1 Định nghĩa hàm tính giá trị lớn nhất của ma trận :

```

int Tinh_Max_MaTran(int a[SIZE][SIZE], int m, int n)
{
    int max, i, j;
    max = a[0][0];
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            if (max < a[i][j])
                max = a[i][j];
    return max;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.2 Bổ sung nguyên mẫu các hàm :

```
int Tinh_Max_MaTran(int a[SIZE][SIZE], int m, int n);
```

- Trong tập tin **menu.h** :

Khai báo thêm biến kq kiểu int để lưu trữ kết quả, bổ sung xử lý chức năng 3 vào case 3:

```

void XuLyMenu(int menu, int a[SIZE][SIZE], int &m, int &n)
{

```

```

// Khai báo biến
int kq;
switch (menu)
{
    //...
    case 3:
        system("CLS");
        cout << endl << "3. Tính giá trị lớn nhất của ma trận";
        kq = Tinh_Max_MaTran(a, m, n);
        cout << "\nMa trận hiện hành:\n";
        XuatMaTran(a, m, n);
        cout << "\nMax(a) = " << kq;
        break;
    //...
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 3..

Bước 6: Bổ sung chức năng 4 (tính giá trị lớn nhất hàng i) vào chương trình.

- Trong tập tin **thuvien.h** :

6.1 Định nghĩa hàm tính giá trị lớn nhất hàng i :

```

int Tinh_Max_hang_i(int a[SIZE][SIZE], int i, int n)
{
    int maxi,j;
    maxi = a[i][0]; //hang i cot 0
    for (j = 1; j < n; j++)
        if (maxi < a[i][j])
            maxi = a[i][j];
    return maxi;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

6.2 Bổ sung nguyên mẫu các hàm :

```
int Tinh_Max_hang_i(int a[SIZE][SIZE], int i, int n);
```

- Trong tập tin **menu.h** :

Trong hàm XuLyMenu, khai báo thêm i để lưu trữ giá trị hàng nhập từ bàn phím, bổ sung xử lý chức năng 4 vào case 4:

```

void XuLyMenu(int menu, int a[SIZE][SIZE], int &m, int &n)
{
    // Khai báo biến
    int kq, i;
    switch (menu)
    {
        //...
        case 4:
            system("CLS");

```

```

cout << endl << "4. Tính giá trị lớn nhất hàng i";
do
{
    cout << "\nChọn hàng i (0 <= i <= " << m - 1 << ") : i = ";
    cin >> i;
} while (i < 0 || i > m - 1);
kq = Tinh_Max_hang_i(a, i, n);
cout << "\nMã trận hiện hành:\n";
XuatMaTran(a, m, n);
cout << "\nMax(hàng " << i << ") = " << kq;
break;
//...
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 4..

Bước 7: Bổ sung chức năng 5 (tính tổng hàng i) vào chương trình.

- Trong tập tin **thuvien.h** :

7.1 Định nghĩa hàm tính tổng hàng i :

```

int Tinh_Tong_Hang_i(int a[SIZE][SIZE], int i, int n)
{
    int j, sum;
    sum = 0;
    for (j = 0; j < n; j++)
        sum += a[i][j];
    return sum;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

7.2 Bổ sung nguyên mẫu các hàm :

```
int Tinh_Tong_Hang_i(int a[SIZE][SIZE], int i, int n);
```

- Trong tập tin **menu.h** :

Trong hàm XuLyMenu bổ sung xử lý chức năng 5 vào case 5:

```

void XuLyMenu(int menu, int a[SIZE][SIZE], int &m, int &n)
{
    // Khai báo biến
    int kq, i;
    switch (menu)
    {
        //...
        case 5:
            system("CLS");
            cout << endl << "5. Tính giá trị lớn nhất hàng i";
            do
            {
                cout << "\nChọn hàng i (0 <= i <= " << m - 1 << ") : i = ";

```

```

        cin >> i;
    } while (i < 0 || i > m - 1);
    kq = Tinh_Tong_hang_i(a, i, n);
    cout << "\nMa tran hien hanh:\n";
    XuatMaTran(a, m, n);
    cout << "\nTong (hang " << i << ") = " << kq;
    break;
//...
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 5.

Bước 8: Bổ sung chức năng 6 (tính giá trị nhỏ nhất cột j) vào chương trình..

- Trong tập tin **thuvien.h** :

8.1 Định nghĩa hàm tính giá trị nhỏ nhất cột j :

```

int Tinh_Min_Cot_j(int a[SIZE][SIZE], int m, int j)
{
    int minj, i;
    minj = a[0][j];//hang i cot 0
    for (i = 1; i < m; i++)
        if (minj > a[i][j])
            minj = a[i][j];
    return minj;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

8.2 Bổ sung nguyên mẫu các hàm :

```
int Tinh_Min_Cot_j(int a[SIZE][SIZE], int m, int j);
```

- Trong tập tin **menu.h** :

Trong hàm XuLyMenu, khai báo thêm j để lưu trữ giá trị cột nhập từ bàn phím, bổ sung xử lý chức năng 6 vào case 6:

```

void XuLyMenu(int menu, int a[SIZE][SIZE], int &m, int &n)
{
    // Khai báo biến
    int kq, i,j;
    switch (menu)
    {
        //...
        case 6:
            system("CLS");
            cout << endl << "6. Tinh gia tri nho nhat cot j ";
            do
            {
                cout << "\nChon cot j (0 <= j <= " << n - 1 << ") : j = ";
                cin >> j;
            } while (j < 0 || j > n - 1);
            kq = Tinh_Min_Cot_j(a, m, j);
            cout << "\nMa tran hien hanh:\n";
    }
}

```

```

        XuatMaTran(a, m, n);
        cout << "\nMax(cot " << j << ") = " << kq;
        break;
    //...
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 6.

Bước 9: Bổ sung chức năng 7 (tính tích cột j) vào chương trình..

- Trong tập tin **thuvien.h** :

9.1 Định nghĩa hàm tính tích cột j :

```

int Tinh_Tich_Cot_j(int a[SIZE][SIZE], int m, int j)
{
    int p, i;
    p = 1;
    for (i = 0; i < m; i++)
        p *= a[i][j];
    return p;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

9.2 Bổ sung nguyên mẫu các hàm :

int Tinh_Tich_Cot_j(int a[SIZE][SIZE], int m, int j);

- Trong tập tin **menu.h** :

Trong hàm XuLyMenu bổ sung xử lý chức năng 7 vào case 7:

```

void XuLyMenu(int menu, int a[SIZE][SIZE], int &m, int &n)
{
    // Khai báo biến
    int kq, i,j;
    switch (menu)
    {
        //...
        case 7:
            system("CLS");
            cout << endl << "7. Tinh tích cot j ";
            do
            {
                cout << "\nChon cot j (0 <= j <= " << n - 1 << ") : j = ";
                cin >> j;
            } while (j < 0 || j > n - 1);
            kq = Tinh_Tich_Cot_j(a, m, j);
            cout << "\nMa tran hien hanh:\n";
            XuatMaTran(a, m, n);
            cout << "\nTich(cot " << j << ") = " << kq;
            break;
        //...
    }
}

```

```
_getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 7.

Bước 10: Bổ sung chức năng 8 (Xuat aij : lon nhat hang i va nho nhat cot j) vào chương trình..

- Trong tập tin **thuvien.h** :

10.1 Định nghĩa hàm Xuat aij : lon nhat hang i va nho nhat cot j :

```
void MaxHang_MinCot(int a[SIZE][SIZE], int m, int n)
{
    int i, j;
    int dau = 0;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            if (a[i][j] == Tinh_Max_Hang_i(a, i, n) && a[i][j] == Tinh_Min_Cot_j(a, m, j))
            {
                dau = 1;
                break;
            }
    if (!dau)
        cout << "\nKhong co phan tu nao tho man dieu kien bai toan";
    else
    {
        cout << "\nCac phan tu a[i][j] tho : Max hang i va Min cot j:\n";
        for (i = 0; i < m; i++)
            for (j = 0; j < n; j++)
                if (a[i][j] == Tinh_Max_Hang_i(a, i, n) && a[i][j] == Tinh_Min_Cot_j(a, m, j))
                    cout << "\na[" << i << "][" << j << "] = " << a[i][j]
                    << " : Max hang " << i << " va Min cot " << j;
    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

10.2 Bổ sung nguyên mẫu các hàm :

```
void MaxHang_MinCot(int a[SIZE][SIZE], int m, int n);
```

- Trong tập tin **menu.h** :

Trong hàm XuLyMenu bổ sung xử lý chức năng 8 vào case 8:

```
void XuLyMenu(int menu, int a[SIZE][SIZE], int &m, int &n)
{
    // Khai báo biến
    int kq, i,j;
    switch (menu)
    {
        //...
        case 8:
            system("CLS");
            cout << endl << "8. Xuat aij : lon nhat hang i va nho nhat cot j";
            cout << "\nMa tran hien hanh:\n";
            XuatMaTran(a, m, n);
            MaxHang_MinCot(a, m, n);
    }
}
```

```

        break;
    //...
}
_getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 8.

Kiểm tra tất cả các chức năng – kết thúc chương trình.

Bài 3: Xoắn ốc

Viết chương trình cho phép người dùng nhập vào số nguyên n . Sau đó, xuất ra màn hình ma trận vuông cấp n sau khi đã điền các số từ 1 đến n^2 theo chiều xoắn ốc như hình dưới đây (với $n = 5$)

Máu chốt của chương trình là viết hàm tạo được ma trận xoắn ốc gồm n^2 số nguyên dương đầu tiên như hình vẽ. sau đó xuất ma trận đã tạo được.

Chương trình sẽ tổ chức như mục 1, phần C, lab 4, chỉ có 2 tập tin : program.cpp và thuvien.h (không có tập tin menu.h vì bài toán không yêu cầu tổ chức tùy chọn menu)

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

Bước 1: Tạo dự án Win32 Console Application mới. Đặt tên là **Lab06_D_Bai3**

Bước 2: Tạo cấu trúc chương trình như đã hướng dẫn trong **mục 1 phần C lab 4** (từ bước 1 đến bước 7).

Tức là ta đã có phần chương trình cốt lõi sau :

- Trong tập tin **program.cpp** ta có nội dung sau :

```

#include <iostream>
#include <conio.h>

using namespace std;

#include "thuvien.h"
void ChayChuongTrinh();
int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    _getch();
}

```

- Tập tin **thuvien.h** là rỗng.

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

Bước 3: Bổ sung thao tác xuất ma trận

- Trong tập tin **program.cpp** :

Bổ sung thư viện **<iomanip>** // do xuất có định dạng

- Trong tập tin **thuvien.h** bổ sung các nội dung:

//Định nghĩa hằng

#define MAX 10 //kích thước tối đa ma trận vuông

//Khai báo nguyên mẫu các hàm xử lý

//... (bổ sung sau)

//Định nghĩa các hàm xử lý

3.1 Hàm xuất ma trận vuông

```
void XuatMaTran(int a[MAX][MAX], int n)
{
    Int i, j;
    for (i = 0; i < n; i++)
    {
        cout << "\n\n\n";
        for (j = 0; j < n; j++)
            cout << setw(5) << a[i][j];
    }
}
```

Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

3.2 Khai báo nguyên mẫu :

void XuatMaTran(int a[MAX][MAX], int n);

Bước 4: Bổ xung thao tác tạo ma trận xoắn ốc

Vấn đề chính là : $\forall value \in [1, n^2]$, tìm hàng, cột $\in [0, n-1]$:
 $a[hàng, cột] = value$ sao cho ma trận tạo được là ma trận xoắn (các
 value tăng dần theo vòng xoắn).

Thuật toán thực hiện theo nhiều vòng xoắn:

- **Vòng xoắn đầu tiên :**

+ Gán n value từ 1 đến n cho hàng 0 (đầu tiên).

(tổng quát, ta gọi là hàng trên)

$a[hàngTren][cột] : cột từ 0 đến n-1; hàngTren = 0;$

Để chuẩn bị cho vòng xoắn sau, hàng tăng lên 1 đơn vị (hangTren++)

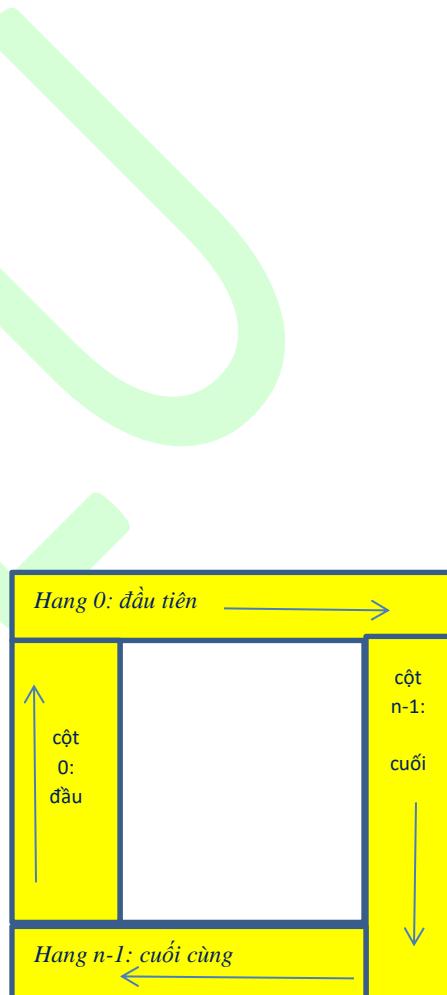
+ Theo chiều xoắn, Gán $n-1$ value kế tiếp cho cột $n-1$ (cột cuối) kể từ hàng 1 (bỏ đi vị trí hàng đầu cột cuối là $a[0][n-1]$ đã gán rồi).

(Tổng quát, ta gọi là cột phải)

$a[hàng][cotPhai] : hàng từ 1 đến n-1; cotPhai = n-1;$

Để chuẩn bị cho vòng xoắn sau, cột phải giảm 1 đơn vị (cotPhai--)

+ Theo chiều xoắn, ta gán ngược $n-1$ value kế tiếp cho hàng $n-1$ (cuối cùng), do $a[n-1][n-1]$ đã được gán rồi.
 (Tổng quát, ta gọi là hàng dưới)



$a[hangDuo][cột] : cột n-2 đến 0; hangDuo = n-1;$

Để chuẩn bị cho vòng xoắn sau, hangDuo giảm 1 đơn vị (hangDuo--)

+ Theo chiều xoắn, ta gán ngược n-2 value kế tiếp cho cột 0 (đầu tiên), do $a[n-1][0]$ và $a[0][0]$ đã được gán rồi.

(Tổng quát, ta gọi là cột trái)

$a[hàng][cotTrai] : hàng n-2 đến 1; cotTrai = 0;$

Để chuẩn bị cho vòng xoắn sau, cotTrai tăng 1 đơn vị (cotTrai+1)

Nhận xét :

- Kết thúc vòng xoắn đầu tiên, các value đã dùng từ 1 đến $4(n-1)$.
- Ma trận sử dụng vòng xoắn đầu cấp n, vòng kế tiếp cấp n-2 (giảm 2 hàng, 2 cột), vòng kế tiếp cấp n-4, ...
- Vòng xoắn từ ngoài vào trong càng nhỏ dần (hàng trên tăng, hàng dưới giảm; cột trái tăng, cột phải giảm), dừng khi mọi vòng xoắn đê đã gan value. (mỗi vị trí của ma trận đã được gán value). Nên cách đơn giản điều khiển vòng lặp dừng là kiểm tra value đã sử dụng đến trị n^2 .

- Các vòng xoắn kế tiếp : ...

Ta có thể viết hàm tạo ma trận xoắn ốc (sắp ma trận n^2 số nguyên dương đầu tiên tăng dần theo chiều xoắn ốc như sau :

Input : n

Output : ma trận xoắn ốc gồm n^2 số nguyên dương đầu tiên

```
void Tao_MaTran_XoanOc(int a[MAX][MAX], int n)
{
    int value, hangTren, hangDuo, cotTrai, cotPhai;
    int i, j;
    //Khai tao gia tri cho hangTren, cotPhai, hangDuo, cotTrai
    hangTren = 0; //hang dau tien
    cotPhai = n - 1; //cot cuoi cung
    hangDuo = n - 1; //hang cuoi cung
    cotTrai = 0; //cot dau tien
    value = 1;

    while (value <= n*n)
    {
        //gan gia tri value cho hang tren
        for (j = cotTrai; (j <= cotPhai) && (value <= n*n); j++) //hang tren
        {
            a[hangTren][j] = value;
            value++;
        }
        if (value > n*n)
            break;
        hangTren++; //chuan bi cho hang ke tiep

        //gan gia tri value cho cot phai
        for (i = hangTren; (i <= hangDuo) && (value <= n*n); i++) //Cot phai
        {
            a[i][cotPhai] = value;
            value++;
        }
        if (value > n*n)
            break;
        hangDuo--;
        cotPhai--;
    }
}
```

```

        value++;
    }

    if (value > n*n)
        break;
cotPhai--; //chuan bi cho cot ke truoc

//Gan gia tri value cho hang duoi
for (j = cotPhai; (j >= cotTrai) && (value <= n*n); j--) //Hang duoi
{
    a[hangDuo][j] = value;
    value++;
}
if (value > n*n)
    break;
hangDuo--; // chuan bi cho hang ke tren

//Gan gia tri value cho cot trai
for (i = hangDuo; (i >= hangTren) && (value <= n*n); i--) //Cot trai
{
    a[i][cotTrai] = value;
    value++;
}
if (value > n*n)
    break;
cotTrai++; //chuan bi cho cot ke tiep
}
//Sau 1 lân xoắn gồm 4 vòng for ma tran bot di 2 hang 2 cot (value tăng theo vòng xoắn)
}

```

Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

4.2 Khai bao nguyên mẫu :

```
void Tao_MaTran_XoanOc(int a[MAX][MAX], int n);
```

Bước 5:

Trong [program.cpp](#), cập nhập lại hàm ChayChuongTrinh, gọi các hàm tạo ma trận xoắn , xuất ma trận xoắn và điều khiển việc lặp thực hiện chương trình.

```

void ChayChuongTrinh()
{
    char kt;
    int a[MAX][MAX], n;
    do
    {
        system("CLS");
        cout << "\nNhap n = ";
        cin >> n;
        Tao_MaTran_XoanOc(a, n);
        system("CLS");
        cout << "\nMa tran xoan oc cap " << n << ":";
        XuatMaTran(a, n);
        _getch();
        cout << "\n\nNua khong, nhan ESC neu khong!\n";
    }
}
```

```

        kt = _getch();
    } while (kt != 27);
}

```

Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

Thực hiện chương trình với n từ 5 đến 10. Kết thúc chương trình.

Bài 4. Các phép toán ma trận

Viết chương trình thực hiện các phép toán sau trên các ma trận:

- Tích hai ma trận cùng cấp
- Hiệu hai ma trận cùng cấp
- Tích hai ma trận

Tạo dự án Win32 Console Application mới. Đặt tên là **Lab06_D_Bai4**.

Chương trình tổ chức như Lab06_D_Bai1. (sinh viên tự viết).

Tham khảo tập tin **thuvien.h** sau :

```

//Dinh nghia hang
#define SIZE 10
//Dinh nghia kieu du lieu moi
typedef int MaTranVuong[SIZE][SIZE];

//Khai bao nguyen mau cac ham:
void NhapMaTran(MaTranVuong a, int n,char kt);
void XuatMaTran(MaTranVuong a, int n);
void TinhTong_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n);
void TinhHieu_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n);
void TinhTich_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n);

//=====
//Dinh nghia cac ham

void NhapMaTran(MaTranVuong a, int n,char kt)
{
    int i, j;
    for (i = 0; i < n; i++) //hang i
        for (j = 0; j < n; j++) //cot j
    {
        cout << endl << kt << "[" << i << "]" << j << "]= ";
        cin >> a[i][j];
    }
}

void XuatMaTran(MaTranVuong a, int n)
{
    int i, j;
    for (i = 0; i < n; i++)
    {
        cout << endl << endl;
        for (j = 0; j < n; j++)
            cout << setw(4) << a[i][j];
    }
}

```

```
//c[i][j] = a[i][j] + b[i][j] ; ∀i,j ∈ {0,..,n-1}
void TinhTong_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n)
{
    int i, j;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            c[i][j] = a[i][j] + b[i][j];
}

//c[i][j] = a[i][j] - b[i][j] ; ∀i,j ∈ {0,..,n-1}
void TinhHieu_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n)
{
    int i, j;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            c[i][j] = a[i][j] - b[i][j];
}

//c[i][j] = a[i][0] b[0][j] + a[i][1] b[1][j] +...+ a[i][n-1] b[n-1][j] ; ∀i,j ∈ {0,..,n-1}
void TinhTich_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n)
{
    int i, j, k;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
        {
            c[i][j] = 0;
            for (k = 0; k < n; k++)
                c[i][j] += a[i][k] * b[k][j];
        }
}
```

E. Bài tập bắt buộc

Tất cả các bài tập phải được tổ chức dưới dạng thư viện hàm và có thể có/không có hệ thống tùy chọn menu.

- Viết chương trình nhập vào một ma trận vuông cấp n với n là số nguyên dương, các phần tử của ma trận là số thực. Xuất ra màn hình ma trận đã nhập, tính và in ra màn hình các giá trị S-T, trong đó:
 - $S = \sum_{i=0}^{n-1} h_i$ với h_i là số dương nhỏ nhất của hàng thứ i .
 - $T = \sum_{j=0}^{n-1} v_j$ với v_j là số âm lớn nhất của cột thứ j .

2. Kiểm tra tính chất ma trận vuông

Viết chương trình thực hiện các thao tác trên các ma trận vuông cấp n . Yêu cầu của chương trình:

- Kiểm tra ma trận có phải là ma trận đối xứng.
- Kiểm tra ma trận có phải là ma trận tam giác trên.
- Kiểm tra ma trận có phải là ma trận tam giác dưới.
- Kiểm tra ma trận có phải là ma trận chéo
- Kiểm tra ma trận có phải là ma trận đơn vị

3. Ma trận vuông

Viết chương trình thực hiện các thao tác trên các ma trận vuông cấp n . Yêu cầu của chương trình:

- Hoán vị hai cột j và h của ma trận, xuất kết quả.
- Hoán vị hai hàng i và k của ma trận, in kết quả.
- Tìm ma trận hoán vị

4. In lịch

Viết chương trình in ra lịch của tháng M trong năm N cho trước theo một trong 2 mẫu trong hình dưới đây.
(xem lại Lab 1, Lab 3 để biết cách tính thứ trong tuần và số ngày trong một tháng.)

Ví dụ: Lịch của tháng 1 năm 2015.

Mon	Tue	Wed	Thu	Fri	Sat	Sun	Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4				1	2	3	
5	6	7	8	9	10	11	4	5	6	7	8	9	10
12	13	14	15	16	17	18	11	12	13	14	15	16	17
19	20	21	22	23	24	25	18	19	20	21	22	23	24
26	27	28	29	30	31		25	26	27	28	29	30	31

5. Đổi đơn vị tiền tệ

Viết chương trình cho phép nhập vào một số tiền M thuộc một trong các loại tiền tệ sau: Đô-la Mỹ, Yên Nhật, Bảng Anh, Euro, Baht Thái, VN Đồng, Nhân dân tệ (Trung Quốc), Won (Hàn Quốc), Ruble (Nga) và rupee (Ấn Độ). Sau đó, xuất ra kết quả đổi M sang các loại tiền tệ khác.

Xem bảng tỷ giá giữa các loại tiền tệ tại: <http://www.xe.com/>.

F. Bài tập làm thêm

1. Bãi mìn

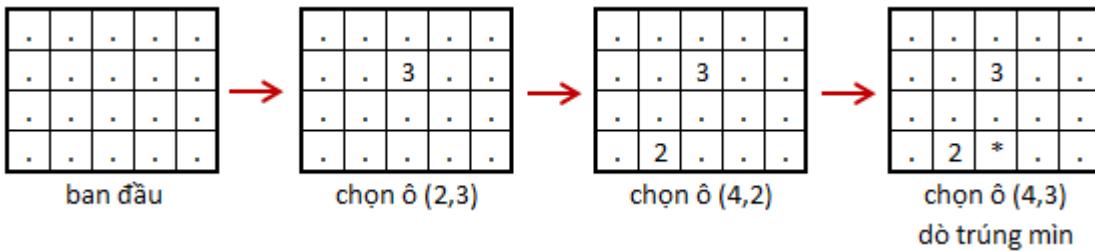
Xem bãi mìn như một lưới (ma trận) các ô. Hãy viết chương trình cho phép người dùng nhập vào 3 giá trị m , n và p , trong đó, m , n là kích thước của bãi mìn, p là số quả mìn. Sau đó, đặt ngẫu nhiên p quả mìn vào các ô trong lưới. Với các ô còn lại, hãy điền vào số quả mìn ở 8 ô xung quanh nó. Xuất ra màn hình bãi mìn đã tạo, ký hiệu quả mìn là dấu *.

Ví dụ: với $m = 4$, $n = 5$, $p = 6$, hình dưới đây cho thấy một kết quả được tạo ra từ chương trình.

ô hiện tại và 8 ô lân cận	Sau khi đặt mìn	Sau khi tính số mìn xung quanh

2. Trò chơi dò mìn

Viết chương trình tạo ra một bãi mìn như trong bài tập 1 ở trên. Ban đầu, chương trình hiển thị bãi mìn chỉ là lưới các dấu chấm. Sau đó, chương trình cho phép người chơi nhập vị trí (hàng, cột) của các ô muốn dò mìn. Nếu người chơi mở trúng ô không có mìn thì hiển thị số lượng mìn ở các ô xung quanh tại vị trí được chọn. Chương trình dừng khi người chơi dò trúng ô có mìn hoặc tất cả các ô không chứa mìn đã được mở và thông báo kết quả.



Cài tiến chương trình trò chơi trên bằng cách cho người chơi k mạng ($k << p$). Nghĩa là người chơi được phép dò trúng k quả mìn trước khi chép (chương trình dừng).

3. X-O-Empty

Cho một ma trận vuông cấp n trong đó, mỗi ô chứa một trong hai giá trị X, O hoặc là ô trống. Viết chương trình tìm ra dãy liên tiếp các ô (theo chiều ngang, chiều dọc, chéo) dài nhất chứa giá trị X.

4. Sudoku (Số độc)

Sudoku là một loại trò chơi logic và cách chơi là điền các số từ 1 tới 9 vào những ô trống (trong một lưới 9x9 ô như hình bên trái) sao cho mỗi cột dọc, mỗi hàng ngang, mỗi vùng nhỏ 3x3 có đủ các số từ 1 tới 9 mà không được lặp lại. Hình bên phải là một cách điền (lời giải) như vậy.

Cho một lưới 9x9 ô chứa các số từ 1 tới 9. Hãy viết chương trình kiểm tra xem đó có phải là một lời giải của câu đó Sudoku hay không.

5	3			7				
6			1	9	5			
	9	8				6		
8			6				3	
4			8	3			1	
7			2				6	
	6				2	8		
		4	1	9			5	
		8			7	9		

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Cài tiến chương trình trên để người dùng có thể chơi Sudoku.

5. Ma phương lẻ

Ma phương là một ma trận vuông cấp n được tạo ra từ một dãy các số nguyên liên tiếp từ 1 tới n^2 trong đó, tổng các số trên mỗi hàng bằng tổng các số trên mỗi cột và bằng tổng các số trên các đường chéo. Giá trị tổng này gọi là hằng số ma phương.

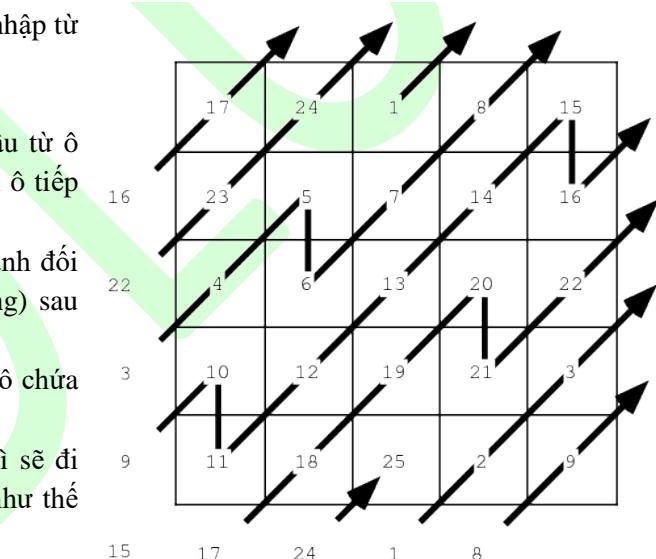
8	1	6	15
3	5	7	15
4	9	2	15
15	15	15	15

30	39	48	1	10	19	28	175
38	47	7	9	18	27	29	175
46	6	8	17	26	35	37	175
5	14	16	25	34	36	45	175
13	15	24	33	42	44	4	175
21	23	32	41	43	3	12	175
22	31	40	49	2	11	20	175
65	65	65	65	65	65	65	65

Viết chương trình xuất ra ma phương bậc n với n lẻ được nhập từ bàn phím.

Hướng dẫn cách lập ma phương lẻ:

- Số 1 luôn viết ở ô chính giữa của dòng 1. Bắt đầu từ ô này, luôn di chuyển theo hướng Đông-Bắc để đến ô tiếp theo.
- Khi mũi tên đi ra ngoài một cạnh thì sẽ đi vào cạnh đối diện với cạnh đó (ví dụ ô 1->2, 3->4) ở cột (hàng) sau (trước).
- Khi mũi tên gặp góc trên cùng bên phải (ở đây là ô chứa số 15) thì sẽ chạy xuống ô nằm ngay bên dưới.
- Khi mũi tên gặp một ô đã có số trong đó rồi thì sẽ đi xuống ô nằm ngay bên dưới (5->6, 20->21). Cứ như thế cho đến khi đi hết các ô trong ma phương.



LAB 7. CÁC KỸ THUẬT XỬ LÝ XÂU KÝ TỰ

THỜI LƯỢNG: 6 TIẾT

A. Mục tiêu

- Giúp sinh viên hiểu rõ và thực hiện thuần thục các kỹ thuật xử lý xâu ký tự (chuỗi).
- Tiếp tục rèn luyện kỹ năng phát triển chương trình từng bước theo chức năng..
- Sau khi hoàn thành bài thực hành này, sinh viên :
 - Nắm vững các khái niệm, định nghĩa và thao tác nhập, xuất xâu ký tự.
 - Nắm vững các kỹ thuật xử lý cơ bản trên xâu ký tự.
 - Biết cách định nghĩa và sử dụng kiểu dữ liệu xâu ký tự bằng từ khóa **typedef**.
 - Phân biệt được mảng 1 chiều và xâu ký tự.
 - Truyền tham số thực, truyền tham biến.

B. Yêu cầu

- Trong phần C (Ôn tập lý thuyết), sinh viên tự đọc.
- Sinh viên cần chuẩn bị bài trước để thực hành đủ khối lượng yêu cầu.

Buổi thực tập thứ 9 (4 tiết) :

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần D (Hướng dẫn thực hành) của lab 7.
 - Yêu cầu lưu trữ :
 - Tạo thư mục MaSV_Lab07_HD chứa trong ổ đĩa qui định.
 - Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - Mỗi project, xóa thư mục debug.
 - Sử dụng lại tập tin word LoiChuongTrinh.docx trong lab6 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - Thu bài : Xem thông báo của giáo viên .

Buổi thực tập thứ 10 (4 tiết) :

• 2 tiết đầu:

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần E (Bắt buộc) của lab 7.
 - Yêu cầu lưu trữ :
 - Tạo thư mục MaSV_Lab07_BB chứa trong ổ đĩa qui định.
 - Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - Mỗi project, xóa thư mục debug.
 - Sử dụng lại tập tin word LoiChuongTrinh.docx từ lab7 (hướng dẫn) để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - Giáo viên sẽ thu bài qua hệ thống thu bài trực tuyến tại phòng lab (thư mục MaSV_Lab07_BB) trước giờ giải lao.

C. Ôn tập lý thuyết

1. Cú pháp khai báo biến chuỗi ký tự

Cú pháp: **char** **Tên_bien_chuoi** [Số_ký_tự_tối_da];
Hoặc: **wchar_t** **Tên_bien_chuoi** [Số_ký_tự_tối_da];

Trong đó:

- **Tên_bien_chuoi**: là tên biến chuỗi ký tự, do người lập trình đặt và tuân theo quy tắc đặt tên.
- **Số_ký_tự_tối_da** là một nguyên dương, cho biết số ký tự tối đa có thể lưu trong biến chuỗi.

2. Cú pháp định nghĩa kiểu dữ liệu chuỗi ký tự

Cú pháp: **typedef char** **Tên_kieu_chuoi** [Số_ký_tự_tối_da];

Ví dụ: **#define MAX 100**
 typedef char Chuoi [MAX];

Khi đó:

- **Chuoi**: trở thành 1 kiểu dữ liệu, là kiểu chuỗi ký tự.
- Ta có thể khai báo các biến thuộc kiểu này: **Chuoi a, b;**

3. Các thao tác nhập - xuất chuỗi ký tự

e. Nhập xâu ký tự từ bàn phím

Với khai báo xâu ký tự: **char a[MAX]**; hoặc **Chuoi a;**

Ta sử dụng 1 trong 2 cách sau để nhập dữ liệu cho **a**:

- **cin >> a;** Cách này không thể nhập chuỗi có chứa khoảng trắng.
- **gets_s(a);** **a** có chứa khoảng trắng, phải dùng hàm **_flushall()**; trước khi gọi hàm **gets_s()**.
- hoặc **gets_s(a, MAX);**

Lưu ý quan trọng:

- Một xâu ký tự bao giờ cũng phải kết thúc bởi ký tự **NULL** (**\0**).
- Đối với các cách trên, khi kết thúc việc nhập (nhấn Enter), trình biên dịch sẽ tự động thêm ký tự **NULL** vào cuối chuỗi.
- Nếu ta dùng cách nhập từng ký tự như mảng một chiều thông thường (bằng 1 vòng lặp for) thì khi đó, **a** chỉ là mảng một chiều có chứa **n** các ký tự khác **NULL**. Để **a** biến thành chuỗi ký tự chứa **n** ký tự khác **NULL**, cần thêm ký tự **NULL** vào cuối mảng, tức là:

```
for (int i=0, i<n; i++)
    cin >> a[i];
a[n] = NULL;
```
- Ký tự đầu tiên của chuỗi **a** bao giờ cũng tương ứng với chỉ số 0, nghĩa là: **a[0]** là ký tự đầu tiên trong chuỗi.
Nếu **a** là chuỗi rỗng thì **a[0] = NULL**.
- Nếu chuỗi **a** có chứa **n** ký tự khác **NULL** (hay chiều dài là **n**) thì **a[n] = NULL**.

f. Xuất giá trị của chuỗi ra màn hình

Dùng lệnh **cout** như khi xuất các biến khác: **cout << a;**

4. Duyệt các ký tự trong chuỗi

Sử dụng tín hiệu kết thúc xâu ký tự là **NULL**, ta thường dùng một trong hai cách sau để duyệt:

<pre>for (int i=0; a[i] != NULL; i++) // Xử lý a[i] ở đây</pre>	<pre>int i=0; while (a[i] != NULL) { // Xử lý a[i] ở đây i++; }</pre>
---	---

5. Một số hàm trong thư viện <string.h> :

Tên hàm	Chức năng	Nguyên mẫu	Sử dụng
<code>gets_s</code>	Nhập dữ liệu cho chuỗi (không quá MAX ký tự)	char * __cdecl <code>gets_s</code> (char * _Buf, _In_rsize_t _Size)	<code>gets_s(a,MAX);</code>
<code>_flushall()</code>	Làm trống (xóa) vùng đệm bàn phím	int __cdecl <code>_flushall()</code>	<code>_flushall();</code> <code>int t = _flushall();</code>
<code>strlen</code>	Tính chiều dài chuỗi	size_t <code>strlen</code> (const char *a);	<code>int l = strlen(a);</code>
<code>strcat_s</code>	Nối chuỗi (nối chuỗi sau vào cuối chuỗi trước)	errno_t __cdecl <code>strcat_s</code> (char * _Dst, _In_rsize_t _SizeInBytes, _In_z const char * _Src)	<code>Strcat_s(a,MAX,b);</code>
<code>strcpy_s</code>	Sao chép chuỗi (chép chuỗi sau sang chuỗi trước với không quá MAX ký tự)	errno_t <code>strcpy_s</code> (char *strDestination, size_t numberOfElements, const char *strSource);	<code>strcpy_s(a,MAX,b)</code>
<code>strcmp</code>	So sánh 2 chuỗi (có phân biệt ký tự thường, hoa)	int <code>strcmp</code> (const char *s1, const char *s2);	$strcmp(s1, s2) = q \begin{cases} < 0; s1 < s2 \\ = 0; s1 \equiv s2; \\ > 0; s1 > s2 \end{cases}$
<code>_stremp<i>i</i></code>	So sánh 2 chuỗi (0 phân biệt ký tự thường, hoa)	int <code>_stremp<i>i</i></code> (const char *s1, const char *s2);	$strempi(s1, s2) = q \begin{cases} < 0; s1 < s2 \\ = 0; s1 \equiv s2; \\ > 0; s1 > s2 \end{cases}$
...			

Ghi chú : có thể dùng

- `cin.getline(chuoi, MAX);` //Nhập chuỗi chuoi không qua MAX ký tự
- `cin.ignore();` //xóa bộ đệm (thay `_flushall()`).

D. Hướng dẫn thực hành

Phần này xây dựng chương trình thực hiện các thao tác trên cấu trúc dữ liệu xâu ký tự với các chức năng cơ bản.

Tiếp tục tổ chức chương trình theo thư viện và có/không có hệ thống menu.

Bài 1:

Viết chương trình thực hiện các thao tác trên các chuỗi (xâu ký tự) , với các chức năng cơ bản :

0. Thoát khỏi chương trình

1. Nhập chuỗi

2. Xem chuỗi
3. Tính chiều dài chuỗi
4. Nối chuỗi
5. Sao chép chuỗi
6. So sánh chuỗi theo thứ tự từ điển (không phân biệt ký tự thường, HOA)
7. So sánh chuỗi theo thứ tự từ điển (có phân biệt ký tự thường, HOA)
(Các chức năng từ 3 đến 7, vừa dùng các hàm thư viện trong string.h, vừa tự cài đặt).

Bước 3. Tạo dự án Win32 Console Application mới. Đặt tên là **Lab07_D_Bai1**

Bước 4. Tạo cấu trúc cho chương trình như đã hướng dẫn trong **mục 2 lab 4** (từ 1- 8 để có cấu trúc nội dung tối thiểu chạy được chương trình).

Tức là ta có kết quả chương trình ở bước này như sau :

- Tập tin **thuvien.h** : Rỗng
- Tập tin **menu.h** : Rỗng
- Tập tin **program.cpp** có nội dung như sau :

```
#include <iostream>
#include <conio.h>
```

```
using namespace std;
```

```
#include "thuvien.h"
```

```
#include "menu.h"
```

```
void ChayChuongTrinh();
```

```
int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    _getch();
}
```

Nhấn Ctrl + F5 để chạy chương trình, sửa lỗi nếu có.

Bước 3:

Bước này ta định nghĩa kiểu dữ liệu mới; tổ chức và vận hành hệ thống menu.

- Trong tập tin **thuvien.h** :

Ta bổ sung định nghĩa hằng, kiểu dữ liệu mới :

Vì chương trình thực hiện các thao tác trên xâu ký tự, nên ta cần định nghĩa một hằng là giá trị kích thước khai báo của xâu. Ngoài ra, ta có thể định nghĩa một kiểu dữ liệu xâu ký tự (lưu ý rằng có thể không cần thực hiện định nghĩa này vì ta có thể làm trực tiếp trên biến xâu ký tự)

//Dinh nghia hang

```
#define MAX 100//kich thuoc khai bao xau ky tu
```

//Dinh nghia kieu du lieu moi:

```
typedef char String[MAX]; //kieu xau ky tu dat ten String
```

```
//Khai báo nguyên mẫu các hàm xử lý, nhập xuất
//bổ sung sau
```

```
//Định nghĩa các hàm xử lý, nhập xuất
//bổ sung sau
```

- Trong tập tin **menu.h** :

```
// Khai báo nguyên mẫu các hàm xử lý menu
//bổ sung sau
//Định nghĩa các hàm xử lý menu
```

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình

```
void XuatMenu()
{
    cout << "\n===== Bang Menu =====";
    cout << "\n0. Thoat khoi chuong trinh";
    cout << "\n1. gets_s : Nhap chuoi";
    cout << "\n2. Xem chuoi";
    cout << "\n3. strlen_Tinh chieu dai chuoi";
    cout << "\n4. strcat_s_Noi chuoi sau vao sau chuoi truoc";
    cout << "\n5. strcpy_s_Chep chuoi sau vao chuoi truoc";
    cout << "\n6. _strcmpi_So sanh chuoi _ khong phan biet KT thuong, HOA";
    cout << "\n7. strcmp_So sanh chuoi _ phan biet KT thuong, HOA";
    //=====
    cout << "\n8. Noi chuoi sau vao sau chuoi truoc";
    cout << "\n9. Chep chuoi sau qua chuoi truoc";
    cout << "\n10. So sanh chuoi _ khong phan biet KT thuong, HOA";
    cout << "\n11. So sanh chuoi _ phan biet KT thuong, HOA";
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách

```
// Input : soMenu = Số lượng menu có thể chọn.
// Output: Số thứ tự menu do người dùng nhập vào.
```

```
int ChonMenu(int soMenu)
{
    int stt;
    for (;;)
    {
        system("CLS");
        XuatMenu();
        cout << "\nNhap 1 so khong khoang [0,...," << soMenu << "] de chon chuc nang, stt = ";
        cin >> stt;
        if (0 <= stt && stt <= soMenu)
            break;
    }
    return stt;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Định nghĩa hàm xử lý menu :

Các thao tác thường thực hiện trên cùng một đầu vào là 2 chuỗi, nên ta bổ sung thêm 2 biến chuỗi kiểu String làm đối của hàm **XuLyMenu**, ngoài tham số đã có là tham số menu.

```
void XuLyMenu(int menu, String a, String b)
{
    //khai bao bien

    switch (menu)
    {
        case 0:
            system("CLS");
            cout << "\n0. Thoat khoi chuong trinh\n";
            break;
        case 1:
            system("CLS");
            cout << "\n1. gets_s_Nhap chuoi a: ";
            break;
        case 2:
            system("CLS");
            cout << "\n2. Xem chuoi";
            break;
        case 3:
            system("CLS");
            cout << "\n3. strlen_Tinh chieu dai chuoi";
            break;
        case 4:
            system("CLS");
            cout << "\n4. strcat_s_Noi chuoi sau vao sau chuoi truoc";
            break;
        case 5:
            system("CLS");
            cout << "\n5. strcpy_s_Chep chuoi sau qua chuoi truoc";
            break;
        case 6:
            system("CLS");
            cout << "\n6. _strncpy _ So sanh chuoi _khong phan biet KT thuong, HOA";
            break;
        case 7:
            system("CLS");
            cout << "\n7. strcmp _ So sanh chuoi _ phan biet KT thuong, HOA";
            //=====
        case 8:
            system("CLS");
```

```
cout << "\n8. Nối chuỗi sau vào sau chuỗi trước";
break;

case 9:
    system("CLS");
    cout << "\n9. Chèp chuỗi sau qua chuỗi trước";
    break;

case 10:
    system("CLS");
    cout << "\n10. So sánh chuỗi _không phân biệt KT thường, HOA";
    break;

case 11:
    system("CLS");
    cout << "\n7. So sánh chuỗi _ phân biệt KT thường, HOA";
    break;

}

 getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```
void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, String a, String b);
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

- Trong tập tin **program.cpp** :
 - + Hàm **ChayChuongTrinh** ta cập nhật lại như sau :

```
void ChayChuongTrinh()
{
    int menu,
        soMenu = 11;
    String a,b;
    do
    {
        system("CLS");
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, a,b);
    } while (menu > 0);

}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra sự vận hành của hệ thống menu.

Kiểm tra chức năng thoát khỏi chương trình (chọn 0).

Bước 4 : Bổ sung vào chương trình các thao tác nhập xuất dữ liệu

Trong bước 4 này, ta làm công việc sau :

- Trong **program.cpp** Bổ sung thêm thư viện **<string.h>**

- Trong tập tin **thuvien.h**, soạn thảo các hàm nhập, xuất chuỗi
- Trong tập tin **menu.h** bổ sung xử lý chức năng nhập chuỗi,xem chuỗi trong hàm **XuLyMenu**.

- Trong tập tin *thuvien.h* :

Bổ sung các hàm nhập xuất :

4.1 Hàm nhập chuỗi

(Sử dụng hàm gets_s trong string.h)

```
void gets_s_NhapChuoi(String a, char kt)
{
    cout << "\nNhap chuoi: " << kt << " = ";
    _flushall();
    gets_s(a,MAX);
}
```

Ghi chú : Có thể không cần viết hàm, mà dùng trực tiếp hàm gets_s

4.2 Hàm xuất dữ liệu ma trận vuông ra màn hình

```
//Xuat chuoi
void XuatChuoi(String a)
{
    cout << a;
}
```

Ghi chú : có thể dùng trực tiếp cout<<.

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.3 Bổ sung nguyên mẫu các hàm :

```
void gets_s_NhapChuoi(String a, char kt);
void XuatChuoi(String a);
```

- Trong tập tin *menu.h* :

Bổ sung xử lý chức năng nhập chuỗi vào case 1, xem chuỗi vào case 2 (Các case từ 3 đến 11 giữ nguyên)

```
void XuLyMenu(int menu, String a, String b)
{
    //khai bao bien
    int kq;
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << "\n0. Thoat khoi chuong trinh\n";
            break;
        case 1:
            system("CLS");
            cout << "\n1. gets_s_Nhap chuoi a: ";
            gets_s_NhapChuoi(a, 'a');
            cout << "\nChuoi a vua nhap: ";
            XuatChuoi(a);

            NhapChuoi(b, 'b');
            cout << "\nChuoi b vua nhap: ";
            XuatChuoi(b);
    }
}
```

```
cout << "\nNhan phim bat ky de tiep tuc";  
  
break;  
  
case 2:  
    system("CLS");  
    cout << "\n2. Xem chuoi";  
    cout << "\nChuoi a: ";  
    XuatChuoi(a);  
    cout << "\nChuoi b: ";  
    XuatChuoi(b);  
    cout << "\nNhan phim bat ky de tiep tuc";  
  
break;  
  
//...  
}  
  
_getch();  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 1, 2.

Trong các bước tiếp theo, ta bổ sung lần lượt từng chức năng vào chương trình :

- Lần lượt soạn thảo từng hàm chức năng trong tập tin **thuvien.h**,
- Lần lượt bổ sung xử lý chức năng trong hàm **XuLyMenu** của **menu.h**,

Bước 5 : Bổ sung chức năng tính chuỗi vào chương trình

- Trong tập tin **thuvien.h** :

5.1 Hàm tính chiều dài chuỗi

(Sử dụng hàm **strlen** trong **string.h**)

```
int strlen_TinhChieuDaiChuoi(String a)  
{  
    int l = strlen(a);  
    return l;  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.2 Bổ sung nguyên mẫu các hàm :

```
int strlen_TinhChieuDaiChuoi(String a);
```

- Trong tập tin **menu.h** :

Trong hàm **XuLyMenu** khai báo thêm biến **kq** kiểu **int** để lưu trữ kết quả tính toán, bổ sung xử lý chức năng tính chiều dài chuỗi vào case 3 (Các case khác giữ nguyên)

```
void XuLyMenu(int menu, String a, String b)  
{  
    //khai bao bien
```

```

int kq;
switch (menu)
{
    //...
    case 3:
        system("CLS");
        cout << "\n3. strlen_Tinh chieu dai chuoi";
        cout << "\nChuoi a: ";
        XuatChuoi(a);
        cout << "\nchieu dai chuoi a: l = " << strlen_TinhChieuDaiChuoi(a);
        cout << "\nChuoi b: ";
        XuatChuoi(b);
        cout << "\nchieu dai chuoi b: l = " << strlen_TinhChieuDaiChuoi(b);
        cout << "\nNhan phim bat ky de tiep tuc";
        break;
    //...
}

} //getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 3.

Bước 6 :

Bổ sung chức năng nối chuỗi vào chương trình

- Trong tập tin *thuvien.h* :

6.1 Hàm nối chuỗi

(Sử dụng hàm *strcat_s* trong *string.h*, nối chuỗi nguồn *b* vào cuối chuỗi đích *a*)

```

void strcat_s_Noi_ChuoiSau_VaoSau_ChuoiTruoc(String a, String b)
{
    strcat_s(a, MAX, b);
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

6.2 Bổ sung nguyên mẫu các hàm :

```
void strcat_s_Noi_ChuoiSau_VaoSau_ChuoiTruoc(String a, String b);
```

- Trong tập tin *menu.h* :

Bổ sung xử lý chức năng nối chuỗi vào case 4 (Các case khác giữ nguyên)

```

void XuLyMenu(int menu, String a, String b)
{
    //khai bao bien
    int kq;
    switch (menu)
    {
        //...
        case 4:
            system("CLS");
            cout << "\n4. strcat_s_Noi chuoi sau vao sau chuoi truoc";
}

```

```
cout << "\nChuoi truoc : a = ";
XuatChuoi(a);
cout << "\nChieu sau : b = ";
XuatChuoi(b);

strcat_s_Noi_ChuoiSau_VaoSau_ChuoiTruoc(a, b);
cout << "\nChuoi truoc sau khi noi : a = ";
XuatChuoi(a);

break;
//...

}

_getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 4.

Bước 7 :

Bổ sung chức năng sao chép chuỗi vào chương trình

- Trong tập tin *thuvien.h* :

7.1 Hàm sao chép chuỗi

(Sử dụng hàm strcpy_s trong string.h)

```
void strcpy_s_Chep_ChuoiSau_Qua_ChuoiTruoc(String a, String b)
{
    strcpy_s(a, MAX, b);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

7.2 Bổ sung nguyên mẫu các hàm :

```
void strcpy_s_Chep_ChuoiSau_Qua_ChuoiTruoc(String a, String b);
```

- Trong tập tin *menu.h* :

Bổ sung xử lý chức năng sao chép chuỗi vào case 5 (Các case khác giữ nguyên)

```
void XuLyMenu(int menu, String a, String b)
{
    //khai bao bien
    int kq;
    switch (menu)
    {
        //...
        case 5:
            system("CLS");
            cout << "\n5. strcpy_s_Chep chuoi sau qua chuoi truoc";
            cout << "\nChuoi sau : b = ";
            XuatChuoi(b);
    }
}
```

```
strcpy_s_Chep_ChuoiSau_Qua_ChuoiTruoc(a, b);
cout << "\nChuoi truoc a,do b chep qua : a = ";
XuatChuoi(a);
break;
//...
}

 getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 5.

Bước 8 :

Bổ sung chức năng so sánh chuỗi không phân biệt ký tự thường hoa vào chương trình

- Trong tập tin *thuvien.h* :

8.1 Hàm so sánh chuỗi không phân biệt ký tự thường hoa
(Sử dụng hàm *_strcmpl* trong *string.h*)

```
//So sanh 2 chuoi theo thu tu tu dien : khong phan biet thuong hoa
int _strcmpl_SoSanhChuoi_KPB(String a, String b)
{
    return _strcmpl(a, b);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

8.2 Bổ sung nguyên mẫu các hàm :

```
int _strcmpl_SoSanhChuoi_KPB(String a, String b);
```

- Trong tập tin *menu.h* :

Bổ sung xử lý chức năng so sánh chuỗi chuỗi vào case 6 (Các case khác giữ nguyên)

```
void XuLyMenu(int menu, String a, String b)
{
    //khai bao bien
    int kq;
    switch (menu)
    {
        //...
        case 6:
            system("CLS");
            cout << "\n6. strcmpl _ So sanh chuoi _khong phan biet KT thuong, HOA";
            cout << "\nChuoi a = ";
            XuatChuoi(a);
            cout << "\nChieu b = ";
            XuatChuoi(b);

            kq = _strcmpl_SoSanhChuoi_KPB(a, b);
            if (kq > 0)
```

```
        cout << "\na > b";
else
    if(kq < 0)
        cout << "\na < b";
    else
        cout << "\na == b";
break;

//...

}

 getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 6.

Bước 9 :

Bổ sung chức năng so sánh chuỗi có phân biệt ký tự thường hoa vào chương trình

- Trong tập tin *thuvien.h* :

9.1 Hàm so sánh chuỗi có phân biệt ký tự thường hoa

(Sử dụng hàm strcmp trong string.h)

```
//So sanh 2 chuoi theo thu tu tu dien : co phan biet thuong hoa
int strcmp_SoSanhChuoi_PB(String a, String b)
{
    return strcmp(a, b);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

9.2 Bổ sung nguyên mẫu các hàm :

```
int strcmp_SoSanhChuoi_PB(String a, String b);
```

- Trong tập tin *menu.h* :

Bổ sung xử lý chức năng so sánh chuỗi chuỗi vào case 6 (Các case khác giữ nguyên)

```
void XuLyMenu(int menu, String a, String b)
{
    //khai bao bien
    int kq;
    switch (menu)
    {
        //...
        case 7:
            system("CLS");
            cout << "\n7. strcmp _ So sanh chuoi _ phan biet KT thuong, HOA";
            cout << "\nChuoi a = ";
            XuatChuoi(a);
            cout << "\nChieu b = ";
            XuatChuoi(b);
```

```

kq = strcmp_SoSanhChuoi_PB(a, b);
if (kq > 0)
    cout << "\na > b";
else
if (kq < 0 )
    cout << "\na < b";
else
    cout << "\na == b";
break;
//...
}

 getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 7.

Bước 10 :

Bổ sung chức năng nối chuỗi (tự viết, không dùng hàm thư viện) vào chương trình
- Trong tập tin **thuvien.h** :

10.1 Hàm nối chuỗi

```

//ham noi chuoi b vao cuoi chuoi a
void Noi_ChuoiSau_VaoSau_ChuoiTruoc(String a, String b)
{
    int i, l;
    l = strlen_TinhChieuDaiChuoi(a);
    for (i = 0; b[i] != NULL; i++)
        a[l++] = b[i];
    a[l] = NULL;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

10.2 Bổ sung nguyên mẫu các hàm :

void Noi_ChuoiSau_VaoSau_ChuoiTruoc(String a, String b);
- Trong tập tin **menu.h** :

Bổ sung xử lý chức năng nối chuỗi vào case 8 (Các case khác giữ nguyên)

```

void XuLyMenu(int menu, String a, String b)
{
    //khai bao bien
    int kq;
    switch (menu)
    {
        //...
        case 8:
            system("CLS");
            cout << "\n4. strcat_s_Noi chuoi sau vao sau chuoi truoc";
            cout << "\nChuoi truoc : a = ";
    }
}

```

```

XuatChuoi(a);
cout << "\nChieu sau : b = ";
XuatChuoi(b);

Noi_ChuoiSau_VaoSau_ChuoiTruoc(a, b);
cout << "\nChuoi truoc sau khi noi : a = ";
XuatChuoi(a);

break;
//...
}

 getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 8.

Bước 11 :

Bổ sung chức năng sao chép chuỗi (không dùng hàm thư viện) vào chương trình

- Trong tập tin *thuvien.h* :

11.1 Hàm sao chép chuỗi

```

//Ham sao chep chuoi b sang a
void Chep_ChuoiSau_Qua_ChuoiTruoc(String a, String b)
{
    int i;
    for (i = 0; (a[i] = b[i]) != NULL; i++);
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

11.2 Bổ sung nguyên mẫu các hàm :

```
void Chep_ChuoiSau_Qua_ChuoiTruoc(String a, String b);
```

- Trong tập tin *menu.h* :

Bổ sung xử lý chức năng sao chép chuỗi vào case 9 (Các case khác giữ nguyên)

```

void XuLyMenu(int menu, String a, String b)
{
    //khai bao bien
    int kq;
    switch (menu)
    {
        //...

        case 9:
            system("CLS");
            cout << "\n9. Chep chuoi sau qua chuoi truoc";
            cout << "\nChuoi sau : b = ";

```

```
XuatChuoi(b);  
  
Chep_ChuoiSau_Qua_ChuoiTruoc(a, b);  
cout << "\nChuoi truoc a,do b chep qua : a = ";  
XuatChuoi(a);  
  
break;  
  
//...  
}  
  
_getch();  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 9.

Bước 12 :

Bổ sung chức năng so sánh chuỗi không phân biệt ký tự thường hoa vào chương trình

- Trong tập tin *thuvien.h* :

```
12.1 Hàm so sánh chuỗi không phân biệt ký tự thường hoa  
//So sanh 2 chuoi theo thu tu tu dien : khong phan biet thuong hoa  
int SoSanhChuoi_KPB(String a, String b)  
{  
    int i;  
    for (i = 0; a[i] != NULL && b[i] != NULL; i++)  
    {  
        if (Chuyen_KT_Hoa(a[i]) < Chuyen_KT_Hoa(b[i]))  
            return -1;  
        if (Chuyen_KT_Hoa(a[i]) > Chuyen_KT_Hoa(b[i]))  
            return 1;  
    }  
    if (a[i] != NULL)  
        return 1;  
    if (b[i] != NULL)  
        return -1;  
    return 0;  
}
```

12.2 Hàm chuyển ký tự thường thành hoa

```
char Chuyen_KT_Hoa(char x)  
{  
    if ('a' <= x && x <= 'z')  
        x = x - 32;  
    return x;  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

12.3 Bổ sung nguyên mẫu các hàm :

```
int SoSanhChuoi_KPB(String a, String b);
char Chuyen_KT_Hoa(char x);
```

- Trong tập tin **menu.h** :

Bổ sung xử lý chức năng so sánh chuỗi case 10 (Các case khác giữ nguyên)

```
void XuLyMenu(int menu, String a, String b)
{
    //khai bao bien
    int kq;
    switch (menu)
    {
        //...

        case 10:
            system("CLS");
            cout << "\n10. So sanh chuoi _khong phan biet KT thuong, HOA";
            cout << "\nChuoi a = ";
            XuatChuoi(a);
            cout << "\nChieu b = ";
            XuatChuoi(b);

            kq = SoSanhChuoi_KPB(a, b);
            if (kq == 1)
                cout << "\na > b";
            else
                if(kq == -1)
                    cout << "\na < b";
                else
                    cout << "\na == b";
            break;

        //...
    }
    _getch();
}
```



Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 10.

Bước 13 :

Bổ sung chức năng so sánh chuỗi có phân biệt ký tự thường hoa vào chương trình

- Trong tập tin **thuvien.h** :

13.1 Hàm so sánh chuỗi có phân biệt ký tự thường hoa

```
//So sanh 2 chuoi theo thu tu tu dien : co phan biet thuong hoa
int SoSanhChuoi_PB(String a, String b)
{
```

```

int i;
for (i = 0; a[i] != NULL && b[i] != NULL; i++)
{
    if (a[i] < b[i])
        return -1;
    if (a[i] > b[i])
        return 1;
}
if (a[i] != NULL)
    return 1;
if (b[i] != NULL)
    return -1;
return 0;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

13.2 Bổ sung nguyên mẫu các hàm :

```
int SoSanhChuoi_PB(String a, String b);
```

- Trong tập tin *menu.h* :

Bổ sung xử lý chức năng so sánh chuỗi chuỗi vào case 11 (Các case khác giữ nguyên)

```

void XuLyMenu(int menu, String a, String b)
{
    //khai bao bien
    int kq;
    switch (menu)
    {
        //...
        case 11:
            system("CLS");
            cout << "\n11. So sanh chuoi _ phan biet KT thuong, HOA";
            cout << "\nChuoi a = ";
            XuatChuoi(a);
            cout << "\nChieu b = ";
            XuatChuoi(b);

            kq = SoSanhChuoi_PB(a, b);
            if (kq == 1)
                cout << "\na > b";
            else
                if (kq == -1)
                    cout << "\na < b";
                else
                    cout << "\na == b";
            break;
        //...
    }
    getch();
}

```

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện các chức năng 11.

Kiểm tra các chức năng của chương trình – Kết thúc chương trình.

Ghi chú :

Thay thế hàm nhập chuỗi - `void gets_s_NhapChuoi(String a, char kt)` – bằng hàm sau đây :

```
//Nhập chuỗi, không dùng gets_s
void NhapChuoi(String a, char kt)
{
    int i = 0;
    char x;
    cout << "\nNhap Chuoi: " << kt << " : ";
    _flushall();
    x = _getch();
    while (x != 13)
    {
        a[i++] = x;
        x = _getch();
    }
    a[i] = NULL;
}
```

Bài 2: (nắn tên)

Từ là một dãy các ký tự liên tiếp không chứa dấu cách (ký tự trắng). Ta xem tên là một xâu ký tự gồm nhiều từ phân cách nhau bởi các dấu cách.

Viết chương trình nắn các tên được nhập từ bàn phím theo quy cách:

- Quy cách 1 : Xóa các ký tự trắng ở đầu và cuối tên.
- Quy cách 2 : Khử bớt các ký tự trắng ở giữa hai tên, chỉ giữ lại một ký tự trắng.
- Quy cách 3 : Các chữ cái đầu mỗi từ viết IN HOA, các chữ còn lại viết in thường.

Ví dụ: (hang trên : tên nhập; hang sau : tên đã nắn theo chuẩn

Da LAT hoaNG Hon

Da Lat Hoang Hon

Ý tưởng giải quyết bài toán :

- Xử lý tên -> xử lý từ -> xử lý ký tự
- Trong quá trình xử lý tên theo quy cách, ta dùng một biến trung gian để lưu trữ kết quả
- Xử lý tên : Khử các ký tự trắng ở đầu và cuối tên. Khử các ký tự trắng giữa 2 từ chỉ để lại một.
- Xử lý từ : Ký tự đầu nắn thành ký hoa, các ký tự còn lại trong từ nắn thành ký tự thường. Chèn ký tự vào cuối biến trung gian...

Nên ta cần thực hiện các công việc sau :

- Chuyển ký tự thường thành hoa.
- Chuyển ký tự hoa thành thường.
- Chèn một ký tự vào cuối xâu ký tự
- Nắn tên theo quy cách
- ...

Chương trình tổ chức theo thư viện hàm (**và không có hệ thống menu.**)

Bước 1. Tạo Project rỗng mới đặt tên **Lab07_D_Bai2**

Bước 2. Tạo cấu trúc cho chương trình như đã hướng dẫn trong **mục 1 lab 4** (từ 1-7 để có cấu trúc nội dung tối thiểu chạy được chương trình, dạng không có hệ thống menu).

Tức là ta có kết quả chương trình ở bước này như sau :

- Tập tin **thuvien.h** : Rỗng (chỉ có các chú thích về cấu trúc văn bản)
- Tập tin **program.cpp** có nội dung như sau :

```
#include <iostream>
#include <conio.h>
using namespace std;
#include "thuvien.h"
void ChayChuongTrinh();
int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    getch();
}
```

Nhấn Ctrl + F5 để chạy chương trình, sửa lỗi nếu có.

Bước 3:

Bước này ta bổ sung thêm các thư viện cần thiết, định nghĩa hằng, kiểu dữ liệu mới, định nghĩa các hàm chúc năng ...

- Trong tập tin **program.cpp** :

Bổ sung thêm thư viện **<string.h>**

- Trong tập tin **thuvien.h** :

+ Bổ sung định nghĩa hằng, kiểu dữ liệu mới :

Vì chương trình thực hiện các thao tác trên xâu ký tự, nên ta cần định nghĩa một hằng là giá trị kích thước khai báo của xâu.

//Dinh nghia hang

```
#define MAX 100//kich thuoc khai bao xau ky tu
```

```
#define CACH ' ' //ky tu trang
```

//Dinh nghia kieu du lieu moi: khong co

//Khai bao nguyen mau cac ham xu ly, nhap xuat

//bổ sung sau

//Dinh nghia cac ham xu ly, nhap xuat : khong (dung ham thu vien trong string.h)

3.1 Định nghĩa hàm chuyển sang ký tự thường

(Nếu là ký tự thường thì giữ nguyên, nếu là HOA thì chuyển sang thường)

```
char Chuyen_KT_Thuong(char x)
```

```
{
```

```
    if ('A' <= x && x <= 'Z')  
        x = x + 32;
```

```
    return x;
```

```
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chuyển sang ký tự HOA

(Nếu là ký tự HOA thì giữ nguyên, nếu là thường thì chuyển sang HOA)

```
char Chuyen_KT_Hoa(char x)
{
    if ('a' <= x && x <= 'z')
        x = x - 32;
    return x;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Chèn 1 ký tự vào cuối một xâu ký tự

```
//Chen kt vao cuoi b
void ChenCuoi(char b[MAX], char kt)
{
    int i;
    for (i = 0; b[i] != NULL; i++);
    b[i++] = kt;
    b[i] = NULL;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Nắn tên theo quy cách

```
void NanTen(char a[MAX])
{
    int i;
    char b[MAX];//Xau trung gian
    b[0] = NULL;
    i = 0;
    //Khu dau cach
    while (a[i] == CACH)
        i++; //a[i] == NULL hay la dau 1 tu
    while (a[i] != NULL) //xu ly tu
    {
        ChenCuoi(b, Chuyen_KT_Hoa(a[i])); // xu ly dau tu: a[i]
        i++; //xet ky tu ke tiep : than tu, CACH hay ket thuc xau
        while (a[i] != CACH && a[i] != NULL) //ky tu trong than tu
        {
            ChenCuoi(b, Chuyen_KT_Thuong(a[i])); // xu ly than tu
            i++;
        } //a[i] == CACH hay a[i] == NULL
        //Da xu ly xong 1 tu
        //Chua ket thuc xau thi tiep tuc xu ly tu tiep theo
        //Tiep tuc vuot dau cach
        while (a[i] == CACH)
            i++; //a[i] == NULL hay la dau 1 tu
        if (a[i] != NULL) //tu vua xu ly chua phai la tu cuoi
            ChenCuoi(b, CACH); // chen cac vao sau b
    }
    strcpy_s(a, MAX, b);
}
```

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.5 Khai báo nguyên mẫu :

```
char Chuyen_KT_Thuong(char x);
char Chuyen_KT_Hoa(char x);
void ChenCuoi(char b[MAX], char kt);
void NanTen(char a[MAX]);
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Bước 4:

Cập nhật lại hàm ChayChuongTrinh() : Nhập chuỗi, xuất chuỗi, gọi hàm NanTen để giải quyết bài toán, điều khiển lặp công việc giải quyết bài toán.

```
void ChayChuongTrinh()
{
```

```
    char a[MAX];
    char thoat;
    do
    {
        system("CLS");
        cout << "\nNhap xau ky tu : ";
        gets_s(a);
        system("CLS");
        cout << "\nTen vua nhap: ";
        cout << a;
        NanTen(a);
        cout << "\nTen da nan : ";
        cout << a;
        cout << endl << "Nua khong ? go ESC neu khong\n";
        thoat = _getch();
    } while (thoat != 27);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả chương trình – Kết thúc chương trình.

E. Bài tập bắt buộc

Tất cả các bài tập phải được tổ chức dưới dạng thư viện hàm và có/không menu chức năng.

1. Xử lý chuỗi

Viết chương trình thực hiện các chức năng sau trên chuỗi:

- Thoat khoi chuong trinh
- Nhap chuoi
- Xem chuoi
- Tinh chieu dai chuoi
- Chèn ký tự x vào đầu chuỗi
- Chèn ký tự x vào cuối chuỗi
- Chen ky tu x vao chuoi a tai vi tri cho truoc
- Xoa ky tu dau chuoi

- Xóa ký tự cuối chuỗi
- Xóa ký tự tại vị trí cho trước
- Cắt ký tự đầu chuỗi rồi chèn vào vị trí cuối chuỗi (đã cắt)
- Cắt ký tự cuối chuỗi rồi chèn vào vị trí đầu chuỗi (đã cắt)
- Xóa tất cả ký tự x cho trước khỏi chuỗi.
- Thay thế tất cả ký tự x trong chuỗi thành ký tự y .

Tạo dự án Win32 Console Application mới. Đặt tên là **Lab07_E_Bai1**

Tổ chức dự án như bài 1 mục D

(Lưu ý là nhập, xuất chuỗi không cần viết hàm, mà có thể dùng : gets_s(a,MAX), cout<<)

Tham khảo tập tin **thuvien.h** sau đây:

//Dinh nghia hang

#define MAX 100

//Dinh nghia kieu du lieu moi

typedef char String[MAX];

//Khai bao nguyen mau

int TinhChieuDaiChuoi(String a);

void ChenDau_KT(char a[MAX], char x);

void ChenCuoi_KT(char a[MAX], char x);

int ChenKT_VT(char a[MAX], char x, int vt);

void XoaDau_KT(char a[MAX]);

void XoaCuoi_KT(char a[MAX]);

int XoaKT_VT(char a[MAX], int vt);

void CatDauChenCuoi(char a[MAX]);

void CatCuoiChenDau(char a[MAX]);

void Xoa_X(char a[MAX], char x);

void Thay_x_Bang_y(char a[MAX], char x, char y);

//Dinh nghia cac ham xu ly

//Tinh chieu dai chuoi

int TinhChieuDaiChuoi(String a)

{

 int i = 0;

 while (a[i])

 i++;

 return i;

}

//Chen ky tu x vao dau chuoi a, ket qua luu tru vao a a

void ChenDau_KT(char a[MAX], char x)

{

 int i;

 int l = TinhChieuDaiChuoi(a);

 for (i = l; i >= 0; i--)

 a[i + 1] = a[i];

 a[0] = x;

}

```

//Chen ky tu x vao cuoi chuoi a, ket qua luu tru vao a a
void ChenCuoi_KT(char a[MAX], char x)
{
    int l = TinhChieuDaiChuoi(a);
    a[l++] = x;
    a[l] = NULL;
}
//Chen ky tu x vao chuoi a tai vi tri vt
//Input : a,x,vt
//output : 1: thanh cong; 0 : khong thanh cong
int ChenKT_VT(char a[MAX], char x, int vt)
{
    int i, l, kq = 1;
    l = TinhChieuDaiChuoi(a);
    if (vt == 0)
    {
        ChenDau_KT(a, x);
    }
    else
    {
        if (vt == l)
        {
            ChenCuoi_KT(a, x);
        }
        else
        {
            if (0 < vt && vt < l)
            {
                for (i = l; i >= vt; i--)
                    a[i + 1] = a[i];
                a[vt] = x;
            }
            else
                kq = 0;
        }
    }
    return kq;
}
//Xoa ky tu dau chuoi a
void XoaDau_KT(char a[MAX])
{
    int i;
    for (i = 0; a[i] != NULL; i++)
        a[i] = a[i + 1];
    a[i - 1] = NULL;
}
//Xoa ky tu cuoi chuoi a
void XoaCuoi_KT(char a[MAX])
{
    int i;
    for (i = 0; a[i] != NULL; i++);
    a[i - 1] = NULL;
}
//CXoa ky tu tai vi tri vt cua chuoi a
//Input : a,vt

```

```

//output : 1; thanh cong; 0 : khong thanh cong
int XoaKT_VT(char a[MAX], int vt)
{
    int i, l, kq = 1;
    l = TinhChieuDaiChuoi(a);
    if (vt == 0)
    {
        XoaDau_KT(a);
    }
    else
    if (vt == l-1)
    {
        XoaCuoi_KT(a);
    }
    else
    if (0 < vt && vt < l-1)
    {
        for (i = l-1; i > vt; i--)
            a[i - 1] = a[i];
        a[l-1] = NULL;
    }
    else
        kq = 0;
    return kq;
}

//Cat ky tu dau chuoi roi chen vao sau chuoi
void CatDauChenCuoi(char a[MAX])
{
    int i;
    char x;
    x = a[0];
    for (i = 1; a[i] != NULL; i++)
        a[i - 1] = a[i];
    a[i - 1] = x;
}

//Cat ky tu cuoi chuoi roi chen vao tai vi tri dau chuoi
void CatCuoiChenDau(char a[MAX])
{
    int i, l;
    char x;
    l = TinhChieuDaiChuoi(a);
    x = a[l-1];
    for (i = l-2; i >= 0; i--)
        a[i+1] = a[i];
    a[0] = x;
}

//Xoa tat ca ky tu x trong chuoi
void Xoa_x(char a[MAX], char x)
{
    int i, h=0;
    for (i = 0; a[i] != NULL; i++)
        if (a[i] != x)

```

```

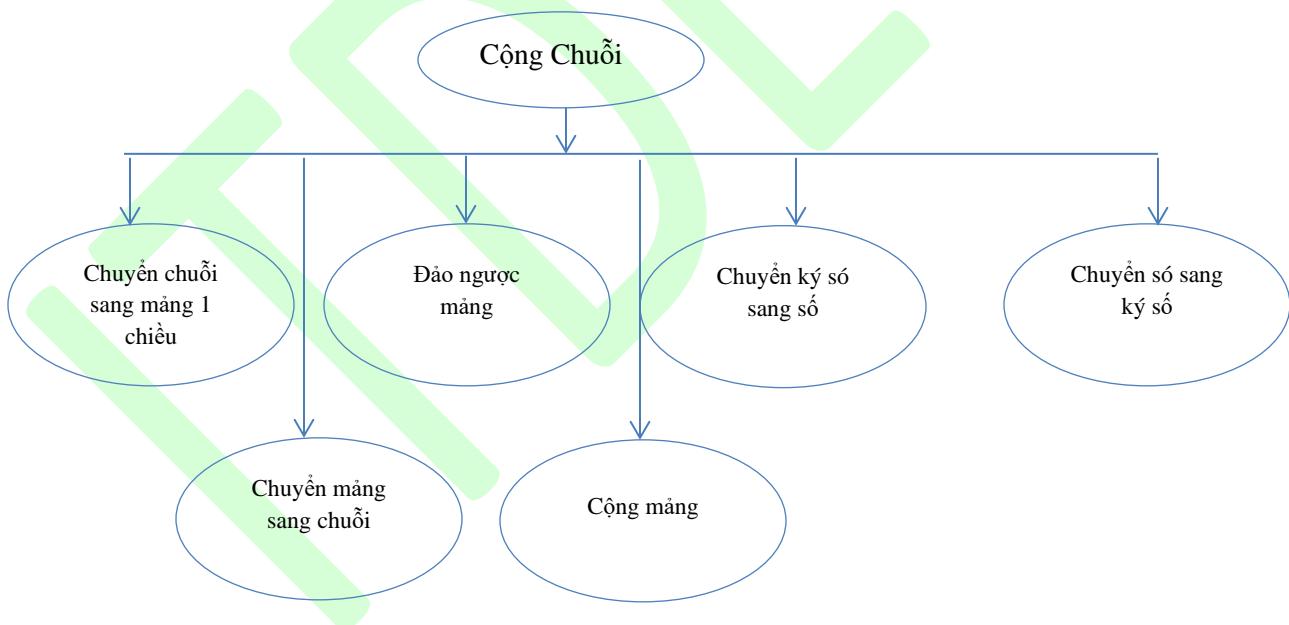
        a[h++] = a[i];
        a[h] = NULL;
    }
//Thay the tat ca cac ky tu x trong chuoi bang ky tu y
void Thay_x_Bang_y(char a[MAX], char x, char y)
{
    int i;
    for (i = 0; a[i] != NULL; i++)
        if (a[i] == x)
            a[i] = y;
}

```

2. Phép cộng số tự nhiên lớn

Cho hai số tự nhiên lớn m, n (không quá 50 chữ số) . Viết chương trình thực hiện phép cộng hai số m, n .
Ý tưởng giải quyết vấn đề :

- Số tự nhiên lớn sẽ được biểu diễn bởi xâu ký tự. Và ta phải thực hiện phép cộng các xâu ký số.*
- Để đơn giản, thay vì ta thực hiện phép cộng các xâu ký số ta chuyển sang thực hiện phép cộng 2 mảng 1 chiều. Nên ta phải chuyển cấu trúc xâu sang mảng 1 chiều.*
- Khi cộng mảng, nếu xuất phát từ cách thức cộng thông thường thì ta thực hiện từ phải sang trái, khi đó sẽ vấp phải trường hợp tràn ô nhớ, tức là ở hàng cao nhất mà có nhớ thì không có nơi để lưu trữ. Để khắc phục điều này, ta đảo ngược mảng và thực hiện từ trái sang phải.*



Tổ chức chương trình bài 2 phần D (hướng dẫn), không có hệ thống menu.

Tham khảo tập tin **thuvien.h** sau đây :

```

//Dinh nghia hang
#define MAX 50
//Dinh nghia kieu du lieu moi
typedef char SoTuNhienLon[MAX];
//-----
int Chuyen_KySo_So(char x)
{

```

```

        return x - '0';
    }
    char Chuyen_So_KySo(int so)
    {
        char x = (char)(so + '0');
        return x;
    }

//Chuyen chuoi sang mang 1 chieu
void Chuyen_Chuoi_Sang_Mang(SoTuNhienLon m, int a[MAX], int &l)
{
    int i;
    l = strlen(m);
    for (i = 0; i < l; i++)
        a[i] = Chuyen_KySo_So(m[i]);
}

//Dao nguoc mang
void DaoNguocMang(int a[MAX], int l)
{
    int i, j, tam;
    for (i = 0, j = l - 1; i < j; i++, j--)
    {
        tam = a[i];
        a[i] = a[j];
        a[j] = tam;
    }
}

//Chuyen mang sang chuoi
void ChuyenMang_Sang_Chuoi(int a[MAX], int l, SoTuNhienLon m)
{
    int i;
    for (i = 0; i < l; i++)
        m[i] = Chuyen_So_KySo(a[i]);
    m[l] = NULL;
}

//Cong 2 mang
void CongMang(int a[MAX], int la, int b[MAX], int lb, int c[MAX], int &lc)
{
    int nho, i;
    lc = la >= lb ? la : lb;

    if (lc > lb)
        for (i = lb; i < lc; i++)
            b[i] = 0;
    if (lc > la)
        for (i = la; i < lc; i++)
            a[i] = 0;

    nho = 0;
    for (i = 0; i < lc; i++)
    {
}

```

```

c[i] = a[i] + b[i] + nho;
if (c[i] > 9)
{
    c[i] = c[i] - 10;
    nho = 1;
}
else
    nho = 0; // tra lai nho = 0;
}
if (nho == 1)
{
    c[lc] = nho;
    lc++;
}
}

//Cong 2 so tu nhien lon
void CongSoTuNhienLon(SoTuNhienLon m, SoTuNhienLon n, SoTuNhienLon t)
{
    int a[MAX], b[MAX], c[MAX];
    int la, lb, lc;
    Chuyen_Chuoi_Sang_Mang(m, a, la);
    Chuyen_Chuoi_Sang_Mang(n, b, lb);

    DaoNguocMang(a, la);
    DaoNguocMang(b, lb);

    CongMang(a, la, b, lb, c, lc);

    DaoNguocMang(c, lc);
    ChuyenMang_Sang_Chuoi(c, lc, t);
}

```

3. Xử lý chuỗi

Viết chương trình thực hiện các chức năng sau trên chuỗi :

- Chuyển tất cả các ký tự trong chuỗi thành ký tự thường
- Chuyển tất cả các ký tự trong chuỗi thành ký tự HOA
- Đảo ngược chuỗi
- Kiểm tra một chuỗi có phải là chuỗi đối xứng (Palindrome).

4. Tìm – Đếm

Viết chương trình thực hiện các thao tác sau trên chuỗi :

- Đếm số lượng ký tự x xuất hiện trong chuỗi
- Xuất các giá trị ký tự phân biệt của chuỗi và số lần xuất hiện tương ứng của nó.
- Tìm vị trí xuất hiện đầu tiên của ký tự *x* trong chuỗi. Nếu không trả về -1
- Tìm vị trí xuất hiện của chuỗi *t* trong chuỗi *s*. Nếu *s* không chứa *t* thì trả về -1.
- Đếm số từ trong chuỗi *s*.
- Đảo vị trí của từ đầu và từ cuối trong chuỗi *s*. Ví dụ: *s* = meo an ca thì kết quả là *s* = ca an meo

5. Chuyển xâu ký số thành số tự nhiên

Viết chương trình đổi một xâu ký số thành giá trị số tương ứng.

Ví dụ : nhập “123456567890”, Xuất : 1234567890

F. Bài tập làm thêm

1. Từ chung

Viết chương trình thực hiện các chức năng sau trên xâu ký tự:

- Liệt kê các ký tự xuất hiện trong cả hai xâu s và t .
- Liệt kê các từ xuất hiện trong cả hai xâu s và t .
- Tìm và liệt kê từ dài nhất trong xâu s .

2. Tách từ

Viết chương trình thực hiện các chức năng sau trên xâu ký tự:

- Tách xâu s thành một mảng các từ dựa vào khoảng trắng. Ví dụ: $s =$ “Khoa Cong nghe Thong tin” thì tách thành mảng gồm các xâu: “Khoa”, “Cong”, “nghe”, “Thong”, “Tin”.
- Kiểm tra xâu s có chứa ký tự số hay không? Nếu có, tách các số đó ra thành một mảng số riêng. Ví dụ: $s =$ “Thang 1 Nam 2015, dan so Viet Nam vao khoang 90 trieu” thì mảng số kết quả gồm các phần tử: 1, 2015, 90.
- Đảo ngược thứ tự các từ trong xâu s . Ví dụ: “Thuong qua Viet Nam” \Rightarrow “Nam Viet qua Thuong”

3. Năm Dương lịch – Âm Lịch

Viết chương trình nhập vào năm Dương lịch, xuất ra năm Âm lịch

Ví dụ:

Input : 2016

Output: Binh Than

4. Nén xâu

Viết chương trình thực hiện việc nén và giải nén một xâu ký tự. Ví dụ: xâu $s =$ “AAABBCCCCC CDDDEEEEEEFFFG” sau khi nén là $s =$ “3A2B6C3D7E3FG”. *Chú ý: nếu ký tự chỉ xuất hiện 1 lần thì không cần nén, không cần ghi số lần xuất hiện ký tự đó. Trong trường hợp này là ký tự G.*

Cho biết cách nén xâu theo cách trên không có tác dụng trong những trường hợp nào?

5. Mã hóa

Viết chương trình mã hóa và giải mã một chuỗi ký tự bằng cách đảo bit ($0 \rightarrow 1$, $1 \rightarrow 0$) hoặc đảo ngược thứ tự các bit của từng ký tự trong xâu.

LAB 8. KIỂU CẤU TRÚC

THỜI LUỢNG: 6 TIẾT

A. Mục tiêu

- Giúp sinh viên hiểu rõ và thực hiện thuần thục các kỹ thuật xử lý trên kiểu dữ liệu cấu trúc.
- Tiếp tục hoàn thiện kỹ năng phát triển một chương trình có cấu trúc theo chức năng, tổ chức theo thư viện hàm và hệ thống tùy chọn menu.
- Sau khi hoàn thành bài thực hành này, sinh viên cần phải:
 - Nắm vững các định nghĩa kiểu cấu trúc, biến cấu trúc, mảng cấu trúc.
 - Nắm vững các thao tác nhập - xuất biến cấu trúc, mảng cấu trúc.
 - Biết cách định nghĩa và sử dụng kiểu dữ liệu **mảng cấu trúc** bằng từ khóa **typedef**.
 - Vận dụng được các kỹ thuật tìm kiếm, sắp xếp vào mảng cấu trúc.
 - Hiểu rõ cơ chế gọi hàm và truyền tham số có kiểu cấu trúc.

B. Yêu cầu

- Trong phần C (hướng dẫn thực hành), sinh viên tự đọc.
- Sinh viên cần chuẩn bị bài trước để thực hành đủ khối lượng yêu cầu.

Buổi thực tập thứ 10 (4 tiết) :

- 2 tiết cuối:

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần D (Hướng dẫn thực hành) của lab 8.
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab08_HD chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Sử dụng lại tập tin word LoiChuongTrinh.docx từ lab7 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - ✓ Thu bài : Xem thông báo của giáo viên .

Buổi thực tập thứ 11 (4 tiết) :

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần E (Bắt buộc) của lab 8. (Để đảm bảo thời gian, sinh viên nên chuẩn bị trước ít nhất 2 bài)
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab08_BB chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Sử dụng lại tập tin word LoiChuongTrinh.docx trong lab5 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.

- ✓ Giáo viên sẽ thu bài qua hệ thống thu bài trực tuyến tại phòng lab (thu thư mục MaSV_Lab08_BB) vào cuối buổi thực tập.

C. Ôn tập lý thuyết

1. Cách định nghĩa kiểu cấu trúc và khai báo biến cấu trúc

Định nghĩa kiểu cấu trúc		
Dạng 1	Dạng 2	Dạng 3
struct TênKiểuCấuTrúc { // Các thành phần };	struct TênKiểuCấuTrúc { // Các thành phần } danh_sách_các_biéñ_CT;	struct { // Các thành phần } danh_sách_các_biéñ_CT;
Ví dụ		
struct NgayThang { unsigned int Ngay; unsigned int Thang; unsigned int Nam; }; // Khai báo 2 biến cấu trúc NhanVien an, binh;	struct NgayThang { unsigned int Ngay; unsigned int Thang; unsigned int Nam; }; NhanVien { unsigned int MaSo; char HoTen[50]; char DiaChi[100]; NgayThang NgaySinh; int SoDT; }; } an, binh;	struct NgayThang { unsigned int Ngay; unsigned int Thang; unsigned int Nam; }; struct { unsigned int MaSo; char HoTen[50]; char DiaChi[100]; NgayThang NgaySinh; int SoDT; }; } an, binh;

Trong đó:

- **TênKiểuCấuTrúc:** là tên kiểu dữ liệu cấu trúc, do lập trình viên tự đặt theo quy tắc đặt tên.
- Mỗi thành phần của cấu trúc gọi là 1 trường, có dạng khai báo như các biến: **KDL tên_trường;**

2. Truy cập đến các thành phần của cấu trúc

Cú pháp:

tên_biéñ_cáú_trúc.tên_trường

Nếu thành phần là một cấu trúc: **tên_biéñ_cáú_trúc.tên_trường_cáú_trúc.tên_trường**

Ví dụ:

- **an.MaSo, binh.HoTen, binh.SoDT, an.DiaChi.**
- **an.NgaySinh.Thang, binh.NgaySinh.Nam.**

3. Nhập, xuất dữ liệu cho biến cấu trúc

g. Nhập giá trị cho từng thành phần của cấu trúc

- cin >> binh.MaSo; Nhập mã số cho nhân viên binh
- cin >> an.NgaySinh.Nam Nhập năm sinh cho nhân viên an
- gets_s(an.DiaChi) Nhập địa chỉ của nhân viên an

h. Xuất giá trị của từng thành phần cấu trúc

- cout << binh.MaSo; Xuất mã số của nhân viên binh ra màn hình
- cout << an.NgaySinh.Nam Xuất năm sinh của nhân viên an
- cout << an.DiaChi Xuất địa chỉ của nhân viên an

4. Khởi đầu (gán) giá trị biến cấu trúc

Khởi đầu cho từng thành phần của cấu trúc, theo thứ tự.

- NgayThang ns = { 31, 12, 1988 };
- NhanVien an = { 10, "Nguyen Van An", "1C NCTru", { 5, 12, 1990 }, 1234567890 };
- NhanVien binh = { 25, "Pham Binh", "100B BTXuan", { 24, 6, 1981 }, 1278934560 };

5. Mảng cấu trúc

Mảng cấu trúc là mảng một chiều chứa các phần tử có cùng kiểu cấu trúc.

Khai báo:

TênKiểuCấuTrúc tên_bien_mảng[Kích_thước];

Ví dụ:

NhanVien dsnv[1000];

Định nghĩa kiểu mảng cấu trúc: **typedef TênKiểuCấuTrúc tên_kiểu_mảng[Kích_thước];**

Ví dụ:

typedef NhanVien MangNV[1000];

MangNV dsnv;

6. Truyền tham số

Tham số hình thức	Tham số thực (đôi số)
Biến cấu trúc	Giá trị cấu trúc của cùng kiểu
Tên mảng cấu trúc	Tên mảng cấu trúc cùng kiểu, cùng kích thước

D. Hướng dẫn thực hành

*Phần này xây dựng chương trình thực hiện các thao tác trên kiểu cấu trúc với các chức năng cơ bản.
Tiếp tục tổ chức chương trình theo thư viện và có/không có hệ thống menu.*

Bài 1: Quản lý nhân viên

Một công ty quản lý nhân viên theo các thông tin:

- Mã nhân viên : một chuỗi có đúng 7 ký tự
- Họ và Chữ lót : chuỗi không quá 20 ký tự
- Tên: chuỗi không quá 7 ký tự
- Ngày sinh : từ 1 đến 2 ký số
- Tháng sinh : từ 1 đến 2 ký số
- Năm sinh : 4 ký số
- Địa chỉ : chuỗi không quá 20 ký tự
- Lương : một số thực dương

Viết chương trình tùy chọn thực hiện trên danh sách các nhân viên, với các chức năng:

- Thoát khỏi chương trình**
- Nhập danh sách nhân viên**

- c. Xem danh sách nhân viên
- d. Thêm 1 nhân viên
- e. Xóa một nhân viên
- f. Sửa thông tin nhân viên
- g. Tìm nhân viên theo mã số
- h. Tìm nhân viên theo tên
- i. Sắp danh sách sinh viên tăng dần theo mã nhân viên
- j. Sắp danh sách nhân viên tăng dần theo tên, theo họ và lương
- k. Tính tổng lương tháng
- l. Liệt kê các nhân viên có lương tháng $\geq x$ (x cho trước)
- m. Liệt kê các nhân viên có năm sinh trong khoảng $[u,v]$

Bước 1. Tạo 1 dự án Win32 Console Application mới. Đặt tên là **Lab08_D_Bai1**

Bước 2. Tạo cấu trúc cho chương trình như bài 1 mục D lab 7 (phần lỗi tối thiểu chạy được của chương trình)

Tức là ta có kết quả chương trình ở bước này như sau :

- Tập tin **thuvien.h** : Rỗng
- Tập tin **menu.h** : Rỗng
- Tập tin **program.cpp** có nội dung như sau :

```
#include <iostream>
#include <conio.h>
using namespace std;

#include "thuvien.h"
#include "menu.h"

void ChayChuongTrinh();

int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    _getch();
}
```

Nhấn Ctrl + F5 để chạy chương trình, sửa lỗi nếu có.

Bước 3:

Bước này ta định nghĩa kiểu dữ liệu mới; tổ chức và vận hành hệ thống menu.

- Trong tập tin **thuvien.h** :

Ta bổ sung định nghĩa hằng cho kích thước khai báo mảng 1 chiều, các kiểu dữ liệu cấu trúc : NhanVien, NgayThangNam .

//Dinh nghia hang

//Dinh nghia hang

#define MAX 100//kich thuoc khai mang cau truc

#define NGANGDOI '='

```
#define NGANGDON '-'  
  
//Dinh nghia kieu du lieu moi : cac kieu cau truc  
  
typedef unsigned short int Byte;  
//Kieu ngay thang nam sinh  
struct NgayThangNam  
{  
    Byte ngaySinh;  
    Byte thangSinh;  
    unsigned int namSinh;  
};  
//Kieu nhan vien  
struct NhanVien  
{  
    char maNV[8];  
    char hoLot[21];  
    char ten[8];  
    NgayThangNam ntns;  
    char diaChi[21];  
    double luong;  
};  
  
//Khai bao nguyen mau cac ham xu ly, nhap xuat  
//bo sung sau  
  
//Dinh nghia cac ham xu ly, nhap xuat  
//bo sung sau  
  
- Trong tap tin menu.h :  
  
// Khai bao nguyen mau cac ham xu ly menu  
//bo sung sau  
// Định nghĩa các hàm xử lý menu  
  
3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình  
void XuatMenu()  
{  
    cout << "\n=====He thong chuc nang=====\n";  
    cout << "n0. Thoat khoi chuon trinh";  
    cout << "n1. Nhap danh sach nhan vien";  
    cout << "n2. xem danh sach nhan vien";  
    cout << "n3. Them mot nhan vien vao cuoi danh sach";  
    cout << "n4. Xoa mot nhan vien theo ma nhan vien";  
    cout << "n5. Sua thong tin nhan vien theo ma nhan vien";  
    cout << "n6. Tim nhan vien theo ma so";  
    cout << "n7. Tim nhan vien theo ten";  
    cout << "n8. Sap danh sach nhan vien tang dan theo ma nhan vien";  
    cout << "n9. Sap danh sach nhan vien tang dan theo ten - ho - luong";  
    cout << "n10. Tinh tong luong thang";  
    cout << "n11. Liet ke cac nhan vien co luong >= x (nhap tu ban phim)";  
    cout << "n12. Liet ke cac nhan vien co nam sinh trong khoang [u,v]";  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách

```
// Input : soMenu = Số lượng menu có thể chọn.  
// Output: Số thứ tự menu do người dùng nhập vào.  
int ChonMenu(int soMenu)  
{  
    int stt;  
    for (;;) {  
        system("CLS");  
        XuatMenu();  
        cout << "\nNhập 1 số khong khoang [0,..," << soMenu << "] để chọn chức năng, stt = ";  
        cin >> stt;  
        if (0 <= stt && stt <= soMenu)  
            break;  
    }  
    return stt;  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Định nghĩa hàm xử lý menu :

Các thao tác thực hiện trên cùng một đầu vào là mảng 1 chiều kiểu NhanVien, kích thước mảng n, nên ta bổ sung thêm biến mảng 1 chiều NhanVien a, số nguyên dương n làm đối của hàm **XuLyMenu**, ngoài tham số đã có là tham số menu.

```
void XuLyMenu(int menu, NhanVien a[MAX], int &n)  
{  
    //khai bao bien  
    switch (menu)  
    {  
        case 0:  
            system("CLS");  
            cout << "\n0. Thoát khỏi chương trình\n";  
            break;  
        case 1:  
            system("CLS");  
            cout << "\n1. Nhập danh sách nhân viên";  
            break;  
        case 2:  
            system("CLS");  
            cout << "\n2. Xem danh sách nhân viên";  
            break;  
        case 3:  
            system("CLS");  
            cout << "\n3. Thêm một nhân viên";  
            break;  
        case 4:  
            system("CLS");  
    }
```

```

cout << "\n4. Xoa mot nhan vien";
break;

case 5:
    system("CLS");
    cout << "\n5. Sua thong tin nhan vien";
    break;

case 6:
    system("CLS");
    cout << "\n6. Tim nhan vien theo ma so";
    break;

case 7:
    system("CLS");
    cout << "\n7. Tim nhan vien theo ten";
    break;

case 8:
    system("CLS");
    cout << "\n8. Sap danh sach nhan vien tang dan theo ma nhan vien";
    break;

case 9:
    system("CLS");
    cout << "\n9. Sap danh sach nhan vien tang dan theo ten - ho - luong";
    break;

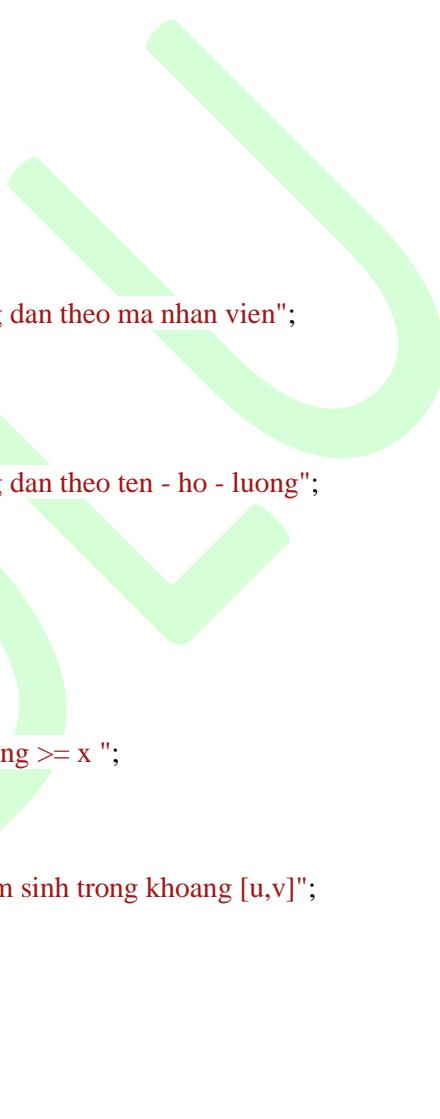
case 10:
    system("CLS");
    cout << "\n10. Tinh tong luong thang";
    break;

case 11:
    system("CLS");
    cout << "\n11. Liet ke cac nhan vien co luong >= x ";
    break;

case 12:
    system("CLS");
    cout << "\n12. Liet ke cac nhan vien co nam sinh trong khoang [u,v]";
    break;
}

_getch();
}

```



Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```

void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, NhanVien a[MAX], int &n);

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

- Trong tập tin **program.cpp** :
- + Hàm **ChayChuongTrinh** ta cập nhật lại như sau :

```
void ChayChuongTrinh()
{
    int menu,
        soMenu = 12,
        n = 0;
    NhanVien a[MAX];
    do
    {
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, a, n);
    } while (menu > 0);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra sự vận hành của hệ thống menu.

Kiểm tra chức năng thoát khỏi chương trình (chọn 0).

Bước 4 : Bổ sung vào chương trình các thao tác nhập xuất dữ liệu

Trong bước 4 này, ta làm công việc sau :

- Trong tập tin **program.cpp** , bổ sung các thư viện cần thiết
- Trong tập tin **thuvien.h** , soạn thảo các hàm nhập, xuất dữ liệu
- Trong tập tin **menu.h** bổ sung xử lý chức năng nhập dữ liệu, xem chuỗi trong hàm **XuLyMenu**.

- Trong tập tin **program.cpp** :

Bổ sung thư viện <string.h>, <iomanip>

- Trong tập tin **thuvien.h** :

Bổ sung các hàm nhập xuất dữ liệu

4.1 Nhập dữ liệu :

//Ham Nhập 1 nhân viên từ bàn phím

```
void Nhap_1NV(NhanVien &p)
{
    cout << "\nMa Nhan Vien (dung 7 ky so) : ";
    _flushall();
    gets_s(p.maNV, 8);

    cout << "\nHo va chu lot: ";
    _flushall();
    gets_s(p.hoLot, 21);

    cout << "\nTen nhan vien: ";
    _flushall();
    gets_s(p.ten, 7);

    cout << "\nNgay sinh: ";
    _flushall();
    cin >> p.ntns.ngaySinh;

    cout << "\nThang sinh: ";
    _flushall();
    cin >> p.ntns.thangSinh;

    cout << "\nNam sinh: ";
```

```
_flushall();
cin >> p.ntns.namSinh;

cout << "\nNhập địa chỉ: ";
_flushall();
gets_s(p.diaChi,20);

cout << "\nNhập lương: ";
_flushall();
cin >> p.luong;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Hàm nhập danh sách nhân viên
void Nhap_DSNV(NhanVien a[MAX], int &n)
{
    int i;
    cout << "\nNhập n = ";
    cin >> n;
    for (i = 0; i < n; i++)
    {
        system("CLS");
        cout << "\nNhập thông tin cho nhân viên thứ " << i + 1 << ":";
        Nhap_1NV(a[i]);
    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.2 Hàm xuất dữ liệu ra màn hình

```
//Xuất ke ngang
void XuatKeNgang()
{
    int i;
    cout << "\n";
    cout << setiosflags(ios::left)
        << ':';
    for (i = 1; i <= 76; i++)
        cout << NGANGDOI;
    cout << ':';
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuất tiêu đề
void XuatTieuDe()
{
    XuatKeNgang();
    cout << endl;
    cout << setiosflags(ios::left)
        << ':'
        << setw(8) << "Mã NV"
}
```

```
<< ':'
<< setw(16) << "Ho"
<< setw(8) << "Ten"
<< ':'
<< setw(12) << "NTN sinh"
<< ':'
<< setw(16) << "Dia Chi"
<< ':'
<< setw(12) << "Luong"
<< ';'
XuatKeNgang();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuat 1 nhan vien
void Xuat_1NV(NhanVien p)
{
    cout << setiosflags(ios::left)
        << ':'
        << setw(8) << p.maNV
        << ':'
        << setw(16) << p.hoLot
        << setw(8) << p.ten
        << ':'
        << setw(2) << p.ntns.ngaySinh
        << '/'
        << setw(2) << p.ntns.thangSinh
        << '/'
        << setw(6) << p.ntns.namSinh
        << ':'
        << setw(16) << p.diaChi
        << ':'
        << setw(12) << setiosflags(ios::fixed) << setprecision(2) << p.luong
        << ';'
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuat DS nhan vien
void Xuat_DSNV(NhanVien a[MAX], int n)
{
    int i;
    XuatTieuDe();
    for (i = 0; i < n; i++)
    {
        cout << endl;
        Xuat_1NV(a[i]);
    }
    XuatKeNgang();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.3 Bổ sung nguyên mẫu các hàm :

```
void Nhap_1NV(NhanVien &p);
void Nhap_1NV(NhanVien &p);
void XuatKeNgang();
void XuatTieuDe();
void Xuat_1NV(NhanVien p);
void Xuat_DSNV(NhanVien a[MAX], int n);
```

- Trong tập tin *menu.h* :

Bổ sung xử lý chức năng nhập dữ liệu vào case 1, xem dữ liệu vào case 2 (Các case từ 3 đến 12 giữ nguyên)

```
void XuLyMenu(int menu, NhanVien a[MAX], int &n)
{
    //khai bao bien
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << "\n0. Thoát khỏi chương trình\n";
            break;
        case 1:
            system("CLS");
            cout << "\n1. Nhập danh sách nhân viên";
            Nhap_DSNV(a, n);
            system("CLS");
            Xuat_DSNV(a, n);
            cout << "\nSố nhân viên trong danh sách : n = " << n;
            break;
        case 2:
            system("CLS");
            cout << "\n2. Xem danh sách nhân viên";
            Xuat_DSNV(a, n);
            cout << "\nSố nhân viên trong danh sách : n = " << n;
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện các chức năng 1, 2.

Bước 5 :

Ta có nhận xét là việc nhập dữ liệu cho danh sách nhân viên (mảng cấu trúc) từ bàn phím rất mất thời gian, đặc biệt là phải lặp lại cho mỗi lần chạy chương trình, nên ta cần có một cách phù hợp để tiết kiệm thời gian. Một trong các cách có thể dùng là khởi tạo dữ liệu ban đầu cho danh sách, sau đó muốn tạo lại danh sách ta có thể chọn chức năng 1 (nhập danh sách nhân viên từ bàn phím)

Ta sửa lại như sau :

- Trong tập tin menu.h, để cho trực quan hàm XuatMenu ta cập nhật lại như sau :

```
void XuatMenu()
{
    cout << "\n=====He thong chuc nang=====";
    cout << "\n0. Thoat khoi chuong trinh";
    cout << "\n1. Nhap danh sach nhan vien";
    cout << "\n2. xem danh sach nhan vien";
    cout << "\n3. Them mot nhan vien vao cuoi danh sach";
    cout << "\n4. Xoa mot nhan vien theo ma nhan vien";
    cout << "\n5. Sua thong tin nhan vien theo ma nhan vien";
    cout << "\n6. Tim nhan vien theo ma so";
    cout << "\n7. Tim nhan vien theo ten";
    cout << "\n8. Sap danh sach nhan vien tang dan theo ma nhan vien";
    cout << "\n9. Sap danh sach nhan vien tang dan theo ten - ho - luong";
    cout << "\n10. Tinh tong luong thang";
    cout << "\n11. Liet ke cac nhan vien co luong >= x ";
    cout << "\n12. Liet ke cac nhan vien co nam sinh trong khoang [u,v]";
    cout << "\n\n";
    cout << "|n  (Du lieu da duoc khai tao, chon 1 neu muon tao lai du lieu.)";
    cout << "\n\n";
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

- Trong tập tin program.cpp, ta khởi tạo dữ liệu cho mảng a và kích thước n trong hàm ChayChuongTrinh:

```
void ChayChuongTrinh()
{
    int menu,
        soMenu = 12,
        n = 10;//khai dau n

    NhanVien a[MAX] = //khai dau a
    {
        { "1234507", "Nguyen Van", "Tan", { 1, 1, 1990 }, "Da Lat", 20345678.0 },
        { "1205507", "Tran Minh", "Hoang", { 10, 10, 1980 }, "Qui Nhon", 14345678.0 },
        { "2234067", "Truong", "Hoang", { 10, 12, 1980 }, "Da Lat", 10342678.0 },
        { "2134167", "Le Tuan", "Ban", { 10, 1, 1975 }, "Quang Nam", 18349678.0 },
        { "1135067", "Tran Minh", "Ban", { 5, 6, 1977 }, "Phu Yen", 12345978.0 },
        { "1034007", "Truong", "Hoang", { 21, 12, 1987 }, "Da Lat", 15348678.0 },
        { "1130160", "Vo Tuan", "Trong", { 1, 11, 1985 }, "Qui Nhon", 20341178.0 },
        { "1235567", "Hoang Van", "Tan", { 15, 12, 1970 }, "Phu Yen", 22300970.0 },
        { "1034327", "Tran Van", "Vu", { 11, 11, 1982 }, "Da Lat", 16348888.0 },
        { "1034467", "Hoang Trong", "Ban", { 12, 1, 1980 }, "Nha Trang", 14365698.0 }
    };

    do
    {
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, a, n);
    } while (menu > 0);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra chức năng 2 với bộ dữ liệu khởi tạo

Trong các bước tiếp theo:

- Ta sử dụng bộ dữ liệu khởi tạo. Khi nào cần thay đổi bộ dữ liệu khác, ta chọn chức năng 1 để nhập lại dữ liệu từ bàn phím
- **Bổ sung lần lượt từng chức năng (từ 3 đến 12) vào chương trình :**
 - o Lần lượt soạn thảo từng hàm chức năng trong tập tin **thuvien.h** .
 - o **Bổ sung xử lý chức năng tương ứng trong hàm XuLyMenu của menu.h,**

Bước 6: Bổ sung chức năng 3 (thêm một nhân viên vào cuối danh sách) vào chương trình

- Trong **thuvien.h** viết hàm thêm nhân viên

6.1 Hàm thêm nhân viên vào cuối danh sách:

```
//Them nhan vien vao cuoi danh sach
void ThemNhanVien_Cuoi(NhanVien a[MAX], int &n, NhanVien p)
{
    a[n++] = p;
}
```

6.2 Khai báo nguyên mẫu :

```
void ThemNhanVien_Cuoi(NhanVien a[MAX], int &n, NhanVien p);
```

- Trong menu.h, bổ sung xử lý hàm thêm nhân viên vào case 3 của hàm XuLyMenu :

```
void XuLyMenu(int menu, NhanVien a[MAX], int &n)
{
    //khai bao bien
    switch (menu)
    {
        //...
        case 3:
            system("CLS");
            cout << "\n3. Them mot nhan vien";
            cout << "\nNhap thong tin nhan vien can chen vao cuoi danh sach:\n";
            Nhap_1NV(p);
            system("CLS");
            cout << "\nDanh sach ban dau :\n";
            Xuat_DSNV(a, n);
            cout << "\nSo nhan vien trong danh sach : n = " << n;
            ThemNhanVien_Cuoi(a, n, p);
            cout << "\nDanh sach ket qua :\n";
            Xuat_DSNV(a, n);
            cout << "\nSo nhan vien trong danh sach sau khi them : n = " << n;
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra chức năng 3.

Bước 7: Bổ sung chức năng 4 (Xóa một nhân viên theo mã nhân viên) vào chương trình

- Trong **thuvien.h** viết hàm xóa nhân viên

7.1 Hàm xóa nhân viên (theo mã nhân viên) :

//Xoa nhan vien theo ma so

void XoaNhanVien_MaNV(NhanVien a[MAX], int &n, char maNV[8])

{

int i,j;

for (i = 0; i < n; i++)

if (strcmp(a[i].maNV, maNV) == 0)

break;

if (i == n)

 {

 cout << "\nKhong co nhan vien nao trong danh sach co ma so " << maNV;

return;

 }

for (j = i + 1; j < n; j++)

 a[j - 1] = a[j];

 n--;

}

7.2 Khai báo nguyên mẫu :

void XoaNhanVien_MaNV(NhanVien a[MAX], int &n, char maNV[8]);

- Trong hàm XuLyMenu của tập tin menu.h :

+ Bổ sung xử lý hàm xóa nhân viên vào case 4 :

void XuLyMenu(int menu, NhanVien a[MAX], int &n)

{

//khai bao bien

char maNV[8];

switch (menu)

 {

//...

case 4:

 system("CLS");

 cout << "\n4. Xoa mot nhan vien theo ma so";

 cout << "\nNhap ma nhan vien can xoa :";

 cin >> maNV;

 cout << "\nDanh sach hien hanh :\n";

 Xuat_DSNV(a, n);

 cout << "\nSo nhan vien trong danh sach : n = " << n;

 XoaNhanVien_MaNV(a, n, maNV);

 cout << "\nDanh sach ket qua :\n";

 Xuat_DSNV(a, n);

 cout << "\nSo nhan vien trong danh sach sau khi xoa : n = " << n;

break;

//...

 }

_getch();

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra chức năng 4.

Bước 8: Bổ sung chức năng 5 (Sửa thông tin nhân viên theo mã nhân viên) vào chương trình :

- Trong **thuvien.h** viết hàm sửa thông tin nhân viên

8.1 Hàm sửa thông tin nhân viên (theo mã nhân viên) :

//Sửa thông tin nhân viên theo mã số

void SuaNhanVien_MaNV(NhanVien a[MAX], int &n, char maNV[8])

{

 int i;

 char tl;

 for (i = 0; i < n; i++)

 if (strcmp(a[i].maNV, maNV) == 0)

 break;

 if (i == n)

 {

 cout << "\nKhông có nhân viên nào trong danh sách có mã số " << maNV;

 return;

 }

 cout << "\nSửa thông tin nhân viên có mã số : " << maNV;

 cout << "\nCó sửa ma nhân viên không ? Nhấn K nếu không: ";

 tl = _getch();

 if (tl != 'K' && tl != 'k')

 {

 cout << "\nMã Nhân Viên : ";

 _flushall();

 gets_s(a[i].maNV, 8);

 }

 cout << "\nCó sửa họ và chung không ? Nhấn K nếu không: ";

 tl = _getch();

 if (tl != 'K' && tl != 'k')

 {

 cout << "\nHọ và chung: ";

 _flushall();

 gets_s(a[i].hoLot, 21);

 }

 cout << "\nCó sửa tên không ? Nhấn K nếu không: ";

 tl = _getch();

 if (tl != 'K' && tl != 'k')

 {

 cout << "\nTên nhân viên: ";

 _flushall();

 gets_s(a[i].ten, 8);

 }

 cout << "\nCó sửa ngày sinh không ? Nhấn K nếu không: ";

 tl = _getch();

 if (tl != 'K' && tl != 'k')

```

{
    cout << "\nNgay sinh: ";
    _flushall();
    cin >> a[i].ntns.ngaySinh;
}

cout << "\nCo sua thang sinh khong ? nhan k neu khong: ";
tl = _getch();
if (tl != 'k' && tl != 'K')
{
    cout << "\nThang sinh: ";
    _flushall();
    cin >> a[i].ntns.thangSinh;
}

cout << "\nCo sua nam sinh khong ? nhan k neu khong: ";
tl = _getch();
if (tl != 'k' && tl != 'K')
{
    cout << "\nNam sinh: ";
    _flushall();
    cin >> a[i].ntns.namSinh;
}

cout << "\nCo sua dia chi khong ? nhan k neu khong: ";
tl = _getch();
if (tl != 'k' && tl != 'K')
{
    cout << "\nNhap dia chi: ";
    _flushall();
    gets_s(a[i].diaChi, 20);
}

cout << "\nCo sua luong khong ? nhan k neu khong: ";
tl = _getch();
if (tl != 'k' && tl != 'K')
{
    cout << "\nNhap luong: ";
    _flushall();
    cin >> a[i].luong;
}
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

8.2 Khai báo nguyên mẫu :

```
void SuaNhanVien_MaNV(NhanVien a[MAX], int &n, char maNV[8]);
```

- Trong hàm XuLyMenu của tập tin menu.h :

+ Bổ sung xử lý hàm sửa thông tin nhân viên vào case 5 :

```
void XuLyMenu(int menu, NhanVien a[MAX], int &n)
{
    //khai bao bien
```

```

char maNV[8];
switch (menu)
{
    //...
    case 5:
        system("CLS");
        cout << "\n5. Sua thong tin nhan vien";
        cout << "\nDanh sach hien hanh :\n";
        Xuat_DSNV(a, n);

        cout << "\nNhap ma nhan vien can sua :";
        cin >> maNV;
        SuaNhanVien_MaNV(a, n, maNV);

        cout << "\nDanh sach ket qua :\n";
        Xuat_DSNV(a, n);
        break;
    //...
}
 getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra chức năng 5.

Bước 9: Bổ sung chức năng 6 (tìm nhân viên theo mã nhân viên) vào chương trình :

- Trong **thuvien.h**

9.1 Hàm tìm nhân viên (theo mã nhân viên) :

```

//Tim nhan vien theo ma so
//Khong co: tra ve -1; co : tra ve i (chi so bieu dien nhan vien tim duoc trong danh sach)
int TimNhanVien_MaNV(NhanVien a[MAX], int n, char maNV[8])
{
    int i, kq = -1;
    for (i = 0; i < n; i++)
    if (strcmp(a[i].maNV, maNV) == 0)
    {
        kq = i;
        break;
    }
    return kq;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

9.2 Khai báo nguyên mẫu :

```
int TimNhanVien_MaNV(NhanVien a[MAX], int n, char maNV[8]);
```

- Trong hàm XuLyMenu của tập tin menu.h :

- + Bổ sung khai báo biến kq kiểu int để lưu trữ kết quả tìm kiếm
- + Bổ sung xử lý hàm tìm nhân viên vào case 6 :

```
void XuLyMenu(int menu, NhanVien a[MAX], int &n)
```

```

{
    //khai bao bien
    char maNV[8];
    int kq;
    switch (menu)
    {
        //...
        case 6:
            system("CLS");
            out << "\n6. Tim nhan vien theo ma so";
            cout << "\nDanh sach hien hanh :\n";
            Xuat_DSNV(a, n);
            cout << "\nNhap ma nhan vien can tim :";
            cin >> maNV;
            kq = TimNhanVien_MaNV(a, n, maNV);

            if (kq == -1)
                cout << "\nKhong co NV nao trong danh sach co ma so : " << maNV;
            else
            {
                cout << "\nthong tin nhan vien co ma so : " << maNV << " :\n";
                XuatTieuDe();
                Xuat_1NV(a[kq]);
            }
            break;
        //...
    }
    getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra chức năng 6.

Bước 10: Bổ sung chức năng 7 (tìm nhân viên theo tên) vào chương trình :

- Trong *thuvien.h*

10.1 Hàm tìm nhân viên (theo tên nhân viên) :

```

//Tim nhan vien theo ten
void TimNhanVien_Ten(NhanVien a[MAX], int n, char ten[7])
{
    int i,
        dem = 0; //luu tru co bao nhieu nhan vien co ten nhu vay
    for (i = 0; i < n; i++)
        if (_strcmpi(a[i].ten, ten) == 0)
            dem++;
    if (!dem)
        cout << "\nTrong danh sach khong co nhan vien nao co ten la " << ten;
    else
    {
        cout << "\nCo " << dem << " nhan vien ten " << ten
            << ", voi thong tin chi tiet nhu sau :\n";
        XuatTieuDe();
    }
}

```

```

        for (i = 0; i < n; i++)
            if (_strcmpl(a[i].ten, ten) == 0)
            {
                cout << endl;
                Xuat_1NV(a[i]);
            }
        XuatKeNgang();
    }
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

10.2 Khai báo nguyên mẫu :

`void TimNhanVien_Ten(NhanVien a[MAX], int n, char ten[7]);`

- Trong hàm XuLyMenu của tập tin menu.h :

+ Bổ sung khai báo chuỗi lưu trữ tên nhân viên cần tìm
+ Bổ sung xử lý hàm tìm nhân viên theo tên vào case 7:

`void XuLyMenu(int menu, NhanVien a[MAX], int &n)`

```

{
    //khai bao bien
    char maNV[8], ten[7];
    int kq;
    switch (menu)
    {
        //...
        case 7:
            system("CLS");
            cout << "\n7. Tim nhan vien theo ten";
            cout << "\nDanh sach hien hanh :\n";
            Xuat_DSNV(a, n);
            cout << "\nNhap ten nhan vien can tim :";
            cin >> ten;
            TimNhanVien_Ten(a, n, ten);
            break;
        //...
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra chức năng 7.

Bước 11: Bổ sung chức năng 8 (sắp danh sách tăng dần theo mã nhân viên) vào chương trình :

- Trong `thuvien.h`

11.1 Hàm sắp danh sách tăng dần theo mã nhân viên:

```

//Sap tang theo Ma Nhan Vien
void SapTang_MaNV(NhanVien a[MAX], int n)
{
    int i, j;

```

```
//Tang theo ma nhan vien
for (i = 0; i < n - 1; i++)
    for (j = i + 1; j < n; j++)
        if (_strcmpi(a[i].maNV, a[j].maNV) > 0)
            HoanVi(a[i], a[j]);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

11.2 Hoàn hoán vị :

```
void HoanVi(NhanVien &p, NhanVien &q)
{
    NhanVien t;
    t = p;
    p = q;
    q = t;
}
```

11.3 Khai báo nguyên mẫu :

```
void SapTang_MaNV(NhanVien a[MAX], int n);
void HoanVi(NhanVien &p, NhanVien &q);
```

- Trong hàm XuLyMenu của tập tin menu.h :

+ Bổ sung xử lý hàm sắp tăng theo tên vào case 8:

```
void XuLyMenu(int menu, NhanVien a[MAX], int &n)
{
    //khai bao bien
    char maNV[8], ten[7];
    int kq;
    switch (menu)
    {
        //...
        case 8:
            system("CLS");
            cout << "\n8. Sap danh sach nhan vien tang dan theo ma nhan vien";
            cout << "\nDanh sach ban dau :\n";
            Xuat_DSNV(a, n);
            SapTang_MaNV(a, n);
            cout << "\nDanh sach ket qua :\n";
            Xuat_DSNV(a, n);
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra chức năng 8.

Bước 12: Bổ sung chức năng 9 (sắp danh sách tăng dần theo tên-họ-lương) vào chương trình :

- Trong **thuvien.h**

12.1 Hàm sắp danh sách tăng dần theo tên-họ-lương:

```
//Sap tang theo Ten-Ho-Luong
void SapTang_Ten_Ho_Luong(NhanVien a[MAX], int n)
{
    int i, j;
    //Tang theo ten
    for (i = 0; i < n - 1; i++)
        for (j = i + 1; j < n; j++)
            if (_strcmpi(a[i].ten, a[j].ten) > 0)
                HoanVi(a[i], a[j]);

    //Ten trung, tang theo ho
    for (i = 0; i < n - 1; i++)
        for (j = i + 1; j < n; j++)
            if (_strcmpi(a[i].ten, a[j].ten) == 0)
                if (_strcmpi(a[i].hoLot, a[j].hoLot) > 0)
                    HoanVi(a[i], a[j]);

    //Ten va ho trung, tang theo luong
    for (i = 0; i < n - 1; i++)
        for (j = i + 1; j < n; j++)
            if (_strcmpi(a[i].ten, a[j].ten) == 0 && _strcmpi(a[i].hoLot, a[j].hoLot) == 0)
                if (a[i].luong > a[j].luong)
                    HoanVi(a[i], a[j]);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

12.2 Khai báo nguyên mẫu :

```
void SapTang_Ten_Ho_Luong(NhanVien a[MAX], int n);
```

- Trong hàm XuLyMenu của tập tin menu.h :

+ Bổ sung xử lý hàm sắp tăng tên-họ-lương theo tên vào case 9:

```
void XuLyMenu(int menu, NhanVien a[MAX], int &n)
{
    //khai bao bien
    char maNV[8], ten[7];
    int kq;
    switch (menu)
    {
        //...
        case 9:
            system("CLS");
            cout << "\n9. Sap danh sach nhan vien tang dan theo ten-ho luong";
            cout << "\nDanh sach ban dau :\n";
            Xuat_DSNV(a, n);
            SapTang_Ten_Ho_Luong (a, n);
            cout << "\nDanh sach ket qua :\n";
            Xuat_DSNV(a, n);
            break;
        //...
    }
    _getch();
```

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra chức năng 9.

Bước 13: Bổ sung chức năng 10 (tính tổng lương) vào chương trình :

- Trong **thuvien.h**

13.1 Hàm tính tổng lương:

```
double TinhTongLuong(NhanVien a[MAX], int n)
{
    int i;
    double tong = 0;
    for (i = 0; i < n; i++)
        tong += a[i].luong;
    return tong;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

13.2 Khai báo nguyên mẫu :

```
double TinhTongLuong(NhanVien a[MAX], int n);
```

- Trong hàm XuLyMenu của tập tin menu.h :

+ Bổ sung xử lý hàm tính tổng lương theo tên vào case 10:

```
void XuLyMenu(int menu, NhanVien a[MAX], int &n)
{
    //khai bao bien
    char maNV[8], ten[7];
    int kq;
    switch (menu)
    {
        //...
        case 10:
            system("CLS");
            cout << "\n10. Tinh tong luong thang";
            cout << "\nDanh sach hien hanh :\n";
            Xuat_DSNV(a, n);
            cout << "\nTong luong thang: tong = " << TinhTongLuong(a, n);
            break;
        //...
    }
    getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra chức năng 10.

Bước 14: Bổ sung chức năng 11 (liệt kê các nhân viên có lương $\geq x$) vào chương trình :

- Trong **thuvien.h**

14.1 Hàm liệt kê các nhân viên có lương $\geq x$:

```

void Xuat_DSNV_Luong_KhongNhoHon_x(NhanVien a[MAX], int n, double x)
{
    int i, kq = 0;
    for (i = 0; i < n; i++)
        if (a[i].luong >= x)
            kq++;
    if (!kq)
        cout << "\nKhong co nhan vien nao co luong >= " << x;
    else
    {
        cout << "\n\nCo " << kq << " nhan vien co luong >= " << x
            << ", voi thong tin chi tiet nhu sau:";
        XuatTieuDe();
        for (i = 0; i < n; i++)
        {
            if (a[i].luong >= x)
            {
                cout << endl;
                Xuat_1NV(a[i]);
            }
        }
        XuatKeNgang();
    }
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

14.2 Khai báo nguyên mẫu :

```
void Xuat_DSNV_Luong_KhongNhoHon_x(NhanVien a[MAX], int n, double x);
```

- Trong hàm XuLyMenu của tập tin menu.h :

+ Bổ sung xử lý hàm liệt kê các nhân viên có lương $\geq x$ vào case 11:

```

void XuLyMenu(int menu, NhanVien a[MAX], int &n)
{
    //khai bao bien
    char maNV[8], ten[7];
    int kq;
    switch (menu)
    {
        //...
        case 11:
            system("CLS");
            cout << "\n11. Liet ke cac nhan vien co luong >= x (nhap tu ban phim)";
            cout << "\nNhap x = ";
            cin >> x;
            cout << "\nDanh sach hien hanh :\n";
            Xuat_DSNV(a, n);
            Xuat_DSNV_Luong_KhongNhoHon_x(a, n, x);
            break;
        //...
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra chức năng 11.

Bước 15: Bổ sung chức năng 12 (liệt kê các nhân viên có năm sinh trong khoảng [u,v], u <= v) vào chương trình :

- Trong **thuvien.h**

15.1 Hàm liệt kê các nhân viên có năm sinh trong khoảng [u,v], u <= v

```
void Xuat_DSNV_NamSinh_u_v(NhanVien a[MAX], int n, unsigned int u, unsigned int v)
{
    int i, kq = 0;
    for (i = 0; i < n; i++)
        if (u <= a[i].ntns.namSinh && a[i].ntns.namSinh <= v)
            kq++;
    if (!kq)
        cout << "\nKhong co nhan vien nao co nam sinh trong khoang [" << u << ", " << v << "]";
    else
    {
        cout << "\n\nCo " << kq << " nhan vien co nam sinh trong khoang [" << u
        << "...," << v << "] : ";
        XuatTieuDe();
        for (i = 0; i < n; i++)
        {
            if (u <= a[i].ntns.namSinh && a[i].ntns.namSinh <= v)
            {
                cout << endl;
                Xuat_1NV(a[i]);
            }
        }
        XuatKeNgang();
    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

15.2 Khai báo nguyên mẫu :

```
void Xuat_DSNV_NamSinh_u_v(NhanVien a[MAX], int n, unsigned int u, unsigned int v);
```

- Trong hàm XuLyMenu của tập tin menu.h :

+ Bổ sung xử lý hàm liệt kê các nhân viên có năm sinh trong khoảng [u,v], u <= v vào case 12:

```
void XuLyMenu(int menu, NhanVien a[MAX], int &n)
{
    //khai bao bien
    char maNV[8], ten[7];
    int kq;
    switch (menu)
    {
        //...
        case 12:
            system("CLS");
            cout << "\n12. Liet ke cac nhan vien co nam sinh trong khoang [u,v],u<=v";
            cout << "\nNhap u = ";
            cin >> u;
            cout << "\nNhap v = ";
            cin >> v;
            system("CLS");
```

```

cout << "\nDanh sach hien hanh :\n";
Xuat_DSNV(a, n);
Xuat_DSNV_NamSinh_u_v(a, n, u, v);
break;
//...
}
 getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra chức năng 12.

Kiểm tra các chức năng chương trình – Kết thúc chương trình.

Ghi chú:

Việc nhập dữ liệu cho mảng cấu trúc từ bàn phím nhiều lần rất mất thời gian, nên để thực hiện chức năng 1, ta có thể thay thế các hàm nhập dữ liệu cho danh sách nhân viên từ bàn phím như đã dùng bằng các hàm tạo dữ liệu sau đây:

```
//Chen 1 nhan vien
void Chen_NV(char maNV[8], char hoLot[15], char ten[7], Byte ngaySinh, Byte thangSinh,
             unsigned int namSinh, char diaChi[15], double luong, NhanVien a[MAX], int &n)
```

```
{
    if (n < MAX)
    {
        strcpy_s(a[n].maNV, 8, maNV);
        strcpy_s(a[n].hoLot, 15, hoLot);
        strcpy_s(a[n].ten, 7, ten);
        a[n].ntns.ngaySinh = ngaySinh;
        a[n].ntns.thangSinh = thangSinh;
        a[n].ntns.namSinh = namSinh;
        strcpy_s(a[n].diaChi, 15, diaChi);
        a[n].luong = luong;
        n++;
    }
}
```

//Tao Danh Sach nhan vien

```
void TaoDanhSachNhanVien(NhanVien a[MAX], int &n)
{
    Chen_NV("1234507", "Nguyen Van", "Tan", 1, 1, 1990, "Da Lat", 20345678.0, a, n);
    Chen_NV("1205507", "Tran Minh", "Hoang", 10, 10, 1980, "Qui Nhon", 14345678.0, a, n);
    Chen_NV("2234067", "Truong", "Hoang", 10, 12, 1980, "Da Lat", 10342678.0, a, n);
    Chen_NV("2134167", "Le Tuan", "Ban", 10, 1, 1975, "Quang Nam", 18349678.0, a, n);
    Chen_NV("1135067", "Tran Minh", "Ban", 5, 6, 1977, "Phu Yen", 12345978.0, a, n);
    Chen_NV("1034007", "Truong", "Hoang", 21, 12, 1987, "Da Lat", 15348678.0, a, n);
    Chen_NV("1130160", "Vo Tuan", "Trong", 1, 11, 1985, "Qui Nhon", 20341178.0, a, n);
    Chen_NV("1235567", "Hoang Van", "Tan", 15, 12, 1970, "Phu Yen", 22300970.0, a, n);
    Chen_NV("1034327", "Tran Van", "Vu", 11, 11, 1982, "Da Lat", 16348888.0, a, n);
    Chen_NV("1034467", "Hoang Trong", "Ban", 12, 1, 1980, "Nha Trang", 14365698.0, a, n);
    Chen_NV("1134467", "Van Quoc", "Cuong", 1, 10, 1960, "Nha Trang", 24300000.0, a, n);
    Chen_NV("1533407", "Truong", "Hoang", 1, 12, 1967, "Da Nang", 16340000.0, a, n);
}
```

Bài 2. Số phân số

$$Q = \left\{ x = \frac{tu}{mau} : tu \in Z, mau \in N^* \right\}$$

Viết chương trình tùy chọn thực hiện trên các số phân số với các chức năng sau :

0. Thoát khỏi chương trình
1. Nhập phân số
2. Xem phân số
3. Rút gọn phân số
4. Cộng 2 phân số
5. Trừ phân số
6. Nhân 2 phân số
7. Chia phân số
8. So sánh 2 phân số

Tạo project với tên : **Lab08_D_Bai2_PhanSo.**

Tổ chức chương trình tùy chọn menu như bài 1.

Tham khảo tập tin *thuvien.h* sau đây :

```
//Khai bao hang : khong co
//Dinh nghia Kieu du lieu : PhanSo
struct PhanSo
{
    int tu;
    unsigned int mau;
};

//Khai bao nguyen mau cac ham chuc nang, nhap xuat
PhanSo Nhap_PS();
void XuatPS(PhanSo a);
unsigned int Tinh_GTTD(int x);
unsigned int UCLN(unsigned int a, unsigned int b);
unsigned int BCNN(unsigned int a, unsigned int b);
void RutGon(PhanSo &a);
void QuyDong_PS(PhanSo &a, PhanSo &b);
PhanSo CongPhanSo(PhanSo a, PhanSo b);
PhanSo TruPhanSo(PhanSo a, PhanSo b);
PhanSo NhanPhanSo(PhanSo a, PhanSo b);
PhanSo ChiaPhanSo(PhanSo a, PhanSo b);
int SoSanhPhanSo(PhanSo a, PhanSo b);
```

//Dinh nghia cac ham

```
//Ham nhap phan so
PhanSo Nhap_PS()
{
    PhanSo a;
    cout << "\nNhập tu so : ";
    cin >> a.tu;
    do
    {
```

```

        cout << "\nNhap mau so (> 0) : ";
        cin >> a.mau;
    } while (a.mau <= 0);
    return a;
}
//Ham xuat phan so
void XuatPS(PhanSo a)
{
    if (a.tu == 0)
    {
        cout << 0;
        return;
    }
    if (a.mau == 1)
    {
        cout << a.tu;
        return;
    }

    cout << a.tu << '/' << a.mau;
}

```

```

//gia tri tuyet doi
unsigned int Tinh_GTTD(int x)
{
    return (x >= 0 ? x : -x);
}

```

```

//Tinh uoc chung lon nhat 2 so nguyen duong
unsigned int UCLN(unsigned int a, unsigned int b)
{
    unsigned int r;
    while (b > 0)
    {
        r = a % b;
        a = b;
        b = r;
    }
    return a;
}

```

```

//Boi chung nho nhat 2 so nguyen duong
unsigned int BCNN(unsigned int a, unsigned int b)
{
    return (a * b) / UCLN(a, b);
}

```

```

//Rut gon phan so
void RutGon(PhanSo &a)
{
    unsigned int d;
    d = UCLN(Tinh_GTTD(a.tu), a.mau);

```

```
a.mau /= d;
a.tu = a.tu/(int)d;
}

//quy đồng mẫu số 2 phân số
void QuyDong_PS(PhanSo &a, PhanSo &b)
{
    unsigned int d;
    RutGon(a);
    RutGon(b);
    d = BCNN(a.mau, b.mau);
    a.tu = a.tu * (d / a.mau);
    b.tu = b.tu * (d / b.mau);
    a.mau = b.mau = d;
}

//c = a+b
PhanSo CongPhanSo(PhanSo a, PhanSo b)
{
    PhanSo c;
    QuyDong_PS(a, b);
    c.tu = a.tu + b.tu;
    c.mau = a.mau;
    RutGon(c);
    return c;
}

//c = a - b
PhanSo TruPhanSo(PhanSo a, PhanSo b)
{
    PhanSo c;
    QuyDong_PS(a, b);
    c.tu = a.tu - b.tu;
    c.mau = a.mau;
    RutGon(c);
    return c;
}

//c = a * b
PhanSo NhanPhanSo(PhanSo a, PhanSo b)
{
    PhanSo c;
    RutGon(a);
    RutGon(b);
    c.tu = a.tu * b.tu;
    c.mau = a.mau * b.mau;
    RutGon(c);
    return c;
}

//c = a / b
PhanSo ChiaPhanSo(PhanSo a, PhanSo b)
{
    PhanSo c;
```

```

RutGon(a);
RutGon(b);
int tu;
int mau;
tu = a.tu * b.mau;
mau = a.mau * b.tu;
if (mau < 0)
{
    tu = -tu;
    mau = -mau;
}
c.tu = tu;
c.mau = (unsigned int) mau;

RutGon(c);
return c;
}

//So sánh 2 phân số
//kq = -1: a < b
//kq = 0 : a = b
//kq = 1 : a > b
int SoSanhPhanSo(PhanSo a, PhanSo b)
{
    int kq;
    QuyDong_PS(a, b);
    if (a.tu < b.tu)
        kq = -1;
    else
        if (a.tu == b.tu)
            kq = 0;
        else
            kq = 1;
    return kq;
}

```

Bài 3:

Bảng điểm môn “**Lập trình cấu trúc**” bao gồm các thông tin :

- Mã sinh viên: một chuỗi có đúng 7 ký tự (số, chữ)
- Họ và chữ lót sinh viên : chuỗi có không quá 15 ký tự
- Tên sinh viên : chuỗi không quá 6 ký tự
- Năm sinh : số nguyên dương (4 chữ số)
- Giới tính : không quá 3 ký tự
- Quê quán (tỉnh, thành phố) : chuỗi không quá 15 ký tự
- Điểm tổng kết : Điểm tổng kết: một số thực trong đoạn [0,..,10]

Điểm tổng kết môn học được tính từ các cột điểm :

- Điểm bài lab : chiếm 20% điểm tổng kết
- Điểm bài KTGK (2 bài) : chiếm 20 % điểm tổng kết
- Điểm kiểm tra cuối kỳ (điểm CK): chiếm 60% điểm tổng kết

Các thông tin còn lại trong bảng điểm được trích từ hồ sơ lý lịch sinh viên.

Viết chương trình (không phải viết hệ thống tùy chọn menu):

- Xuất ra Danh sách sinh viên gồm các thông tin : mã sinh viên, họ tên sinh viên, giới tính, quê quán, lớp
 - Tính điểm tổng kết và xuất ra bảng điểm chi tiết môn học gồm các thông tin : Mã sinh viên, Điểm bài lab, điểm KTGK, Điểm CK, Điểm tổng kết.
 - Xuất bảng điểm chính thức gồm các thông tin về sinh viên và điểm tổng kết môn học.
- Danh sách sinh viên cho trong bảng sau :

Danh sach sinh vien mon LTCT:

:Ma NV	:Ho	Ten	:GT	:NS	:Que quan	:Lop
:1512967	:Trieu	Minh	:Nu	:1997	:Ninh Thuan	:CTK39
:1510279	:Nguyen Van	Tan	:Nam	:1997	:Da Lat	:CTK39
:1512555	:Tran Tuan	Ngoc	:Nam	:1996	:Khanh Hoa	:CTK39
:1412120	:Vu Thi	Hoa	:Nu	:1996	:Binh Dinh	:CTK38
:1313320	:Le Ngoc	Minh	:Nam	:1995	:Can Tho	:CTK37
:						
:1510214	:Dinh Thi	Ngoc	:Nu	:1997	:Da Lat	:CTK39
:1512128	:Ta Van	Ton	:Nam	:1997	:Binh Dinh	:CTK39
:1512868	:Doan	Du	:Nam	:1996	:Da Lat	:CTK39
:1512887	:Uuong Ngoc	Yen	:Nu	:1997	:Da Nang	:CTK39
:1514205	:Uuong Trung	Duong	:Nam	:1995	:Phu Yen	:CTK39
:						
:1510192	:Nhac Linh	San	:Nu	:1997	:Da Lant	:CTK39
:1312890	:Hoang	Dung	:Nu	:1995	:Da Nang	:CTK37
:1444405	:Truong Vo	Ky	:Nam	:1996	:Binh Dinh	:CTK38
:1512988	:Vi Tieu	Bao	:Nam	:1996	:Da Lat	:CTK39
:						
So luong sinh vien trong danh sach : n = 14						

- Bảng điểm chi tiết môn học cho trong bảng sau:

Bang diem chi tiet mon LTCT:

:Ma NV	:Diem Lab	:Diem KTGK	:Diem Cuoi Ky	:Diem TK
:1512967	:9.0	:8.0	:6.0	:7.0
:1510279	:10.0	:6.0	:5.0	:6.2
:1512555	:8.0	:5.0	:7.0	:6.8
:1412120	:9.5	:9.0	:8.0	:8.5
:1313320	:8.0	:8.0	:4.0	:5.6
:				
:1510214	:8.5	:8.0	:6.0	:6.9
:1512128	:9.0	:9.0	:9.0	:9.0
:1512868	:10.0	:8.5	:8.5	:8.8
:1512887	:7.0	:7.0	:7.0	:7.0
:1514205	:8.0	:8.0	:8.5	:8.3
:				
:1510192	:10.0	:9.0	:9.0	:9.2
:1312890	:9.0	:9.0	:8.0	:8.4
:1444405	:7.0	:5.0	:5.0	:5.4
:1512988	:8.5	:8.0	:8.0	:8.1
:				
So luong sinh vien trong danh sach : n = 14				

- Hai bảng trên có cùng thứ tự trong cột mã sinh viên.

Chương trình tổ chức theo thư viện hàm (*và không có hệ thống menu.*)

Bước 1. Tạo Project rỗng mới đặt tên **Lab08_D_Bai3**

Bước 2. Tạo cấu trúc cho chương trình như đã hướng dẫn trong **mục 1 lab 4** (từ 1-7 để có cấu trúc nội dung tối thiểu chạy được chương trình, dạng không có hệ thống menu).

Tức là ta có kết quả chương trình ở bước này như sau :

- Tập tin **thuvien.h** : Rỗng (chỉ có các chú thích về cấu trúc văn bản)
- Tập tin **program.cpp** có nội dung như sau :

```
#include <iostream>
#include <conio.h>
using namespace std;
#include "thuvien.h"
void ChayChuongTrinh();
int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    _getch();
}
```

Nhấn Ctrl + F5 để chạy chương trình, sửa lỗi nếu có.

Bước 3:

Bước này ta bổ sung thêm định nghĩa hằng, kiểu dữ liệu mới,

- Trong tập tin **thuvien.h :**

+ Bổ sung định nghĩa hằng, kiểu dữ liệu mới :

```
//Dinh nghia hang
#define MAX 100//kich thuoc khai mang cau truc
#define NGANGDOI '=' //ke ngang doi khi xuat du lieu
#define NGANGDON '-' //Ke ngang don khi xuat du lieu

//Dinh nghia kieu du lieu moi : cac kieu cau truc
//Kieu Sinh Vien
struct SinhVien
{
    char maSV[8];
    char hoLot[15];
    char ten[8];
    char gioiTinh[4];
    unsigned int namSinh;
    char queQuan[15];
    char lop[6];
};

struct BangDiem_ChiTiet
{
    char maSV[8];
    double diemLab;
    double baiKTGK;
    double baiKTCK;
    double diemTK;
};
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

//Khai bao nguyen mau

//Định nghĩa các hàm

Bước 4 : Bổ sung các hàm nhập xuất dữ liệu và kiểm tra kết quả xuất danh sách sinh viên

4.1 Bổ sung các thư viện cần thiết :

- Trong tập tin *program.cpp* :

Bổ sung thêm thư viện *<string.h>*, *<iomanip>*

4.2 Định nghĩa các hàm nhập dữ liệu :

```
//=====================================================================//  
////////// Tao Danh sach sinh vien ////////////  
//=====================================================================//  
//Chen 1 sinh vien  
void Chen_SV(char maSV[8], char hoLot[15], char ten[7], char gioiTinh[3], unsigned int namSinh,  
            char queQuan[15], char lop[7], SinhVien a[MAX], int &n)  
{  
    if (n < MAX)  
    {  
        strcpy_s(a[n].maSV, 8, maSV);  
        strcpy_s(a[n].hoLot, 15, hoLot);  
        strcpy_s(a[n].ten, 8, ten);  
        strcpy_s(a[n].gioiTinh, 4, gioiTinh);  
        a[n].namSinh = namSinh;  
        strcpy_s(a[n].queQuan, 15, queQuan);  
        strcpy_s(a[n].lop, 6, lop);  
  
        n++;  
    }  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

//Tao Danh Sach sinh vien

```
void TaoDanhSachSinhVien(SinhVien a[MAX], int &n)  
{  
    n = 0;  
    Chen_SV("1512967", "Trieu", "Minh", "Nu", 1997, "Ninh Thuan", "CTK39", a, n);  
    Chen_SV("1510279", "Nguyen Van", "Tan", "Nam", 1997, "Da Lat", "CTK39", a, n);  
    Chen_SV("1512555", "Tran Tuan", "Ngoc", "Nam", 1996, "Khanh Hoa", "CTK39", a, n);  
    Chen_SV("1412120", "Vo Thi", "Hoa", "Nu", 1996, "Binh Dinh", "CTK38", a, n);  
    Chen_SV("1313320", "Le Ngoc", "Minh", "Nam", 1995, "Can Tho", "CTK37", a, n);  
    Chen_SV("1510214", "Dinh Thi", "Ngoc", "Nu", 1997, "Da Lat", "CTK39", a, n);  
    Chen_SV("1512128", "Ta Van", "Ton", "Nam", 1997, "Binh Dinh", "CTK39", a, n);  
    Chen_SV("1512868", "Doan", "Du", "Nam", 1996, "Da Lat", "CTK39", a, n);  
    Chen_SV("1512887", "Vuong Ngoc", "Yen", "Nu", 1997, "Da Nang", "CTK39", a, n);  
    Chen_SV("1514205", "Vuong Trung", "Duong", "Nam", 1995, "Phu Yen", "CTK39", a, n);  
    Chen_SV("1510192", "Nhac Linh", "San", "Nu", 1997, "Da Lant", "CTK39", a, n);  
    Chen_SV("1312890", "Hoang", "Dung", "Nu", 1995, "Da Nang", "CTK37", a, n);
```

```
Chen_SV("1444405", "Truong Vo", "Ky", "Nam", 1996, "Binh Dinh", "CTK38", a, n);
Chen_SV("1512988", "Vi Tieu", "Bao", "Nam", 1996, "Da Lat", "CTK39", a, n);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.3 Đinh nghĩa các hàm xuất dữ liệu

```
//==============================================================================
// Xuat Danh sach sinh vien /////////////////////////////////
//==============================================================================
```

```
//Xuat ke ngang doi
void XuatKeNgangDoi()
```

```
{
    int i;
    cout << "\n";
    cout << setiosflags(ios::left)
        << ':';
    for (i = 1; i <= 66; i++)
        cout << NGANGDOI;
    cout << ':';
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuat ke ngang don
void XuatKeNgangDon()
```

```
{
    int i;
    cout << "\n";
    cout << setiosflags(ios::left)
        << ':';
    for (i = 1; i <= 66; i++)
        cout << NGANGDON;
    cout << ':';
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuat tieu de
void XuatTieuDe()
```

```
{
    XuatKeNgangDoi();
    cout << endl;
    cout << setiosflags(ios::left)
        << ':'
        << setw(8) << "Ma NV"
        << ':'
        << setw(15) << "Ho"
        << setw(7) << "Ten"
        << ':'
        << setw(5) << "GT"
        << ':'
        << setw(5) << "NS"
```

```
<< ' '
<< setw(15) << "Que quan"
<< ' '
<< setw(6) << "Lop"
<< ' ';
XuatKeNgangDoi();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

//Xuat 1 sinh viên

```
void Xuat_1SV(SinhVien p)
{
    cout << setiosflags(ios::left)
        << ' '
        << setw(8) << p.maSV
        << ' '
        << setw(15) << p.hoLot
        << setw(7) << p.ten
        << ' '
        << setw(5) << p.gioiTinh
        << ' '
        << setw(5) << p.namSinh
        << ' '
        << setw(15) << p.queQuan
        << ' '
        << setw(6) << p.lop
        << ' ';
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

//Xuat DS sinh viên

```
void Xuat_DSSV(SinhVien a[MAX], int n)
{
    int i;
    XuatTieuDe();
    for (i = 0; i < n; i++)
    {
        cout << endl;
        Xuat_1SV(a[i]);
        if ((i + 1) % 5 == 0)
            XuatKeNgangDon();
    }
    XuatKeNgangDoi();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.4 Khai báo nguyên mẫu các hàm nhập, xuất :

```
void Chen_SV(char maSV[8], char hoLot[15], char ten[7], char gioiTinh[3], unsigned int namSinh,
            char queQuan[15], char lop[7], SinhVien a[MAX], int &n);
void TaoDanhSachSinhVien(SinhVien a[MAX], int &n);
void XuatKeNgangDoi();
void XuatKeNgangDon();
void XuatTieuDe();
void Xuat_1SV(SinhVien p);
```

```
void Xuat_DSSV(SinhVien a[MAX], int n);
```

4.5 Kiểm tra chức năng nhập, xuất danh sách sinh viên

- Trong tập tin **program.cpp**, ta cập nhật hàm **ChayChuongTrinh** :

```
void ChayChuongTrinh()
{
    int n;
    SinhVien a[MAX];

    system("CLS");
    TaoDanhSachSinhVien(a, n);
    cout << "\nDanh sach sinh vien mon LTCT:\n";
    Xuat_DSSV(a, n);
    cout << "\nSo luong sinh vien trong danh sach : n = " << n;
    getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra các chức năng tạo và xuất danh sách sinh viên.

Bước 5: Tạo bảng điểm chi tiết và tính điểm tổng kết môn học

- Trong **thuvien.h**, ta bổ sung các hàm sau :

5.1 Tạo bảng điểm chi tiết :

```
//=====================================================================
// /////////////////////////////////////////////////////////////////// Tao bang diem chi tiet ///////////////////////////////////////////////////////////////////
//=====================================================================

//Chen diem 1 sinh vien
void Chen_Diem(char maSV[8], double diemLab, double baiKTGK, double baiKTCK, BangDiem_ChiTiet bd[MAX], int &n)
{
    if (n < MAX)
    {
        strcpy_s(bd[n].maSV, 8, maSV);
        bd[n].diemLab = diemLab;
        bd[n].baiKTGK = baiKTGK;
        bd[n].baiKTCK = baiKTCK;
        bd[n].diemTK = 0; //khai tao
        n++;
    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

//Tạo bảng điểm chi tiết : chưa tính điểm tổng kết

```
void TaoBangDiem_ChiTiet(BangDiem_ChiTiet bd[MAX], int &n)
{
    n = 0;
    Chen_Diem("1512967", 9, 8, 6, bd, n );
```

```

Chen_Diem("1510279", 10, 6, 5, bd, n);
Chen_Diem("1512555", 8, 5, 7, bd, n);
Chen_Diem("1412120", 9.5, 9, 8, bd, n);
Chen_Diem("1313320", 8, 8, 4, bd, n);
Chen_Diem("1510214", 8.5, 8, 6, bd, n);
Chen_Diem("1512128", 9, 9, 9, bd, n);
Chen_Diem("1512868", 10, 8.5, 8.5, bd, n);
Chen_Diem("1512887", 7, 7, 7, bd, n);
Chen_Diem("1514205", 8, 8, 8.5, bd, n);
Chen_Diem("1510192", 10, 9, 9, bd, n);
Chen_Diem("1312890", 9, 9, 8, bd, n);
Chen_Diem("1444405", 7, 5, 5, bd, n);
Chen_Diem("1512988", 8.5, 8, 8, bd, n);
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

//Tinh diem tong ket mon hoc

```

void TinhDiemTongKet(BangDiem_ChiTiet bd[MAX], int &n)
{
    int i;
    double tam;
    for (i = 0; i < n; i++)
    {
        tam = bd[i].diemLab * 0.2 + bd[i].baiKTGK * 0.2 + bd[i].baiKTCK * 0.6;
        bd[i].diemTK = tam;
    }
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.2 Các hàm xuất bang điểm chi tiết:

```

//=====
// Xuat bang diem chi tiet
//=====

```

```

//Xuat tieu de bang diem chi tiet
void XuatTieuDe_bd()
{
    XuatKeNgangDoi();
    cout << endl;
    cout << setiosflags(ios::left)
        << ':'
        << setw(8) << "Ma NV"
        << ':'
        << setw(13) << "Diem Lab"
        << ':'
        << setw(13) << "Diem KTGK"
        << ':'
        << setw(13) << "Diem Cuoi Ky"
        << ':'
        << setw(15) << "Diem TK"
        << ':';
    XuatKeNgangDoi();
}

```

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuat diem 1 sinh vien
void Xuat_Diem_1SV(BangDiem_ChiTiet p)
{
    cout << setiosflags(ios::left)
        << ' '
        << setw(8) << p.maSV
        << ' '
        << setw(13) << setiosflags(ios::fixed) << setprecision(1) << p.diemLab
        << ' '
        << setw(13) << setiosflags(ios::fixed) << setprecision(1) << p.baiKTGK
        << ' '
        << setw(13) << setiosflags(ios::fixed) << setprecision(1) << p.baiKTCK
        << ' '
        << setw(15) << setiosflags(ios::fixed) << setprecision(1) << p.diemTK
        << ' ';
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuat bang diem chi tiet
void Xuat_BangDiem_ChiTiet(BangDiem_ChiTiet bd[MAX], int n)
{
    int i;
    XuatTieuDe_bd();
    for (i = 0; i < n; i++)
    {
        cout << endl;
        Xuat_Diem_1SV(bd[i]);
        if ((i + 1) % 5 == 0)
            XuatKeNgangDon();
    }
    XuatKeNgangDoi();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.3 Khai báo nguyên mẫu :

```
void Chen_Diem(char maSV[8], double diemLab, double baiKTGK, double baiKTCK, BangDiem_ChiTiet bd[MAX], int &n);
void TaoBangDiem_ChiTiet(BangDiem_ChiTiet bd[MAX], int &n);
void TinhDiemTongKet(BangDiem_ChiTiet bd[MAX], int &n);
void XuatTieuDe_bd();
void Xuat_Diem_1SV(BangDiem_ChiTiet p);
void Xuat_BangDiem_ChiTiet(BangDiem_ChiTiet bd[MAX], int n);
```

5.4 Kiểm tra chức năng tạo và xuất bảng điểm chi tiết :

- Trong tập tin **program.cpp**, ta cập nhật lại hàm **ChayChuongTrinh** :

```
void ChayChuongTrinh()
{
    int n;
```

```

SinhVien a[MAX];
BangDiem_ChiTiet bd[MAX];

system("CLS");
TaoDanhSachSinhVien(a, n);
cout << "\nDanh sach sinh vien mon LTCT:\n";
Xuat_DSSV(a, n);
cout << "\nSo luong sinh vien trong danh sach : n = " << n;
_getch();

system("CLS");
TaoBangDiem_ChiTiet(bd, n);
TinhDiemTongKet(bd, n);
cout << "\nBang diem chi tiet mon LTCT:\n ";
Xuat_BangDiem_ChiTiet(bd, n);
cout << "\nSo luong sinh vien trong danh sach : n = " << n;
_getch();

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra các chức năng tạo và xuất bảng điểm chi tiết.

```

Bước 6: Xuất bảng điểm chính thức là bảng gồm các thông tin sinh viên và điểm tổng kết.

6.1 Hàm xuất bảng điểm chính thức:

```

//=====
//Xuất bảng điểm chính thức
//=====

void XuatBangDiem_ChinhThuc(SinhVien a[MAX], BangDiem_ChiTiet bd[MAX], int n)
{
    //xuat tiêu đề
    cout << endl << "=====\n";
    cout << setiosflags(ios::left)
        << ':';
    cout << endl << "=====\n";
    cout << setiosflags(ios::left)
        << ':' << setw(8) << "Ma SV"
        << ':' << setw(15) << "Ho"
        << ':' << setw(7) << "Ten"
        << ':' << setw(5) << "GT"
        << ':' << setw(5) << "NS"
        << ':' << setw(15) << "Que quan"
        << ':' << setw(6) << "Lop"
        << ':' << setw(8) << "Diem TK"
        << ':\n';
    cout << endl << "=====\n";
    int i;

```

```

for (i = 0; i < n; i++)
{
    cout << endl;
    cout << setiosflags(ios::left)
        << ':'
        << setw(8) << a[i].maSV
        << ':'
        << setw(15) << a[i].hoLot
        << setw(7) << a[i].ten
        << ':'
        << setw(5) << a[i].gioiTinh
        << ':'
        << setw(5) << a[i].namSinh
        << ':'
        << setw(15) << a[i].queQuan
        << ':'
        << setw(6) << a[i].lop
        << ':'
        << setw(8) << setiosflags(ios::fixed) << setprecision(1) << bd[i].diemTK
        << ':';
    if ((i + 1) % 5 == 0)
        cout << endl << "-----";
}
cout << endl << "=====";
}

```

6.2 Khai báo nguyên mẫu:

```
void XuatBangDiem_ChinhThuc(SinhVien a[MAX], BangDiem_ChiTiet bd[MAX], int n);
```

6.3. Kiểm tra chức năng xuất bảng điểm chính thức:

- Trong tập tin **program.cpp**, ta cập nhật lại hàm **ChayChuongTrinh** :

```

void ChayChuongTrinh()
{
    int n;
    SinhVien a[MAX];
    BangDiem_ChiTiet bd[MAX];

    system("CLS");
    TaoDanhSachSinhVien(a, n);
    cout << "\nDanh sach sinh vien mon LTCT:\n";
    Xuat_DSSV(a, n);
    cout << "\nSo luong sinh vien trong danh sach : n = " << n;
    _getch();

    system("CLS");
    TaoBangDiem_ChiTiet(bd, n);
    TinhDiemTongKet(bd, n);
    cout << "\nBang diem chi tiet mon LTCT:\n ";
    Xuat_BangDiem_ChiTiet(bd, n);
    cout << "\nSo luong sinh vien trong danh sach : n = " << n;
    _getch();
}

```

```

system("CLS");
cout << "\nBang diem chinh thuc mon LTCT:\n";
XuatBangDiem_ChinhThuc(a, bd, n);
cout << "\nSo luong sinh vien trong danh sach : n = " << n;
cout << endl;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra chức năng xuất bảng điểm chính thức.

Kiểm tra thực hiện các yêu cầu ban đầu – kết thúc chương trình.

Bài 4:

Đa thức một biến bậc n, hệ số thực có dạng:

$$F(x) = c_0 + c_1x^1 + \dots + c_nx^n ; c_i \in \mathbb{R}, \forall i; c_n \neq 0.$$

Có thể biểu diễn bởi một mảng các đơn thức có dạng $c_i x^i$.

Mỗi đơn thức là một cấu trúc gồm 2 trường: **Hệ số và số mũ**.

Viết chương trình tùy chọn thực hiện các chức năng dưới đây:

0. Thoát khỏi chương trình
1. Nhập đa thức
2. Xuất đa thức
3. Cộng 2 đa thức
4. Trừ đa thức
5. Nhân 2 đa thức
6. Tính giá trị của đa thức tại x

Tạo project rỗng mới, đặt tên **Lab08_D_Bai4**.

Tổ chức chương trình tùy chọn menu như bài 1.

- Tham khảo tập tin **thuvien.h** sau đây :

```

//Dinh nghia hang
#define MAX 100
//Dinh nghia kieu du lieu
//Kieu don thuc
struct DonThuc
{
    double heSo;
    int mu;
};

//Kieu da thuc : mang cac don thuc
typedef DonThuc DaThuc[MAX];

//Khai bao nguyen mau
void RutGon_DT(DaThuc A, int &n);
void Khu_HS0(DaThuc A, int &n);
void SapTangTheoMu(DaThuc A, int n);
void DinhDang_DT(DaThuc A, int &n);

void NhapDaThuc(DaThuc A, int &n, char kt);
void XuatDaThuc(DaThuc A, int n);

```

```

void Tong2DaThuc(DaThuc A, int m, DaThuc B, int n, DaThuc C, int &k);
void Hieu2DaThuc(DaThuc A, int m, DaThuc B, int n, DaThuc C, int &k);
void Nhan_DonThuc(DaThuc A, int m, DonThuc p, DaThuc C, int &k);
void NhanDaThuc(DaThuc A, int m, DaThuc B, int n, DaThuc C, int &k);
double x_Mu_m(double x, int m);
double A_x(DaThuc A, int m, double x);

```

//Dinh nghia cac ham

```

//=====================================================================
//Dinh dang da thuc : Rut gon - Khu he so 0 - Sap tang theo so mu :
//=====================================================================

//Rut gon da thuc : don cac don thuc co so mu bang nhau lai voi nhau
void RutGon_DT(DaThuc A, int &n)
{
    DaThuc B; //Bien trung gian
    int i;//duyet A
    j; //duyet B
    int h = 0;//kich thuoc khoi tao cua B
    kq;
    for (i = 0; i < n; i++)
    {
        kq = 0;
        for (j = 0; j < h; j++)
            if (A[i].mu == B[j].mu)
            {
                kq = 1;
                break;
            }
        if (kq == 1)
            B[j].heSo += A[i].heSo;
        else
            B[h++] = A[i];
    }
    n = h;
    for (j = 0; j < h; j++) //Gan B cho A
        A[j] = B[j];
}

//Loai he so 0
void Khu_HS0(DaThuc A, int &n)
{
    int i,
        h = 0;
    DaThuc B;
    for (i = 0; i < n; i++)
        if (A[i].heSo != 0)
            B[h++] = A[i];
    n = h;
    for (i = 0; i < n; i++)
        A[i] = B[i];
}

//Sap tang theo so mu
void SapTangTheoMu(DaThuc A, int n)

```

```

{
    int i, j;
    DonThuc t;
    for (i = 0; i < n - 1; i++)
        for (j = i + 1; j < n; j++)
            if (A[i].mu > A[j].mu)
            {
                t = A[i];
                A[i] = A[j];
                A[j] = t;
            }
}
void DinhDang_DT(DaThuc A, int &n)
{
    if (n == 0)
    {
        cout << "\nDa thuc rong!";
        _getch();
        return;
    }
    RutGon_DT(A, n);
    Khu_HS0(A, n);
    SapTangTheoMu(A, n);
}
//=====================================================================
//Nhập, Xuất da thục : da thục đã được định dạng
//=====================================================================
//Nhập da thục
void NhapDaThuc(DaThuc A, int &n, char kt)
{
    int i;
    cout << "\nNhập kích thước mảng : ";
    cin >> n;
    for (i = 0; i < n; i++)
    {
        cout << "\nDa thục " << kt << "[" << i << "] :";
        cout << "\nNhập hệ số != 0 : hệ số = ";
        cin >> A[i].heSo;
        cout << "\nNhập số mu : Số mu = ";
        cin >> A[i].mu;
    }
    DinhDang_DT(A, n);
}

//Xuất Da thục : mu biểu diễn bảng ^
void XuatDaThuc(DaThuc A, int n)
{
    if (n == 0)
    {
        cout << "\nDa thục rong!";
    }
}

```

```

        _getch();
        return;
    }
    int i = 0;
    //xử lý đơn thuần, số mũ có thể == 0
    if (A[i].mu == 0) //so mu = 0
        cout << A[i].heSo;
    else // so mu > 0 : >= 1
    {
        if (A[i].mu == 1) // so mu = 1
        {
            if (A[i].heSo == 1) //he so == 1
                cout << "x ";
            else // he so !=1
            if (A[i].heSo == -1)
                cout << "-x ";
            else
                cout << A[i].heSo << "x ";
        }
        else //so mu > 1
        {
            if (A[i].heSo == 1) //he so == 1
                cout << "x^" << A[i].mu << " ";
            else // he so !=1
            if (A[i].heSo == -1)
                cout << "-x^" << A[i].mu << " ";
            else
                cout << A[i].heSo << "x^" << A[i].mu << " ";
        }
    }
    //Xử lý các đơn thuần tiếp theo
    i++;
    while (i < n)
    {
        //so mu > 0
        if (A[i].mu == 1) // so mu = 1
        {
            if (A[i].heSo == 1) //he so == 1
                cout << "+x ";
            else // he so !=1
            if (A[i].heSo == -1)
                cout << "-x ";
            else
                if (A[i].heSo > 0)
                    cout << "+" << A[i].heSo << "x ";
                else
                    cout << A[i].heSo << "x ";
        }
        else //so mu > 1
        {
            if (A[i].heSo == 1) //he so == 1
                cout << "+x^" << A[i].mu << " ";
            else // he so !=1
            if (A[i].heSo == -1)

```

```

cout << "-x^" << A[i].mu << " ";
else
if (A[i].heSo > 0)
    cout << "+" << A[i].heSo << "x^" << A[i].mu << " ";
else // (he so < 0)
    cout << A[i].heSo << "x^" << A[i].mu << " ";
}
i++;
}
}

//=====:
// Cac phep toan da thuc      :
//=====:
// C = A + B
void Tong2DaThuc(DaThuc A, int m, DaThuc B, int n, DaThuc C, int &k )
{
    int i = 0, j = 0;
    k = 0;
    DonThuc x;

    while (i < m && j < n)
    {
        if (A[i].mu < B[j].mu)
        {
            C[k++] = A[i];
            i++;
        }
        else
        if (B[j].mu < A[i].mu)
        {
            C[k++] = B[j];
            j++;
        }
        else // 2 mu bang nhau
        {
            x.heSo = A[i].heSo + B[j].heSo;
            x.mu = A[i].mu;
            if (x.heSo != 0) // khu he so 0
                C[k++] = x;
            i++;
            j++;
        }
    }

    while (i < m)
    {
        C[k++] = A[i];
        i++;
    }

    while (j < n)
    {
        C[k++] = B[j];
    }
}

```

```

        j++;
    }
    DinhDang_DT(C, k);
}

//C = A - B
void Hieu2DaThuc(DaThuc A, int m, DaThuc B, int n, DaThuc C, int &k)
{
    int i = 0, j = 0;
    k = 0;
    DonThuc x;

    while (i < m && j < n)
    {
        if (A[i].mu < B[j].mu)
        {
            C[k++] = A[i];
            i++;
        }
        else
        if (B[j].mu < A[i].mu)
        {
            x.mu = B[j].mu;
            x.heSo = -B[j].heSo;
            C[k++] = x;
            j++;
        }
        else //2 mu bang nhau
        {
            x.heSo = A[i].heSo - B[j].heSo;
            x.mu = A[i].mu;
            if (x.heSo != 0) // khu he so 0
                C[k++] = x;
            i++;
            j++;
        }
    }

    while (i < m)
    {
        C[k++] = A[i];
        i++;
    }

    while (j < n)
    {
        x.mu = B[j].mu;
        x.heSo = -B[j].heSo;
        C[k++] = x;
        j++;
    }
    DinhDang_DT(C, k);
}

```

```
//Nhận một dãy thực với một đơn vị
void Nhan_DonThuc(DaThuc A, int m, DaThuc p, DaThuc C, int &k)
{
    int i;
    DaThuc x;
    k = 0;
    for (i = 0; i < m; i++)
    {
        x.mu = A[i].mu + p.mu;
        x.heSo = A[i].heSo * p.heSo;
        C[k++] = x;
    }
    DinhDang_DT(C, k);
}
```

```
//C = AB
void NhanDaThuc(DaThuc A, int m, DaThuc B, int n, DaThuc C, int &k)
{
    DaThuc T1, T2;
    int i, j, h1, h2;
    Nhan_DonThuc(B, n, A[0], C, k);
```

```
for (i = 1; i < m; i++)
{
    Nhan_DonThuc(B, n, A[i], T1, h1);
    Tong2DaThuc(C, k, T1, h1, T2, h2);
    for (j = 0; j < h2; j++)
        C[j] = T2[j];
    k = h2;
}
DinhDang_DT(C, k);
```

```
double x_Mu_m(double x, int m)
{
    double t = 1;
    int i;
    for (i = 1; i <= m; i++)
        t *= x;
    return t;
}
```

```
//Tính A(x)
double A_x(DaThuc A, int m, double x)
{
    double v = 0, t;
    int i;
    for (i = 0; i < m; i++)
    {
        t = A[i].heSo * x_Mu_m(x, A[i].mu);
        v += t;
    }
    return v;
}
```

- Để khởi đầu thời gian nhập dữ liệu, có thể khởi đầu các đa thức A, B trong hàm ChayChuongTrinh() của tập tin **program.cpp**. Trong quá trình thực hiện chương trình, muốn thay đổi bộ dữ liệu mới có thể chọn thực hiện chức năng 1 để nhập đa thức từ bàn phím.

```
void ChayChuongTrinh()
{
    int soMenu = 6,
        menu;

    int m = 3, n = 4; //khai báo m, n

    //khai báo A, B
    DaThuc      A = { {1,1}, {4,2}, {-5,6} };
    DaThuc      B = { {1, 0 }, {-4, 2}, {3, 3 }, {6, 4 } };

    do
    {
        system("CLS");
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, A,m, B, n);
    } while (menu > 0);
}
```

E. Bài tập bắt buộc

Tất cả các bài tập phải được tổ chức dưới dạng thư viện hàm và có menu chức năng.

1. Quản lý sinh viên

Bảng điểm sinh viên chứa các thông tin :

- Mã sinh viên : Chuỗi có đúng 7 ký tự
- Họ và Tên sinh viên : Chuỗi có không quá 25 ký tự
- Năm sinh : số nguyên dương gồm 4 ký số
- Lớp : Chuỗi có không quá 10 ký tự
- Điểm Trung Bình: Số thực trong khoảng [0,..,10] có 1 số lẻ
- Xếp loại học lực : Chuỗi có không quá 5 ký tự

Viết chương trình tùy chọn menu thực hiện các chức năng :

0. Thoát khỏi chương trình
1. Tạo bảng điểm sinh viên
2. Xem bảng điểm sinh viên.
3. Xem bảng điểm đầy đủ - có kết quả XLHT

4. Xuất bảng điểm sinh viên theo từng lớp
5. Sắp bảng điểm sinh viên giảm dần theo điểm trung bình
6. Sắp xếp và xuất bảng điểm tăng dần theo tên, nếu trùng tên, sắp tăng theo mã sinh viên.
7. Tìm và xuất thông tin của sinh viên có tên cho trước.
8. Tìm và xuất thông tin của các sinh viên có điểm trung bình cao nhất.
9. Xếp loại học lực của sinh viên dựa vào điểm trung bình. Biết rằng:
 - o Giỏi: $8.5 \leq \text{Điểm trung bình} \leq 10$
 - o Khá: $7.0 \leq \text{Điểm trung bình} < 8.5$
 - o Trung bình: $5.5 \leq \text{Điểm trung bình} < 7.0$
 - o Yếu: $4.0 \leq \text{Điểm trung bình} < 5.5$
 - o Kém: $0.0 \leq \text{Điểm trung bình} < 4.0$

Bộ dữ liệu sử dụng trong chương trình:

:Ma SV	:Ho va Ten	:Nam sinh	:Lop	:Diem TB	:XLoai
:1534507	:Duong Qua	:1997	:CTK39	:9.0	:
:1405507	:Tu Ma Ngoc Yen	:1996	:CTK38	:6.5	:
:1334067	:Quach Tinh	:1996	:CTK37	:8.0	:
:1300167	:Hoang Dung	:1995	:CTK37	:7.2	:
:1534507	:Lam Xung	:1997	:CTK39	:4.0	:
:					
:1405507	:Hoang Dung	:1996	:CTK38	:8.5	:
:1334067	:Vuong Ngoc Yen	:1995	:CTK37	:7.0	:
:1334167	:Lenh Ho Xung	:1995	:CTK37	:5.6	:
:1530070	:Trieu Minh	:1997	:CTK39	:3.0	:
:1534123	:Nhac Dung	:1997	:CTK39	:2.0	:
:					
:1534456	:Ha Thai Xung	:1997	:CTK39	:6.0	:
:1512307	:Uy Tieu Bao	:1997	:CTK39	:5.0	:
:1400067	:Chau Ba Thong	:1996	:CTK38	:6.5	:
:1530000	:Cong Tang Ton Nu Thu Huong	:1997	:CTK39	:8.0	:
:1401234	:Ton That Du	:1996	:CTK38	:9.5	:
:					
:1434167	:Doan DU	:1996	:CTK38	:5.6	:
:1333217	:Hoang Trong Yen	:1995	:CTK37	:7.0	:
:1531117	:Au Duong Phong	:1997	:CTK39	:5.5	:

2. Quản lý thuê bao điện thoại

Viết chương trình quản lý các khách hàng thuê bao điện thoại. Trong đó, với mỗi thuê bao, cần quản lý những thông tin sau:

- Mã số thuê bao: là một số nguyên dương.
- Họ và tên: là một chuỗi chứa tối đa 50 ký tự.
- Địa chỉ: là một chuỗi chứa tối đa 100 ký tự.
- Số điện thoại: là một chuỗi chứa 10 hoặc 11 chữ số và một dấu chấm(.) để phân cách mã vùng với số điện thoại. Ký tự đầu tiên là ký tự 0.
- Ngày hợp đồng: cho biết ngày, tháng, năm hợp đồng thuê bao.

Chương trình cho phép người dùng thực hiện những chức năng dưới đây:

0. Thoát khỏi chương trình
1. Nhập danh sách thuê bao
2. Xem danh sách thuê bao
3. Tìm số điện thoại khi biết tên
4. Tìm thông tin thuê bao khi biết số điện thoại

5. Xuất các thuê bao có cùng địa chỉ
6. Sắp xếp các thuê bao tăng dần theo mã vùng.

3. Vector trong không gian 3 chiều

Viết chương trình thực hiện các phép toán trên vector trong không gian 3 chiều theo các yêu cầu dưới đây. Giả sử cho 2 vector có tọa độ như sau: $\vec{u} = (x_u, y_u, z_u)$ và $\vec{v} = (x_v, y_v, z_v)$

1. Tính độ dài vector \vec{u} theo công thức $|\vec{u}| = \sqrt{x_u^2 + y_u^2 + z_u^2}$.
2. Kiểm tra hai vector \vec{u} và \vec{v} có bằng nhau không.
3. Tính tổng của hai vector \vec{u} và \vec{v} theo công thức $\vec{u} + \vec{v} = (x_u + x_v, y_u + y_v, z_u + z_v)$
4. Tính hiệu của hai vector \vec{u} và \vec{v} theo công thức $\vec{u} - \vec{v} = (x_u - x_v, y_u - y_v, z_u - z_v)$
5. Tính tích của một số thực với một vector theo công thức $k \cdot \vec{u} = (k \cdot x_u, k \cdot y_u, k \cdot z_u)$
6. Tính tích vô hướng của hai vector \vec{u} và \vec{v} theo công thức $\vec{u} \cdot \vec{v} = x_u x_v + y_u y_v + z_u z_v$
7. Kiểm tra hai vector \vec{u} và \vec{v} có vuông góc với nhau hay không? Biết rằng, hai vector \vec{u} và \vec{v} vuông góc nếu tích vô hướng của chúng bằng 0.

F. Bài tập làm thêm

1. Số phức

Số phức là số có dạng $a + bi$, trong đó a và b là các số thực, i là đơn vị ảo với $i^2 = -1$. Trong biểu thức này, a gọi là phần thực và b gọi là phần ảo của số phức. Số phức có thể được biểu diễn trên mặt phẳng phức với trục hoành là trục thực và trục tung là trục ảo. Do đó, một số phức $a + bi$ có thể được xác định bằng một điểm có tọa độ (a, b) . Một số phức nếu có phần thực bằng 0 thì được gọi là **số thuần ảo**, nếu phần ảo bằng 0 thì trở thành số thực.

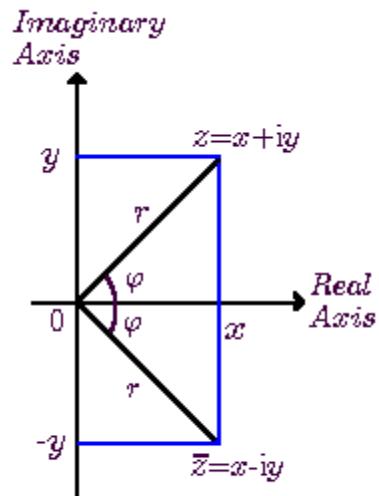
Hãy viết chương trình thực hiện các chức năng sau trên số phức:

- Tạo số phức bằng cách nhập phần tự và phần ảo từ bàn phím.
- Xuất số phức ra màn hình.
- Kiểm tra hai số phức có bằng nhau hay không.
- Tính độ lớn (magnitude) hay mô-đun của số phức $z = a + bi$ theo công thức $|z| = \sqrt{a^2 + b^2}$.
- Tìm số phức liên hợp của số phức $a + bi$.
- Tìm số đối của số phức $a + bi$.
- Tìm số nghịch đảo của số phức (khác 0) $a + bi$ theo công thức: $\frac{a}{a^2+b^2} - \frac{b}{a^2+b^2}i$
- Kiểm tra một số phức $a + bi$ có phai là số thuần ảo hay không.
- Cộng hai số phức theo công thức: $(a+bi) + (c+di) = (a+c) + (b+d)i$.
- Trừ số phức cho một số phức theo công thức: $(a+bi) - (c+di) = (a-c) + (b-d)i$.
- Nhân một số thực với một số phức.
- Nhân hai số phức theo công thức: $(a+bi) * (c+di) = (ac-bd) + (ad+bc)i$.
- Chia một số phức cho một số phức:

$$\frac{a+bi}{c+di} = \frac{(a+bi)(c-di)}{(c+di)(c-di)} = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}i$$

- Tìm Argumen của số phức $z = a + bi$. Biết Argumen(z) chính là góc hợp bởi chiều dương của trục Ox và vector \overrightarrow{OM} với M(a,b) là tọa độ của z khi biểu diễn trên mặt phẳng tọa độ.
- Xuất số phức $z = a + bi$ ở dạng lượng giác theo công thức:

$$z = r(\cos\varphi + i \sin\varphi) \text{ với } r = |z|, \varphi = \text{Argumen}(z)$$



2. Quản lý sách

Viết chương trình quản lý một thư viện sách (gọi chung là tài liệu) đáp ứng các chức năng sau:

- Nhập một danh sách tài liệu.
- Xuất danh sách tài liệu ra màn hình.
- Tính tổng giá tất cả các tài liệu
- Tìm danh mục sách được xuất bản bởi **nhaXb** vào **namXb** cho trước.
- Tìm những bài báo khoa học có sự tham gia của tác giả **tacGia** cho trước.
- Thống kê số lượng tài liệu theo mỗi loại.
- Xem thông tin những tài liệu có giá đắt nhất.
- Liệt kê các tài liệu theo từng năm xuất bản.
- Tìm tài liệu có nhiều tác giả nhất.
- Xem thông tin tài liệu theo mã số tài liệu (**maTL**) cho trước.
- Sắp xếp các tài liệu tăng dần theo tựa đề.

Biết rằng, với mỗi tài liệu, cần quản lý các thông tin sau:

- Mã tài liệu: là một chuỗi chứa tối đa 10 ký tự.
- Tựa đề
- Loại tài liệu: thuộc một trong các loại sau: Sách, Báo khoa học, Tạp chí, Luận văn.
- Năm xuất bản
- Tác giả: lưu danh sách tác giả của tài liệu (phân tách nhau bởi dấu phẩy), trường này không dùng cho Tạp chí.
- Nhà xuất bản: không dùng cho Luận văn.
- Giá (mua về hoặc đèn bù nếu mất): tính theo VNĐ.

3. Cửa hàng bách hóa

Viết chương trình quản lý các mặt hàng trong một cửa hàng bách hóa theo các yêu cầu sau:

- Nhập một danh sách mặt hàng
- Xuất các mặt hàng ra màn hình theo thứ tự ABC của tên mặt hàng.
- Tính tổng giá trị của tất cả các mặt hàng.
- Cho biết các mặt hàng có giá trị lớn nhất trong cửa hàng.
- Xuất ra những mặt hàng đã hết hạn sử dụng.
- Sắp xếp và xuất các mặt hàng theo từng loại.
- Tìm những mặt hàng còn hạn sử dụng không quá 30 ngày.
- Liệt kê những mặt hàng đã bán hết (số lượng tồn bằng 0).
- Tính số lượng mặt hàng bánh và kẹo.
- Tìm mặt hàng bia có giá rẻ nhất.

Với mỗi mặt hàng, cần quản lý các thông tin sau:

- Tên mặt hàng
- Đơn giá (tính theo VNĐ)
- Số lượng
- Đơn vị tính (gói, hộp, thùng, kg, ...)
- Hạn sử dụng (kiểu ngày tháng)
- Loại mặt hàng (bánh, kẹo, đồ uống, sữa, bia, ...)

LAB 9. BIẾN ĐỘNG VÀ KIỂU CON TRỎ

THỜI LƯỢNG: 6 TIẾT

A. Mục tiêu

- Giúp sinh viên hiểu rõ và thực hiện thuần thục các kỹ thuật xử lý trên con trỏ.
- Sau khi hoàn thành bài thực hành này, sinh viên cần:
 - Nắm vững các khái niệm và cách định nghĩa kiểu con trỏ, khai báo biến con trỏ.
 - Nắm vững các kỹ thuật xử lý cơ bản trên kiểu con trỏ.
 - Hiểu rõ cách cấp phát bộ nhớ, truy cập phần tử và thu hồi vùng nhớ cấp phát cho các cấu trúc dữ liệu động : mảng động 1 chiều, mảng động 2 chiều, xâu ký tự động, cấu trúc động, ...
 - Cài đặt được các thao tác cơ bản trên các cấu trúc dữ liệu động. bằng con trỏ.
 - Hiểu rõ cơ chế gọi hàm, truyền tham trị và truyền tham biến. Phân biệt được các đối vào, đối ra của hàm.

B. Yêu cầu

- Trong phần C (hướng dẫn thực hành), sinh viên tự đọc.
- Sinh viên cần chuẩn bị bài trước để thực hành đủ khối lượng yêu cầu.

Buổi thực tập thứ 12 (4 tiết) :

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần D (Hướng dẫn thực hành) của lab 9.
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab09_HD chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Sử dụng lại tập tin word LoiChuongTrinh.docx từ lab 8 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - ✓ Thu bài : Xem thông báo của giáo viên .

Buổi thực tập thứ 13 (4 tiết) :

* 2 tiết đầu:

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần E (Bắt buộc) của lab 9.
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab09_BB chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Sử dụng lại tập tin word LoiChuongTrinh.docx trong lab5 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - ✓ Giáo viên sẽ thu bài qua hệ thống thu bài trực tuyến tại phòng lab (thu thư mục MaSV_Lab09_BB) vào trước giờ giải lao.

C. Ôn tập lý thuyết

1. Kiểu con trỏ và biến con trỏ

Cú pháp khai báo biến con trỏ: **KDL** * **Tên_bien_con_trо;**

Ví dụ: **int** *px; **double** *pd; **NhanVien** *nv;

Cú pháp định nghĩa kiểu con trỏ: **typedef KDL * Tên_kiểu_con_trо**

Ví dụ: **typedef int** *IntPtr;
typedef Node *NodePointer;

2. Các phép toán trên con trỏ

- Lấy nội dung tại địa chỉ mà con trỏ px trỏ tới:
- Phép gán địa chỉ cho con trỏ cùng kiểu:
- Phép gán hai con trỏ cùng kiểu:
- Mọi con trỏ đều có thể nhận giá trị NULL:
- Cấp phát vùng nhớ cho biến con trỏ:
- Thu hồi vùng nhớ cho biến con trỏ:

*** px**
int *px, x = 3; **px** = &x;
int *py; **py** = px;
py = **NULL**;
px = **new KDL**;
delete px;

3. Mảng động một chiều

Cấp phát động cho mảng một chiều thông qua con trỏ (hay dùng con trỏ để cài đặt mảng 1 chiều).

- Khai báo
- Cấp phát vùng nhớ
- Thu hồi (giải phóng) vùng nhớ
- Truy cập đến phần tử trong mảng

KDL ***tên_bien_mảng**;
tên_bien_mảng = **new KDL [Kích_thước_mảng]**;
delete []**tên_bien_mảng**;

Truy cập đến giá trị	a[i]	*(a+i)
Truy cập đến địa chỉ	&a[i]	a+i

4. Mảng động hai chiều (ma trận)

Cấp phát động cho mảng hai chiều thông qua con trỏ

- Khai báo
- Cấp phát vùng nhớ
- Thu hồi (giải phóng) vùng nhớ
- Truy cập đến phần tử ở hàng i, cột j trong mảng theo chỉ số (ví dụ: a[i][j]) hoặc sử dụng con trỏ *(a + i*n + j).

KDL ***tên_bien_mảng**;
tên_bien_mảng = **new KDL [Số_dòng * Số_cột]**;
delete []**tên_bien_mảng**;

5. Chuỗi ký tự động

Cấp phát động cho chuỗi ký tự thông qua con trỏ

- Khai báo
- Khởi tạo
- Cấp phát vùng nhớ
- Thu hồi (giải phóng) vùng nhớ
- Truy cập đến từng ký tự (phần tử) giống như mảng động một chiều.

char ***tên_bien_chuỗi**;
tên_bien_mảng = **NULL**;
tên_bien_mảng = **new char [Số_ký_tự_tối_đa]**;
delete []**tên_bien_mảng**;

6. Con trỏ cấu trúc

Giả sử ta có một kiểu dữ liệu cấu trúc có tên là KCT.

- Khai báo

KCT ***tên_bien**;

- Cấp phát vùng nhớ tên biến = **new KCT**;
- Thu hồi (giải phóng) vùng nhớ **delete** tên biến;
- Truy cập đến các thành phần (trường) của cấu trúc: tên biến **->** Tên_thành_phần

Ví dụ:

```
PhanSo *p;
p = new PhanSo;
p->TuSo = 2;
```

```
p->MauSo = 5;
cout << p->TuSo << '/' << p->MauSo;
delete p;
```

7. Mảng động cấu trúc

- Khai báo
- Cấp phát vùng nhớ
- Thu hồi (giải phóng) vùng nhớ

KCT *tên biến;
tên biến = **new KCT [Kích_thước]**;
delete [] tên biến;

Chú ý:

- Theo cách khai báo này, mỗi phần tử của mảng là một cấu trúc, không phải con trỏ cấu trúc.

8. Truyền tham số

a. Một số điểm cần lưu ý

- Muốn hàm trả về một giá trị là mảng hoặc xâu thì khai báo kiểu trả về là kiểu con trỏ.
- Muốn tham số thực giữ lại giá trị đã bị thay đổi (bên trong hàm) khi chương trình ra khỏi hàm thì phải dùng cách truyền tham biến. Nghĩa là đối số phải là con trỏ hoặc tham chiếu.

b. Cách truyền tham số

Tham số hình thức	Tham số thực (Đối số)	Ghi chú
Biến	Giá trị	Truyền tham trị
Con trỏ	Con trỏ Địa chỉ của biến Tên mảng một chiều Tên mảng hai chiều (phải ép kiểu) Tên xâu ký tự	Truyền tham biến hay truyền bằng biến
Tham chiếu	Giá trị	
Tên mảng	Tên mảng	
Tên xâu ký tự	Tên xâu ký tự	

D. Hướng dẫn thực hành

Bài 1. Mảng động 1 chiều

Cho mảng số nguyên a có n phần tử. Viết chương trình xuất ra các giá trị phân biệt trong a và số lần xuất hiện của mỗi phần tử này.

Yêu cầu:

- Cài đặt mảng động
- Tạo project rỗng, đặt tên **MaSV_Lab09_D_Bai1**.
- Chương trình tổ chức theo thư viện hàm, không có hệ thống menu.
- Tham khảo hàm chức năng tương ứng trong tập tin **thuvien.h** sau đây :

//Xuat ca gia tri phan biet trong mang va so lan xuat hien cua moi phan tu

void XuatGiaTri_SoLan_PhanBiet(int *a, int n)

{

 int i;//duyet a

```

dau; //danh dau a[i] da co trong b chua
int *b, //luu cac gia tri phan biet
    j;//duyet b
*c, //luu so lan xuat hien cua cac b[j]
    m;// kich thuoc cua b, c
m = 0;
b = new int[n];

for (i = 0; i < n; i++)
{
    dau = 1;
    for (j = 0; j < m && dau; j++)
        dau = dau && (*a + i) != *(b + j);
    if (dau)
    {
        *(b + m) = *(a + i);
        m++;
    }
}
c = new int[n];
int k;
for (k = 0; k < m; k++)
    *(c + k) = 0;

for (j = 0; j < m; j++)
{
    for (i = 0; i < n; i++)
        if (a[i] == *(b+j))
        {
            *(c+j) = *(c + j) + 1;
        }
}
cout << "\nCac gia tri phan biet trong a va so lan xuat hien cac gia tri nay:\n";
for (k = 0; k < m; k++)
{
    cout << "\nGia tri " << *(b+k) << " xuat hien " << *(c+k) << " lan.";
}

delete[]b;
delete[]c;
}

//=====

```

Bài 2. Mảng động 2 chiều

Viết chương trình tùy chọn thực hiện các phép toán trên ma trận vuông các số nguyên :

0. Thoát khỏi chương trình
1. Nhập tự động ma trận
2. Xem mảng
3. Cộng ma trận

4. Trừ ma trận
5. Nhân ma trận

Yêu cầu:

- Cài đặt mảng động

Tạo project rỗng, đặt tên ***MaSV_Lab09_D_Bai2***.

Chương trình tổ chức theo thư viện hàm và có hệ thống menu, phát triển như bài 1 mục D lab 8.

Tham khảo tập tin ***thuvien.h*** sau đây :

```
//Dinh nghia hang
//Dinh nghia kieu du lieu moi
typedef int *MaTranVuong;

//Khai bao nguyen mau cac ham
void NhapMaTran(MaTranVuong &a, int n);
void XuatMaTran(MaTranVuong a, int n);
/*
void TinhTong_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n);
void TinhHieu_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n);
void TinhTich_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n);
*/
//=====
//Dinh nghia cac ham
void NhapMaTran(MaTranVuong &a, int n, char kt)
{
    int i, j;
    for (i = 0; i < n; i++) //hang i
        for (j = 0; j < n; j++) //cot j
    {
        cout << endl << kt << "[" << i << "]" << j << "=";
        cin >> *(a + i*n + j);
    }
}
void XuatMaTran(MaTranVuong a, int n)
{
    int i, j;
    for (i = 0; i < n; i++)
    {
        cout << endl << endl;
        for (j = 0; j < n; j++)
            cout << setw(4) << *(a + i*n + j);
    }
}

MaTranVuong TinhTong_2_MaTran(MaTranVuong a, MaTranVuong b, int n)
{
    int i, j;
    MaTranVuong c;
    c = new int[n*n];
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            *(c + i*n + j) = *(a + i*n + j) + *(b + i*n + j);
    return c;
}
```

MaTranVuong TinhHieu_2_MaTran(**MaTranVuong** a, **MaTranVuong** b, **int** n)

```
{
    MaTranVuong c;
    int i, j;
    c = new int[n*n];
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            *(c + i*n + j) = *(a + i*n + j) - *(b + i*n + j);

    return c;
}
```

MaTranVuong TinhTich_2_MaTran(**MaTranVuong** a, **MaTranVuong** b, **int** n)

```
{
    MaTranVuong c;
    int i, j, k;
    c = new int[n*n];
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
        {
            *(c + i*n + j) = 0;
            for (k = 0; k < n; k++)
                *(c + i*n + j) += *(a + i*n + k) * *(b + k*n + j);
        }
    return c;
}

//=====
```

Bài 3. Xâu ký tự động

Viết chương trình tùy chọn thực hiện các chức năng sau đây trên chuỗi động :

0. Thoát khỏi chương trình
1. Nhập chuỗi
2. Xem chuỗi
3. Tính chiều dài chuỗi
4. Chèn ký tự x vào chuỗi tại vị trí cho trước
5. Xóa ký tự tại vị trí cho trước
6. Cắt ký tự cuối chèn vào vị trí đầu
7. Xóa tất cả các ký tự x

Yêu cầu:

- Cài đặt chuỗi động

Tạo project rỗng, đặt tên **MaSV_Lab09_D_Bai3**.

Chương trình tổ chức theo thư viện hàm và có hệ thống menu, phát triển như bài 1 mục D lab 8.

Tham khảo tập tin **thuvien.h** sau đây :

```
#define MAX 100
//Dinh nghia kieu du lieu moi
typedef char *ChuoiDong;

//Khai bao nguyen mau
```

```
//Dinh nghia cac ham xu ly
//Tinh chieu dai chuoi
int TinhChieuDaiChuoi(ChuoiDong a)
{
    int i = 0;
    while (*(a+i) !=NULL)
        i++;
    return i;
}

//Chen ky tu x vao chuoi a tai vi tri vt
//Input : a,x,vt
//output : 1; thanh cong; 0 : khong thanh cong
int ChenKT_VT(ChuoiDong a, char x, int vt)
{
    int i, h, kq;
    h = TinhChieuDaiChuoi(a);
    if (vt < 0 || vt > h)
        kq = 0;
    else
    {
        for (i = h; i >= vt; i--)
            *(a + i + 1) = *(a + i);
        *(a + vt) = x;
        kq = 1;
    }
    return kq;
}

//Xoa ky tu tai vi tri vt cua chuoi a
//Input : a,vt
//output : 1; thanh cong; 0 : khong thanh cong
int XoaKT_VT(ChuoiDong a, int vt)
{
    int i, h, kq;
    h = TinhChieuDaiChuoi(a);
    if (vt < 0 || vt > h - 1)
    {
        kq = 0;
    }
    else
    {
        for (i = vt; i < h; i++)
            *(a + i) = *(a + i + 1);
        kq = 1;
    }
    return kq;
}

//Cat ky tu cuoi chuoi roi chen vao tai vi tri dau chuoi
void CatCuoiChenDau(ChuoiDong a)
{
    int i, h;
    char x;
```

```
h = TinhChieuDaiChuoi(a);
x = *(a + h - 1);
for (i = h - 2; i >= 0; i--)
    *(a + i + 1) = *(a + i);
*(a + 0) = x;
}

//Xoa tat ca ky tu x trong chuoi
void Xoa_x(ChuoiDong a, char x)
{
    int i, h = 0;
    for (i = 0; a[i] != NULL; i++)
        if (*(a + i) != x)
    {
        *(a + h) = *(a + i);
        h++;
    }
    *(a + h) = NULL;
}

//=====
```

Bài 4. Cấu trúc động

Bảng điểm môn học của sinh viên chứa các thông tin sau :

- Mã sinh viên : chuỗi có đúng 7 ký tự
- Học và chữ lót của sinh viên : chuỗi có không quá 14 ký tự
- Tên sinh viên : chuỗi có không quá 7 ký tự
- Giới tính : chuỗi từ 2 đến 3 ký tự.
- Năm sinh : số nguyên dương 4 ký số
- Quê quán : chuỗi có không quá 14 ký tự
- Lớp : chuỗi có 5 ký tự
- Điểm : số thực từ 0 đến 10

Viết chương trình tùy chọn trên danh sách sinh viên với các chức năng :

0. **Thoát khỏi chương trình**
1. **Tạo danh sách sinh viên**
2. **Xem danh sách sinh viên**
3. **Xuất danh sách sinh viên giảm dần theo điểm**
4. **Xem danh sách sinh viên theo lớp**
5. **Xuất danh sách sinh viên theo lớp và giảm dần theo điểm**
6. **Thông kê chất lượng học tập sinh viên theo lớp (Giỏi,Khá,TB,Yếu, Kém = ?)**

Tiêu chuẩn xếp loại học tập của sinh viên (dựa vào điểm) như sau :

- **Giỏi** : Điểm ≥ 8.5
- **Khá** : $7 \leq \text{Điểm} < 8.5$
- **TB** : $5.5 \leq \text{Điểm} < 7$
- **Yếu** : $4 \leq \text{Điểm} < 5.5$
- **Kém** : Điểm < 4

Yêu cầu:

- Cài đặt mảng cấu trúc động
- Sử dụng bộ dữ liệu cho sẵn kèm theo sau đây:

:Ma	:NU	:Ho	Ten	:GT	:NS	:Que quan	:Lop	:Diem :
:1512967	:Trieu		Minh	:Nu	:1997	:Ninh Thuan	:CTK39	:5.5 :
:1410279	:Hoang	Duoc	Su	:Nam	:1995	:Da Lat	:CTK38	:6.0 :
:1512555	:Au	Duong	Phong	:Nam	:1996	:Khanh Hoa	:CTK39	:7.5 :
:1412120	:Vu	Thi	Yen	:Nu	:1996	:Binh Dinh	:CTK38	:3.0 :
:1313320	:Le	Ngoc	Minh	:Nam	:1995	:Can Tho	:CTK37	:2.5 :
:								
:1510214	:Dinh	Thi	Yen	:Nu	:1997	:Da Lat	:CTK39	:9.0 :
:1512887	:Vuong	Ngoc	Yen	:Nu	:1997	:Da Nang	:CTK39	:6.0 :
:1414245	:Vuong	Trung	Duong	:Nam	:1996	:Phu Yen	:CTK38	:7.0 :
:1510192	:Nhac	Linh	San	:Nu	:1997	:Da Lat	:CTK39	:3.2 :
:1312890	:Hoang		Dung	:Nu	:1995	:Da Nang	:CTK37	:8.0 :
:								
:1510192	:Cao	Vien	Vien	:Nu	:1997	:Da Lat	:CTK39	:9.6 :
:1444405	:Truong	Uo	Ky	:Nam	:1996	:Binh Dinh	:CTK38	:7.0 :
:1412988	:Ui	Tieu	Bao	:Nam	:1996	:Da Nang	:CTK38	:8.5 :
:1312990	:Duong		Qua	:Nam	:1995	:Da Lat	:CTK37	:3.5 :
:1333993	:Chau	Ba	Thong	:Nam	:1994	:Quang Ngai	:CTK37	:9.1 :
:								
:1512128	:Ta	Van	Ton	:Nam	:1997	:Binh Dinh	:CTK39	:9.3 :
:1400128	:Vien	Thua	Chi	:Nam	:1997	:Binh Thuan	:CTK38	:5.3 :
:1512868	:Doan		Du	:Nam	:1996	:Da Lat	:CTK39	:8.5 :
:								

Bước 1. Tạo 1 dự án Win32 Console Application mới. Đặt tên là **Lab09_D_Bai4**

Bước 2. Tạo cấu trúc cho chương trình như bài 1 mục D lab 8 (phần lõi tối thiểu chạy được của chương trình)
Tức là ta có kết quả chương trình ở bước này như sau :

- Tập tin **thuvien.h** : Rỗng
- Tập tin **menu.h** : Rỗng
- Tập tin **program.cpp** có nội dung như sau :

```
#include <iostream>
#include <conio.h>
using namespace std;

#include "thuvien.h"
#include "menu.h"

void ChayChuongTrinh();

int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    _getch();
}
```

Nhấn Ctrl + F5 để chạy chương trình, sửa lỗi nếu có.

Bước 3:

Bước này ta định nghĩa kiểu dữ liệu mới; tổ chức và vận hành hệ thống menu.

- **Trong tập tin *thuvien.h* :**

Ta bổ sung định nghĩa hằng biểu thị số lượng tối đa sinh viên trong danh sách sinh viên, kiểu dữ liệu cấu trúc *SinhVien*

//**Dịnh nghĩa hang**

//**Dịnh nghĩa hang**

#define MAX 100 //So luong toi da sinh vien trong danh sach

#define NGANGDOI '=' //dau bang : gach 2 hang

#define NGANGDON '-' //dau tru : gach 1 hang

//**Dịnh nghĩa kieu du lieu moi : kieu cau truc Sinh vien**

//**Kieu Sinh Vien**

struct **SinhVien**

{

char maSV[8];

char hoLot[14];

char ten[7];

char gioiTinh[4];

unsigned int namSinh;

char queQuan[14];

char lop[7];

double diem;

};

//**Khai báo nguyên mẫu các hàm xử lý, nhập xuất**

//bổ sung sau

//**Dịnh nghĩa các hàm xử lý, nhập xuất**

//bổ sung sau

- **Trong tập tin *menu.h* :**

//**Khai báo nguyên mẫu các hàm xử lý menu**

//bổ sung sau

//**Định nghĩa các hàm xử lý menu**

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình

void XuatMenu()

{

 cout << "\n=====He thong chuc nang=====";

 cout << "\n0. Thoat khoi chuong trinh";

 cout << "\n1. Tao danh sach sinh vien";

 cout << "\n2. xem danh sach sinh vien";

 cout << "\n3. Sap danh sach sinh vien giam dan theo diem";

 cout << "\n4. xem danh sach sinh vien theo lop";

 cout << "\n5. Xuat danh sach sinh vien theo lop va co diem giam dan";

 cout << "\n6. Thong ke chat luong sinh vien theo lop";

 cout << "\n=====";

}

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách

```
// Input : soMenu = Số lượng menu có thể chọn.  
// Output: Số thứ tự menu do người dùng nhập vào.  
int ChonMenu(int soMenu)  
{  
    int stt;  
    for (;;) {  
        system("CLS");  
        XuatMenu();  
        cout << "\nNhap 1 so khong khoang [0,..." << soMenu << "] de chon chuc nang, stt = ";  
        cin >> stt;  
        if (0 <= stt && stt <= soMenu)  
            break;  
    }  
    return stt;  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Định nghĩa hàm xử lý menu :

Các thao tác thực hiện trên cùng một đầu vào là mảng 1 chiều kiểu NhanVien,kích thước mảng n, nên ta bổ sung thêm biến mảng 1 chiều NhanVien a, số nguyên dương n làm đối của hàm **XuLyMenu**, ngoài tham số đã có là tham số menu. Ở đây ta dùng biến động mảng 1 chiều kiểu NhanVien và cài đặt bằng con trỏ .

```
void XuLyMenu(int menu, SinhVien *a, int &n)  
{  
    //khai bao bien  
    switch (menu) {  
        case 0:  
            system("CLS");  
            cout << "\n0. Thoat khoi chuong trinh\n";  
            break;  
        case 1:  
            system("CLS");  
            cout << "\n1. Tao danh sach nhan vien";  
            break;  
        case 2:  
            system("CLS");  
            cout << "\n2. Xem danh sach nhan vien";  
            break;  
        case 3:  
            system("CLS");  
            cout << "\n3. Sap danh sach sinh vien giam dan theo diem";  
            break;  
        case 4:  
            system("CLS");  
            cout << "\n4. xem danh sach sinh vien theo lop";  
            break;  
        case 5:
```

```
        system("CLS");
        cout << "\n5. Xuat danh sach sinh vien theo lop va co diem giam dan";
        break;
    case 6:
        system("CLS");
        cout << "\n6. Thong ke chat luong sinh vien theo lop";
        break;
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```
void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, SinhVien * a, int &n);
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

- Trong tập tin **program.cpp** :
 - + Hàm **ChayChuongTrinh** ta cập nhật lại như sau :

```
void ChayChuongTrinh()
{
    int menu,
        soMenu = 6,
        n = 0;
    SinhVien *a;
    a = new SinhVien[MAX];
    do
    {
        system("CLS");
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, a, n);
    }
    while(menu > 0);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra sự vận hành của hệ thống menu.

Kiểm tra chức năng 0 (thoát khỏi chương trình).

Bước 4 : Bổ sung vào chương trình các thao tác nhập xuất dữ liệu

Trong bước 4 này, ta làm công việc sau :

- Trong tập tin **program.cpp** , bổ sung các thư viện cần thiết
- Trong tập tin **thuvien.h** , soạn thảo các hàm tạo và xuất danh sách sinh viên
- Trong tập tin **menu.h** bổ sung xử lý chức năng tạo, xuất dữ liệu trong hàm **XuLyMenu**.

- Trong tập tin **program.cpp** :

Bổ sung thư viện **<string.h>**, **<iomanip>**

- Trong tập tin **thuvien.h** :

Bổ sung các hàm tạo danh sách sinh viên và xuất danh sách sinh viên ra màn hình

4.1 Các hàm tạo dữ liệu

Ta dùng cách sau để tạo danh sách sinh viên theo bộ dữ liệu cho trước và tránh việc nhập từ bàn phím để khỏi mất thời gian.

```
=====  
==//  
/////////// Tao Danh sach sinh vien ///////////  
=====  
==//  
//Chen 1 sinh vien  
void Chen_SV(char *maSV, char *hoLot, char *ten, char *gioiTinh, unsigned int namSinh,  
            char *queQuan, char *lop, double diem, SinhVien *a, int &n)  
{  
    if (n < MAX)  
    {  
        strcpy_s((a + n)->maSV, 8, maSV);  
        strcpy_s((a + n)->hoLot, 14, hoLot);  
        strcpy_s((a + n)->ten, 7, ten);  
        strcpy_s((a + n)->gioiTinh, 4, gioiTinh);  
        (a+n)->namSinh = namSinh;  
        strcpy_s((a + n)->queQuan, 15, queQuan);  
        strcpy_s((a + n)->lop, 6, lop);  
        (a + n)->diem = diem;  
        n++;  
    }  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Tao Danh Sach sinh vien  
void TaoDanhSachSinhVien(SinhVien *a, int &n)  
{  
    Chen_SV("1512967", "Trieu", "Minh", "Nu", 1997, "Ninh Thuan", "CTK39",5.5, a, n);  
    Chen_SV("1410279", "Hoang Duoc", "Su", "Nam", 1995, "Da Lat", "CTK38",6, a, n);  
    Chen_SV("1512555", "Au Duong", "Phong", "Nam", 1996, "Khanh Hoa", "CTK39",7.5, a, n);  
    Chen_SV("1412120", "Vo Thi", "Yen", "Nu", 1996, "Binh Dinh", "CTK38", 3, a, n);  
    Chen_SV("1313320", "Le Ngoc", "Minh", "Nam", 1995, "Can Tho", "CTK37", 2.5,a, n);  
    Chen_SV("1510214", "Dinh Thi", "Yen", "Nu", 1997, "Da Lat", "CTK39",9, a, n);  
    Chen_SV("1512887", "Vuong Ngoc", "Yen", "Nu", 1997, "Da Nang", "CTK39",6, a, n);  
    Chen_SV("1414245", "Vuong Trung", "Duong", "Nam", 1996, "Phu Yen", "CTK38",7, a, n);  
    Chen_SV("1510192", "Nhac Linh", "San", "Nu", 1997, "Da Lat", "CTK39", 3.2,a, n);  
    Chen_SV("1312890", "Hoang", "Dung", "Nu", 1995, "Da Nang", "CTK37", 8, a, n);  
    Chen_SV("1510192", "Cao Vien", "Vien", "Nu", 1997, "Da Lat", "CTK39", 9.6, a, n);  
    Chen_SV("1444405", "Truong Vo", "Ky", "Nam", 1996, "Binh Dinh", "CTK38",7, a, n);  
    Chen_SV("1412988", "Vi Tieu", "Bao", "Nam", 1996, "Da Nang", "CTK38", 8.5,a, n);  
    Chen_SV("1312990", "Duong", "Qua", "Nam", 1995, "Da Lat", "CTK37", 3.5, a, n);  
    Chen_SV("1333993", "Chau Ba", "Thong", "Nam", 1994, "Quang Ngai", "CTK37", 9.1, a, n);  
    Chen_SV("1512128", "Ta Van", "Ton", "Nam", 1997, "Binh Dinh", "CTK39", 9.3, a, n);  
    Chen_SV("1400128", "Vien Thua", "Chi", "Nam", 1997, "Binh Thuan", "CTK38", 5.3, a, n);  
    Chen_SV("1512868", "Doan", "Du", "Nam", 1996, "Da Lat", "CTK39", 8.5, a, n);  
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.2 Các hàm xuất danh sách sinh viên ra màn hình

```
//=====
==//
/////////////////////// Xuat Danh sach sinh vien ///////////////////
//=====
==/
```

```
//Xuat ke ngang doi
void XuatKeNgangDoi()
{
    int i;
    cout << "\n";
    cout << setiosflags(ios::left)
        << ':';
    for (i = 1; i <= 68; i++)
        cout << NGANGDOI;
    cout << ':';
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuat ke ngang don
void XuatKeNgangDon()
{
    int i;
    cout << "\n";
    cout << setiosflags(ios::left)
        << ':';
    for (i = 1; i <= 68; i++)
        cout << NGANGDON;
    cout << ':';
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuat tieu de
void XuatTieuDe()
{
    XuatKeNgangDoi();
    cout << endl;
    cout << setiosflags(ios::left)
        << ':'
        << setw(7) << "Ma NV"
        << ':'
        << setw(15) << "Ho"
        << setw(7) << "Ten"
        << ':'
        << setw(4) << "GT"
        << ':'
        << setw(4) << "NS"
        << ':'
        << setw(14) << "Que quan"
        << ':'
```

```
<< setw(6) << "Lop"
<< ':'
<< setw(5) << "Diem"
<< ';';

XuatKeNgangDoi();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuat 1 sinh viên
void Xuat_1SV(SinhVien p)
{
    cout << setiosflags(ios::left)
        << ':'
        << setw(7) << p.maSV
        << ':'
        << setw(15) << p.hoLot
        << setw(7) << p.ten
        << ':'
        << setw(4) << p.gioiTinh
        << ':'
        << setw(4) << p.namSinh
        << ':'
        << setw(14) << p.queQuan
        << ':'
        << setw(6) << p.lop
        << ':'
        << setw(5) << setiosflags(ios::fixed) << setprecision(1) << p.diem
        << ';';
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

```
//Xuat DS sinh viên
void Xuat_DSSV(SinhVien *a, int n)
{
    int i;
    XuatTieuDe();
    for (i = 0; i < n; i++)
    {
        cout << endl;
        Xuat_1SV(*(a+i));
        if ((i + 1) % 5 == 0)
            XuatKeNgangDon();
    }
    XuatKeNgangDoi();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.3 Bổ sung nguyên mẫu các hàm :

```
void Chen_SV(char *maSV, char *hoLot, char *ten, char *gioiTinh, unsigned int namSinh,
            char *queQuan, char *lop, double diem, SinhVien *a, int &n);
```

```
void TaoDanhSachSinhVien(SinhVien *a, int &n);
```

```
void XuatKeNgangDoi();
void XuatKeNgangDon();
void XuatTieuDe();
void Xuat_1SV(SinhVien p);
void Xuat_DSSV(SinhVien *a, int n);
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

- Trong tập tin **menu.h** :

Bổ sung xử lý chức năng tạo danh sách sinh viên vào case 1, xem danh sách sinh viên vào case 2 (Các case từ 3 đến 6 giữ nguyên)

```
void XuLyMenu(int menu, SinhVien *a, int &n)
{
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << "\n0. Thoát khỏi chương trình\n";
            delete []a;
            break;
        case 1:
            system("CLS");
            cout << "\n1. Tao danh sach sinh vien";
            TaoDanhSachSinhVien(a, n);
            system("CLS");
            cout << "\n          DANH SACH SINH VIEN :\n";
            Xuat_DSSV(a, n);
            cout << "\nSo sinh vien trong danh sach : n = " << n;
            break;
        case 2:
            system("CLS");
            cout << "\n2. Xem danh sach sinh vien";
            cout << "\n          DANH SACH SINH VIEN :\n";
            Xuat_DSSV(a, n);
            cout << "\nSo sinh vien trong danh sach : n = " << n;
            break;
        //...
    }
    getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện các chức năng 1, 2.

Các bước tiếp theo, bổ sung định nghĩa các hàm chức năng trong tập tin **thuvien.h, bổ sung xử lý chức năng trong hàm **XuLyMenu** trong tập tin **menu.h****

Bước 5 : Bổ sung chức năng 3 – sắp DSSV giảm dần theo điểm – vào chương trình

- Trong tập tin **thuvien.h** , soạn thảo hàm sắp danh sách sinh viên giảm theo điểm

- Trong tập tin **menu.h** bổ sung xử lý chức năng sắp danh sách sinh viên giảm theo điểm trong hàm **XuLyMenu**.

- Trong tập tin *thuvien.h* :

5.1 Hàm sắp danh sách sinh viên giảm theo điểm

```
//Sap DSSV giam theo diem
void Sap_DSSV_GiamDiem(SinhVien *a, int n)
{
    SinhVien t;
    int i, j;
    if (n == 0)
    {
        cout << "\nDS rong!";
        return;
    }
    else
    {
        for (i = 0; i < n - 1; i++)
            for (j = i + 1; j < n; j++)
                if ((a + i)->diem < (a + j)->diem)
                {
                    t = *(a + i);
                    *(a + i) = *(a + j);
                    *(a + j) = t;
                }
    }
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.2 Bổ sung khai báo nguyên mẫu hàm

```
void Sap_DSSV_GiamDiem(SinhVien *a, int n);
```

- Trong tập tin *menu.h* :

Bổ sung xử lý chức năng 3 vào case 3 (Các case khác giữ nguyên)

```
void XuLyMenu(int menu, SinhVien *a, int &n)
{
    switch (menu)
    {
        //...
        case 3:
            system("CLS");
            cout << "\n3. Sap danh sach sinh vien giam dan theo diem";
            cout << "\nDanh sach ban dau :\n";
            Xuat_DSSV(a, n);
            Sap_DSSV_GiamDiem(a, n);
            cout << "\nDanh sach sinh vien giam theo diem :\n";
            Xuat_DSSV(a, n);
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 3.

Bước 6 Bổ sung chức năng 4 – xem DSSV theo lớp – vào chương trình

- Trong tập tin **thuvien.h**, soạn thảo các hàm xem danh sách sinh viên theo lớp
- Trong tập tin **menu.h** bổ sung xử lý chức năng sắp xem danh sách sinh viên theo lớp trong hàm **XuLyMenu**.

- Trong tập tin *thuvien.h* :

6.1 Các hàm xem danh sách sinh viên theo lớp

```
//=====
==/
////////// Danh sach lop /////////////
//=====
==/
//Tao danh sach lop tu danh sach sinh vien : dsLop
void DSSV_Lop(SinhVien *a, int n, char lop[7], SinhVien *dsLop, int &h)
{
    int kq = n;
    int i;
    for (i = 0; i < n; i++)
        if (_strcmpi((a + i)->lop, lop) == 0)
    {
        kq = i;
        break;
    }
    if (kq == n)
        h = 0;
    else
    {
        h = 0;
        for (i = kq; i < n; i++)
            if (_strcmpi((a + i)->lop, lop) == 0)
        {
            *(dsLop + h) = *(a + i);
            h++;
        }
    }
}

//Xuat ds lop
void Xuat_DSSV_Lop(SinhVien *a, int n, char lop[7])
{
    SinhVien *dsLop;
    int i, h;
    dsLop = new SinhVien[MAX];
    DSSV_Lop(a, n, lop, dsLop, h);
    if (h == 0)
        cout << "\nKhong co lop " << lop << " trong DSSV";
    else
    {
```

```

cout << "\n\nDanh sach sinh vien thuoc lop " << lop << " :\n";
XuatTieuDe();
for (i = 0; i < h; i++)
{
    cout << endl;
    Xuat_1SV(*(dsLop + i));
}
XuatKeNgangDoi();
cout << "\nCo " << h << " sinh vien thuoc lop " << lop;
}
delete[]dsLop;
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

6.2 Bổ sung khai báo nguyên mẫu hàm

```

void DSSV_Lop(SinhVien *a, int n, char lop[7], SinhVien *dsLop, int &h);
void Xuat_DSSV_Lop(SinhVien *a, int n, char lop[7]);

```

- Trong tập tin **menu.h** :

Bổ sung xử lý chức năng 4 vào case 4 (Các case khác giữ nguyên)

```

void XuLyMenu(int menu, SinhVien *a, int &n)
{
    switch (menu)
    {
        //...
        case 4:
            system("CLS");
            cout << "\n4. Xuat danh sach sinh vien theo lop";
            cout << "\nDanh sach ban dau :\n";
            Xuat_DSSV(a, n);
            _getch();

            Xuat_DSSV_Lop(a, n, "CTK39");
            _getch();
            Xuat_DSSV_Lop(a, n, "CTK38");
            _getch();
            Xuat_DSSV_Lop(a, n, "CTK37");
            _getch();
            Xuat_DSSV_Lop(a, n, "CTK36");
            break;
        //...
    }
    _getch();
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 4

Bước 7. Bổ sung chức năng 5 – xuất DSSV theo lớp và có điểm giảm dần vào chương trình.

- Trong tập tin **thuvien.h**, soạn thảo các hàm sắp danh sách sinh viên giảm dần theo điểm của các lớp

- Trong tập tin **menu.h** bổ sung xử lý chức năng xem danh sách sinh viên theo lớp và có điểm giảm trong hàm **XuLyMenu**.

- Trong tập tin *thuvien.h* :

7.1 Các hàm xuất danh sách sinh viên theo lớp và có điểm giảm dần

```
//=====
//===== Xuất Danh sach lop co diem giam dan =====
//=====

//Sap sinh vien thuoc lop giam dan theo diem
void Sap_DSSV_Lop_GiamDiem(SinhVien *a, int n, char lop[7], SinhVien *dsLop, int &h)
{
    SinhVien t;
    int i, j;
    DSSV_Lop(a, n, lop, dsLop, h);
    if (h == 0)
        return; // Khong co lop trong DSSV
    else
    {
        for (i = 0; i < h - 1; i++)
            for (j = i + 1; j < h; j++)
                if ((dsLop + i)->diem < (dsLop + j)->diem)
                {
                    t = *(dsLop + i);
                    *(dsLop + i) = *(dsLop + j);
                    *(dsLop + j) = t;
                }
    }
}

//Xuat DS lop giam dan theo diem
void Xuat_DSSV_Lop_Giam_Diem(SinhVien *a, int n, char lop[7])
{
    SinhVien *dsLop;
    int i, h;
    dsLop = new SinhVien[MAX];
    Sap_DSSV_Lop_GiamDiem(a, n, lop, dsLop, h);
    if (h == 0)
        cout << "\nKhong co lop " << lop << " trong DSSV";
    else
    {
        cout << "\n\nDanh sach sinh vien thuoc lop " << lop << " giam dan theo diem:\n";
        XuatTieuDe();
        for (i = 0; i < h; i++)
        {
            cout << endl;
            Xuat_1SV(*(dsLop + i));
        }
        XuatKeNgangDoi();
        cout << "\nCo " << h << " sinh vien thuoc lop " << lop;
    }
    delete[]dsLop;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

7.2 Bổ sung khai báo nguyên mẫu hàm

```
void Sap_DSSV_Lop_GiamDiem(SinhVien *a, int n, char lop[7], SinhVien *dsLop, int &h);
void Xuat_DSSV_Lop_Giam_Diem(SinhVien *a, int n, char lop[7]);
```

- Trong tập tin **menu.h** :

Bổ sung xử lý chức năng 5 vào case 5 (Các case khác giữ nguyên)

```
void XuLyMenu(int menu, SinhVien *a, int &n)
{
    switch (menu)
    {
        //...
        case 5:
            system("CLS");
            cout << "\n5. Xuat danh sach sinh viên theo lớp và giam dan theo diem";
            cout << "\nDanh sach ban dau :\n";
            Xuat_DSSV(a, n);
            Xuat_DSSV_Lop_Giam_Diem(a, n, "CTK39");
            _getch();
            Xuat_DSSV_Lop_Giam_Diem(a, n, "CTK38");
            _getch();
            Xuat_DSSV_Lop_Giam_Diem(a, n, "CTK37");
            _getch();
            Xuat_DSSV_Lop_Giam_Diem(a, n, "CTK36");
            //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 5

Bước 8. Bổ sung chức năng 6 – thống kê chất lượng sinh viên theo lớp vào chương trình.

- Trong tập tin **thuvien.h**, soạn thảo các hàm **thống kê chất lượng sinh viên theo lớp**
- Trong tập tin **menu.h** bổ sung xử lý chức năng **thống kê chất lượng sinh viên theo lớp** trong hàm **XuLyMenu**.

- Trong tập tin **thuvien.h** :

8.1 Hàm **thống kê chất lượng sinh viên theo lớp và xuất kết quả ra màn hình**

```
//=====================================================================
/////////////////Thong ke theo lop chat luong sinh vien va xuat ket qua ra man hinh /////////////
//=====================================================================
```

```
void ThongKe_ChatLuong(SinhVien *a, int n, char lop[7])
{
    SinhVien *dsLop;
    int h = 0;
    dsLop = new SinhVien[MAX];
    DSSV_Lop(a, n, lop, dsLop, h);
    int i;
```

```

int kem = 0,
yeu = 0,
trungBinh = 0,
kha = 0,
gioi = 0;
//Thong ke
if (h == 0)
{
    cout << "\nDanh sach " << lop << " rong!";
    getch();
}
else
{
    for (i = 0; i < h; i++)
    {
        if ((dsLop + i)->diem >= 8.5)
            gioi++;
        else
            if (7 <= (dsLop + i)->diem && (dsLop + i)->diem < 8.5)
                kha++;
            else
                if (5.5 <= (dsLop + i)->diem && (dsLop + i)->diem < 7)
                    trungBinh++;
                else
                    if (4 <= (dsLop + i)->diem && (dsLop + i)->diem < 5.5)
                        yeu++;
                    else
                        if (0 <= (dsLop + i)->diem && (dsLop + i)->diem < 4)
                            kem++;
    }
    //Xuat ds lop tuong ung
    Xuat_DSSV_Lop(a, n, lop);
    //Xuat ket qua thong ke lop
    cout << "\nThong ke chat luong lop " << lop << " :\n";
    cout << "\nSo sinh vien gioi : Gioi = " << gioi << ", ti le (%): ti le = "
        << setiosflags(ios::fixed) << setprecision(1) << (double)gioi / h;
    cout << "\nSo sinh vien kha : Kha = " << kha << ", ti le (%): ti le = "
        << setiosflags(ios::fixed) << setprecision(1) << (double)kha / h;
    cout << "\nSo sinh vien TB : TB = " << trungBinh << ", ti le (%): ti le = "
        << setiosflags(ios::fixed) << setprecision(1) << (double)trungBinh / h;
    cout << "\nSo sinh vien yeu : Yeu = " << yeu << ", ti le (%): ti le = "
        << setiosflags(ios::fixed) << setprecision(1) << (double)yeu / h;
    cout << "\nSo sinh vien kem : kem = " << kem << ", ti le (%): ti le = "
        << setiosflags(ios::fixed) << setprecision(1) << (double)kem / h;
    }
}

```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

8.2 Bổ sung khai báo nguyên mẫu hàm

void ThongKe_ChatLuong(SinhVien *a, int n, char lop[7]);

- Trong tập tin **menu.h** :

Bổ sung xử lý chức năng 6 vào case 6 (Các case khác giữ nguyên)

```
void XuLyMenu(int menu, SinhVien *a, int &n)
{
    switch (menu)
    {
        //...
        case 6:
            system("CLS");
            cout << "\n6. Thong ke chat luong lop";
            cout << "\nDanh sach ban dau :\n";
            Xuat_DSSV(a, n);
            _getch();
            system("CLS");
            ThongKe_ChatLuong(a, n, "CTK39");
            _getch();
            ThongKe_ChatLuong(a, n, "CTK38");
            _getch();
            ThongKe_ChatLuong(a, n, "CTK37");
            _getch();
            ThongKe_ChatLuong(a, n, "CTK36");
            break;
        //...
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 6.

Kiểm tra tất cả chức năng chương trình – Kết thúc chương trình.

E. Bài tập bắt buộc

Tất cả các bài tập phải được tổ chức dưới dạng thư viện hàm, có/không có hệ thống menu và cài đặt cấu trúc dữ liệu động.

1. Mảng động 1 chiều

Viết chương trình tùy chọn thực hiện các chức năng sau đây trên mảng số nguyên :

0. Thoát khỏi chương trình
1. Nhập tự động mảng
2. Xem mảng
3. Tính giá trị lớn nhất
4. Tính tổng các phần tử của mảng
5. Đếm các số dương trong dãy
6. Đếm số lần xuất hiện của x trong dãy

Yêu cầu:

- Cài đặt mảng động
- Các thuật toán cài đặt dưới hình thức đệ quy

2. Xâu ký tự

Viết chương trình thực hiện các thao tác sau trên chuỗi động:

- Đếm số khoảng trắng (gồm cả ký tự cách và Tab) trong chuỗi.

- Tìm vị trí xuất hiện của chuỗi t trong chuỗi s . Nếu s không chứa t thì trả về -1.
- Tìm vị trí xuất hiện đầu tiên của ký tự X cho trước trong chuỗi s .
- Đảo vị trí của từ đầu và từ cuối trong chuỗi s . Ví dụ: $s = \text{meo an ca}$ thì kết quả là $s = \text{ca an meo}$
- Đổi ký tự đầu tiên trong chuỗi s sang chữ HOA, còn lại là chữ thường.
- Liệt kê từng ký tự và số lần xuất hiện của chúng trong chuỗi s (không phân biệt hoa-thường).
- Đếm số từ trong chuỗi s .

3. Mảng 2 chiều động

Viết chương trình thực hiện các thao tác sau trên ma trận vuông động:

- Kiểm tra ma trận có phải là đối xứng
- Kiểm tra ma trận có phải là tam giác trên.
- Kiểm tra ma trận có phải là tam giác dưới.
- Kiểm tra ma trận có phải là tam giác chéo
- Kiểm tra ma trận có phải là tam giác đơn vị

4. Quản lý sách

Viết chương trình quản lý một thư viện sách (gọi chung là tài liệu) đáp ứng các chức năng sau:

- Nhập danh sách tài liệu.
- Xem danh sách tài liệu
- Tính tổng giá tất cả các tài liệu
- Tìm danh mục sách được xuất bản bởi $nhaXb$ vào $namXb$ cho trước.
- Tìm những bài báo khoa học có sự tham gia của tác giả $tacGia$ cho trước.
- Thống kê số lượng tài liệu theo mỗi loại.
- Liệt kê các tài liệu theo từng năm xuất bản.
- Xem thông tin tài liệu theo mã số tài liệu ($maTL$) cho trước.
- Sắp xếp các tài liệu tăng dần theo tựa đề.

Biết rằng, với mỗi tài liệu, cần quản lý các thông tin sau:

- Mã tài liệu: là một chuỗi chứa tối đa 10 ký tự.
- Tựa đề
- Loại tài liệu: thuộc một trong các loại sau: Sách, Báo khoa học, Tạp chí, Luận văn.
- Năm xuất bản
- Tác giả: lưu họ và tên tác giả của tài liệu (phân tách nhau bởi dấu phẩy), trường này không dùng cho Tạp chí.
- Nhà xuất bản: không dùng cho Luận văn.
- Giá (mua vè hoặc đền bù nếu mất): tính theo VNĐ.

F. Bài tập làm thêm

Bài 1. Bài cào

Bài cào là một kiểu chơi bài bằng bộ bài tây 52 lá. Bài được chia cho từng người, mỗi người 3 lá. Điểm của người chơi trong mỗi ván là số lẻ của tổng điểm 3 lá bài. Ví dụ, nếu tổng điểm 3 lá bài là 27 thì người đó được 7 điểm, nếu tổng là 10 điểm thì được 0 điểm. Cách tính điểm của các lá bài như sau:

- Các lá 2, 3, ..., 10 mỗi lá có điểm tương ứng với con số đó, bắt kè lá bài màu gì.
- Lá A có điểm là 1, các lá J, Q, K đều được tính là 10 điểm.

Sau khi tính điểm và trình bài, ai có số điểm cao nhất là thắng ván đó. Trường hợp đặc biệt, ai sở hữu được cả 3 lá bài đều là bài tây (J, Q hoặc K) hoặc cả 3 lá đều cùng điểm số thì thắng ngay ván đó, không cần tính điểm. Nếu có từ 2 người trở lên có cùng điểm số cao nhất thì tiền cược được chia đều.

Viết chương trình minh họa trò chơi theo mô tả trên. Trong đó, máy tính đóng vai trò người chia bài. Sau mỗi ván, máy phải xáo bài trước khi chia. Có tất cả 10 người chơi, mỗi người được cấp một số tiền M. Chương trình sẽ dừng khi chỉ còn 1 người đủ tiền đặt cược hoặc khi người dùng chọn chức năng thoát chương trình. Tiền cược quy định cho mỗi ván là C với $C \leq M/10$. Những người chơi có số tiền bé hơn C không được phép tham gia tiếp.

Chương trình phải có các chức năng sau: thiết lập mức cược C, chia bài, tính điểm và thoát chương trình.

Bài 2. Ma trận xoắn ốc

Viết chương trình cho phép người dùng nhập vào số nguyên n . Sau đó, xuất ra màn hình ma trận vuông cấp n sau khi đã điền các số từ 1 đến n^2 theo chiều xoắn ốc.

- Tính trung bình cộng của các phần tử trong mảng
- Tính tổng bình phương của các phần tử trong mảng
- Tính độ lệch lớn nhất giữa 2 phần tử nằm liền tiếp nhau
- Tính tổng lớn nhất của k phần tử liên tiếp (có thể tính xoay vòng, ví dụ cho $n=10$, $k=4$, vị trí bắt đầu tính là 8 thì sẽ tính tổng của các phần tử thứ 8, thứ 9, thứ 0 và thứ 1).
- Tìm phần tử xuất hiện nhiều nhất và số lần xuất hiện của nó.
- Tìm số âm lớn nhất và vị trí của nó.
- Đếm và xuất các phần tử xuất hiện ít nhất k lần với k cho trước.
- Đếm số lần xuất hiện của phần tử x kể từ vị trí vt cho trước.
- Xáo trộn các phần tử trong mảng một cách ngẫu nhiên.
- Sắp các số nguyên tố nằm đầu mảng và tăng, các số còn lại nằm ở cuối và giảm dần.

LAB 10. THUẬT TOÁN ĐỆ QUY

THỜI LƯỢNG: 4 TIẾT

A. Mục tiêu

- Tiếp cận các thuật toán đệ quy.
- Sau khi hoàn thành bài thực hành này, sinh viên :
 - Hiểu rõ hoạt động của các hàm đệ quy
 - Cấu trúc của hàm đệ quy
 - Cài đặt các thuật toán đệ quy
 - Tiếp tục thao tác trên các cấu trúc dữ liệu cơ bản động cài đặt bằng con trỏ.

B. Yêu cầu

- Trong phần C (hướng dẫn thực hành), sinh viên tự đọc.
- Sinh viên cần chuẩn bị bài trước để thực hành đủ khối lượng yêu cầu.

Buổi thực tập thứ 13 (4 tiết) :

- 2 tiết cuối:

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần D (hướng dẫn thực hành) của lab 10.
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab10_HD chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Sử dụng lại tập tin word LoiChuongTrinh.docx trong lab5 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - ✓ Giáo viên sẽ thu bài qua hệ thống thu bài trực tuyến tại phòng lab (thu thư mục MaSV_Lab10_HD) vào cuối buổi thực tập.

Buổi thực tập thứ 14 (4 tiết) :

- 2 tiết đầu:

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Tất cả các bài trong phần E (Bắt buộc) của lab 10.
 - Yêu cầu lưu trữ :
 - ✓ Tạo thư mục MaSV_Lab10_BB chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Sử dụng lại tập tin word LoiChuongTrinh.docx trong lab5 để bổ sung thêm các lỗi mới xảy ra khi chạy chương trình và cách giải quyết tương ứng.
 - ✓ Giáo viên sẽ thu bài qua hệ thống thu bài trực tuyến tại phòng lab (thu thư mục MaSV_Lab10_BB) trước giờ giải lao.

C. Ôn tập lý thuyết

1. Cấu trúc của một hàm đệ quy :

KDL Tên_Hàm (Các tham số)

{

Các biến cục bộ

if(điều kiện dừng)

Xử lý trường hợp đặc biệt;

else

if (điều kiện hợp lệ)

Xử lý đệ quy;

//Liên hệ đệ quy

}

2. Phân loại các thuật toán đệ quy :

- Đệ quy tuyến tính :

Trong mô tả thuật toán (hàm), thuật toán (hàm) chỉ có 1 lần gọi chính nó.

- Đệ quy nhị phân :

Trong mô tả thuật toán (hàm), thuật toán (hàm) có 2 lần gọi chính nó.

- Đệ quy phi tuyến :

Trong mô tả thuật toán, thuật toán (hàm) có lặp một số lần gọi chính nó.

- Đệ quy hỗn hợp :

Đây là trường hợp 2 thuật toán đệ quy (hàm đệ quy) gọi lẫn nhau.

D. Hướng dẫn thực hành

Bài 1.(Đệ quy tuyến tính)

Viết chương trình tùy chọn thực hiện các chức năng sau đây trên mảng số nguyên :

- Thoát khỏi chương trình
- Nhập tự động mảng
- Xem mảng
- Tính tổng các phần tử trong mảng
- Tính tích các phần tử trong mảng
- Tính giá trị nhỏ nhất của mảng
- Tính giá trị lớn nhất trong mảng
- Tính tổng các số nguyên tố trong mảng
- Tìm vị trí cuối cùng x xuất hiện trong mảng, nếu có
- Đếm số đường chạy của mảng

Yêu cầu:

- Cài đặt mảng động
- Các thuật toán cài đặt dưới hình thức đệ quy

Tạo project rỗng, đặt tên **MaSV_Lab10_D_Bai1**.

Chương trình tổ chức theo thư viện hàm và có hệ thống menu.

Tham khảo tập tin **thuvien.h** sau đây :

```
#define TAB '\t'  
typedef int *DayDong;
```

```
void NhapTuDong(DayDong a, int n)
{
    int i;
    // Gieo số ngẫu nhiên đầu tiên
    srand((unsigned)time(NULL));
    // Duyệt qua từng phần tử từ vị trí 0 tới n-1
    for (i = 0; i<n; i++)
    {
        // Sinh một số ngẫu nhiên trong phạm vi
        // [-20..20) rồi gán cho phần tử thứ i
        *(a+i) = -20 + rand() % 40;
    }
}

void XuatMang(DayDong a, int n)
{
    int i;
    for (i = 0; i<n; i++)
        cout << *(a+i) << TAB;
}

//tinh tong
int TinhTong(DayDong a, int n)
{
    int kq;
    if ( n==1)
        kq = *a;
    else
        if (n >1)
            kq = TinhTong(a,n-1) + *(a+n-1);
    return kq;
}

//tinh tích
int TinhTich(DayDong a, int n)
{
    int kq;
    if ( n==1)
        kq = *a;
    else
        if (n >1)
            kq = *(a+n-1) * TinhTich(a,n-1);
    return kq;
}

//Tinh min
int TinhMin(DayDong a, int n)
{
    int kq;
    if (n == 1)
        kq = *(a+0);
    else
        if (n > 1)
```

```

        if (TinhMin(a, n - 1) > *(a + n - 1))
            kq = *(a + n - 1);
        else
            kq = TinhMin(a, n - 1);
    return kq;
}

```

```

//Tinh max
int TinhMax(ĐayDong a, int n)
{
    int kq;
    if (n == 1)
        kq = *(a + 0);
    else
        if (n > 1)
            if (TinhMax(a, n - 1) < *(a + n - 1))
                kq = *(a + n - 1);
            else
                kq = TinhMax(a, n - 1);
    return kq;
}

```

```

//Tinh so duong chay
int TinhSo_DC(ĐayDong a, int n)
{
    int kq;
    if (n == 1)
        kq = 1;
    else
        if (n > 1)
            if (*(a + n - 1) < *(a + n - 2))
                kq = TinhSo_DC(a, n - 1) + 1;
            else
                kq = TinhSo_DC(a, n - 1);
    return kq;
}

```

```

//Kiem tra nguyen to
int KiemTra_NT(int x)
{
    int i, m,
        kq;
    if (x < 2)
        kq = 0;
    else
    {
        m = (int)sqrt((double)x);
        kq = 1;
        for (i = 2; i <= m; i++)
            if (x % i == 0)
            {
                kq = 0;
                break;
            }
    }
}
```

```

        }
    }
    return kq;
}

//Tinh tong nguyen to
int TinhTong_NT(DayDong a, int n)
{
    int kq;
    if (n == 1)
        if (KiemTra_NT(*a))
            kq = *a;
        else
            kq = 0;
    else
        if (n > 1)
            if (KiemTra_NT(*(a + n - 1)))
                kq = TinhTong_NT(a, n - 1) + *(a + n - 1);
            else
                kq = TinhTong_NT(a, n - 1);
    return kq;
}

//vi tri cuoi cung x xuat hien
int Tim_Cscc(DayDong a, int n, int x)
{
    int kq;
    if (n == 1)
        if (*a == x)
            kq = 0;
        else
            kq = -1;
    else
        if (n > 1)
            if (*(a + n - 1) == x)
                kq = n - 1;
            else
                kq = Tim_Cscc(a, n - 1, x);
    return kq;
}

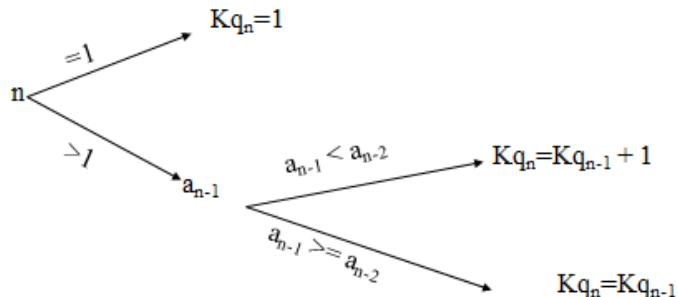
//Dem so duong chay
int DemSo_DC(DayDong a, int n)
{
    int kq;
    if (n == 1)
        kq = 1;
    else
        if (n > 1)
            if (*(a + n - 1) < *(a + n - 2) )
                kq = DemSo_DC(a, n - 1) + 1;
            else
                kq = DemSo_DC(a, n - 1);
    return kq;
}

```

Ghi chú:

Đường chạy :dãy con tăng dần lớn nhất trong dãy

Cây liệt kê:



Bài 2. (đệ quy nhị phân)

Viết chương trình tùy chọn thực hiện các chức năng sau đây:

0. Thoát khỏi chương trình
1. Xuất n số Fibonacci đầu tiên
2. Xuất tổ hợp chẵp k trong n.
3. Tìm giá trị Min, Max trong đoạn a[l..r] của mảng a[1..n]

Yêu cầu:

- Các thuật toán cài đặt dưới hình thức đệ quy

Tạo project rỗng, đặt tên **MaSV_Lab10_D_Bai2**.

Chương trình tổ chức theo thư viện hàm và có hệ thống menu.

Tham khảo tập tin **thuvien.h** sau đây :

```

//So fibonacci thu n
int Fib(int n)
{
    int kq;
    if (n == 0)
        kq = 0;
    else
        if (n == 1)
            kq = 1;
        else
            if (n > 1)
                kq = Fib(n - 1) + Fib(n - 2);
    return kq;
}
  
```

//k so Fibonacci dau tien : dem tu 0

```

void Xuat_K_Fib(int k)
{
    int i;
    for (i = 0; i < k; i++)
        cout << Fib(i) << '\t';
  
```

```

}

//Tinh to hop chap k trong n : Cnk
int Tinh_Cnk(int n, int k)
{
    int kq;
    if (k == 0 || k == n)
        kq = 1;
    else
        if (0 < k && k < n)
            kq = Tinh_Cnk(n - 1, k) + Tinh_Cnk(n - 1, k - 1);
    return kq;
}
void MinMax(int a[MAX], int l, int r, int &Min, int &Max)
{
    int Min1, Min2, Max1, Max2;
    if (l == r)
    {
        Min = a[l];
        Max = a[l];
    }
    else
    {
        MinMax(a, l, (l + r) / 2, Min1, Max1);
        MinMax(a, (l + r) / 2 + 1, r, Min2, Max2);
        if (Min1 < Min2)
            Min = Min1;
        else
            Min = Min2;
        if (Max1 > Max2)
            Max = Max1;
        else
            Max = Max2;
    }
}

```

Bài 3 : (đệ quy phi tuyến)

Liệt kê các hoán vị của n số nguyên dương đầu tiên.

Tạo project rỗng, đặt tên ***MaSV_Lab10_D_Bai3***.

Chương trình tổ chức theo thư viện hàm và không có hệ thống menu.

Tham khảo tập tin ***thuvien.h*** sau đây :

```
#define MAX 10
int a[MAX], n ; // khai bao toan cuc
void HV(int &x, int &y)
{
    int t;
    t = x;
    x = y;
    y = t;
}
```

```
void LietKe_HoanVi(int k)
{
    int j;
    if (k == 1)
    {
        cout << endl;
        for (j = 0; j < n; j++)
            cout << a[j] << '\t';
    }
    else
        for (j = k - 1; j >= 0; j--)
    {
        HV(a[k - 1], a[j]);
        LietKe_HoanVi(k - 1);
        HV(a[j], a[k - 1]);
    }
}
```

E. Bài tập bắt buộc

Trong các bài sau đây, các thuật toán viết dưới dạng đệ quy.

Bài 1:

Viết chương trình tùy chọn thực hiện các chức năng sau :

1. Đảo ngược số nguyên dương n
2. Đếm số lượng các chữ số trong số nguyên dương n
3. Tìm chữ số có giá trị lớn nhất trong số nguyên dương n.
4. Đổi sang hệ nhị phân số nguyên dương n
5. Tìm chữ số đầu tiên trong n.

Bài 2.

Viết chương trình tùy chọn thực hiện các chức năng sau :

1. Tính tổng n số lẻ dương đầu tiên.
2. Tính tích n số lẻ dương đầu tiên.
3. Tính tổng : $1+(1.2)+(1.2.3) + \dots + (1.2\dots.n)$

Bài 3:

Tính ước chung lớn nhất của 2 số nguyên dương.

Bài 4.

Xuất xâu ký tự đảo ngược.

Bài 5.

Viết chương trình tùy chọn thực hiện các chức năng sau :

1. Đếm số lần xuất hiện của x trong dãy a.
2. Đếm số các số nguyên tố trong dãy a.

Bài 6. Viết chương trình giải bài toán Tháp Hà Nội.

LAB 11 :LẬP TRÌNH VỚI NHẬP XUẤT CÁC TẬP TIN VĂN BẢN THỜI LƯỢNG: 4 TIẾT

A. Mục tiêu

- Cung cấp sinh viên kiến thức về đọc và ghi file văn bản
- Sinh viên thực hiện được thao tác đọc nội dung file văn bản lưu trữ vào các cấu trúc dữ liệu phù hợp :Mảng 1 chiều, Mảng 2 chiều (ma trận), mảng cấu trúc.

B. Yêu cầu

Buổi thực tập thứ 14 (4 tiết) :

- 2 tiết cuối:

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Các bài trong phần D (hướng dẫn luyện tập)
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục MaSV_HoVaTen_Lab11_D chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Gửi cho giáo viên qua mail theo thời gian qui định sau;
 - ✓ Địa chỉ mail : giaotiep2008@gmail.com

Buổi thực tập thứ 15 (4 tiết) :

- 2 tiết đầu:

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Các bài trong phần E (Bắt buộc)
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục MaSV_HoVaTen_Lab11_E chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Xem thông báo của giáo viên

- 2 tiết cuối:

Ôn tập

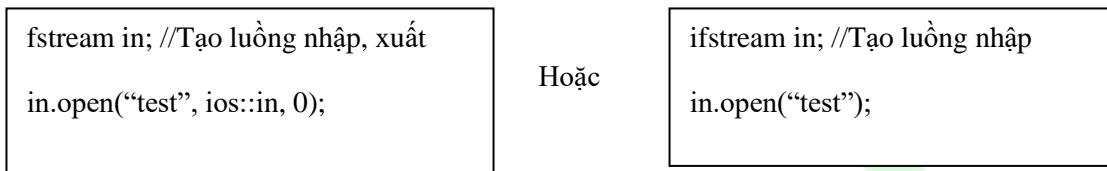
C. Hướng dẫn lý thuyết

1. Khai báo thư viện sử dụng <fstream>
2. Sử dụng luồng ghi/đọc:
 - Đối tượng
 - **fstream** //Luồng đọc và ghi
 - **ofstream** // Luồng ghi
 - **ifstream** //Luồng đọc
3. Tạo luồng
 - **fstream Tên_Luong;** //Nhập /Xuất.

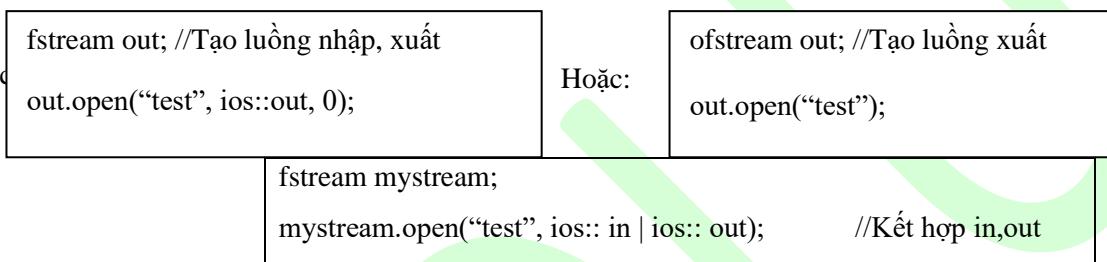
- ifstream Tên_Luong; //Nhập .
- ofstream Tên_Luong; //Xuất.

4. Mở tập tin (đọc – nhập, ghi-xuất)

a. Mở để đọc:



b. Mở để ghi: Các cách mở sau là tương đương:



d. Kết hợp tạo luồng vừa mở luôn tập tin.

Khởi tạo một đối tượng fstream ngay khi nó được khai báo.

- Đọc:

fstream in ("test",ios::in);
hoặc
ifstream in("test")

- Ghi:

fstream out ("test",ios::out);
hoặc
ofstream out("test");

Tóm lại, các cách viết sau là tương đương:

fstream in; in.open("test",ios::in);	fstream in ("test",ios::in);	ifstream in("test");
fstream out; out.open("test",ios::out);	fstream out ("test",ios::out);	ofstream out("test");

5. Kết quả của việc mở tập tin tin.

Nếu việc mở tập tin (đọc hay ghi, gắn với luồng đã mở) không thành công, luồng sẽ trả về giá trị 0

6. Kiểm tra mở tập tin có thành công hay không:

Xem khai báo:

```
fstream mystream;  
mystream.open("test", ios:: in | ios:: out);
```

Cách 1:

Kiểm tra giá trị của luồng (Giá trị luồng trả về 0 nếu không mở được tập tin)

```
if ( !mystream)  
{  
    cout<<"\nTập tin không mở được!";  
    exit(1); //return;  
}
```

Cách 2:

Dùng hàm thành phần fail(). Hàm trả về giá trị 1 nếu việc mở tập tin không thành công.

```
if (mystream.fail())  
{  
    cout<< "\nTập tin không mở được!";  
    exit(1); //return;  
}
```

7. Kiểm tra hết tập tin.

Dùng hàm thành phần eof() của luồng ?fstream

$$?fstreamName.eof() = \begin{cases} 1; & \text{Hết dữ liệu} \\ 0; & \text{Ngược lại} \end{cases}$$

trong đó ? là i, o hoặc **không có ký tự nào**

Với khai báo:

```
ifstream in("test");  
if(!in.eof()) //chưa hết dữ liệu  
{  
    //Xử lý dữ liệu  
}
```

8. Thảo tác đọc dữ liệu từ tập tin, ghi dữ liệu vào tập tin.

Với các khai báo:

```
ifstream in("test1");  
ofstream out("test2");  
int sonhap;
```

- Đọc:

```
in>>sonhap; // đọc một giá trị dữ liệu trong test 1, lưu trữ trong sonhap
```

- Ghi:

```
out<<sonhap; //ghi sonhap vao test2
```

9. Đóng tập tin:

Dùng hàm thành phần close()

```
mystream.close();
```

Lưu ý: Đối với tập tin, đã có thao tác mở thì nên có thao tác đóng.

D. Hướng dẫn luyện tập

Bài 1: (Chuyển dữ liệu của Tập tin theo định dạng vào Mảng 1 chiều)

Đọc dữ liệu của tập tin văn bản có giá trị là các số nguyên , ghi vào mảng một chiều.

Giả sử nội dung tập tin văn bản \text1.txt có định dạng sau:

9
1 2 3 4 5 6 7 8 9

Trong đó:

- Hàng 1: chứa số lượng các phần tử của tập tin (9)
- Hàng 2: Các giá trị của tập tin.

Viết chương trình đọc các giá trị trong tập tin \text1.txt rồi lưu trữ vào mảng 1 chiều các số nguyên.

Tạo project **Lab01_P2_Bai01** theo cấu trúc như hướng dẫn trong lab01_P1 (không có tập tin menu.h) :

- Nội dung trong tập tin Thuvien.h :

```
#define MAX 100
void File_Array(char *filename, int a[MAX], int &n);

//Đọc dữ liệu trong tập tin filename ghi vào mảng a
void File_Array(char *filename, int a[MAX], int &n)
{
    int i;
    ifstream in(filename); //Mở tập tin filename để đọc
    if (!in) //Kiểm tra việc mở tập tin
    {
        cout<<"\nLoi mo file !";
        return;
    }

    in>>n; //đọc dữ liệu đầu tiên của tập tin (hàng đầu), lưu vào n.
            //n là kích thước mảng
    for (i = 0; i < n; i++)
    {
        in>>a[i]; //lần lượt đọc dữ liệu tập tin ghi vào mảng 1 chiều arr
    }
    in.close();
}
```

- Nội dung trong tập tin Program.cpp :

```
#include <iostream>
#include <fstream>
using namespace std;
#include "Thuvien.h"
void ChayChuongTrinh();

int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    int n, a[MAX];
    char filename[MAX];
    system("cls");
    cout<<"Nhập tên file mở để đọc:";
    cin>>filename;
    File_Array(filename,a,n);
    cout<<n<<endl;
    for (int i = 0; i < n; i++)
        cout<<a[i]<<'\t';
    system("PAUSE");
}
```

- Tạo và soạn thảo nội dung tập tin “Text1.txt” :

Thao tác tương tự như tạo các tập tin .cpp, .h , mô tả vắn tắt như sau :

- Nhấp phải chuột vào thư mục **ReSource Files**, chọn **Add > New Items**, chọn **Utility > Text File (.txt)**
- Trong khung Name : đặt tên tập tin là “Text1.txt”
- Chọn add. Đã tạo xong tập tin có tên Text1.txt
- Soạn thảo nội dung tập tin theo yêu cầu trong vùng soạn thảo tập tin. Lưu ý là 2 giá trị dữ liệu khác nhau phải tách biệt bởi 1 ký tự tách (ký tự trắng, tab, . . .)

Ghi chú :

Hàm chuyển dữ liệu tập tin vào mảng 1 chiều có thể viết dạng khác :

```
int File_Array(char *filename, int a[MAX], int &n)
{
    ifstream in(filename);
    if (!in)
        return 0; //Không thành công
```

```

in >> n;
for (int i = 0; i < n; i++)
    in >> a[i];
in.close();
return 1;//Thanh cong
}

```

- Lưu ý : viết lại cách sử dụng hàm File_Array trong hàm main() cho cách viết này.

Bài 2: (Chuyển dữ liệu của Mảng 1 chiều các số vào Tập tin)

Đọc dữ liệu của mảng 1 chiều rồi ghi vào tập tin “\kq1.txt” theo định dạng :

- Hàng đầu : số phần tử của tập tin (kích thước của mảng)
- Các hàng sau là các phần tử của tập tin (các phần tử của mảng)

Xuất tập tin “\kq1.txt” ra mà hình để kiểm tra.

Mảng a :

```
int a[9] = {1,2,3,4,5,6,7,8,9};
```

Tập tin “\kq1.txt” sẽ như sau :

```

9
1   2   3   4   5   6   7   8   9
//Tạo Project như trên
//hàm chuyển dữ liệu của mảng 1 chiều vào Tập tin)
void Write_Int(int a[], int n, char *filename)
{

```

```

    ofstream out(filename); //Mo de ghi
    if (!out)
    {
        cout<<"\nLoi mo file !";
        return;
    }
    out<<n; //ghi kích thước mảng vào tập tin
    out<<'n'; //xuống hàng

```

```

    for(int i = 0; i < n;i++)
    {
        out<<a[i];//lần lượt ghi a[i] vào tập tin
        out<<'\t'; //2 giá trị trong tập tin cách 1 tab
    }
    out.close();
    cout<<"\nghi xong dữ liệu vào tệp "<<filename;
}

```

//ham xuất dữ liệu của tập tin ra màn hình

```

void File_Display1(char *filename)
{

```

```

int n, x;
ifstream in(filename); //Mo de doc
if (!in)
{
    cout << "\nLoi mo file!";
    return;
}
in >> n;
cout << endl << n << endl;
for (int i = 0; i < n; i++)
{
    in >> x;
    cout << x << '\t';
}
in.close();
}

```

Bài 3 : (Chuyển dữ liệu của Tập tin theo định dạng vào Mảng 1 chiều)

Định dạng tập tin chỉ chứa dữ liệu – không có số phần tử tập tin

Giả sử tập tin văn bản \text2.txt có nội dung là các số nguyên sau:

1 2 3 4 5 6 7 8 9

Viết chương trình đọc các giá trị trong tập tin \text2.txt rồi lưu trữ vào mảng 1 chiều các số nguyên.

//Tạo Project như trên

```

// hàm đọc dữ liệu của tập tin, ghi vào mảng 1 chiều.
//Tập tin có định dạng : chỉ chứa dữ liệu – không có số phần tử tập tin ở hàng đầu)
int File_Array1(char *filename, int a[MAX], int &n)
{
    ifstream in(filename); //Mo de doc
    if (!in)
        return 0; //Khong thanh cong
    n = 0;
    while (!in.eof())
    {
        in >> a[n];
        n++;
    }
    in.close();
    return 1; //Thanh cong
}

```

- Tạo và soạn thảo nội dung tập tin “Text2.txt” theo yêu cầu.

Bài 4: (Chuyển dữ liệu của Tập tin vào Ma trận vuông)

Đọc dữ liệu tập tin (hàng đầu của tập tin là cấp ma trận vuông) rồi ghi vào ma trận vuông.

Giả sử tập tin văn bản \text3.txt có nội dung sau:

```
3  
1     2     3  
4     5     6  
7     8     9
```

Trong đó:

- Hàng 1: Chỉ cấp của ma trận vuông (số phần tử của tập tin).
- Các hàng sau là các phần tử của tập tin

Viết hàm đọc các giá trị trong tập tin \text3.txt rồi lưu trữ vào ma trận vuông cấp n.

Tạo project **Lab01_P2_Bai04** theo cấu trúc như hướng dẫn trong lab01_P1 (không có tập tin menu.h) :

- Nội dung trong tập tin Thuvien.h :

```
#define MAX 100
```

```
int File_Mat(char *filename, int a[MAX][MAX], int &n);  
int File_Mat(char *filename, int a[MAX][MAX], int &n)  
{  
    ifstream in(filename); //Mo de doc  
    if (!in)  
        return 0;  
    in>>n;  
    int i, j;  
    for (i = 0; i < n; i++)  
        for (j = 0; j < n; j++)  
            in>>a[i][j];  
    in.close();  
    return 1;  
}
```

- Nội dung tập tin program.cpp như sau:

```
#include <iostream>  
#include <fstream>  
using namespace std;  
#include "Thuvien.h"  
void ChayChuongTrinh();  
int main()  
{  
    ChayChuongTrinh();  
    return 1;  
}  
  
void ChayChuongTrinh()  
{  
    int n, a[MAX][MAX], i, j;
```

```

char filename[80];
system("cls");
cout<<"Nhập tên file mở để đọc:"; cin>>filename;
File_Mat(filename,a,n);
cout<<"\n\n="<<n;
cout<<endl;
for ( i = 0; i < n; i++)
{
    cout<<"\n";
    for (j = 0; j < n; j++)
        cout<<a[i][j]<<'t';
}
system("PAUSE");
}

```

- Tạo và soạn thảo nội dung tập tin “Text3.txt” theo yêu cầu:

Bài 5: (Chuyển dữ liệu ma trận vuông vào Tập tin theo định dạng)

Duyệt ma trận vuông, ghi dữ liệu của ma trận vào tập tin theo định dạng như sau:

- Dòng 1: n //Cấp ma trận vuông
- Các dòng sau: Ghi các giá trị ma trận theo n dòng và n cột tương ứng. Các giá trị cách nhau 1 tab.

Chẳng hạn:

		3
1	2	3
4	5	6
7	8	9

Sau đó xuất tập tin ra màn hình.

//Tạo Project như trên

//Hàm lưu ma trận vuông vào tập tin

```

void Mat_File(char *filename, int a[MAX][MAX], int n)
{
    ofstream out(filename);
    if (!out)
    {
        cout<<"\nLoi mo file !";
        return;
    }
    out<<n;
    int i, j;
    for(i = 0; i < n; i++)
    {
        out<<endl;
        for(j = 0; j < n; j++)

```

```

        out<<a[i][j]<<"\t";
    }
    out.close();
    cout<<"\nLuu file thanh cong!";
}
//Xuất tập tin ma tran vuong ra màn hình
void File_Display2(char *filename)
{
    int n, x;
    ifstream in(filename); //Mo de doc
    if (!in)
    {
        cout << "\nLoi mo file !";
        return;
    }
    in >> n;
    cout << n;
    for (i = 0; i < n; i++)
    {
        cout<<endl;
        for (j = 0; j < n; j++)
        {
            in >> x;
            cout << x << '\t';
        }
    }
    in.close();
}

```

Bài 6: (Chuyển dữ liệu của tập tin cấu trúc vào Mảng cấu trúc)

Giả sử tập tin văn bản `\tnhanvien.txt` chứa những thông tin về nhân viên của một công ty. Mỗi dòng trong tập tin chứa các thông tin: Mã số, họ tên, Ngày tháng năm sinh, địa chỉ, Lương.

Viết hàm đọc các thông tin trong tập tin `|nhanvien.txt` rồi lưu trữ vào mảng 1 chiều các cấu trúc có trường tương ứng

- Tập tin `nhanvien.txt` :

252348	Vo_Minhh_Duong	3	1	1980	Da_Lat	15000543
113454	Hoang_Hoa_Tra	12	12	1987	Khanh_Hoa	18183210
213000	Truong_Thanh_Hang	2	2	1980	Phu_Yen	15000543
121234	Nguyen_Thi_Hang	10	8	1987	Binh_Dinh	20232343
222345	Le_Van_Trong	12	5	1991	Hue	18183210
200320	Van_Minhh_Duong	10	10	1988	Da_Lat	18183210
150034	Tran_Hoang_Hoa	15	10	1996	Binh_Dinh	18183210
213400	Ly_Thi_Hang	18	7	1988	Khanh_Hoa	20232343
203000	Dinh_Thai_Duong	21	12	1980	Da_Lat	15000543
201145	Le_Van_Tam	25	6	1988	Hue	18183210
202220	Trinh_Hoai_Duong	19	10	1991	Da_Lat	18183210
165234	Nguyen_Minhh_Tam	22	9	1980	Binh_Dinh	20232343

Tạo project Lab02_Bai6 theo cấu trúc như hướng dẫn trong lab 01 (không có tập tin menu.h) :

- Nội dung trong tập tin Thuvien.h :

```
#define MAX 20
#define THOAT 0
struct DATE
{
    int ngay;
    int thang;
    int nam;
};
struct NHANVIEN
{
    int ms;
    char hoten[MAX];
    DATE ntn;
    char diachi[MAX];
    double luong;
};
void Xuat(NHANVIEN ds[MAX], int n);
int Read_struct(char *filename, NHANVIEN ds[MAX], int &n);

/******************
int Read_struct(char *filename, NHANVIEN ds[MAX], int &n)
{
    ifstream in(filename);
    if (!in)
        return 0;
    n = 0;
    while (!in.eof())
    {
        in >> ds[n].ms;
        in >> ds[n].hoten;
        in >> ds[n].ntn.ngay;
        in >> ds[n].ntn.thang;
        in >> ds[n].ntn.nam;
        in >> ds[n].diachi;
        in >> ds[n].luong;
        n++;
    }
    in.close();
    return 1;
}
void Xuat(NHANVIEN ds[MAX], int n)
{
    cout << setiosflags(ios::left);
    cout << setw(20) << "MS"
        << setw(20) << "Ho Ten"
        << setw(20) << "NTN Sinh"
        << setw(20) << "Dia chi"
        << setw(20) << "Luong";
    cout << endl;
    for (int i = 0; i < n; i++)
    {
        cout << setw(20) << ds[i].ms
            << setw(20) << ds[i].hoten
            << setw(2) << ds[i].ntn.ngay << '/'
            << setw(2) << ds[i].ntn.thang << '/'
            << setw(14) << ds[i].ntn.nam
            << setw(20) << ds[i].diachi
            << setiosflags(ios::fixed) << setprecision(2) << setw(20) << ds[i].luong;
        cout << endl;
    }
}
```

}

- Nội dung tập tin Program.cpp :

```
#include <iostream>
#include <fstream>
#include <string.h>
#include<iomanip>
using namespace std;
#include "Thuvien.h"

void ChayChuongTrinh();

int main()
{
    ChayChuongTrinh();
    return 1;
}

void ChayChuongTrinh()
{
    char filename[80];
    NHANVIEN ds[MAX];
    int n, kq;
    do
    {
        system("cls");
        cout << "Nhập tên file mở để đọc:"; cin >> filename;
        kq = Read_struct(filename, ds, n);
        if (!kq)
        {
            cout << "\nNhập lại tập tin";
            system("PAUSE");
        }
    } while (!kq);
    system("PAUSE");
}
```

- Tạo và soạn thảo nội dung tập tin “nhanvien.txt” theo yêu cầu:

Lưu ý :

- Thông tin mỗi nhân viên sẽ nằm trên 1 hàng.
- Trên mỗi hàng, các thông tin đặc trưng khác nhau của nhân viên phải tách biệt bằng ký tự tách (ký tự trắng, tab,...), tốt nhất sử dụng các tab để đóng thảng các thông tin nhân viên cùng tính chất thành các cột.
- Các chuỗi biểu diễn thông tin đặc trưng của nhân viên gồm nhiều từ, các từ phải kết nối với nhau bằng một ký tự kết nối như dấu gạch dưới (_).

E. Bài tập

Bài tập:

Viết chương trình tùy chọn thực hiện các thao tác :

0. Thoát khỏi chương trình
1. Chuyển dữ liệu của tập tin (các số nguyên) vào mảng 1 chiều

2. Chuyển dữ liệu của tập tin (cấu trúc) vào mảng cấu trúc

Lưu ý :

Khi thực hiện các thao tác trên cần xuất các mảng ra màn hình.

- Tập tin \text1.txt cho thắc tác 1 (hang đầu là số phần tử của tập tin):

8

12 2 8 5 1 6 4 15

- Tập tin \text2.txt cho thắc tác 1 (Các giá trị đều là dữ liệu của tập tin):

12 2 8 5 1 6 4 15

- Tập tin \text3.txt cho thắc tác 2:

:Ma_Nu	:Ho	tLot	Ten	:NTNSinh	:Dia Chi	:Luong
2523480	Vo	Minh	Duong	3	1	1980
1134547	Hoang	Hoa	Tra	12	12	1987
2130007	Truong	Thanh	Hang	2	2	1980
1212345	Nguyen	Thi	Hang	10	8	1987
2223453	Le	Van	Trong	12	5	1991
2003205	Van	Minh	Duong	10	10	1988
1500348	Tran	Hoang	Hoa	15	10	1996
2134007	Ly	Thi	Hang	18	7	1988
2030007	Dinh	Thai	Duong	21	12	1980
2011453	Le	Van	Tam	25	6	1988
2022205	Trinh	Hoai	Duong	19	10	1991
1652345	Nguyen	Minh	Tam	22	9	1980

MỤC LỤC

Lab 01	LÀM QUEN VỚI C++ VÀ VISUAL STUDIO	3
Lab 02	ĐỊNH NGHĨA HÀM VÀ GỌI HÀM	25
Lab 03	CÁC CÂU LỆNH ĐIỀU KHIỂN	39
Lab 04	TỔ CHỨC THƯ VIỆN HÀM VÀ MENU	59
Lab 05	CÁC KỸ THUẬT XỬ LÝ MẢNG MỘT CHIỀU	89
Lab 06	CÁC KỸ THUẬT XỬ LÝ MẢNG HAI CHIỀU (MA TRẬN)	131
Lab 07	CÁC KỸ THUẬT XỬ LÝ XÂU KÝ TỰ	166
Lab 08	KIẾU CẤU TRÚC	194
Lab 09	BIÊN ĐỘNG VÀ KIẾU CON TRỎ	244
Lab 10	THUẬT TOÁN ĐỆ QUY	269
Lab 11	LẬP TRÌNH VỚI NHẬP XUẤT CÁC TẬP TIN VĂN BẢN	277

