TRƯỜNG ĐẠI HỌC ĐÀ LẠT
KHOA CÔNG NGHỆ THÔNG TIN

GIÁO TRÌNH
NGUYÊN LÝ LẬP TRÌNH CẤU TRÚC

Trần Tuấn Minh – Nguyễn Thị lương
Đà Lạt 2020

MỤC LỤC

LỜI NÓI ĐẦU

Giáo trình " Lập trình cấu trúc với C/C++ " là giáo trình nhập môn về lập trình trong hệ thống đào tạo tín chỉ tại khoa Công nghệ thông tin trường Đại học Đà Lạt. Giáo trình trình bày trong 5 tín chỉ, gồm 3 tín chỉ Lý thuyết và 2 tín chỉ thực hành.
Về phương pháp lập trình, giáo trình giới thiệu cho sinh viên phương pháp lập trình cấu trúc; về thể hiện, chương trình được tổ chức trong Microsoft Visual Studio 2005

Nội dung giáo trình gồm 10 chương:

Từ chương 1 đến chương 4: Giới thiệu mở đầu, các khái niệm và các phép toán cơ bản trong C/C++.

Chương 5: Các câu lệnh điều khiển

Chương này giới thiệu các cấu trúc điều khiển trong C/C++

Chương 6: Các cấu trúc dữ liệu cơ bản

Trong các kiểu dữ liệu có cấu trúc, chương này trình bày các kiểu dữ liệu tĩnh có cấu trúc cơ bản như Mảng, xâu ký tự, Cấu trúc, . . .

Chương 7: Con trỏ

Chương này tiếp cận khái niệm con trỏ, tổ chức các kiểu dữ liệu động bằng con trỏ, tiếp cận với cách truyền tham số bằng biến

Chương 8: Thuật toán đệ quy

Chương này trình bày cách tổ chức các hàm đệ quy.

Chương 9: Các nguyên lý lập trình cấu trúc

Chương này giới thiệu các nguyên lý thường dùng khi tổ chức chương trình theo phương pháp lập trình cấu trúc như: nguyên lý phân rã theo chức năng, thiết kế từ trên xuống, từ dưới lên, làm mịn dần từng bước, . . .

Chương 10: Lập trình với tập tin

Chương này giới thiệu các kỹ thuật lập trình trên các tập tin nhị phân, văn bản: đọc, ghi với các cấu trúc dữ liệu cơ bản.

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.

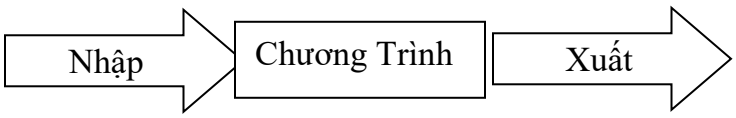
Cuối cùng, Chúng tôi cảm ơn sự động viên, giúp đỡ nhiệt thành của các bạn đồng nghiệp trong khoa Công nghệ thông tin để tập giáo trình này được hoàn thành.

Đà Lạt, tháng 9 năm 2020
Các tác giả

CHƯƠNG 1. CÁC KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH

1.1 Mở đầu

Lập trình thực chất là điều khiển - bằng một ngôn ngữ lập trình cụ thể - các xử lý thông tin trên máy tính điện tử theo yêu cầu của bài toán đặt ra.
Kết quả của lập trình là chương trình được hợp thức hóa.
Thông tin gửi đến chương trình, chương trình xử lý, kết quả sẽ được gửi đi.



Để lập trình phải biết cách tổ chức dữ liệu (cấu trúc dữ liệu) và cách thức xử lý dữ liệu(thuật toán) để tạo ra chương trình mong muốn.
K. Wirth đã đưa ra công thức:

CHƯƠNG TRÌNH = CẤU TRÚC DỮ LIỆU + THUẬT TOÁN

Có nhiều cách tổ chức dữ liệu cũng như có nhiều thuật toán để giải một bài toán . Đưa ra cách tổ chức dữ liệu tốt nhất và chỉ ra thuật toán tốt nhất là công việc của người lập trình.
Các phương pháp lập trình thường được sử dụng là lập trình có cấu trúc và lập trình theo hướng đối tượng.
Các phương pháp lập trình này phản ánh quan niệm lập trình là một hoạt động khoa học và có phương pháp chứ không phải là một công việc ngẫu hứng.
Đặc trưng của lập trình có cấu trúc là chương trình phải có cấu trúc.
Tính cấu trúc của chương trình thể hiện trên các mặt sau:

- Cấu trúc về mặt dữ liệu:

Từ những dữ liệu đã có , có thể xây dựng những dữ liệu có cấu trúc phức tạp hơn.

- Cấu trúc về mặt lệnh:

Từ những lệnh đơn giản đã có, có thể xây dựng được những lệnh có cấu trúc phức tạp hơn.

- Cấu trúc về mặt chương trình:

Một chương trình lớn có thể phân rã thành nhiều modul (hay các chương trình con) độc lập, mỗi chương trình con lại có thể chia ra thành các chương trình con khác... nên chương trình được tổ chức thành một hệ phân cấp. Nhờ vậy mà một chương trình lớn, phức tạp được phân thành những modul chương trình đơn giản, dễ viết, dễ đọc, dễ sửa...

1.2 Thuật toán

1.2.1 Định nghĩa trực quan thuật toán

Thuật toán là dãy hữu hạn các thao tác, sắp xếp theo trình tự xác định, được đề ra nhằm giải quyết một lớp bài toán nhất định.
Các thao tác sẽ biến đổi trạng thái bài toán trước khi thực hiện thao tác thành trạng thái kết quả.
Dãy tuần tự các thao tác trong thuật toán biến đổi trạng thái ban đầu của bài toán thành trạng thái cuối cùng của bài toán.

1.2.2 Các đặc trưng của thuật toán

- Tính xác định.
- Tính dừng (hữu hạn).
- Tính đúng đắn.

1.2.3 Đặc tả thuật toán

Đặc tả thuật toán (hay đặc tả bài toán) là định rõ lớp bài toán mà một thuật toán giải quyết, do đó nó cần chỉ ra các đặc điểm sau:

1. Các đối tượng và phương tiện của Thuật toán cần sử dụng (nhập).
2. Điều kiện ràng buộc (nếu có) trên các đối tượng và phương tiện đó.
3. Các sản phẩm, kết quả (xuất).
4. Các yêu cầu trên sản phẩm kết quả. Thường xuất hiện dưới dạng quan hệ giữa sản phẩm kết quả và các đối tượng,phương tiện sử dụng.

Ta viết:

INPUT : (1) và (2);
OUTPUT: (3) và (4);

1.2.4 Độ phức tạp của Thuật toán

Mỗi Thuật toán đều cần thời gian và các nguồn lực khác để giải quyết một bài toán cụ thể. Các tiêu hao đó đặc trưng độ phức tạp của thuật toán.
Có nhiều thuật toán cùng giải một bài toán, ta mong muốn có được thuật toán hiệu quả hơn theo nghĩa tiêu hao ít hơn một trong các loại nguồn lực.
Đối với máy tính, ta quan tâm đến thời gian và kích thước bộ nhớ mà thuật toán sử dụng.

1.2.5 Các ví dụ

Ví dụ 1.1 :

Bài toán bắp cải (BC), gà (G) và sói (S)

a. Nội dung bài toán.
Có một dòng sông với BC (Bắp cải), G (Gà) và S (Sói) đều đang ở B1 (bờ trái). Một người chèo đò (CĐ) muốn mang tất cả qua B2 (bờ phải). Đò nhỏ nên mỗi lần chỉ có thể chở thêm được 1 trong 3 đối tượng. Nhưng không thể nào để trên 1 bờ chỉ có:

- BC và G
- hoặc G và S

mà không có người chèo đò, vì như vậy G sẽ ăn BC hoặc S sẽ ăn G.

Hãy chỉ cách thực hiện (thuật toán) của người chèo đò.

b. Mô tả thuật toán.
INPUT: - CĐ,BC,G,S ở (B1)
OUTPUT: - CĐ,BC,G,S ở (B2)

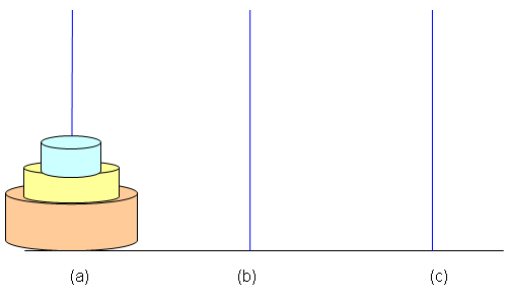
Mô tả xử lý:

TT Nhập	Đ/T Nhập	Thao tác	Đ/T Xuất	TT xuất
B1,CĐ,BC,G, S	CĐ,G	CĐ chở G sang B2	CĐ,G	CĐ, G, B2
B2, CĐ, G	CĐ	CĐ về B1	CĐ	CĐ, BC,S, B1
B1,BC,S,CĐ	CĐ,BC	CĐ chở BC sang B2	CĐ,BC	CĐ,BC,G,B2
B2,CĐ,BC,G	CĐ,BC	CĐ chở G sang B1	CĐ,G	CĐ,G,S,B1
B1,CĐ,G,S	CĐ,S	CĐ chở S sang B2	CĐ,S	BC,S,B2
B2,BC,S	CĐ	CĐ về B1	CĐ	CĐ,G,B1
B1,CĐ,G	CĐ,G	CĐ chở G sang B2	CĐ,G	CĐ,BC,S,G,B2

Ví dụ 1.2 (Bài toán Tháp Hà nội)

a. Nội dung bài toán.

Có 3 cọc (a) , (b), (c), Trên cọc (a) có 3 đĩa kích thước khác nhau: lớn (L), vừa (V), nhỏ (N), với L > V > N. Đĩa nhỏ nằm trên đĩa lớn như hình vẽ, ký hiệu N -> V -> L .



Cần chuyển cả 3 đĩa sang cột (b) với ràng buộc như sau:

- Mỗi lần chỉ chuyển được 1 đĩa từ cọc này sang cọc khác.
- Không được để đĩa lớn trên đĩa nhỏ.
- Khi chuyển đĩa từ cọc này sang cọc khác có thể dùng cọc trung gian thứ 3.

b. Mô tả thuật toán.

INPUT: - 3 đĩa có kích thước L,V,N nằm trên cọc (a).
- đĩa nhỏ nằm trên đĩa lớn.

OUTPUT: - 3 đĩa có kích thước L,V,N nằm trên cọc (b).
- đĩa nhỏ nằm trên đĩa lớn.

Mô tả xử lý:

TT Nhập	Đ/T Nhập	Thao tác	Đ/T Xuất	TT xuất
(a),L,V,N N -> V -> L	N	Chuyển N sang (b)	N	N, (b)
(a), L,V V -> L	V	Chuyển V sang (c)	V	V, (c)
(b), N	N	Chuyển N sang (c)	N	N,V,(c) N -> V

(a), L	L	Chuyển L sang (b)	L	L,(b)
(c), N,V N -> V	N	Chuyển N sang (a)	N	N,(a)
(c),V	V	Chuyển V sang (b)	V	V,L,(b) V -> L
(a), N	N	Chuyển N sang (b)	N	L,V,N, (b) N -> V -> L

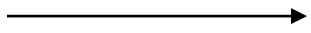
1.3 Diễn đạt thuật toán

1.3.1. Ngôn ngữ tự nhiên

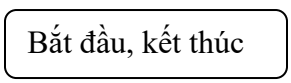
Mô tả các bước thực hiện của thuật toán dưới dạng văn bản bằng ngôn ngữ tự nhiên như tiếng Việt, Anh,. . .

1.3.2. Lưu đồ

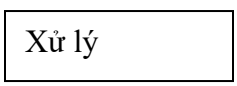
Sơ đồ toàn cảnh có cấu trúc biểu diễn các bước thực hiện của thuật toán, trong đó sử dụng các hình vẽ có quy ước sau:



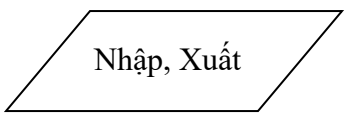
Mũi tên chỉ luồng lưu chuyển của dữ liệu.



Điểm bắt đầu, kết thúc

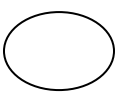


Hình chữ nhật biểu diễn các bước xử lý



Hình bình hành biểu thị nhập xuất

Hình thoi biểu thị thời điểm ra quyết định



Hình tròn biểu thị điểm ghép nối

1.3.3. Mã giả

Dựa vào cú pháp và ngữ nghĩa của một ngôn ngữ lập trình nào đó (chẳng hạn C, Pascal, ...). Cho nên mã giả cũng dựa trên các cơ sở sau đây:

- Ký tự
- Các từ
- Các xâu ký tự (Chuỗi)
- Hằng

- Biến
- Kiểu dữ liệu
- Lệnh gán
- Khối lệnh
- Các cấu trúc điều khiên
- Câu lệnh trả về
- . . .

1.3.4. Ngôn ngữ lập trình

Sử dụng các ngôn ngữ lập trình cấp cao để mổ tả, như C/C++, Pascal,… Trong trường hợp này ta đã cài đặt thuật toán theo ngôn ngữ tương ứng.

1.3.5. Các ví dụ

Ví dụ 1.3:

Thuật toán giải phương trình bậc 2 trên R:

$ax^2 + bx + c = 0$ $(a,b,c \in R; a \neq 0)$

a. Dùng ngôn ngữ tự nhiên:

a. Dạng ngôn ngữ tự nhiên:

INPUT : a, b, c; a≠0

OUTPUT : Một trong các kết quả sau

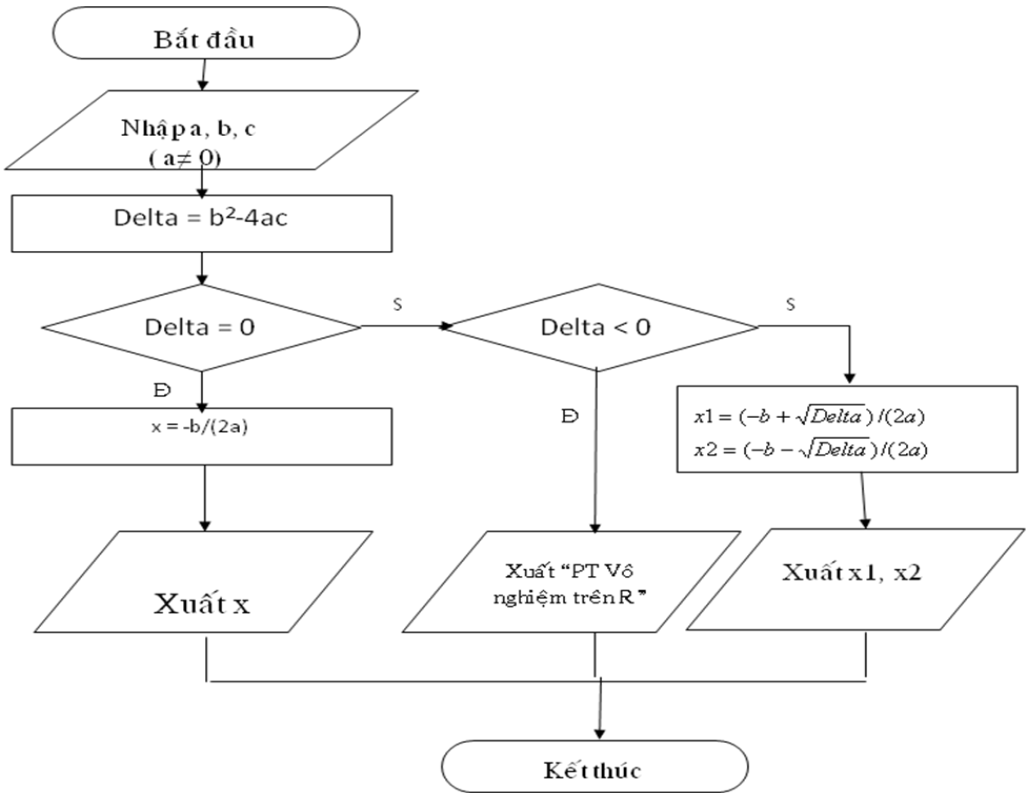
- Thông báo PT vô nghiệm trên R.
- Xuất nghiệm kép x
- Xuất 2 nghiệm phân biệt x_1, x_2

Mô tả:

- Bước 1: Tính delta $= b^2 - 4ac$;
- Bước 2: So sánh deltavà 0.
- Bước 3: Căn cứ vào kết quả bước 2, rẽ nhánh theo các trường hợp sau:
  - 3.1 Trường hợp delta < 0: Thông báo PT vô nghiệm trên R.
  - 3.2 Trường hợp delta = 0:
    - Xuất nghiệm kép
  - 3.3 Trường hợp delta > 0: $x = -\frac{b}{2a}$;
    - Xuất 2 nghiệm phân biệt:

$$x1 = \frac{-b + \sqrt{delta}}{2a}; x2 = \frac{-b - \sqrt{delta}}{2a}$$

b. Dạng lưu đồ:



Ví dụ 1.4 (Thuật toán đổi chỗ):

Hoán đổi 2 giá trị A, B cho nhau.

INPUT A,B;

OUTPUT A ,B;

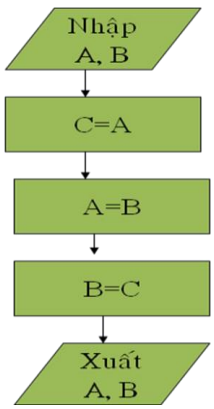
Mô tả thuật toán:

(Sử dụng thêm biến trung gian C)

C = A; // Gán giá trị của A cho C

A = B; // Gán giá trị của B cho A

B = C; // Gán giá trị của C cho B



1.4 Chương trình

1.4.1. Khái quát về chương trình.

Thuật toán phải được diễn đạt sao cho máy tính có thể hiểu và thi hành được. Ngôn ngữ lập trình (programming language) được sử dụng vào mục đích này.

Ngôn ngữ lập trình là tập hợp các qui tắc chặt chẻ về cú pháp (syntax), về ngữ nghĩa (semantic) cho phép tạo ra các văn bản để diễn đạt thuật toán **.**

Một văn bản như vậy gọi là chương trình (program).

Chương trình viết bằng ngôn ngữ lập trình cấp cao gọi là chương trình nguồn (Source program)

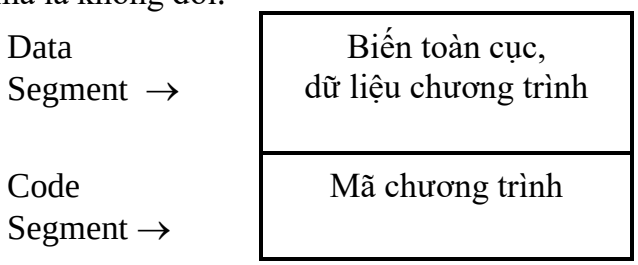
Chương trình viết bằng ngôn ngữ máy gọi là chương trình đích (Target program).

Máy chỉ có thể thi hành chương trình dưới dạng ngôn ngữ máy, vì vậy chương trình nguồn muốn được khai thác phải được chuyển đổi thành chương trình đích tương

đương. Chương trình có nhiệm vụ chuyển đổi chương trình nguồn thành chương trình đích tương đương được gọi là chương trình dịch.
Có 2 loại dịch khác nhau: biên dịch (Complier) và thông dịch (Interpreter).

1.4.2. Mã và dữ liệu

Một chương trình bất kỳ bao gồm hai phần: mã và dữ liệu.
Khi nạp một chương trình đã được dịch sang mã máy vào RAM, phần RAM chứa đoạn mã máy là các lệnh thể hiện thao tác được gọi là code segment. Trong code segment chứa phần mã của chương trình.
Phần RAM chứa các dữ liệu là đối tượng của các thao tác được gọi là data segment. Trong data segment chứa phần dữ liệu của chương trình.
Trong các lần chạy khác nhau, chỉ có phần dữ liệu của chương trình là thay đổi, còn phần mã là không đổi.



1.5 Giới thiệu C/C++

Ý tưởng quan trọng nhất của C xuất phát từ ngôn ngữ BCPL do Martin Richards thiết kế. Anh hưởng của BCPL lên C gián tiếp thông qua ngôn ngữ B do Ken Thompson viết năm 1970 cho hệ thống UNIX đầu tiên trên máy PDP-7.
Từ ngôn ngữ B, Dennish Ritchie và Brian Kernighan phát triển thành ngôn ngữ C vào những năm 1970 tại phòng thí nghiệm của hãng AT & T để phát triển cốt lõi của hệ điều hành UNIX.
Trong nhiều năm, chuẩn cho C trên thực tế là một phiên bản được cung cấp cùng với hệ điều hành Unix version 5. Nó được mô tả lần đầu tiên trong cuốn:

” The C programming language “

của Dennish Ritchie và Brian Kernighan.
Năm 1983 một hội đồng được thành lập để tạo ra một chuẩn cho C, gọi là chuẩn ANSI (American National Standards Institute : Viện định chuẩn Quốc gia Mỹ). Sau 6 năm, chuẩn cuối cùng ANSI C được đề nghị vào tháng 12/1989, và bản đầu tiên được dùng vào năm 1990. Ngày nay các trình biên dịch C chính đều giữ đúng theo ANSI chuẩn.
C là một ngôn ngữ có khả năng tổ hợp những thành phần tốt nhất của ngôn ngữ bậc cao và sự điều khiển linh hoạt của ngôn ngữ assembly .
Năm 1987 hãng Borland đã đưa ngôn ngữ C vào thị trường của IBM-PC thông qua Turbo C.
Vào những năm 1980, Bjarne Stroustrup đã cho ra đời ngôn ngữ C++ bằng cách cài vào ngôn ngữ C khái niệm lập trình hướng đối tượng.
Năm 1988, hãng Zortech giới thiệu một trình biên dịch C++ cho các máy tính MS-DOS.

Cho tới nay đã xuất hiện nhiều phiên bản C++ trong môi trường Microsoft Windows: Borland C++, Turbo C++ của hãng Borland; Visual C++ của Microsoft.
Hiện nay có nhiều chương trình lớn được viết hay được viết lại bằng C/C++:

- Hệ điều hành UNIX.
- Hệ điều hành Windows.
- Hệ quản trị cơ sở dữ liệu Dbase.
- Các chương trình soạn thảo văn bản.
- Các bảng tính điện tử. . .

1.6 Chương trình C++

1.6.1. Chương trình đầu tiên

```
//Chuong trinh C++ dau tien trong moi truong Windows
// tạo bằng Windows 32 Console Application trong MS Visual 2005
#include<iostream>
using namespace std;
int main()
{
        cout<<"Chuong trinh C++ dau tien trong moi truong Windows!\n";
        return 0;
}
```

Chương trình sẽ xuất ra màn hình chuỗi sau:
Chuong trinh C++ dau tien trong moi truong Windows!

1.6.2 Các thao tác Nhập, Xuất dữ liệu

1. Nhập dữ liệu từ bàn phím:
Mệnh đề nhập có thể mô tả như sau:

```
cin >> Bien;  //Nhập giá trị 1 biến
cin >> Bien1 >> Bien2 >> ... >> Bienn;//Nhập giá trị n biến.
```

Trong đó:

cin là đối tượng được khai báo trong <iostream>.
>> là toán tử nhập.
Biến-i là biến mà đối với loại của nó >> được định nghĩa.

2. Xuất dữ liệu ra màn hình:
Mệnh đề xuất có thể mô tả như sau:

```
cout << Bt;    // cho 1 biểu thức
cout << Bt1 << Bt2 << ... << Btn;//n biểu thức
```

Trong đó:

cout là đối tượng đã được khai báo trong <iostream>.
<< là toán tử xuất.
Biểu-Thức-i là biểu thức mà đối với kiểu của nó << được định nghĩa.

3. Định dạng hiển thị dữ liệu
Có thể chỉnh dạng kết xuất bằng cách chèn vào các xử lý dạng thức (format manipulators) từ tập tin header <iomanip>

setw (Width)	Hiển thị giá trị kế tiếp trong một vùng với độ rộng Width được chỉ ra.
setprecision (Precision)	**Hiển thị các giá trị với độ chính xác Precision được chỉ ra (mặc nhiên là 6).**
setiosflags (FlagList)	Đặt các cờ định dạng trong FlagList, trong đó FlagList là một dãy gồm một hoặc nhiều cờ, cách nhau bởi ký hiệu | như là:Flag1 | Flag2 | ... | Flagn.

Sau đây là một số các cờ định dạng sẵn:

ios::showpoint	**Hiển thị dấu chấm thập phân và các số 0 kéo theo.**
ios::fixed	Hiển thị các giá trị thực ở dạng dấu chấm cố định.
ios::scientific	**Hiển thị các giá trị thực ở dạng dấu chấm di động.**
ios::left	Hiển thị các giá trị canh trái trong một vùng.
ios::right	Hiển thị các giá trị canh phải trong một vùng.

CHƯƠNG 2. CÁC KIỂU DỮ LIỆU CƠ BẢN TRONG C++

2.1 Các Yếu Tố Cơ Bản Của Ngôn Ngữ C++

2.1.1 Ký hiệu cơ sở

Ngôn ngữ C++ được xây dựng từ bộ ký hiệu cơ sở sau:

- Bộ 26 chữ cái La-Tinh viết thường (nhỏ): a,b,...,z.
- Bộ 26 chữ cái La-Tinh viết hoa (lớn): A,B,...,Z.
- Bộ 10 chữ số hệ thập phân: 0,1,...,9.
- Bộ dấu các toán tử số học: + - * /
- Bộ dấu các toán tử so sánh: < > =
- Ký tự gạch nối: _ (Khác dấu trừ -).
- Các ký hiệu khác: ' " ; ,.: [ ] # $ & { } % ! . . .

Đặc biệt có khoảng trắng dùng để ngăn cách các từ (phím Space). Các ký hiệu cơ sở đều có trên bàn phím.

2.2.2 Các từ

Từ trong C++ được xây dựng bởi các ký hiệu cơ sở trên. Có 2 loại từ:Từ khóa và tên.

a. Từ khóa (Key Word)

Là những từ có ý nghĩa hoàn toàn xác định, chúng thường được dùng để khai báo các kiểu dữ liệu, để viết các toán tử, và các câu lệnh. Sau đây là các từ khóa trong Borland C++ (Turbo C++):

Asm	auto	break	Case	catch	char
class	const	continue	Default	delete	do
double	else	enum	Extern	float	for
friend	goto	if	Inline	int	long
new	operator	private	Protected	public	register
return	short	signed	Sizeof	static	struct
switch	template	this	Throw	try	typedef
union	unsigned	virtual	Void	volatile	while
cdecl	_cs	_ds	_es	_export	far
huge	interrupt	_loadds	Near	pascal	_regparam
_saveregs	_seg	_ss			

b. Tên hoặc danh hiệu (identifier):

Là từ do người sử dụng tự đặt để giải quyết bài toán của mình. Từ tự đặt dùng để đặt tên cho hằng, biến, hàm, tên kiểu dữ liệu mới,...

Tên được đặt theo quy tắc: phải bắt đầu bằng một chữ cái hoặc dấu gạch nối,sau đó là các chữ cái, chữ số hoặc dấu gạch nối, và không được trùng với từ khóa.

Tên có thể viết bằng chữ thường hoặc chữ hoa. Trong C++ có phân biệt chữ thường và chữ hoa.

2.3 Các Kiểu Dữ Liệu Cơ Bản Trong C++

Trong C++ có 5 kiểu dữ liệu cơ bản là:

- Các kiểu Ký tự.
- Các kiểu nguyên.
- Kiểu Số thực dấu chấm động độ chính xác đơn
- Kiểu Số thực dấu chấm động độ chính xác kép
- Kiểu void.

Các kiểu dữ liệu khác đều dựa vào các kiểu dữ liệu trên.

2.3.1 Ký tự

a. Ký tự 8 bit

Một giá trị ký tự có kiểu dữ liệu khai báo bằng từ khóa char, được lưu trử trong 8 bit và biểu diễn thông qua bảng mã ASCII .

Chẳng hạn:

Ký tự	Mã ASCII (hệ 10)
0	48
1	49
A	65
a	97
. . . .	

Có các kiểu ký tự 8 bit tương ứng với các từ khóa :

signed char (như char, có dấu)

unsigned char (Không dấu).

Sau đây là bảng kích thước, phạm vi biểu diễn của các kiểu ký tự:

KIỂU	Phạm vi biểu diễn	Kích thước	Số ký tự
Char	-128 $\rightarrow$ 127	1 byte	256
signed char	-128 $\rightarrow$ 127	1 byte	256
unsigned char	0 $\rightarrow$ 255	1 byte	256

b. Ký tự UNICODE

Các ký tự Unicode trong C/C++ được định nghĩa bởi: wchar_t

Mỗi ký tự Unicode rộng 16 bit.

c. Kiểu TCHAR

Dùng chung cho char và wchar_t (cho ký tự 8 bit hay unicode 16 bit).

2.3.2 Kiểu nguyên

Trong C++ cho phép sử dụng các kiểu số nguyên được khai báo bởi từ khóa int, hoặc đi kèm theo int với các từ khóa long, short, unsigned.

Số lượng bit được dùng để lưu trử một giá trị int phụ thuộc vào kích thước từ (word) của máy. Thường thì máy 16-bit sẽ dùng 16 bit để lưu trử một giá trị int , trong khi đó máy 32-bit sẽ dùng 32 bit .

Kích thước và phạm vi biểu diễn của chúng được cho trong bảng sau:

Kiểu	Phạm vi biểu diễn	Kích thước
Int	Chiếm 1 từ của máy	
short int , short	-32768 $\rightarrow$ 32767 ($-2^{15} \rightarrow 2^{15}-1$)	16 bit

unsigned short int	$0 \rightarrow 65535$ $(0 \rightarrow 2^{16}-1)$	16 bit
long int , long	-2147483648 $\rightarrow$ 2147483647 $(-2^{31} \rightarrow 2^{31}-1)$	32 bit
unsigned long int	$0 \rightarrow 4294967295$ $(0 \rightarrow 2^{32}-1)$	32 bit
unsigned int	Số nguyên không âm , chiếm 1 từ của máy.	

2.3.3 Kiểu số thực

C++ cho phép sử dụng 3 kích thước giá trị thực, tương ứng với 3 từ khóa:

- float
- double
- long double

Kích thước và phạm vi biểu diễn của chúng được cho trong bảng sau:

Kiểu	Ý nghĩa	Phạm vi biểu diễn	Kích thước	Độ chính xác
float	Số thực chính xác đơn	-3.4E+38 $\rightarrow$ 3.4E+38	32 bit	6 số thập phân
double	Số thực chính xác kép	-1.7E+308 $\rightarrow$ 1.7E+308,0	64 bit	10 số thập phân
long double		Kích thước 96 bit hoặc	128 bit	

2.4 Các Hằng

2.4.1. Định nghĩa

Hằng là các đại lượng mà giá trị của nó không thay đổi trong quá trình tính toán. Ta thường dùng các ký tự hoa để biểu diễn các hằng ký hiệu.

2.4.2 Các loại hằng

Trong C++ có các loại hằng sau đây:

1. Hằng số thực

Giá trị được lấy là float và double. Viết theo 2 cách:

a. Dạng thập phân (dấu chấm cố định):

Bao gồm: Phần nguyên, dấu chấm thập phân, phần phân.

Phần nguyên	•	Phần phân

b. Dạng khoa học hay dạng mũ (dấu chấm động)

Có 3 thành phần: Phần định trị , ký hiệu E hoặc e , và phần bậc.

Phần định trị là một số nguyên hoặc số thực dạng thập phân. Phần bậc là một số nguyên. Hai phần này cách nhau bởi ký tự E hoặc e.

Phần định trị	E hoặc e	Phần bậc

Ví dụ 2.1:

```
12.234E-3        // biểu diễn giá trị 0.012234
0.35E4           // biểu diễn giá trị 3500.0
```

-12.22e-3 // biểu diễn giá trị -0.01222
1e6 // biểu diễn giá trị 1 000 000

2. Hằng nguyên

a. Hằng int

Là số nguyên có kiểu int .

b. Hằng long

Biểu diễn: thêm L hoặc l vào sau hằng int.

c. Hằng unsigned

- Biểu diễn: thêm u vào sau hằng int.
- Có giá trị từ 0 đến 65535.

d. Hằng int hệ 8

- Hằng int hệ 8 luôn nhận giá trị dương.
- Dạng biểu diễn: $0c_1c_2c_3$ …
  Với: c_i là một số nguyên trong khoảng từ 0 đến 7.

Ví dụ 2.2:

- Hằng int hệ 8: 0345
- Giá trị của nó trong hệ 10: 3*8*8 + 4*8 + 5 = 229.

e. Hằng int hệ 16:

Trong hệ này sử dụng 16 ký tự:

0,1,2,3,4,5,6,7,8,9,
a hoặc A
b hoặc B
c hoặc C
d hoặc D
e hoặc E
f hoặc F

Dạng biểu diễn: 0x $c_1c_2c_3$ … hoặc 0X $c_1c_2c_3$ …

Trong đó c_i là một chữ số hệ 16.

Ví dụ 2.3:

Các hằng nguyên hệ 16 : 0xa3 ; 0Xa3 ; 0xA3 ; 0XA3 là như nhau.
Giá trị của nó trong hệ 10 là: 10*16 + 3 = 163.

2.4.3 Hằng ký tự

Là một ký tự được viết trong 2 dấu nháy đơn.

Chẳng hạn ‘a’ , ‘A’, ‘3’, ‘+’ . . . Trình biên dịch của C++ sẽ lưu trữ các hằng này bằng cách dùng các mã số ASCII (hệ 10) của nó, tương ứng là 97,65, 43, 51 . . .

- Ký tự có thể biểu diễn bởi hệ 8:
  Cách viết: ‘\$c_1c_2c_3$’ , trong đó c_i là các ký hiệu trong hệ 8.
- Ký tự có thể biểu diễn bởi hệ 16:
  Cách viết: ‘\x$c_1c_2c_3$’ hoặc ‘\X$c_1c_2c_3$’ trong đó c_i là các ký hiệu trong hệ 16.

Ký tự	Hệ 8		Hệ 16		Mã ASCII hệ 10
	Mã ASCII	Biểu diễn	Mã ASCII	Biểu diễn	
‘a’	141	‘\141’	61	‘\x61’	97
‘A’	101	‘\101’	41	‘\x41’	65

- Đối với một số hằng ký tự đặc biệt, ta sử dụng cách viết sau (thêm dấu \):

Cách viết	Ký tự
'\' '	'
'\" '	"
'\\'	\
'\n'	\n (chuyển dòng)
'\0'	\0 (NULL)
'\t'	Tab
'\b'	Backspace
'\r'	CR (Về đầu
'\f'	LF (sang trang)

Ghi chú:

- Hằng ký tự có thể tham gia vào các biểu thức như mọi số nguyên khác.

Ví dụ: '9' -'0' = 57 -48 = 9

- Cần phân biệt:

'0' : là ký số 0 có mã ASCII hệ 10 là 48.
'\0': là ký tự NULL có mã ASCII hệ 10 là 0.

2.4.4 Hằng xâu ký tự (Chuỗi)

Là một dãy ký tự được bao trong 2 dấu nháy kép.
Ví dụ 2.4:

"Da Lat"
"" // Xâu rỗng

Ghi chú:
Xâu ký tự được lưu trử trong máy dưới dạng một mảng các ký tự. Trình biên dịch tự động thêm ký tự NULL '\0' (được xem là dấu hiệu kết thúc xâu) vào cuối mỗi xâu.

2.4.5 Biểu thức hằng

Biểu thức hằng chỉ bao gồm các hằng. Các biểu thức như vậy được xác định vào lúc biên dịch.

2.4.6 Định nghĩa một hằng

- Dùng chỉ thị #define (Có thể định nghĩa lại giá trị hằng):
  - Cách viết: #define TÊN-HẰNG GIÁ_TRỊ_HẰNG
  - Tác dụng: TÊN-HẰNG sẽ được thay thế bởi GIÁ_TRỊ_HẰNG cho phần còn lại của văn bản chương trình.

Ví dụ 2.5:

#define MAX 100 // Thay thế MAX bằng 100

- Dùng từ khóa const (Không định nghĩa lại được giá trị hằng)
  - Cú pháp: Const kiểu TÊN-HẰNG = GIÁ_TRỊ_HẰNG;

- Tác dụng: Cho phép định nghĩa một hằng ký hiệu có tên là TÊN-HẰNG biểu thị một gia trị là GIÁ_TRỊ_HẰNG và sau này không thể sửa đổi GIÁ_TRỊ_HẰNG của TÊN-HẰNG được .

Ví dụ 2.6:

Const double PI = 3.1416;

2.5 Biến

2.5.1 Định nghĩa

Biến là một phần bộ nhớ được đặt tên, được dùng, để giữ một giá trị mà có thể thay đổi trong chương trình.
Vậy biến gắn với tên và kiểu dữ liệu, có giá trị thay đổi trong quá trình tính toán.

2.5.2 Khai báo biến

Mỗi biến phải được khai báo trước khi sử dụng. Cú pháp khai báo như sau:

- Khai báo một biến:
  Kdl Bien;
- Khai báo nhiều biến cùng một kiểu:
  Kdl Bien1, Bien2, Bien3;

Trong đó:

- Kdl là kiểu dữ liệu nào đó như char, int, double,...
- Bien, Bien1,... là tên chỉ tên của biến.

Giữa Kdl và Tên biến phải cách nhau ít nhất 1 khoảng trắng. Trong phần tên biến, nếu có nhiều biến thì giữa 2 biến phải tách ra bởi dấu phảy (,).
Ví dụ 2.7:

- Khai báo các biến nguyên kiểu int:
  int a, b, c ;
- Khai báo các biến thực kiểu double:
  double x, y ;
- Khai báo một biến ký tự:
  char Kt ;

2.5.3 Khởi đầu cho các biến

Nếu trong khai báo, ngay sau tên biến ta đặt dấu = (phép gán) và một giá trị dữ liệu tương ứng thì đó chính là cách vừa khai báo vừa khởi đầu cho 1 biến.
Ví dụ 2.8:

- Khởi đầu biến ký tự
  char Kt = ‘c’ ;
- Khởi đầu biến thực x:
  double t, x = 3.2, y ;

2.5.4 Lấy địa chỉ cho biến

Mỗi biến được cấp phát một vùng nhớ gồm một số byte liên tiếp. Số hiệu của byte đầu chính là địa chỉ của biến.
Để nhận địa chỉ biến ta dùng toán tử & với cú pháp: &Bien.

2.6 Câu lệnh gán

Có dạng:

```
b = bt;
```

Trong đó b là biến. bt là một biểu thức (một công thức toán học nào đó). Trước tiên tính biểu thức bt và sau đó gán giá trị tính được cho biến b.

Ví dụ 2.9:

```
int x ;
x = 3 ; //gán 3 cho x, khi đó x mang giá trị là 3
```

2.7 Từ khóa: typedef

Từ khóa typedef được dùng để đổi lại tên một kiểu dữ liệu đã có như char, int, float, mảng... thành một tên mới.

Cách viết là đặt từ khóa typedef vào trước 1 khai báo thông thường.

Ví dụ 2.10:

```
typedef int intgia ; //đổi tên int thành ingia
typedef float fl; // đổi float thành fl
```

CHƯƠNG 3. CÁC TOÁN TỬ TRONG C++

Trong C++ có các loại toán tử: phép gán, các toán tử số học, các toán tử quan hệ và logic, các toán tử thao tác trên Bit...

3.1 Các toán tử số học

C++ có 8 toán tử số học:

Toán tử	Ý nghĩa	Ví dụ	Ghi chú
-	Lấy đối	-a ; -(a+b)	Toán tử 1 ngôi
--	Tự giảm dần	--x	Toán tử 1 ngôi
++	Tự tăng dần	++x	Toán tử 1 ngôi
+	Cộng	a + b	Toán tử 2 ngôi
-	Trừ	a –b	Toán tử 2 ngôi
*	Nhân	a * b	Toán tử 2 ngôi
/	Chia	a /b	Toán tử 2 ngôi
%	Chia lấy phần dư	a%b	Toán tử 2 ngôi

3.1.1 Phép chia /

Là toán tử 2 ngôi, toán hạng là các số (thực hay nguyên).

Kết quả của phép chia:

- Nếu cả 2 toán hạng đều là số nguyên thì kết quả là là số nguyên (chặt cụt phần phân).
- Nếu có ít nhất 1 toán hạng là số thực thì kết quả một số thực .

Ví dụ 3.1:

```
int x = 3;
float y = 3.;
```

Kết quả của x/2 là 1 (số nguyên).

Kết quả của y/2 là 1.5 (số thực).

3.1.2 Phép chia nguyên lấy phần dư %

Là toán tử 2 ngôi, toán hạng là số nguyên , kết quả cho ra số nguyên.

Ví dụ 3.2:

- 13 % 3 = 1
- -13 % 3 = -1

Ghi chú:

Kết quả luôn cùng dấu với số bị chia.

3.1.3 Toán tử tăng ++

Là toán tử 1 ngôi, toán hạng là nguyên hay thực .

- Dạng viết 1: ++ n ;
  Tác dụng: n tăng thêm 1 trước khi sử dụng giá trị n.
- Dạng viết 2: n++ ;
  Tác dụng: n tăng thêm 1 sau khi sử dụng giá trị n.

Ví dụ 3.3:

n = 5;			
x = n++ ;	//x = n; //n = n+1;	x = ++n ;	//n = n+1; //x = n;
Kết quả: n = 6; x = 5		Kết quả: n = 6; x = 6	

3.1.4 Toán tử giảm --

Là toán tử 1 ngôi, toán hạng là nguyên hay thực .

- Dạng viết 1: -- n ;
  Tác dụng: n giảm bớt 1 trước khi sử dụng giá trị n.
- Dạng viết 2: n-- ;
  Tác dụng: n giảm bớt 1 sau khi sử dụng giá trị n.

Ví dụ 3.4:

n = 5;			
x = n-- ;	//x = n; //n = n-1;	x = --n ;	//n = n-1; //x = n;
Kết quả: n = 4; x = 5		Kết quả: n = 4; x = 4	

3.1.5 Thứ tự ưu tiên các toán tử số học

Ưu tiên của các toán tử số học được cho trong bảng sau đây theo thứ tự từ trên xuống dưới. Các toán tử cùng độ ưu tiên sẽ được thực hiện từ trên trái sang phải.

Ưu tiên	Toán tử
1	++ -- -
2	* / %
3	+ -

Ghi chú:

Các ký hiệu trong các toán tử ++, -- không được viết rời nhau.

3.2 Toán tử quan hệ và logic

Các toán tử quan hệ và logic thường được sử dụng chung với nhau, được dùng để tạo ra các kết quả đúng, sai. Trong C++, mọi số khác 0 đều được coi là giá trị đúng (true), giá trị duy nhất sai (false) mang hình thức số 0.

3.2.1 Các toán tử quan hệ

Các toán tử quan hệ được dùng để so sánh 2 giá trị với nhau.
Sau đây là bảng các toán tử quan hệ và ý nghĩa của chúng:

Toán tử	Ý nghĩa	Ví dụ	
>	Lớn hơn	a > b	3>7 có giá trị 0
>=	Lớn hơn hay bằng	a >= b	3 >= 7 có gía trị 0
<	Nhỏ hơn	a < b	3<7 có giá trị 1
<=	Nhỏ hơn hay bằng	a <= b	3 <= 7 có gía trị 1
==	Bằng nhau	a == b	3 == 7 có giá trị 0
!=	Khác nhau	a != b	3 != 7 có giá trị 1

Ghi chú:

Các ký hiệu trong các toán tử >=,<=, ==, != không được viết rời.

3.2.2 Các toán tử logic

Các toán tử logic được dùng để nối kết hai giá trị, hoặc trong trường hợp phủ định sẽ tạo một giá trị đảo ngược. Các giá trị có thể nguyên hay thực.
Trong C++ có 3 toán tử logic:

- Phép phủ định 1 ngôi: !
- Phép và (AND): &&
- - Phép hoặc (OR): ||

Ý nghĩa của các toán tử được cho trong bảng sau:

A	B	!a	a&&b	a||b
Khác không (1)	Khác không (1)	0	1	1
Khác không (1)	Bằng không	0	0	1
Bằng không	Khác không (1)	1	0	1
Bằng không	Bằng không	1	0	0

3.2.3 Thứ tự ưu tiên của các toán tử quan hệ và logic

Ưu tiên	Toán tử
1	!
2	> >= < <=
3	== !=
4	&& ||

3.3 Các Toán Tử Thao Tác Bit

Khác với các ngôn ngữ lập trình cấp cao khác, C++ cung cấp nhiều toán tử có thể tác động tới những bit thực sự ở bên trong 1 biến.
Các toán tử thao tác bit chỉ dùng cho số nguyên hoặc ký tự.
Các toán tử thao tác bit bao gồm:

Toán tử	Ý nghĩa	Ví dụ
&	Phép và (AND) theo bit	a&b
|	Phép hoặc (OR) theo bit	a|b
^	Phép hoặc loại trừ (XOR) theo bit	a^b
<<	Dịch trái (Shift left)	a << 4
>>	Dịch phải (Shift right)	a>>4
~	Lấy phần bù theo bit (Not)	~a

Ghi chú:

Các ký hiệu trong các toán tử >>,<< không được viết rời.

3.3.1 Bảng giá trị của các toán tử ~, &, |, ^

a	b	~a	a&b	a|b	a^b
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0

3.3.2 Các toán tử dịch chuyển:

Các toán tử dịch chuyển trái << , hoặc phải >> đều có thể dịch chuyển tất cả các bit nằm trong 1 byte hay 1 từ - theo một số lượng ấn định ở bên phải .
Chẳng hạn:

n<<2	Dịch chuyển các bit của biến n sang trái 2 bit
n>>2	Dịch chuyển các bit của biến n sang phải 2 bit

1. Toán tử dịch chuyển trái:

Khi thực hiện toán tử dịch chuyển trái các bit ở bên phải của biến sẽ được điền vào các giá trị 0.
Ví dụ 3.5:
short int n = 201;
n << 2;
Đổi n sang hệ 2:

Hệ 2	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1

Dịch chuyển trái 2 bit:

Hệ 2	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	0

Đổi n sang hệ 10: 804

2. Toán tử dịch chuyển phải:

C++ sẽ thực hiện tùy theo kiểu dữ liệu của toán hạng bên trái: có kiểu unsigned (int, long, char) hay có kiểu signed (int, long, char).
a. Toán hạng bên trái có kiểu unsigned (int, long, char).

Phép dịch chuyển phải sẽ điền giá trị 0 vào các bit bên trái.
Ví dụ 3.6:

unsigned short int n = 39470; //hệ 10;
n = n >> 2;

- Đổi n sang hệ 2: 1001 1010 0010 1110
- Dịch chuyển trái 2 bit: 0010 0110 1000 1011
- Đổi n sang hệ 10: 9867

b. Toán hạng bên trái có kiểu signed (int, long, char).

Phép dịch chuyển phải sẽ điền bit dấu (số 1) vào các bit bên trái.

Ví dụ 3.7:

short int n = 26066; //hệ 10;

n = n >> 2;

- Đổi n sang hệ 2: 0110 0101 1101 0010
- Dịch chuyển phải 2 bit: 1111 1001 0111 0100
- Đổi n sang hệ 10: 6516

Ghi chú:

Thực chất phép dịch trái là hình thức nhân 2 liên tiếp , còn phép dịch phải là hình thức chia 2 liên tiếp.

Tức là: $x = a << n \equiv x = a*(2^n)$

$x = a >> n \equiv x = a/(2^n)$

3.4 Các toán tử khác

3.4.1 Toán tử sizeof

Toán tử sizeof cho ta kích thước (tính theo byte) của 1 kiểu dữ liệu cũng như một đối tượng dữ liệu. Cách viết toán tử như sau:

- sizeof (kiểu dữ liệu)
- sizeof (đối tượng dữ liệu)

Kiểu dữ liệu có thể là các kiểu chuẩn như int, float hoặc kiểu dữ liệu được định nghĩa bằng từ khóa typedef.

Đối tượng dữ liệu có thể là biến, mảng, cấu trúc,...(tên của vùng nhớ dữ liệu).

Ví dụ 3.8:

- int m;
  sizeof(m) = 4;
- sizeof(int) = 4;
- sizeof(double) = 8;

3.4.2 Toán tử ( )

Dùng để xác định trình tự ưu tiên các thành phần trong biểu thức.

Nếu có nhiều toán tử () lồng nhau thì thực hiện ưu tiên từ trong ra ngoài.

Nếu có nhiều toán tử () rời nhau thì thực hiện từ trái sang phải.

3.4.3 Toán tử dấu phảy ‘ , ‘ (Comma operator)

Được dùng để tạo sự thi hành tuần tự cho các thao tác, thường dùng trong câu lệnh for hay biểu thức vế phải của câu lệnh gán.

Trong vế phải câu lệnh gán, thì giá trị toàn thể biểu thức là giá trị của biểu thức cuối cùng trong danh sách các biểu thức được tách biệt bởi dấu phảy.

3.5 Biểu Thức

Biểu thức là một sự kết hợp giữa các toán tử và các toán hạng để diễn đạt một công thức toán học nào đó. Toán hạng có thể xem là các đại lượng có một giá trị nào đó, và theo nghĩa đó, toán hạng có thể là biến, hằng, hàm...

Mỗi biểu thức có một giá trị, và nói chung cái gì có giá trị đều được xem là biểu thức. Như vậy, các biến, hằng, hàm... đều có thể xem là biểu thức.

Khi viết biểu thức có thể dùng các dấu ngoặc tròn (,) để thể hiện đúng trình tự tính toán trong biểu thức.

Biểu thức được phân loại theo kiểu giá trị: nguyên, thực. Trong các mệnh đề logic, biểu thức được phân thành đúng (giá trị khác không), sai (giá trị bằng không).

Biểu thức thường được dùng trong:

- Vế phải của lệnh gán.
- Làm tham số thực sự của hàm.
- Làm chỉ số. . .

Ta giới thiệu các biểu thức: biểu thức gán, biểu thức điều kiện.

3.5.1 Biểu thức gán

a) Một dạng viết khác trong câu câu lệnh gán: op=

Trong đó "op" là toán tử hai ngôi (+, -, *, /, %, &, |, <<, >>, ^)

Nếu e1,e2 là biểu thức thì e1 op= e2 tương đương với e1 = e1 op e2 nhưng cách viết trước hiệu quả hơn.

Ví dụ 3.9:

n = n+1;	Có thể viết	n +=1;
x = x*(y+3)	Có thể viết	x *= y+3
a = a/(x-1)	Có thể viết	a /=x-1;

b) Biểu thức gán là biểu thức có dạng: v = e

Trong đó v là biến, e là một biểu thức. Giá trị của biểu thức gán là giá trị của e, kiểu của nó là kiểu của v, (nếu viết v = e; ta có câu lệnh gán).

Biểu thức gán có thể sử dụng trong các toán tử và các câu lệnh như các biểu thức khác.

Ví dụ 3.10:

```
int x = 10;
x = (x = x-5, 25/x);
```

Giá trị đầu của x là 10.

Vế trái, biểu thức gán đầu tiên thực hiện gán x-5 cho x, vậy giá trị biểu thức gán x = x-5 này là 5. Thực hiện biểu thức cuối 25/x, lấy 25 chia cho 5, kết quả cuối cùng là 5 đem gán cho vế trái là x.

Nên x = 5.

3.5.2 Toán tử tam phân ?: (thể hiện biểu thức điều kiện)

Biểu thức điều kiện thể hiện bởi toán tử tam phân có dạng:

e1 ? e2: e3

Trong đó, e1, e2, e3 là các biểu thức nào đó.

$$\textbf{Giá trị của biểu thức điều kiện bằng:}\begin{cases}\textbf{Giá trị của e2 nếu e1 khác 0 (Đúng),}\\ \textbf{Giá trị của e3 nếu e1 bằng 0 (Sai).}\end{cases}$$

Kiểu của biểu thức điều kiện phải là kiểu cao nhất trong các kiểu của e2 và e3. Chẳng hạn nếu kiểu của e2 là int, kiểu của e3 là float thì kiểu của biểu thức điều kiện là float.

Biểu thức điều kiện có thể sử dụng như một biểu thức bất kỳ khác.

Ví dụ 3.11:

- (a >= b) ? a: b ≡ Max(a,b)
- (a <= b) ? a: b ≡ Min(a,b)
- (a >= 0) ? a: -a ≡ | a |

3.5.3. Chuyển đổi kiểu dữ liệu

a) Trong biểu thức:

Khi các toán hạng có kiểu khác nhau xuất hiện trong biểu thức, chúng được chuyển một cách tự động về một kiểu chung theo một số qui tắc.

- Trong biểu thức có chứa các biến kiểu char và int thì tất cả các biến kiểu char được chuyển thành int.
- Các toán tử số học có hai toán hạng (+,-,*,/,%) và nếu chúng khác nhau thì kiểu "thấp hơn" sẽ nâng thành kiểu "cao hơn".

Cụ thể:

- char và short chuyển thành int
- float chuyển thành double
- int chuyển thành foat, double . . .

b) Trong câu lệnh gán

Các phép chuyển kiểu giá trị cũng được thực hiện trong phép gán, giá trị của vế phải được chuyển đổi thành kiểu của vế trái (Kiểu kết quả là kiểu của vế trái):

- int có thể chuyển thành float.
- float chuyển thành int bằng cách cắt bỏ phần phân.
- double chuyển thành float bằng cách làm tròn.
- long int chuyển thành short int bằng cách bỏ các bit cao vượt quá

Ví dụ 3.12:

Câu lệnh	Tác dụng
char c, h; short int x, y; float f;	
c = x;	Các bit cao bên trái của biến nguyên x bị chặt bỏ để lại c với 8 bit thấp. Nếu đã có 0 <= x <= 256 thì x và ch có cùng giá trị, ngược lại, giá trị của ch chỉ phản ánh các bit thấp của x.
x = f;	x chỉ nhận phần nguyên của f.
f = h;	f chuyển giá trị nguyên 8 bit được lưu trong h thành giá trị như vậy trong dấu chấm động.
f = y;	f chuyển giá trị nguyên 16 bit được lưu trong y thành giá trị như vậy trong dấu chấm động.

c) Trong việc truyền tham số cho hàm

Vấn đề chuyển đổi kiểu cũng xảy ra khi truyền tham số cho hàm.

d) Sử dụng phép ép kiểu:

Việc chuyển kiểu tường minh (hay gọi là cast) được thực hiện theo cú pháp:

(Kdl) biểu_thức ;

Cast (sắc thái) tạo giá trị của biến theo kiểu đúng nhưng nội dung thực sự của biến là không thay đổi.

Ví dụ 3.13:

```
int  p = 3, q = 2, r;
float x, y;
```

```
x = p/q;
y = (float)p/q;
r = (float)p/q;
```
Khi đó: x =1 và y = 1.5, r =1.

3.6 Độ ưu tiên của các toán tử

Được trình bày trong bảng sau:

Ưu tiên	Toán tử	Trình tự kết hợp	Ghi chú
1	() [ ] ->	Trái qua phải	
2	! ~ & * - ++ -- (type) sizeof	Phải qua trái	*: con trỏ ; & lấy địa chỉ.
3	* / %	Trái qua phải	*: Phép nhân
4	+ -	Trái qua phải	-: PT 2ngôi
5	<< >>	Trái qua phải	
6	< <= > >=	Trái qua phải	
7	== !=	Trái qua phải	
8	&	Trái qua phải	&: AND theo bit
9	^	Trái qua phải	
10	|	Trái qua phải	
11	&&	Trái qua phải	
12	||	Trái qua phải	
13	?:	Phải qua trái	
14	Op=	Phải qua trái	
15	,	Trái qua phải	

Ghi chú:

- Các toán tử cùng mức ưu tiên (nằm trêm 1 dòng) thì trình tự tính toán được chỉ ra trong cột “trình tự kết hợp”.
- Một số các toan tử chưa được giới thiệu, chẳng hạn:
- Trong ưu tiên 1:
  [ ]: dùng đẻ biểu diễn phần tử mảng.
  ->: Biểu diễn thành phần của cấu trúc thông qua con trỏ.
- Trong ưu tiên 2:
  *: khai báo con trỏ.

BÀI TẬP

Bài 1:

Hãy cho biết giá trị của j sau đoạn chương trình:

```
int j;
char c='1';
j=(c<='9') &&(c>='0');
```

Bài 2:

Cho *b* bằng 5 và *c* bằng 8. Hãy cho biết giá trị của *a,b,c* sau khi thi hành riêng biệt từng dòng lệnh sau:

1. *a=b++ + c++;*
2. *a=b++ + ++c;*
3. *a=++b + c++;*
4. *a=++b + ++c;*
5. *b=a++ + ++a;*
6. *a += a+= a*

Bài 3:

Các câu lệnh sau làm gì?

```
a^ =b;
b^=a;
a^=b;
a ^= b^= a^= b;
```

Bài 4:

1. Hãy cho biết giá trị của b và a sau đoạn trình:

```
int a,b=2;
b=(a=3,(5*b)+(a*=b));
```

2. Hãy cho biết giá trị của n và x sau đoạn trình:

```
int n,x=2;
x=x-1;
n=(n=5,n*=10+x++);
```

3. Hãy cho biết giá trị của a, b sau tùng câu lệnh:

```
int a=1,b= a ? 1:2;
b+=1;
a=(b==2)?1:2;
a=(b=2)?1:3;
a=(b=2)?1:2;
```

*Bài 5***:**

Viết hàm tính 2n

CHƯƠNG 4. HÀM VÀ CHƯƠNG TRÌNH

4.1 Cấu trúc chung của chương trình C++

Một chương trình C++ là một đề án (dự án, project) có thể gồm nhiều tập tin chương trình nguồn, mỗi tập tin chương trình là một văn bản chứa một dãy các chỉ thị và các chỉ thị điều khiển biên dịch. Các chỉ thị được phân thành 2 loại:

- Chỉ thị kiểu: Gồm định nghĩa các kiểu dữ liệu mới, biến, hằng và hàm.
- Chỉ thị thực hiện (câu lệnh): Được định nghĩa bằng những phép toán hay việc xử lý thực hiện trên các biến của chương trình.

Tất cả các chỉ thị đều phải kết thúc bằng dấu ; (chấm phảy).

Cả 2 loại chỉ thị này có thể hợp với nhau bằng một cú pháp qui định để hình thành một chỉ thị duy nhất được gọi là khối lệnh. Một khối lệnh được đặt trong cặp dấu ngoặc nhọn: { các chỉ thị }

4.1.1 Sơ đồ tổng quát của chương trình C++

Dạng thức tổng quát của một đề án chỉ có một tập tin chương trình:

```
// Các chỉ thị điều khiển biên dịch
```

```
// Các định nghĩa toàn cục
```

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

```
// Hàm main
int main() // void main()
{
// dãy tuần tự các lệnh
}
```

```
//Phần định nghĩa hàm.
KDL  Ham(Danh_Sach_Cac_Doi)
{
// dãy tuần tự các lệnh
}
. . . . .
```

4.1.2 Một số quy tắc cần nhớ khi viết chương trình

- Quy tắc 1: Mỗi dòng có thể viết 1 hay nhiều chỉ thị.
- Quy tắc 2: Mỗi chỉ thị phải kết thúc bằng dấu chấm phảy (;).
- Quy tắc 3: Quy tắc viết lời giải thích. Các lời giải thích viết:
  - Trên nhiều dòng, một dòng hoặc trên 1 phần của dòng phải đặt vào giữa các dấu /* và */.
  - Trên một dòng hoặc trên phần còn lại của một dòng phải đặt sau //

  Các lời giải thích được trình biên dịch bỏ qua.
- Quy tắc 4: Quy tắc sử dụng các hàm chuẩn:
  Trước khi sử dụng một hàm chuẩn nào cần phải biết nó nằm trong tập tin thư viện nào của C++ để khai báo, và khai báo bằng chỉ thị biên dịch #include:

  #include <Ten_Tap_Tin> // không có dấu chấm phảy như câu lệnh

  Chẳng hạn, khi sử dụng hàm cout, cin. Vì các hàm này nằm trong thư viện vào ra chuẩn iostream, nên ta cần khai báo:

#include < iostream >

4.2 Hàm.

C/ C++ đưa ra khái niệm hàm và trở thành công cụ mạnh mẽ để hỗ trợ cho phương pháp lập trình có cấu trúc. Hàm có thể xem là một đơn vị độc lập của chương trình.
Các hàm có vai trò ngang nhau, vì vậy không cho phép xây dựng một hàm bên trong các hàm khác.
Các hàm thường che dấu những chi tiết thực hiện đối với các phần khác trong chương trình, do đó làm chương trình sáng sủa, dễ sửa đổi.
Một hàm có thể là:

- Nằm ngay trong modul văn bản (có các khai báo,các lệnh có trong hàm) hoặc được đưa một cách tự động vào văn bản của chương trình (bằng đường dẫn #include) hay được dịch riêng rẽ (sẽ được nối kết vào chương trình trong giai đoạn liên kết)
- Được gọi từ chương trình chính, hoặc từ một hàm khác, hoặc từ chính nó (đệ quy).
- Có hay không có đối (tham số hình thức).
- Có hay không có giá trị trả về. . .

Một hàm:

- Chỉ có 1 đầu vào ({).
- Có thể có nhiều điểm ra (return hay }).

4.2.1 Cấu trúc của 1 hàm

Mọi hàm đều có dạng:

```
[Kdl]  Ten_Ham ([danh_sách_kiểu_và_Đối]) // Dòng tiêu đề
{
        // Các chỉ thị về kiểu
        // Các câu lệnh;
        [return [biểu_thức];]
}
```

Trong đó:
a) Dòng tiêu đề:
Chứa các thông tin:

- Kdl: Kiểu dữ liệu của hàm,
- Ten_Ham: Tên của hàm, kiểu và tên đối.
- Danh_sách_kiểu_và_Đối: Là các đối và kiểu tương ứng của nó.

Cuối dòng tiêu đề không có dấu chấm phảy (;) như kết thúc câu lệnh.
1. Kdl: Có thể có hoặc không.
- Trương hợp không:
Không có kiểu dữ liệu của hàm, Một cách ngầm định C++ coi đó là kiểu int.
- Trường hợp có:
Kdl là kiểu dữ liệu của hàm, có thể là bất kỳ kiểu dữ liệu nào ngoại trừ mảng. C++ sẽ dùng từ khóa void để chỉ kiểu dữ liệu của hàm trong trường hợp hàm không trả về giá trị nào cả. Cần lưu ý là:
    - Nếu hàm trả về kiểu int, ta có thể không khai báo (ngầm định là kiểu int).

- Nếu trả về khác kiểu int, ta phải khai báo tường minh Kiểu dữ liệu của hàm.

2. Ten_Ham: Là tên của hàm do người lập trình tự đặt theo quy tắc đặt tên.
3. Danh sách kiểu và đối:

Có thể có hoặc không.

- *Trường hợp không*: Nếu không có đối thì vẫn phải giữ các dấu ngoặc tròn ().Trong trường hợp này có thể thay bằng từ khóa void.
- *Trường hợp có*: Đó là các đối của hàm. Trước mỗi đối có kiểu dữ liêu tương ứng của nó, nếu có nhiều đối và kiểu tương ứng của nó, thì chúng phải cách nhau dấu phảy (,).

b. Thân hàm

Thân hàm bắt đấu bằng dấu ngoặc nhọn mở{, tiếp theo là các chỉ thị về kiểu (nếu có), các câu lệnh trong đó có thể có hay không câu lệnh return và kết thúc bằng dấu ngoặc nhọn đóng.

c. Câu lệnh return

Trong thân hàm có thể có hoặc không có câu lệnh này. Trong trường hợp có, có thể có một câu lệnh return, hoặc nhiều câu lệnh return ở những nơi khác nhau,

Nếu không có câu lệnh return thì chương trình sẽ ra khỏi hàm khi gặp dấu ngoặc nhọn đóng cuối cùng } của thân hàm để trở về nơi gọi nó.

Nếu có, Câu lệnh "return" là cơ chế chuyển giá trị từ hàm được gọi về nơi gọi. Khi gặp câu lệnh return máy sẽ không thực hiện các câu lệnh sau nó trong hàm chứa câu lệnh này.

Dạng tổng quát của câu lệnh này là:

return [Bt];

Ý nghĩa của các dạng có thể minh họa như sau:

- Dạng 1: **return;**
  Dùng để thoát ra khỏi 1 hàm và trở về hàm đã gọi nó, mà không trả về một giá trị nào. Trường hợp này khai báo kiểu dữ liệu của hàm là void. Câu lệnh return có thể dùng để ra khỏi thân switch, các vòng lặp.
- Dạng 2: **return Bt;**
  hoặc **return (Bt);**
  Giá trị của biểu thức Bt sẽ được chuyển kiểu cho phù hợp với kiểu của hàm trước khi gán cho hàm, và ra khỏi hàm chuyển về nơi gọi nó.

Ví dụ 4.1:

Viết hàm tính giá trị lớn nhất của 2 số thực.

Input a,b ∈ R

Output Max(a, b) ∈ R

```
double Max(double a, double b)
{
    double Gtln;
    Gtln = (a > b) ? a:b;
    return Gtln;
}
```

Ghi chú:

Hàm trả về một giá trị, ta nên khai báo một biến để lưu trử giá trị đó.

Ví dụ 4.2:

Viết hàm đổi ký số thành số

```
Input x
Output KS_So(x)
int KS_So(char x)
{
    int GT_So;
     GT_So =  x – 48;//x – ‘0’
    return GT_So;
}
```

Ghi chú:

Ký số x	‘0’	‘1’	‘2’	‘3’	‘4’	‘5’	‘6’	‘7’	‘8’	‘9’
Mã ASCII (hệ 10)	48	49	50	51	52	53	54	55	56	57

Ví dụ 4.3:

Viết hàm: $f(x) = \begin{cases} 1; x \text{ lẻ} \\ 0; x \text{ chẵn} \end{cases}$

```
int f(int x)
{
    int Kq;
    Kq = x % 2;
    return Kq;
}
```

Ví dụ 4.4:

Hàm xuất ra màn hình các chức năng của chương trình.

```
void Menu()
{
    cout<<" \n ****Bảng Menu ********** ";
    cout<<" \n1.**chức năng 1  ********** ";
    cout<<" \n2.**chức năng 2  ********** ";
    cout<<" \n3.**chức năng 3  ********** ";
    cout<<" \n4.****  Thoát   ********** ";
    cout<<"\n ********************** *";
}
```

Ví dụ 4.5:

Viết hàm chọn menu.

```
Input
Output Chon
int Chon_Menu()
{
    int Chon;
    cout<<”\nChon chuc nang nao ? “;
    cin>>Chon;
    return Chon;
}
```

Ví dụ 4.6:

Tính giá trị tuyệt đối của số nguyên.

```
input a ∈ Z
output | a | ∈ N
int gttd(int a)
{
    int Kq =  (a > 0) ? a: -a;
    return  Kq;
}
```

4.2.2 Truyền tham số

Có 2 cách: Truyền bằng trị và truyền bằng biến.

Phần này giới thiệu truyền bằng trị.

Tham số của hàm luôn được truyền theo tham trị, với hình thức lấy các giá trị cùng kiểu với đối gán cho đối.

Khi đó các tham số thực (là các giá trị gán cho đối) không bị thay đổi giá trị khi chương trình ra khỏi hàm.

Ví dụ 4.7:

- Max(3.5, 6.4)
  Các số thực 3.5 và 6.4 gán lần lượt cho các đối thực a, b.
- KS_So('0')
  Hằng ký tự '0' gán cho đối ký tự x.

4.2.3 Lời gọi hàm

- Lời gọi hàm được viết như sau:
  Ten_Ham([Danh sách các tham số thực])

Trong đó:

- Số tham số thực phải bằng số các đối.
- Kiểu của tham số thực phải phù hợp với kiểu của đối tương ứng.

Ví dụ 4.8 (về lời gọi hàm):

- Max(3.5, 6.4)
- KS_So('0')
- f(2)
- Menu()

Ghi chú: Lời gọi hàm có sử dụng dạng truyền tham số cho hàm.

4.2.4 Sử dụng hàm

- Hàm được sử dụng thông qua lời gọi hàm, cách sử dụng như sau:
  - Nếu hàm trả về kiểu void, Viết lời gọi hàm như câu lệnh (thêm dấu ; cuối cùng), tức là:
    Ten_Ham([Danh sách các tham số thực]);
  - Nếu hàm trả về khác kiểu void, lời gọi hàm được sử dụng:
    - Như một toán hạng trong biểu thức.
    - Vế phải câu lệnh gán.

- In giá trị của hàm. . .

Ví dụ 4.9 (về cách sử dụng lời gọi hàm):

- Menu();
- cout<<Max(3.5, 6.4);
- double Kq = 3*Max(3.5, 6.4);
- cout<<KS_So('0');

4.2.5 Khai báo nguyên mẫu của hàm (prototype).

Trước khi sử dụng một hàm, ta có thể khai báo hoặc không khai báo nguyên mẫu của hàm trong chương trình.Vị trí khai báo có thể là ngoài tất cả các hàm, hoặc trong hàm... thường là đầu chương trình tại các định nghĩa toàn cục.

Dạng khai báo nguyên mẫu hàm là:

Kdl tên_hàm (Danh_sách_Kiểu_và _Đối); // Có dấu ;

Ví dụ 4.10:

float max(float a, float b);

Đối với C++, Các trường hợp sau nhất thiết phải khai báo nguyên mẫu:

- Vị trí của hàm đặt sau hàm main().
- Các hàm - không phải hàm main() - gọi lẫn nhau.

Dù rằng có các trường hợp không nhất thiết phải khai báo nguyên mẫu, nhưng tốt hơn cả là ta vẫn khai báo để trình biên dịch dễ phát hiện lỗi khi gọi hàm.

4.2.6 Hoạt động của hàm

Khi gặp một lời gọi hàm thì hàm bắt đầu thực hiện. Quá trình diễn ra theo trình tự:

a) Cấp phát bộ nhớ cho các đối và các biến địa phương.
b) Gán giá trị của các tham số thực cho các đối tương ứng.
c) Thực hiện các câu lệnh trong thân hàm.
d) Khi gặp câu lệnh return hoặc dấu } cuối cùng của thân hàm thì máy sẽ xóa các đối, các biến địa phương và ra khỏi hàm.

4.2.7 Hàm inline

Hàm có ưu điểm là được dùng lại chứ không phải viết lại cho các ứng dụng khác, tuy nhiên nó cũng có một nhược điểm là làm chậm chương trình một các đáng kể, ngay cả khi thân hàm chỉ có một vài câu lệnh, vì phải sao các đối, cất giữ trên thanh ghi, chương trình phải nhảy tới một vị trí mới.

Hàm inline đưa ra một giải pháp khắc phục tình trạng này, và xác định bởi từ khóa inline.

Một hàm được xác định là inline thì sẽ mở rộng “”trên dòng” tại điểm gọi, và làm cho phí tổn của hàm được loại bỏ.

Cách viết là chỉ thêm inline trước kiểu trả về của hàm.

4.3 Một số thư viện trong C++

Một trong những tính năng thuận tiện của C++ là cung cấp kèm theo một số lượng lớn các thư viện. Thư viện chứa các hằng, các hàm thực hiện các thao tác chuyên biệt nào đó. Muốn sử dụng thư viện, ta chỉ cần tham chiếu đến tập tin giao

diện gọi là tập tin tiêu đề (tập tin header, có đuôi dạng.h) của thư viện đó, bằng cách sử dụng chỉ thị: #include <Ten_Tep>
Sau đây là một số thư viện thường dùng:

Tên tập tin tiêu đề	Hàm	Công dụng
iostream	Chứa các đối tượng cin, cout, . . .	Nhập xuất chuẩn
iomanip	Chứa các định dạng kết xuất	
math.h	double log(double x)	Lnx
	double pow(double x, double y)	x^y
	double sqrt(double x)	$\sqrt{x}$
	double exp(double x)	e^x
	(Còn chứa các hàm lượng giác ,. . .)	

4.4 Tổ chức chương trình C++ bằng Win32 Console Application trong môi trường MS Microsoft Visual Studio 2005 .

4.4.1 Tập tin đề án

Một chương trình trong Windows tạo bằng Win32 Console Application trong môi trường MS Microsoft Visual Studio 2005 là một đề án (project) bao gồm các thành phần sau:

a. Chương trình nguồn: Các tập tin dạng .*cpp, .*h. . .

- Tập tin *.h chứa các định nghĩa hằng số, biến, hàm,...

Đó là các tập tin tiêu đề của các thư viện, do ta tự xây dựng hoặc của ngôn ngữ lập trình.

- Tập tin *.CPP có thể có một hay nhiều, nhưng trong đó phải có một tâp tin chứa hàm main. Tập tin *.cpp chứa các xử lý hoặc cài đặt hàm chức năng.

b. Tập khai báo tài nguyên

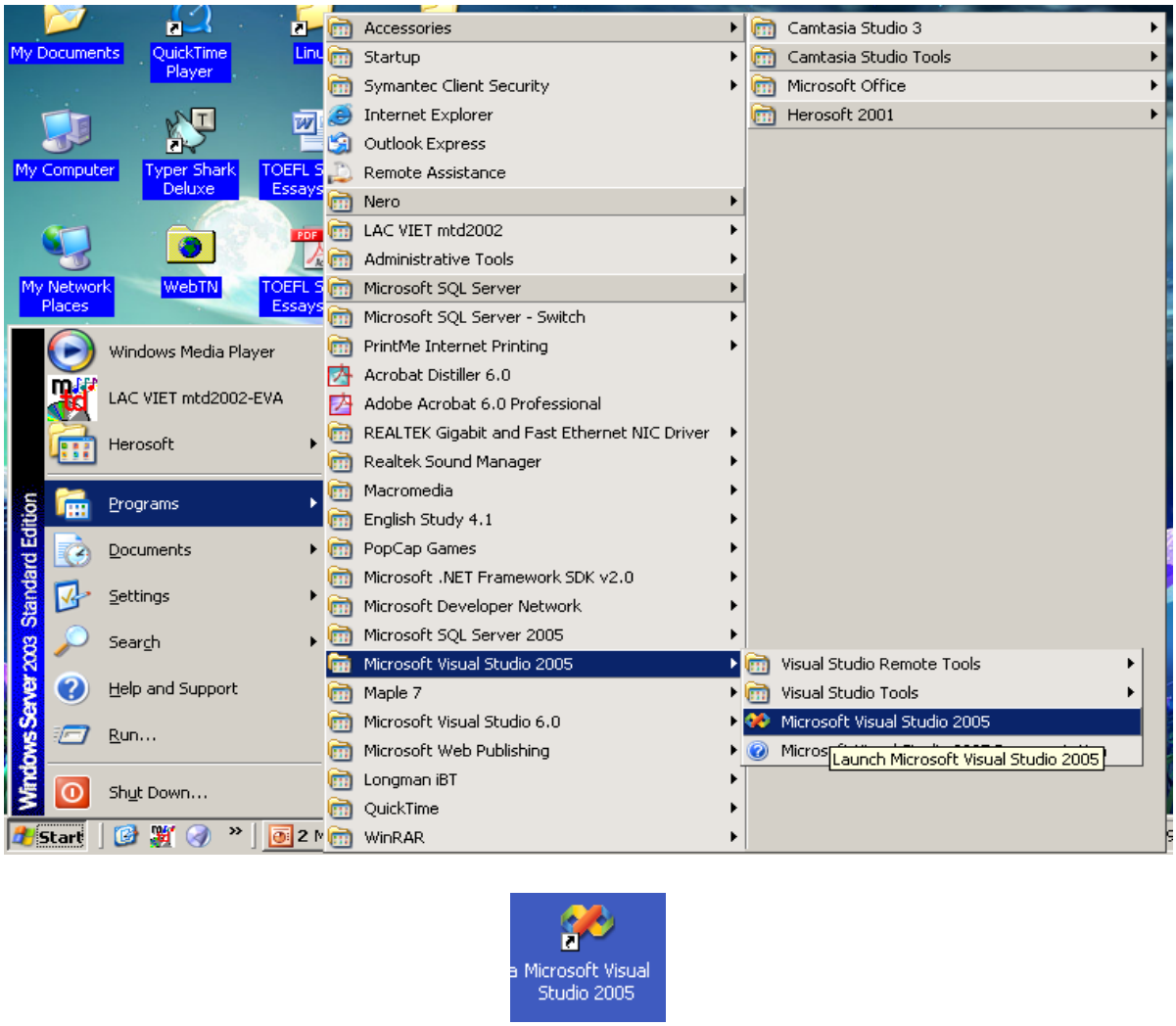
c. Tập tin đề án có dạng .dsw, .sln

4.4.2 Tạo đề án trong Console application trên Microsoft Visual Studio 2005

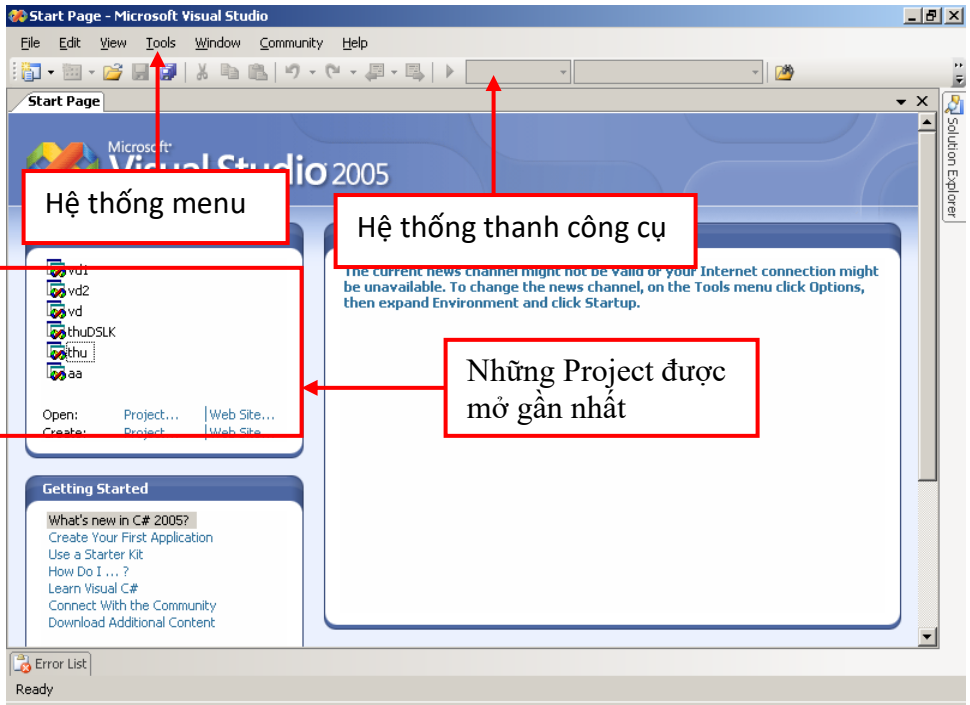
Trường hợp đề án gồm 1 tập tin chương trình *.cpp, tập tin chương trình có thể có một hay nhiều hàm. Chú ý rằng trường hợp nào cũng phải có hàm main(), hàm main() sẽ khởi động các hàm khác để thực hiện các công việc của chương trình.

Các bước tạo đề án:

Bước 1: Start → programs → Microsoft Visual Studio 2005 → Microsoft Visual Studio 2005 (hoặc chọn icon)

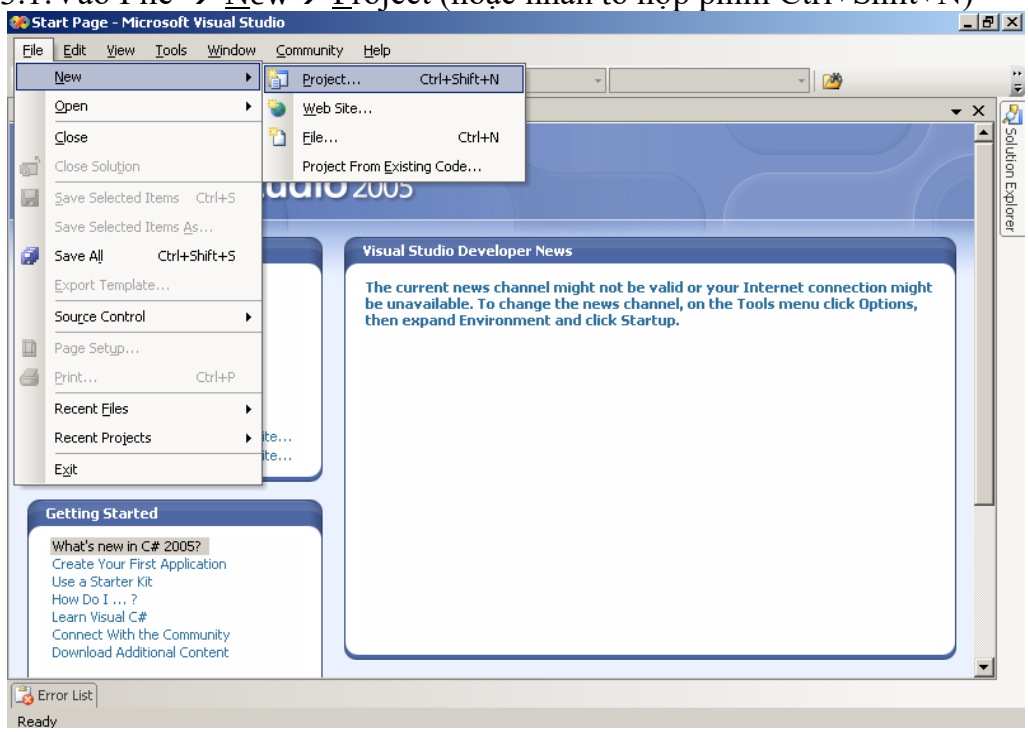


Bước 2: Giao diện chương trình chính khi chạy chương trình

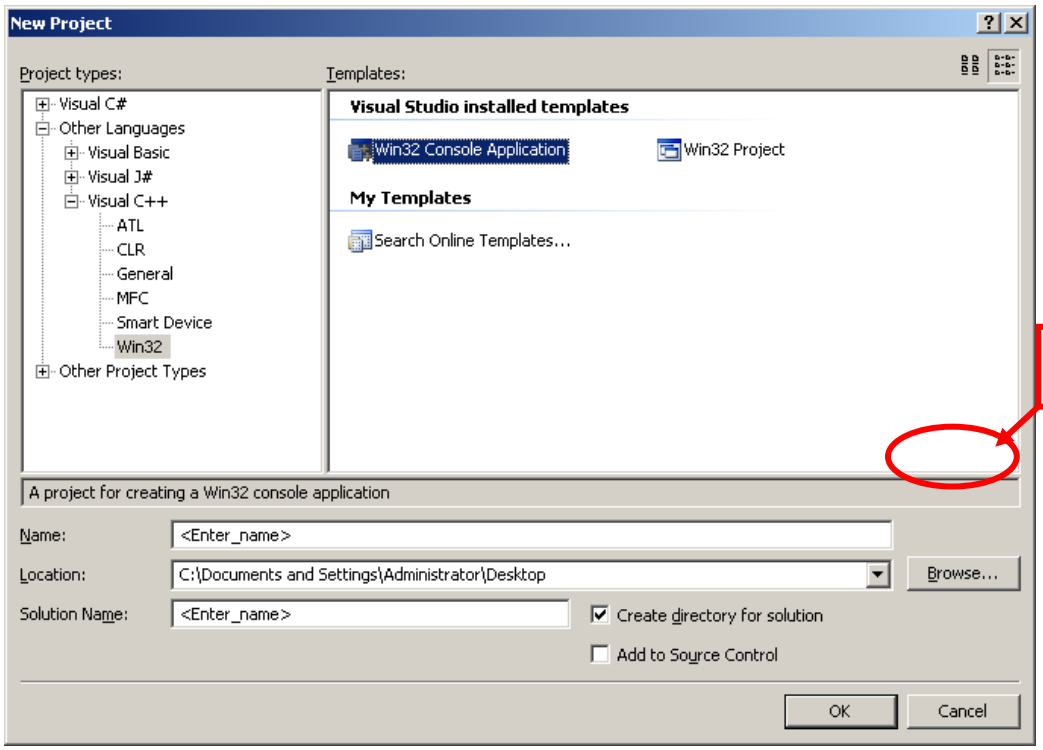


Bước 3: Tạo một Project mới

Bước 3.1:Vào File → New→ Project (hoặc nhấn tổ hợp phím Ctrl+Shilt+N)

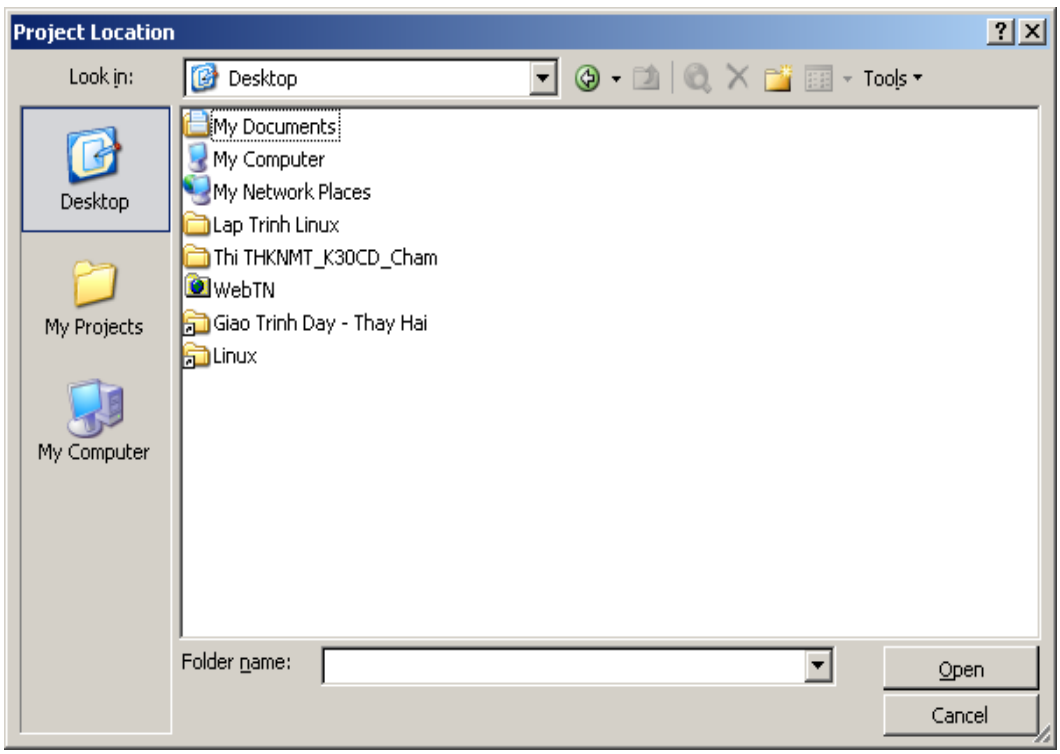


Bước 3.2: chọn Other Languages → Visual C++ → Win32 → Win32 Console Application

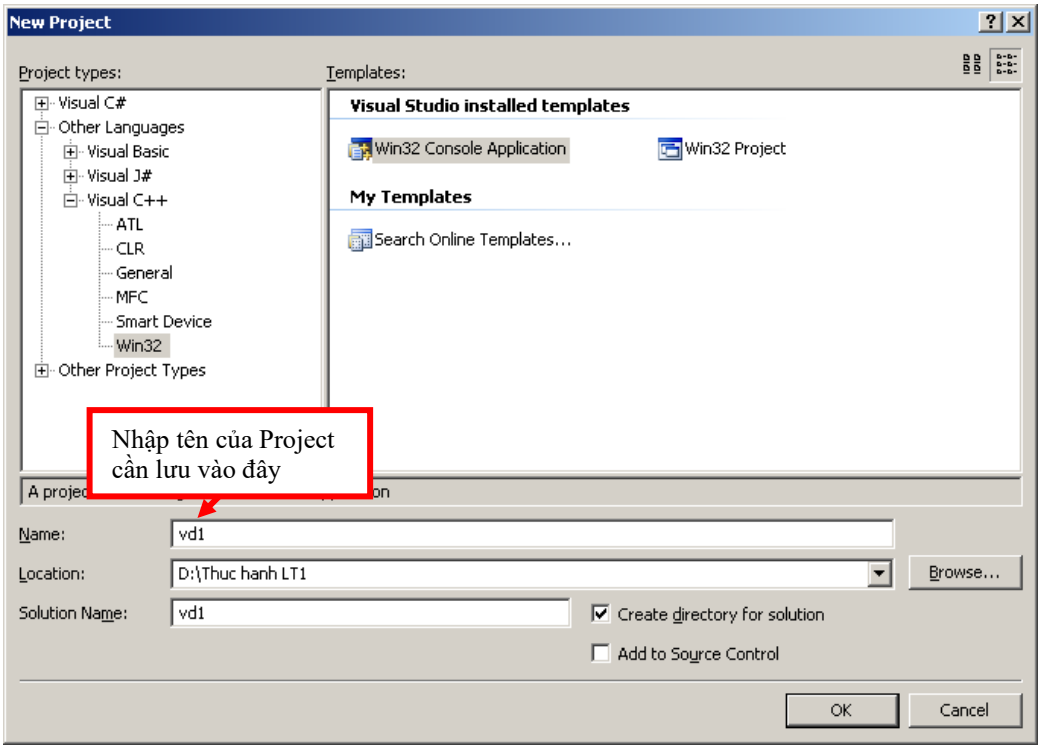


Bước 3.3:Chọn thư mục cần lưu.Sau khi chọn xong thư mục cần lưu

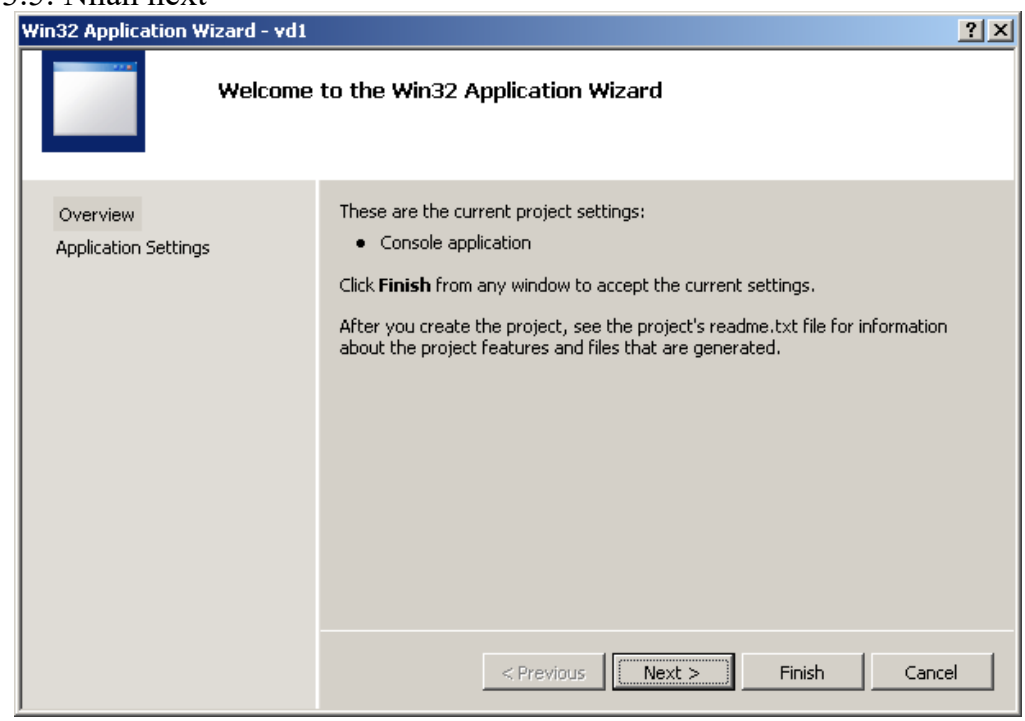
→ nhấn OK để đồng ý (nhấn Cancel để thoát).



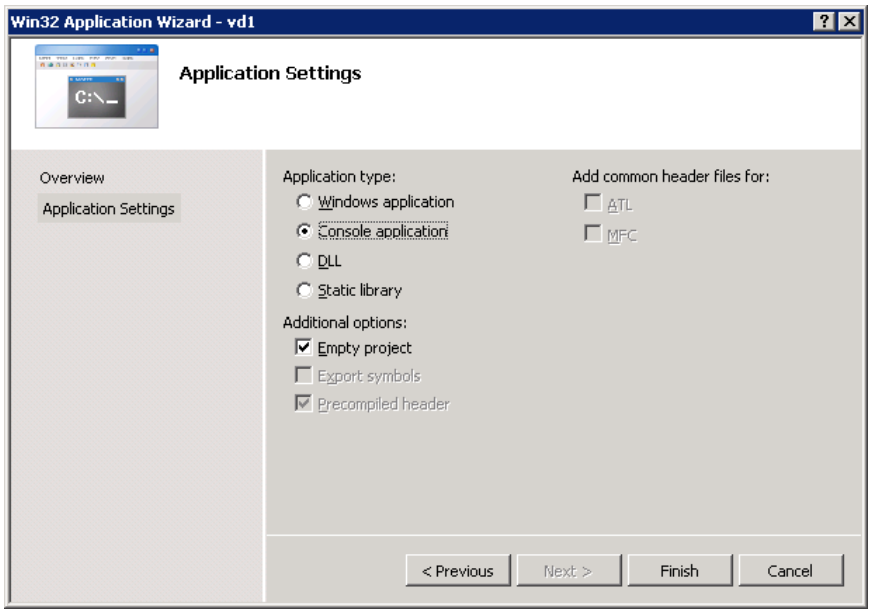
Bước 3.4: Nhập tên Project cần lưu → nhấn OK.



Bước 3.5: Nhấn next

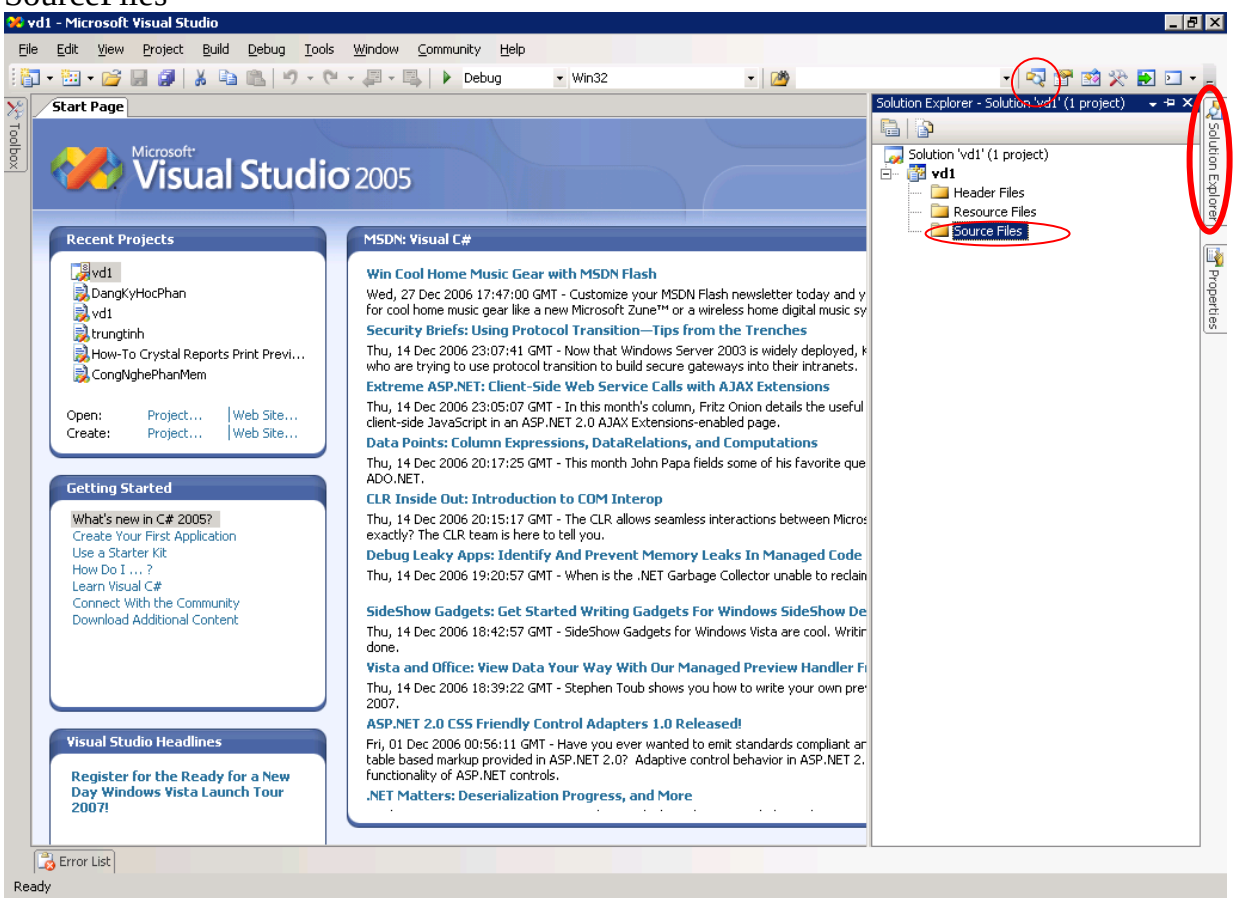


Bước 3.6: Chọn Empty Project và nhấn Finish để hoàn tất việc tạo một Project Console Application



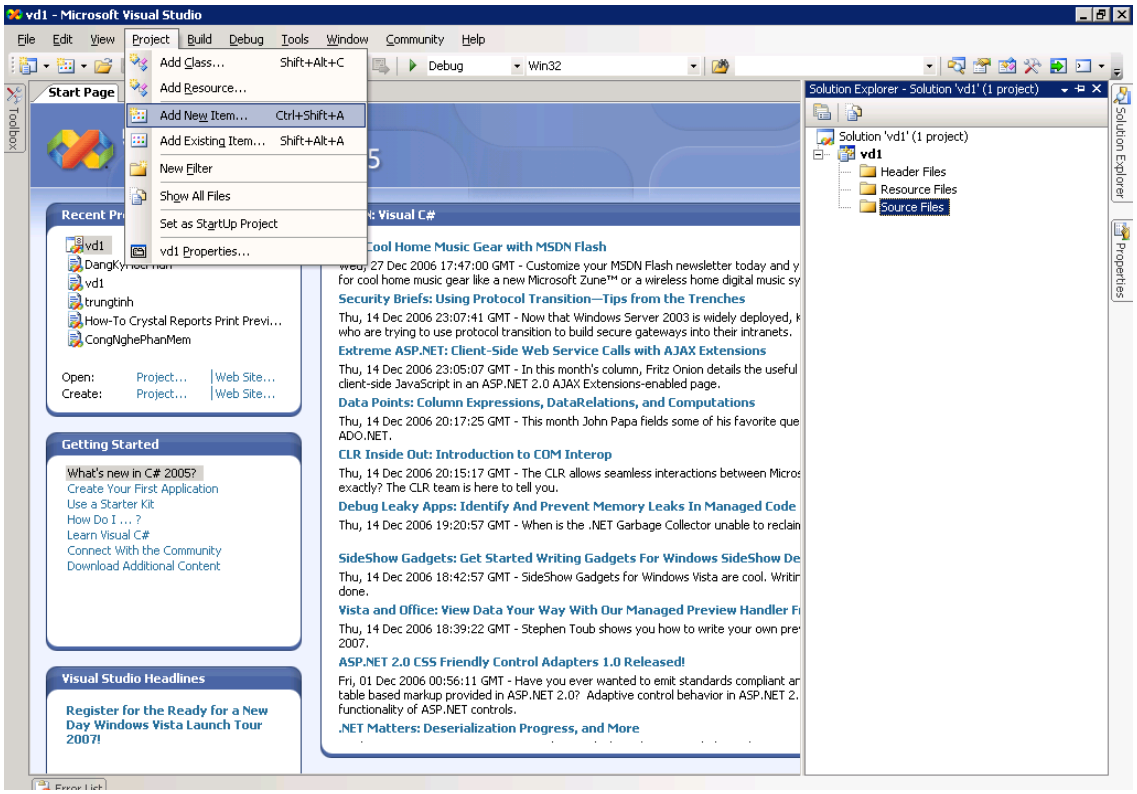
Bước 4: Thêm một file vào

Bước 4.1: Click solution Explorer hoặc chọn View →Solution Explorer và chọn SourceFiles

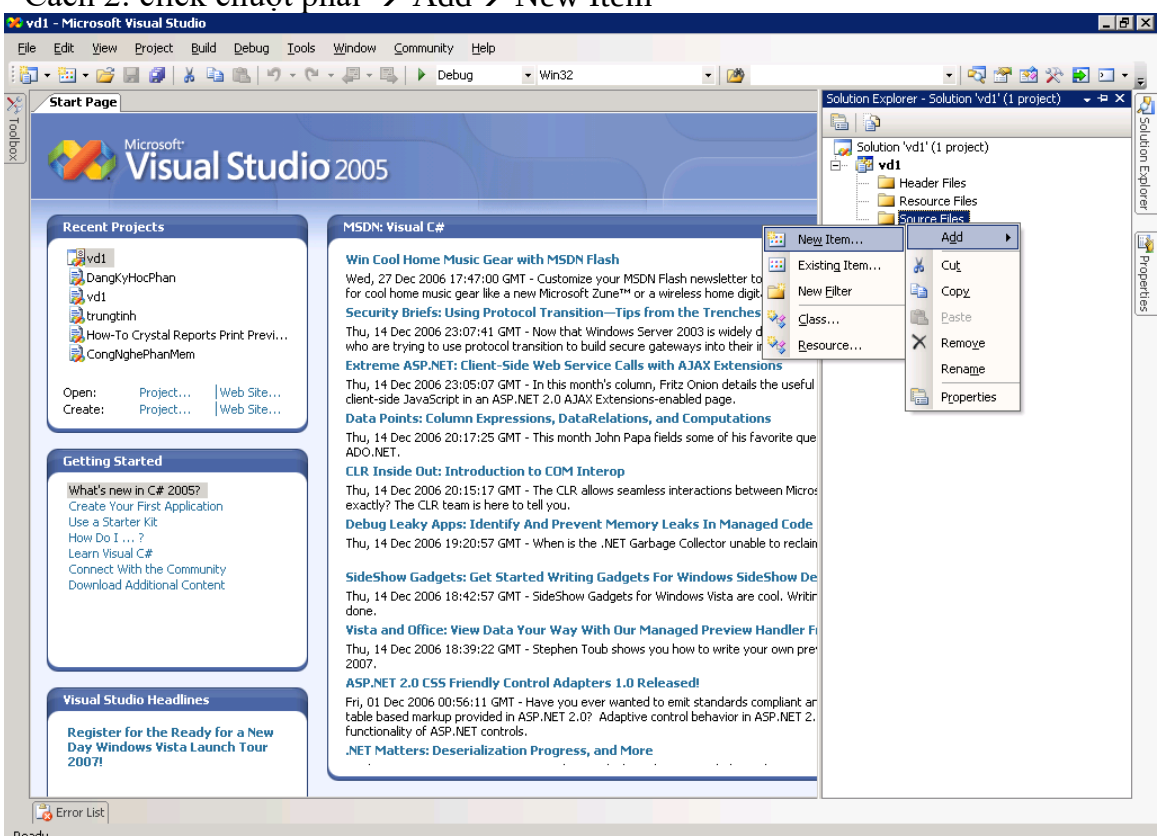


Bước 4.2: click chọn SourceFiles → Thêm file .cpp để viết code bằng 2 cách:

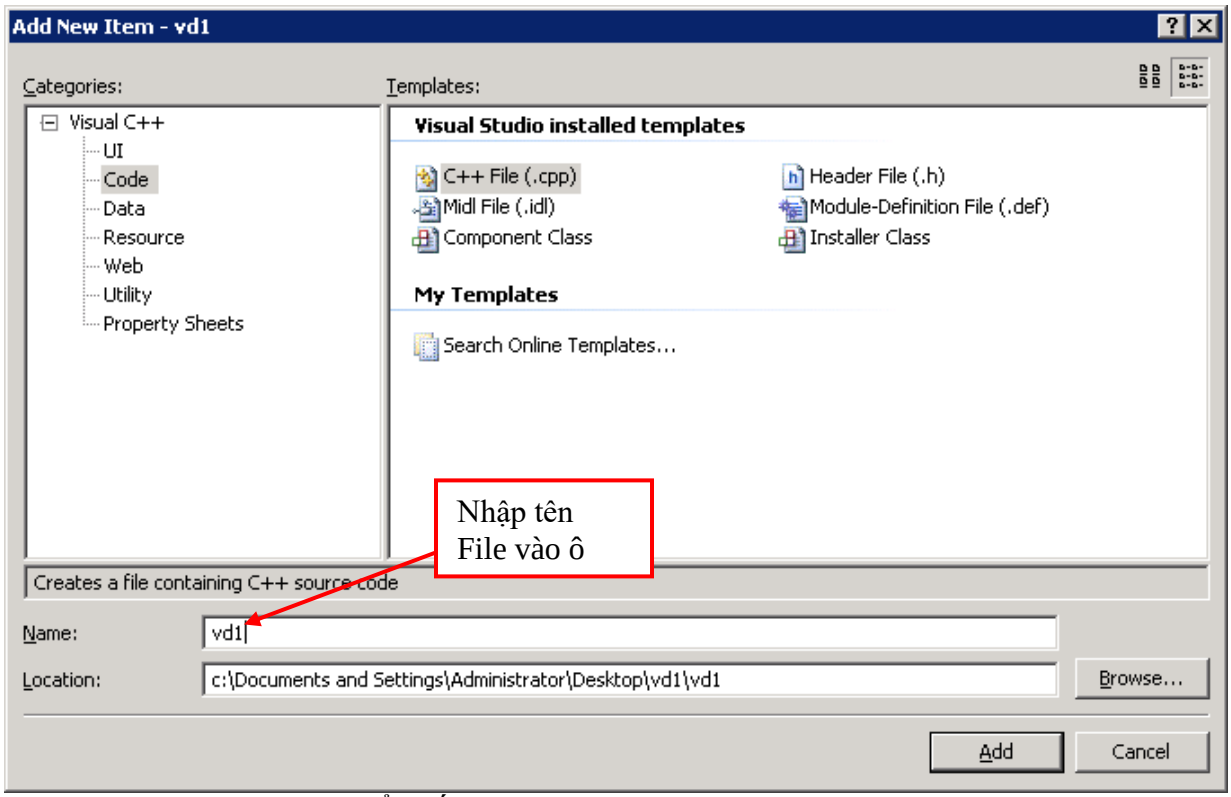
- Cách 1: vào Project → Add new item Hoặc nhấn Ctrl+Shilt+A



- Cách 2: click chuột phải → Add→ New Item

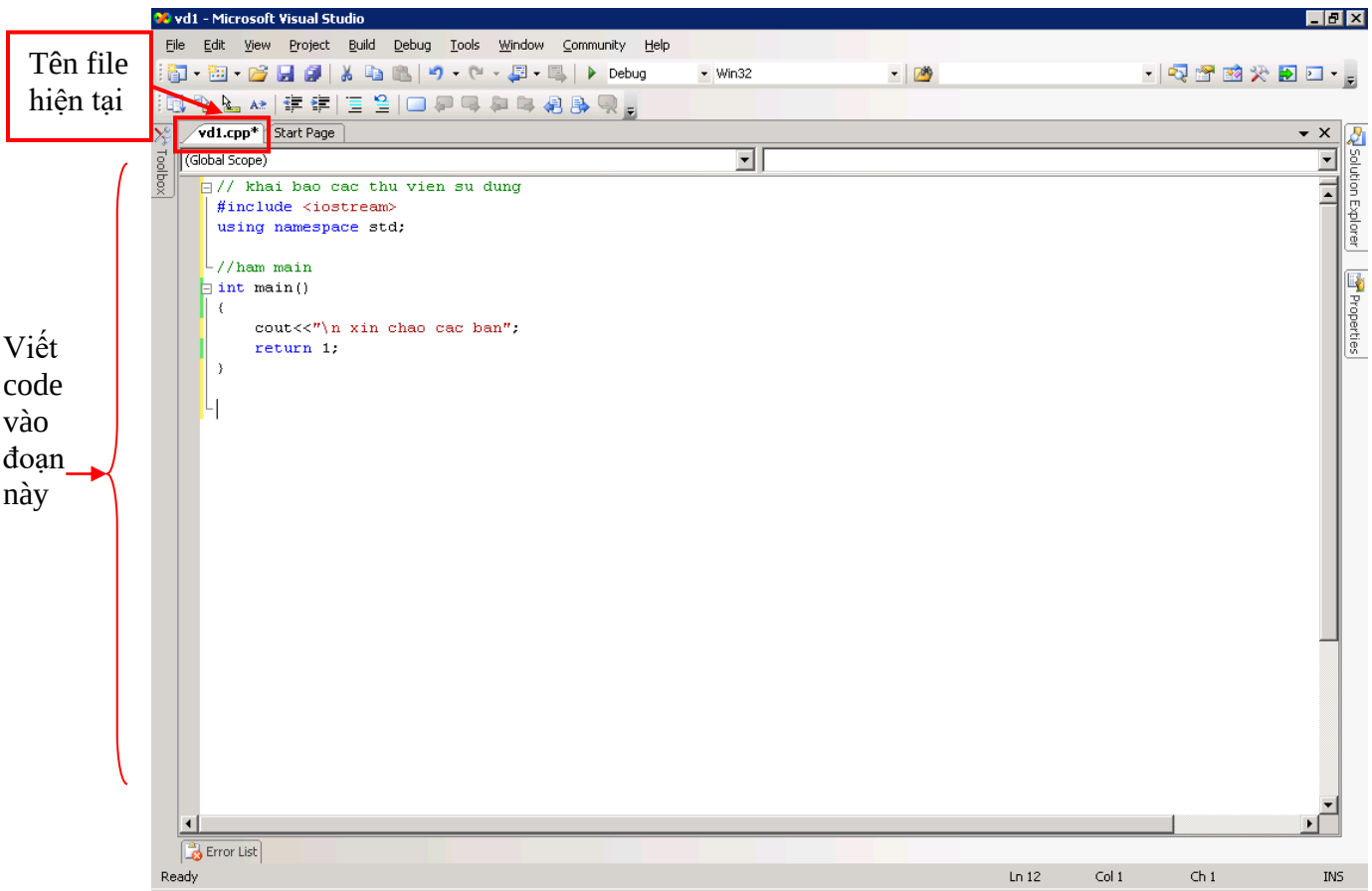


Bước 4.3: Chọn Code→ chọn C++File(.cpp) và nhập tên file



Bước 4.3: Click vào Add để chấp nhận thêm một file mới.

Bước 5: Nhập code (ví dụ nhập đoạn code như sau)

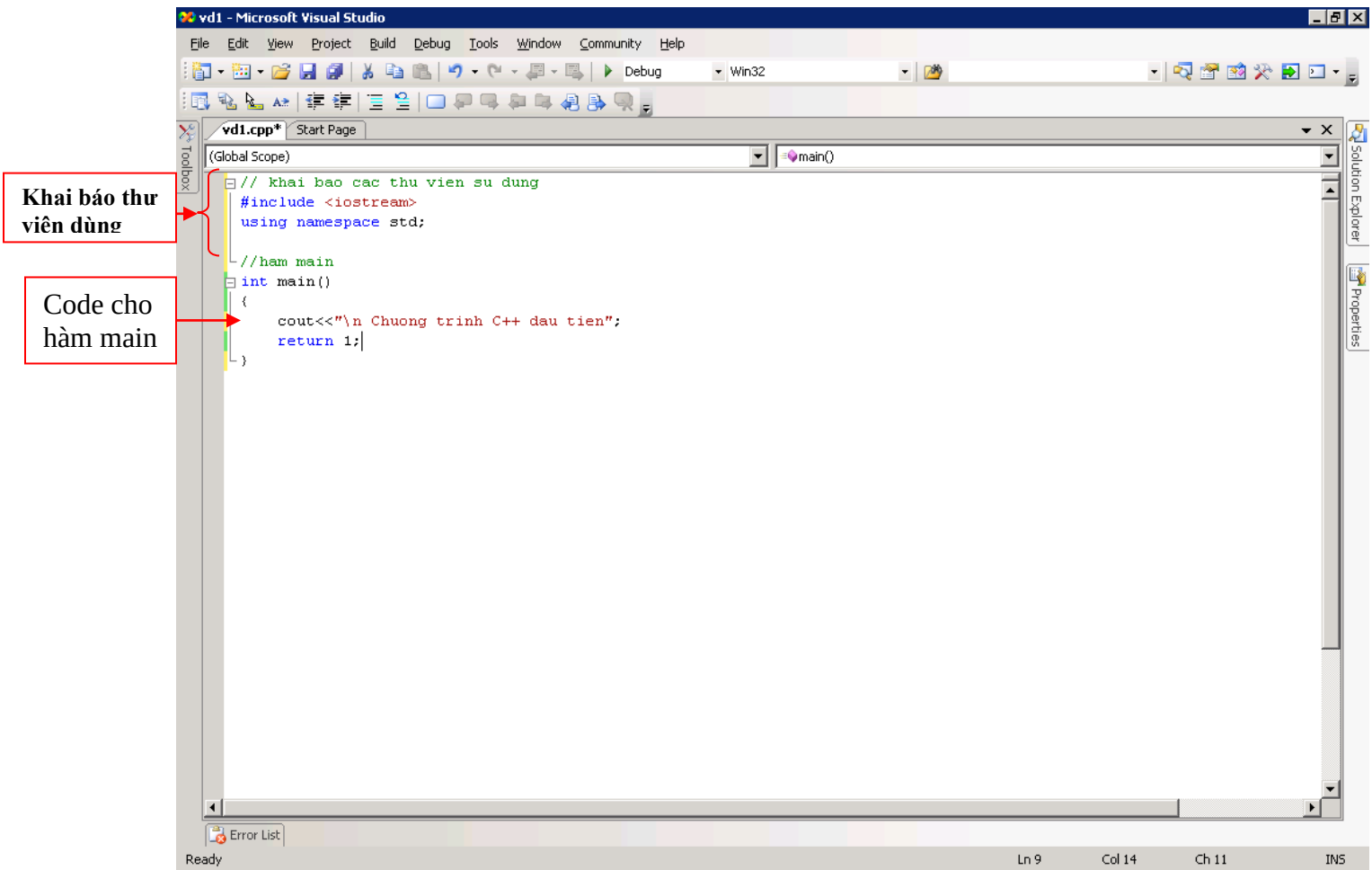


Bước 6: Để thực thi chương trình thực hiện các bước sau:

- Nhấn Ctrl+F7 để biên dịch chương trình (Compile)
- Nếu không có lỗi, nhấn F5 để thực thi chương trình(Execute)

Lưu ý: các chức năng trên có thể thực hiện bằng cách nhấn vào các biểu tượng tương ứng ở góc phải trên màn hình.

Ví dụ: để xuất chuỗi "Chuong trinh C++ dau tien". Ta nhập đoạn code sau và nhấn F5 (Ctrl+F5) để chạy chương trình.



Ví dụ 4.11:

Viết chương trình hiển thị (có định dạng) trên 1 hàng:

MSO: 10 cột
HOTEN: 22 cột
LOP: 10 cột
DTB: 10 cột
TICHLUY: 10 cột

Thực hiện:

Bước 1: Tạo Project rỗng với tên "Dinhdang".

Bước 2: Tạo tập tin chương trình Hienthi.cpp

Bước 3: Nhập đoạn code sau vào tập tin Hienthi.cpp:

```
#include<iostream>
#include<iomanip> //Chứa các định dạng
using namespace std;
int main()
{
        cout <<setiosflags(ios::left)  //Canh trái
             <<setw(12)<<"MASO"  //độ rộng 12 cột
             <<setw(22)<<"HOTEN"
             <<setw(10)<<"LOP"
             <<setw(10)<<"DTB"
             <<setw(10)<<"TICHLUY";
        return 0;
}
```

Ví dụ 4.12:

Viết chương trình tính và xuất ra diện tích tam giác:

S = $\sqrt{p(p-a)(p-b)(p-c)}$, a, b ,c được nhập từ bàn phím, p = $\frac{a+b+c}{2}$

(Sử dụng hàm tính căn bậc 2: *sqrt* trong thư viện math.h)

Thực hiện:

Bước 1: Tạo Project rổng với tên "DTTamgiac".

Bước 2: Tạo tập tin chương trình Dttg.cpp

Bước 3: Nhập đoạn code sau vào tập tin Dttg.cpp:

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
        double a, b, c, p, S;
        cout<<"\nNhap cac canh tam giac (so thuc duong):";
        cin>>a>>b>>c;
        p = (a+b+c)/2;
        S = sqrt(p*(p-a)*(p-b)*(p-c));
        cout<<"\nDien tich tam giac: S = "<<S;
        return 0;
}
```

BÀI TẬP

Bài 1:

Viết chương trình khai báo 3 biến: x kiểu số thực, c kiểu ký tự, i kiểu số nguyên.Nhập, xuất giá trị cho các biến đó.

Bài 2:

Viết chương trình nhập vào 2 biến số nguyên x, y. Tính giá trị của x+y, xuất kết quả ra màn hình

Bài 3:

Viết chương trình tính chu vi, diện tích của hình chữ nhật với chiều dài, rộng nhập từ bàn phím.

Bài 4:

Khai báo hằng PI có giá trị 3.14 sử dụng hằng PI để tính diện tích hình tròn với bán kính được nhập từ bàn phím.

Bài 5:

Khai báo hằng MAX có giá trị 60. Nhập số giây, quy đổi thời gian giây thành giờ, phút, giây. Xuất kết quả ra màn hình dưới dạng: gio:phut:giay

Bài 6:

Khai báo biến x, y kiểu số nguyên. Khởi gán x =20, y=6. Thực hiện các câu lệnh sau và xuất kết quả của x, y trước và sau khi thực hiện từng câu lệnh này.

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

Bài tập 7:

Viết chương trình nhập vào một số nguyên và xuất ra màn hình
(Chú ý: bạn hãy nhập số lớn và giải thích kết quả).

Bài 8:

Viết chương trình tính $x^2 + y^5$, với x và y là 2 số thực được nhập từ bàn phím. (Sử dụng hàm tính tính mũ: *pow* trong thư viện math.h)

Bài 9:

Viết chương trình tính x^n, với x là số thực và n là số nguyên được nhập từ bàn phím. (Sử dụng hàm tính tính mũ: *pow* trong thư viện math.h)

CHƯƠNG 5: CÁC CÂU LỆNH ĐIỀU KHIỂN

5.1 Câu lệnh if

5.1.1 Cú pháp

Dạng 1: if (Bt) Kl	Nếu biểu thức Bt đúng thực hiện Kl

Dạng 2:
if (Bt)
Kl1
else
Kl2
	Nếu biểu thức Bt đúng
	thực hiện Kl1
	Ngược lại
	thực hiện Kl2

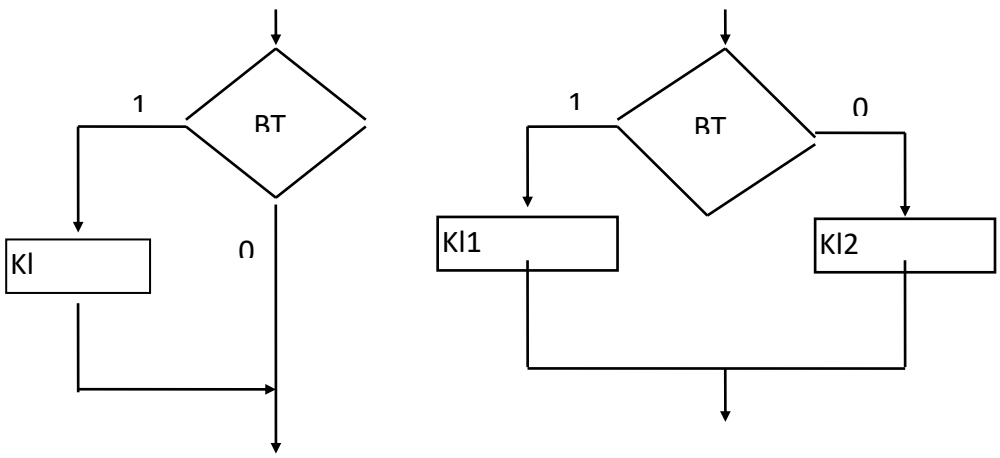
- Bt: là biểu thức có giá trị Đ (khác 0), hay sai (bằng 0).
- Kl: Khối lệnh.

5.1.2 Hoạt động

Trước tiên biểu thức Bt được xác định giá trị.

- Dạng 1:
  Nếu Bt đúng (có giá trị khác 0) thì thực hiện Kl. Nếu sai (giá trị bằng 0) thì khối lệnh được bỏ qua.
- Dạng 2:
  Nếu Bt đúng (có giá trị khác 0) thì thực hiện Kl1. Nếu sai (giá trị bằng 0) thì thực hiện Kl2.

5.1.3 Lưu đồ



Ghi chú:

Nếu có nhiều cấu trúc if - else lồng nhau, để xác định một "else" là của "if" nào ta theo qui tắc: "else" là của một "if" gần với nó nhất mà không có "else".

Tức là:	Tương đương với:

if (Bt1)
  if (Bt2)
  Kl2
  else
  Kl3
else
  Kl1
	if (Bt1)
	{ if (Bt2)
	  Kl2
	  else
	  Kl3
	}
	else
	  Kl1

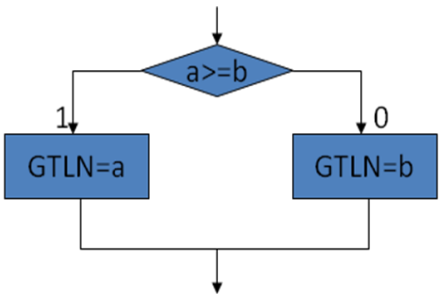
Ví dụ 5.1:

Tính giá trị lớn nhất 2 số thực.

- Mã giả:

```
input    a, b
output  Max(a,b)
Max (a, b) ≡
        if (a >=b)
                GTLN = a;
        else
                GTLN = b;
        EndIf
        return GTLN;
```

Lưu đồ



Cài đặt:

```
double Max (double a, double b)
{
        double GTLN;
        if (a >=b)
                GTLN = a;
        else
                GTLN = b;
        return GTLN;
}
```

Ví dụ 5.2:

Viết hàm giải phương trình bậc 2 trên R

```
void ptb2 ( double a, double b, double c)
{
        double Delta, x1, x2, x;
        Delta = b*b -4*a*c;
```

```
        if (Delta == 0)
        {
                x = -b/(2*a);
                cout<<"\nx = "<<x;
        }
        else
                if (Delta < 0)
                        cout<<"\nPhuong trinh Khong co nghiem thuc";
                else
                {
                        x1 = (-b + sqrt(Delta))/(2*a);
                        x2 = (-b - sqrt(Delta))/(2*a);
                        cout<<"\nPT co 2 nghiem phan biet: "
                            <<"\nx1 = "<<x1
                            <<"\nx2 = "<<x2;
                }
}
```

Ví dụ 5.3:

Viết hàm phân loại tam giác theo yêu cầu

$$f(a,b,c)=\begin{cases} 0\ ;\ Nếu\ a,b,c\ không\ là\ 3\ cạnh\ của\ tam\ giác. \\ 1;\ Nếu\ a,b,c\ là\ 3\ cạnh\ của\ tam\ giác\ đều, \\ 2\ ;\ Nếu\ a,b,c\ là\ 3\ cạnh\ của\ tam\ giác\ cân, \\ 3\ ;\ Nếu\ a,b,c\ là\ 3\ cạnh\ của\ tam\ giác\ vuông, \\ 4\ ;\ Nếu\ a,b,c\ là\ 3\ cạnh\ của\ tam\ giác\ vuông\ cân, \\ 5\ ;\ Nếu\ a,b,c\ là\ 3\ cạnh\ của\ tam\ giác\ thường, \end{cases}$$

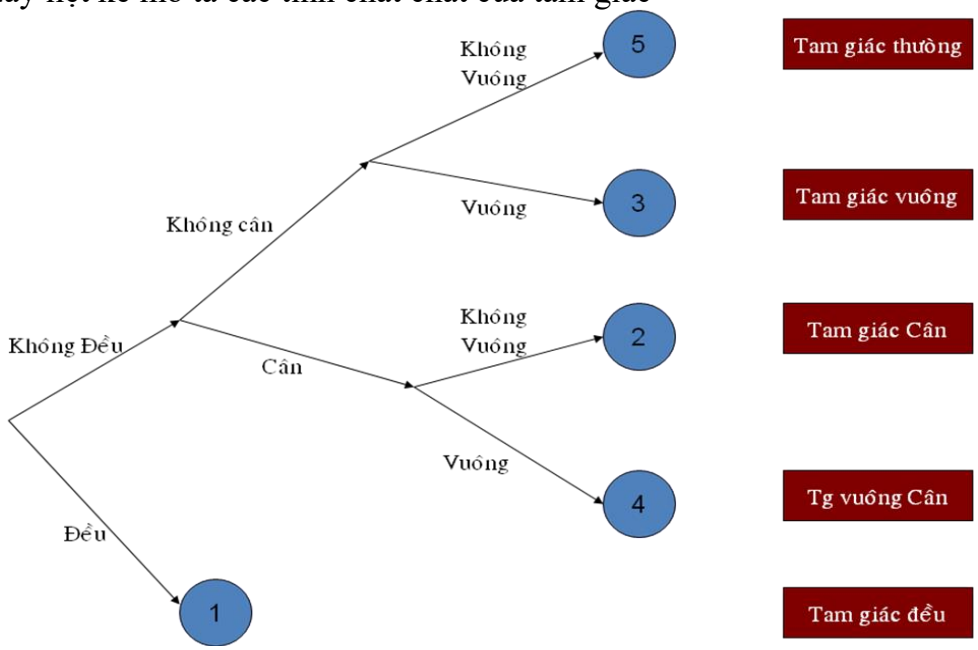
Cài đặt:

```
int Pltg(double a, double b, double c)
{
        int Kq = 0;//Khong la tam giac
        if (a+b > c && a+c > b && b+c > a) // La tam giac
        {
                if(a==b && b == c) // Tam giac deu
                        Kq = 1;
                else // Khong la tg deu
                        if(a == b || b == c || a == c) // Tg can
                        {       // Tg vuong
                                if(a*a ==b*b+c*c || b*b ==a*a+c*c || c*c== a*a+b*b)
                                        Kq = 4;
                                else // Khong vuong
                                        Kq = 2;
                        }
                        else // Khong can
                                if(a*a ==b*b+c*c || b*b ==a*a+c*c || c*c== a*a+b*b)
                                        Kq = 3; // Tg vuong
```

```
                    else // Khong vuong
                        Kq = 5;
        }
        return Kq;
}
```

Ghi chú:

Cây liệt kê mô tả các tính chất chất của tam giác



5.2 Câu lệnh switch

Câu lệnh cho phép chọn một trong nhiều nhánh rẽ.

5.2.1 Cú pháp

```
switch ( Bt)
{
        case H1:    NL1
                    [ break;]
        case H2:    NL2
                    [ break;]
                .....
        case Hn:    NLn
                    [ break;]
        [default:   NLn+1]
}
```

Trong đó Bt là nguyên, các Hk có thể là hằng nguyên, hằng ký tự, hoặc biểu thức hằng.

5.2.2 Hoạt động của câu lệnh switch

- Trước tiên biểu thức nguyên được tính trị.

- Nếu trị biểu thức bằng Hk thì máy sẽ thực hiện NLk.

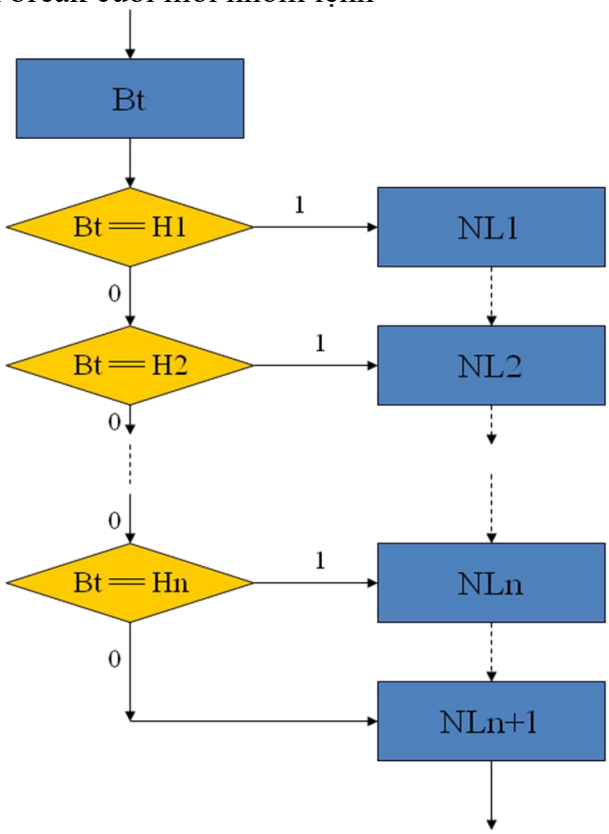
Sau đó:

- Nếu không có câu lệnh break thì thực hiện tiếp NL k+1 của nhãn case Hk+1... cho tới khi gặp lệnh break hoặc gặp dấu } cuối cùng thì ra khỏi switch.
- Nếu có câu lệnh break thì ra khỏi switch.

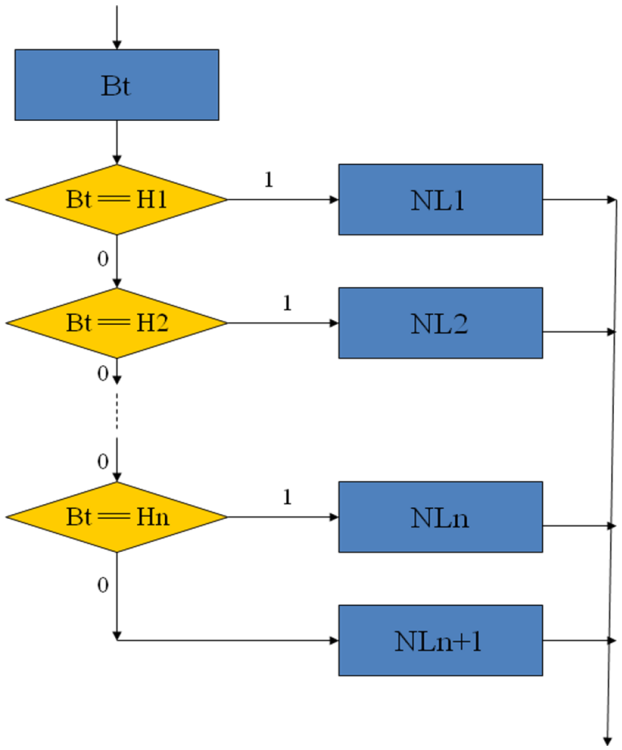
- Khi giá trị biểu thức khác tất cả các Hk thì:
  - Khi có default thì máy nhảy tới thực hiện NLn+1 trong default.
  - Khi không có default thì ra khỏi câu lệnh switch.

5.2.3 Lưu đồ: (Có thành phần default)

a) Không có câu lệnh break cuối mỗi nhóm lệnh



b) Có câu lệnh break cuối mỗi nhóm lệnh:



Ví dụ 5.4:

Viết hàm trả về số ngày của 1 tháng trong 1 năm khi biết tháng và năm.

Đặc tả:

Input thang,
nam;

Output So_Ngay;

Cài đặt:

```
int  SoNgay(int thang,  int nam)
{
        int So_Ngay   , nhuan;
        switch (thang)
        {
        // tháng bằng: 1,3,5,7,8,10,12
        case 1:
            So_Ngay    = 31;
            break;
        case 3:
            So_Ngay    = 31;
            break;
        case 5:
            So_Ngay    = 31;
            break;
        case 7:
            So_Ngay    = 31;
            break;
        case 8:
            So_Ngay    = 31;
            break;
```

```
            case 10:
                So_Ngay    = 31;
                break;
            case 12:
                So_Ngay    = 31;
                break;
            // Các tháng bằng: 4,6,9,11
            case 4:
                So_Ngay    = 30;
                break;
            case 6:
                So_Ngay    = 30;
                break;
            case 9:
                So_Ngay    = 30;
                break;
            case 11:
                So_Ngay    = 30;
                break;
            //Thang 2
            case 2:
                nhuan = ((nam% 4 == 0)&&  (nam %100 != 0) )||
                                                (nam % 400  == 0);
                if (nhuan)
                    So_Ngay    = 29;
                else
                    So_Ngay    = 28;
            }
            return So_Ngay;
        }
}
```

Ví dụ 5.5:

Viết hàm thông báo kết quả phân loại tam giác (ví dụ 5.3)

```
void Thongbao(int Kq)
{
        switch (Kq)
        {
            case  0:
                cout<<"\nKhong phai 3 canh tam giac!";
                break;
            case  1:
                cout<<"\ntam giac deu":
                break;
            case  2:
                cout<<"\ntam giac can":
                break;
            case  3:
```

```
                cout<<"\ntam giac vuong":
                break;
        case  4:
                cout<<"\ntam giac vuong can":
                break;
        case  5:
                cout<<"\ntam giac thuong":
                break;
    }
}
```

Ghi chú:

Gọi hàm phân loại tam giác (với tham số thực a, b, c):

- int Kq = Pltg(a, b, c);
- giá trị Kq này truyền cho hàm Thongbao.

5.3 Câu lệnh for

5.3.1 Cú pháp

Câu lệnh for cho ta một cách xây dựng vòng lặp, có dạng sau:

```
for(bt1 ; bt2 ; bt3)
        Khối lệnh      // Câu lệnh
```

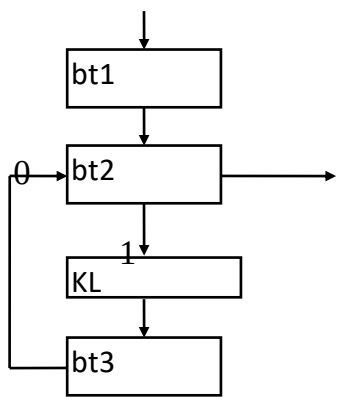
Trong đó:

- Mỗi thành phần bt là một biểu thức hay là một dãy các biểu thức được phân tách bởi dấu phảy.
- Bất kỳ một thành phần nào trong 3 thành phần bt cũng có thể vắng mặt, nhưng các dấu chấm phảy (;) luôn luôn có.

Chức năng của các thành phần bt thường là:

- bt1 là một phép gán để tạo ra giá trị ban đầu cho biến điều khiển.
- bt2 là một một quan hệ logic biểu thị điều kiện để tiếp tục vòng lặp.
- bt3 là một phép gán dùng để thay đổi giá trị của biến điều khiển.

5.3.2 Hoạt động của câu lệnh for

Hoạt động theo các bước: 1. Xác định biểu thức 1. 2. Xác định biểu thức 2. 3. Căn cứ vào giá trị đúng (khác 0), sai (bằng 0), sẽ lựa chọn một trong 2 nhánh: - Nếu biểu thức 2 sai , sẽ ra khỏi câu lệnh for và chuyển đến câu lệnh sau thân for. - Nếu biểu thức 2 đúng, sẽ thực hiện các câu lệnh trong thân for.Khi gặp dấu ngoặc nhọn đóng } cuối cùngcủa thân for,hoặc gặp câu lệnh continue máy sẽ chuyển sang bước 4 (khởi đầu lại).	

4. Tính biểu thức 3, sau đó quay trở lại bước 2 để bắt đầu một vòng mới của vòng lặp.	

Ví dụ 5.6:

Viết hàm thống kê có bao nhiêu điểm >= i, với i =0,...10, từ n điểm nguyên từ 0 đến 10 đọc vào từ bàn phím và xuất kết quả ra mà hình.

```
void ThongKe(int n)
{
	int x, //Diem doc vao tu ban phim
	d0= 0, //so diem 0
	d1=0,
	d2=0,
	d3=0,
	d4=0,
	d5=0,
	d6=0,
	d7=0,
	d8=0,
	d9=0,
	d10=0;
	int i;
	for(i = 1; i <= n; i++)
	{
		cout<<"\nNhap x = ";
		cin>>x;
		switch(x)
		{
			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++;
		}
	}
	cout<<"\nSo diem >=0: "<<do;
	cout<<"\nSo diem >=1: "<<d1;
	cout<<"\nSo diem >=2: "<<d2;
	cout<<"\nSo diem >=3: "<<d3;
	cout<<"\nSo diem >=4: "<<d4;
```

```
        cout<<"\nSo diem >=5: "<<d5;
        cout<<"\nSo diem >=6: "<<d6;
        cout<<"\nSo diem >=7: "<<d7;
        cout<<"\nSo diem >=8: "<<d8;
        cout<<"\nSo diem >=9: "<<d9;
        cout<<"\nSo diem >=10: "<<d10;
}
```

Ví dụ 5.7:

Viết hàm tính tổng n số nguyên dương đầu tiên.

Đặc tả:
input n;
output s =1+2+...+n
Cài đặt:

```
int Tong (int n)
{
        int i,s = 0;
        for (i =1; i <= n ; i++)
                s += i;
        return s;
}
```

Ví dụ 5.8:

Viết hàm tính giai thừa n.

Đặc tả:
input n;
output s =1.2...n
Cài đặt:

```
int Giaithua (int n)
{
        int i,T;
        if (n==0)
                T = 1;
        else
                if( n > 0)
                {
                        T = 1;
                        for (i =1; i <= n ; i++)
                                T*= i;
                }
        return T;
}
```

5.4 Câu lệnh while (Lặp với điều kiện được kiểm tra trước)

5.4.1 Cú pháp

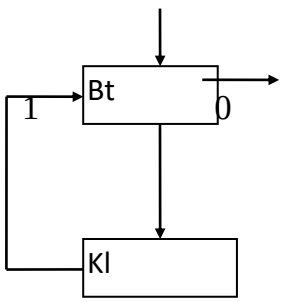
while (Bt)

Kl // Thân vòng lặp

Trong đó:

- Bt có thể là 1 biểu thức hay là một dãy các biểu thức được phân tách nhau bởi dấu phảy.
- Giá trị của Bt là Đúng (khác 0) hoặc sai (bằng 0).

5.4.2 Lưu đồ

Ngữ nghĩa: Trong khi Bt còn đúng Thực hiện Kl	

5.4.3 Hoạt động của câu lệnh while

Theo trình tự:

1. Xác định giá trị của Bt.
2. Tùy thuộc vào tính đúng sai của bt, máy sẽ lựa chọn 1 trong 2 nhánh:

a. Nếu bt có giá trị 0 (sai) máy sẽ ra khỏi vòng lặp và chuyển tới câu lệnh sau thân while.

b. Nếu bt có giá trị khác 0 (đúng) máy sẽ thực hiện các câu lệnh trong thân while. Khi gặp dấu ngoặc nhọn } đóng cuối cùng của thân while máy sẽ trở lại bước 1; hoặc gặp câu lệnh continue máy sẽ chuyển đến đầu một vòng lặp mới của vòng lặp, tức là máy sẽ bỏ qua các câu lệnh còn lại trong thân vòng lặp để trở về bước tính và kiểm tra bt.

Ví dụ 5.9:

Viết hàm kiểm tra một số nguyên dương có phải là số nguyên tố.

Đặc tả:

input n
Output 1; Nếu n nguyên tố
0; Ngược lại;

```
int Kt_Nt (int n)
{
	int  m, i, Nt;
	if ( n <= 1)
		Nt = 0;
	else
	{
		Nt = 1;
		i = 2;
		m = (int)sqrt(n);
		while (i <= m && Nt)
```

```
            {
                if( n % i == 0)
                    Nt = 0;
                i++;
            }
        }
        return  Nt;
}
```

Ví dụ 5.10:

Tìm ước chung lớn nhất 2 số nguyên dương

```
int UCLN(int a, int b)
{
    int r;
    while(b)
    {
      r = a%b;
      a = b;
      b = r;
    }
    return a;
}
```

5.5 Câu lệnh do.. while (Lặp với điều kiện được kiểm tra sau)

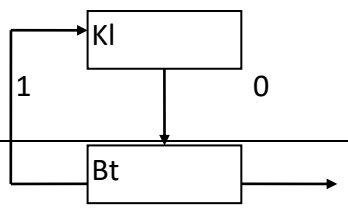
5.5.1 Cú pháp

```
do
    Kl      // thân vòng lặp
while(bt);
```

Trong đó:

- Bt có thể là 1 biểu thức hay là một dãy các biểu thức được phân tách bởi dấu phảy.
- Giá trị của Bt là Đúng (khác 0) hoặc sai (bằng 0).

5.5.2 Lưu đồ

Ngữ nghĩa: Thực hiện Kl trong khi Bt còn đúng	

5.5.3 Hoạt động của câu lệnh do

Theo trình tự:

1. Thực hiện khối lệnh trong thân do.
2. Khi gặp dấu ngoặc nhọn } đóng cuối cùng của thân do máy sẽ tính và xác định giá trị của Bt sau từ khóa while.
3. Tùy thuộc vào tính đúng sai của bt, máy sẽ lựa chọn 1 trong 2 nhánh:

a. Nếu Bt có giá trị bằng 0 (sai) máy sẽ ra khỏi vòng lặp và chuyển tới câu lệnh sau thân do.

b. Nếu Bt có giá trị khác 0 (đúng) máy sẽ trở về bước 1 để tiếp tục thực hiện vòng mới của vòng lặp.

5.11 Ví dụ:

Viết đoạn trình điều khiển việc nhập một số thực a khác 0.

```
do
{
        cout<< « \n Nhap a khac 0: a = « ;
        cin>>a;
}
while (a==0) ;
```

5.6 Câu lệnh goto và nhãn

5.6.1 Nhãn

Nhãn là một tên và có dấu hai chấm (:) đứng sau.
Nhãn có thể được gán cho bất kỳ câu lệnh nào trong chương trình.
Chẳng hạn: tiep_tuc: s + = x;
nghĩa là: tiep_tuc là nhãn câu lệnh gán s += x.

5.6.2 Câu lệnh goto

1. Cú pháp:

```
goto nhãn;
```

2. Tác dụng:

chương trình sẽ nhảy tới thực hiện câu lệnh có nhãn viết sau từ khóa goto;

Ghi chú:

1. Câu lệnh goto và nhãn chỉ nằm trong 1 hàm. goto chỉ cho phép nhảy từ vị trí này đến vị trí khác trong thân 1 hàm. Nó không thể nhảy từ hàm này sang hàm khác.
2. Không cho phép dùng câu lệnh goto nhảy từ ngòai khối lệnh vào trong khối lệnh. Nhưng ngược lại, tức là nhảy từ trong ra ngoài khối lệnh là hợp lệ.
3. Câu lệnh goto thường được dùng để thóat khỏi các câu lệnh switch, và các vòng lặp hoặc kết hợp với if tạo ra vòng lặp.

5.7 Các câu lệnh break, continue

5.7.1 break

Break cho phép ra khỏi switch , và các vòng lặp for, while, do mà không cần kiểm tra điều kiện kết thúc của vòng lặp.
Lệnh break cho phép thoát ra ngay khỏi vòng lặp bên trong nhất chứa lệnh break;
Mọi câu lệnh break có thể thay bằng câu lệnh goto và nhãn thích hợp.

5.7.2 continue

Lệnh continue tạo ra việc bắt đầu lặp lại của vòng lặp chứa nó.
Trong while và do, lệnh continue chuyển điều khiển về thực hiện ngay phần kiểm tra.
Đối với for, điều khiển được chuyển về bước khởi đầu lại. Continue chỉ áp dụng cho vòng lặp, không cho switch.

Ví dụ 5.12:

Viết hàm tính giá trị trung bình của các số không âm trong n số thực đọc vào từ bàn phím.

Input n số thực x;

Output Ttb = (Tổng các thực không âm)/(Số lượng các số không âm)

```
double Gttb(int  n)
{
        int  i, dem;
        double x,Ttb,sum;
        dem = 0;
        sum = 0;
        for(i=1; i<=n; i++)
        {
                cout<<"\nNhap x: x = ";cin>>x;
                if ( x < 0)
                        continue; //Bỏ qua các câu lệnh sau
                sum += x;
                dem++;
        }
        if (dem == 0)
                Ttb = 0;
        else
                Ttb =sum/dem;
          return Ttb;
}
```

5.8 Câu lệnh rỗng

Lệnh rỗng là lệnh chỉ có dấu chấm phảy (;)

(Lệnh này thường được sử dụng trong thân các vòng lặp mà ở đó không muốn có 1 lệnh nào.)

5.9 Vòng lặp vô hạn

Thường có các dạng sau:

a) Dạng for:

```
for (; ;)
        khối lệnh
```

Mệnh đề trong phần thân vòng lặp thường là mệnh đề kép chứa:

- Những mệnh đề mà nó thực hiện lặp lại để giải quyết bài toán.
- Một mệnh đề mà nó sẽ kết thúc sự thực hiện vòng lặp khi một điều kiện nào đó được thỏa. Mệnh đề này thường là if kết hợp với break.

b) Dạng while:

```
while(1)
        khối lệnh
```

c) Dạng do..while:

```
do
        khối lệnh
```

```
while(1);
```

Khi sử dụng vòng lặp ta cần lưu ý đến điều kiện lặp có thể dừng vòng lặp được không. Các vòng lặp dạng trên muốn dừng, cần phải có sự can thiệp của các câu lệnh break, goto, return …

Ví dụ 5.13:

```
        Hàm chọn menu.
    int Chon_Menu()
    {
        int chon;
        for (; ;)
        {
           Menu();
           cout<<"\nNhap Chon (1 <= Chon <= 6: ";
           cin>>Chon;
           if(1<= Chon && Chon<= 6)
               break;
        }
        return Chon;
    }
```

Ví dụ 5.14: tổ chức Menu chương trình

Viết chương trình thực hiện các thao tác trên số nguyên dương. Yêu cầu của chương trình là:

- In ra màn hình menu có các chức năng sau:

1. Tính $S_1 = \sum_{i=1}^{n} \frac{1}{i}$;
2. Tính $S_2 = \sum_{i=1}^{n} \frac{i+1}{i^2}$;
3. Tính $S_3 = \sum_{i=1}^{n} \frac{(-1)^i i}{i+1}$

- Muốn thực hiện thao tác nào thì chọn chức năng tương ứng của menu .

Thực hiện:

Bước 1: Tạo Project với tên "CT_Menu".

Bước 2: Tạo tập tin chương trình Menu.cpp

Bước 3: Trong tập tin Menu.cpp, soạn code theo cấu trúc:

```
// Chèn các tập tin thư viện cần thiết
//Chuong trinh tinh tong
#include <iostream>
using namespace std;

//Khai bao nguyen mau
//Cac ham to chuc menu
void Menu();
```

```
int ChonMenu();
void XL_Menu(int n, int Chon);
//Cac ham chuc nang
double Tong1(int n);
double Tong2(int n);
double Tong3(int n);
void main()
{
	int Chon, n;
	cout<<"\nNhap n = ";
	cin>>n;
	do
	{
		Chon = ChonMenu();
		XL_Menu(n,Chon);
	}
	while(1);
}
void Menu()
{
	cout<<"\n	BANG MENU		";
	cout<<"\n1. Tinh S1";
	cout<<"\n2. Tinh S2";
	cout<<"\n3. Tinh S3";
	cout<<"\n4. Thoat khoi chuong trinh!!!";
}
int ChonMenu()
{
	int Chon;
	for(;;)
	{
		Menu();
		cout<<"\nNhap Chon tu 1 -> 4: ";
		cin>>Chon;
		if (1 <= Chon && Chon <= 4)
			break;
	}
	return Chon;
}
void XL_Menu(int n, int Chon)
{
	switch(Chon)
	{
		case 1:
			cout<<"\n1.Tong S1:\n";
			cout<<"\nS1 = "<<Tong1(n);
			cout<<'\n';
```

```
                break;
            case 2:
                cout<<"\n1.Tong S2:\n";
                cout<<"\nS2 = "<<Tong2(n);
                cout<<'\n';
                break;
            case 3:
                cout<<"\n1.Tong S3:\n";
                cout<<"\nS3 = "<<Tong3(n);
                cout<<'\n';
                break;
            case 4:
                cout<<"\n4. Thoat khoi CT!\n";
                exit(1);
        }
}

double Tong1(int n)
{
        double S = 0;
        int i;
        for(i = 1; i <= n; i++)
                S += 1/(double)i;
        return S;
}

double Tong2(int n)
{
        double S = 0;
        int i;
        for(i = 1; i <= n; i++)
                S += (double)(i+1)/(i*i);
        return S;
}

double Tong3(int n)
{
        double S = 0;
        int i;
        for(i = 1; i <= n; i++)
                if(i%2==0)
                        S += (double)(i)/(i+1);
                else
                        S -= (double)(i)/(i+1);
        return S;
}
```

BÀI TẬP

Bài 1:

Viết chương trình giải phương trình bậc nhất ax+b=0.

Bài 2:

Viết chương trình tính giá trị lớn nhất của 4 số nguyên a, b, c, d được nhập từ bàn phím.

Bài 3:

Viết chương trình tìm ước số chung lớn nhất của 2 số nguyên dương a, b.

Bài 4:

Viết chương trình thực hiện các phép toán số học.

Nhập vào 2 số thực x, y và ký tự K, Tính:

K = '*', ketqua = a*b

K = '+', ketqua = a+b

K = '-', ketqua = a-b

K = '/', ketqua = a/b

Bài 5: (Trò chơi đoán số)

Máy tính " nghĩ ra " một số ngẫu nhiên (số nguyên) và không hiển thị số đó. Người chơi đoán số ấy bằng cách gõ từ bàn phím một số.

- Nếu số ấy nhỏ hơn số do máy tính nghĩ ra thì máy tính hiển thị thông báo câu "Nhỏ hơn".
- Nếu số ấy lớn hơn số do máy tính nghĩ ra thì máy tính hiển thị thông báo câu "Lớn hơn".

Giả sử người chơi chỉ có thể đoán nhiều nhất là k lần.

Nếu cả k lần người chơi đều đoán sai, chương trình hiển thị thông báo "Bạn đã thua", ngược lại người chơi thắng cuộc.

Viết chương trình thể hiện trò chơi.

Lưu ý:

Tạo số ngẫu nhiên (cho các lần thực hiện CT) sử dụng các hàm sau trong thư viện <**stdlib.h**>**:**

```
void srand( unsigned int seed );
int rand( void );
//srand((unsigned)time(NULL));
//X = rand();
time_t time(time_t *timer); //trong time.h
```

Bài 6:

Viết chương trình xuất bảng cửu chương từ 1 đến 9

Bài 7:

Xuất các hình:

```
**********          *          *****
*        *          **         ****
*        *          ***        ***
.        .          . . .      …

**********          ******     *
```

CHƯƠNG 6: CÁC CẤU TRÚC DỮ LIỆU CƠ BẢN

6.1 Mảng

6.1.1 Khái niệm

- Mảng là một tập các biến cùng kiểu được gọi chung bằng một tên.
- Các phần tử của mảng được được truy cập đến bởi chỉ số (index) của mảng.
- Mảng có thể có 1 chiều hay nhiều chiều. Dữ liệu của mảng được chứa trong một vùng nhớ liên tục.

6.1.2 Mảng 1 chiều

I) Khai báo:

KDL Ten_Mang[KT];

Trong đó:

- KDL là kiểu dữ liệu của mảng, có thể là char, int, float,...
- Ten_Mang là một tên, chỉ tên của mảng.
- KT: Là một số nguyên dương chỉ kích thước khai báo của mảng: xác định số các phần tử của mảng.

Ví dụ 6.1:

int a[10];

Trong đó:

- KDL: int (Kiểu dữ liệu của các phần tử của mảng).
- Ten_Mang: a
- Kích thước: 10

Khai báo trên xác định mảng 1 chiều tên là a, có 10 phần tử, mỗi phần tử của mảng là số nguyên.

II) Chỉ số của mảng:

Mỗi phần tử của mảng được xác định bởi chỉ số của mảng. Chỉ số của mảng phải có giá trị int không vượt quá kích thước của mảng. Chỉ số đầu tiên của mảng luôn là 0. Ký pháp a[i] để chỉ phần tử thứ i của mảng a.

Các phần tử của a được đánh số (mặc định) bởi: a[0], a[1],...

III) Lấy địa chỉ của mảng:

Các phần tử của mảng 1 chiều có địa chỉ liên tiếp nhau trong bộ nhớ.

Lấy địa chỉ của phần tử mảng 1 chiều bằng phép toán &, với cú pháp:

&a[i] // a[i] là phần tử thứ i của mảng a

Ghi chú: Địa chỉ đầu của mảng là tên mảng, vậy ta có: a == &a[0]

IV) Kích thước bộ nhớ (số bytes) được sử dụng để lưu trữ mảng 1 chiều là:

Tong (Bytes) = sizeof(KDL)* KT

V) Một số thao tác thường gặp trên mảng một chiều:

1. Nhập và xuất dữ liệu cho mảng 1 chiều:

Thường liên kết với vòng lặp for.

Ví dụ 6.2:

- Hàm nhập dữ liệu cho mảng 1 chiều có n phần tử là số nguyên

```
void Nhap (int a[MAX], int n)
{
```

```
        int i;
        for (i = 0; i < n; i++)       //Duyệt mảng theo chỉ số
        {
            cout<<"\na["<< i<<"] = ";
            cin>>a[i];      //Nhập dữ liệu cho từng phần tử
        }
}
```

- Hàm xuất mảng 1 chiều n số nguyên ra màn hình

```
void Xuat (int a[MAX], int n)
{
        int i;
        for (i = 0; i < n; i++)//Duyệt mảng theo chỉ số
            cout<<a[i]<<'\t';
}
```

2. Sắp xếp một mảng theo thứ tự tăng hay giảm.

Có nhiều thuật toán sắp xếp, ở đây ta giới thiệu thuật toán nổi bọt (buble)

Ví dụ 6.3:

Sắp tăng dần dãy n số nguyên theo thuật toán nổi bọt

```
void Buble (int a[MAX], int n)
{
        int tam, i;
        for (i = 0;i < n-1; i++)
            for (j = i + 1; j < n; j++)
                if(a[i] > a[j] )
                {
                    tam = a[i];
                    a[i] = a[j];
                    a[j] = tam;
                }
}
```

3. Gán mảng:

Hai mảng a, b cùng kiểu, cũng không thể thực hiện trực tiếp việc gán a cho b bằng câu lệnh gán:

```
b = a;
```

Ta chỉ thực hiện được việc gán a cho b bằng cách gán giá trị từng phần tử của a tương ứng cho từng phần tử của b.

Ví dụ 6.4:

Cài đặt hàm gán mảng a cho mảng b, kích thước mảng là n:

```
void gan(int a[], int b[], int n)
{
    for (int i = 0; i < n; i++)
      b[i] = a[i];
}
```

VI) Hàm và mảng 1 chiều

- Hàm không thể trả về một trị là mảng 1 chiều.

- Đối của hàm là tên của mảng 1 chiều: Khi đó tham số thực truyền cho đối tương ứng cũng là tên của mảng 1 chiều, cùng kiểu và cùng kích thước với đối. Tên của tham số thực và tên của đối có thể trùng nhau hoặc khác nhau.

Đối	Tham số thực
Tên mảng 1 chiều	Tên mảng 1 chiều (cùng kiểu, kích thước với đối)

6.1.3 Mảng 2 chiều

Trong các mảng nhiều chiều, hình thức đơn giản nhất là mảng 2 chiều.

I) Cách tiếp cận và khai báo:

Mảng 2 chiều là mảng 1 chiều của mảng 1 chiều.

Cú pháp khai báo như sau:

KDL Ten_Mang[KT1][KT2];

trong đó:

- KDL: kiểu của các phần tử của mảng.
- Ten_Mang: Tên của mảng 2 chiều
- KT1: Kích thước chiều 1, là số nguyên dương.
- KT2: Kích thước chiều 2, là số nguyên dương.

Ví dụ 6.5:

double a[2][3];

Với khai báo này xác định một mảng 2 chiều, các phần tử của mảng có kiểu double, kích thước chiều 1 là 2, kích thước chiều 2 là 3.

II) Chỉ số của mảng:

Mảng 2 chiều được xem như là một ma trận có KT1 hàng (dòng), KT2 cột.

Ta dùng ký pháp a[i][j] để chỉ phần tử tổng quát hàng i cột j. Các chỉ số i, j là các số nguyên không âm không vượt quá các kích thước tương ứng. Mặc định các chỉ số i, j bắt đầu từ 0, tuy nhiên ta có thể ấn định chỉ số đầu tiên bằng một giá trị khác, chẳng hạn a[1][1].

Theo ví dụ trên, Mảng a có 6 phần tử có kiểu double , được đánh số và sắp xếp như sau:

a[0][0]	a[0][1]	a[0][2]
a[1][0]	a[1][1]	a[1][2]

Mảng hai chiều a biểu diễn 1 ma trận 2 hàng 3 cột. Phần tử tổng quát hàng i cột j là:

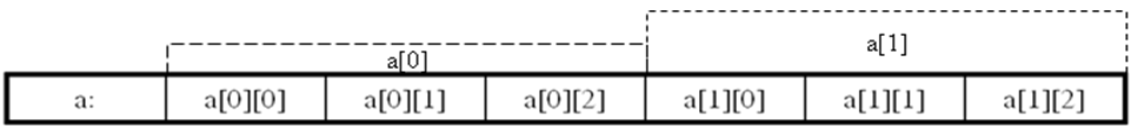
a[i][j] ; với 0 <= i < 2, 0 <= j < 3.

Ghi chú:

KT1, KT2 là các số nguyên dương xác định trước, gọi là kích thước khai báo. Trong thực tế, mỗi khi sử dụng, ta thường dùng số hàng m <= KT1 và số cột n <= KT2. Nếu số hàng = số cột, ta gọi là ma trận vuông.

III. Lưu trữ trong bộ nhớ:

Mảng 2 chiều lưu trữ trong máy trên một vùng nhớ liên tục theo hàng, chẳng hạn với mảng a trên, a lưu trử trong bộ nhớ như sau:



IV) Công thức tính số bytes cần thiết trong bộ nhớ lưu trư mảng 2 chiều:

Tong (Bytes) = sizeof(KDL)* KT1* KT2.

Trong ví dụ trên, có:

- sizeof(double) = 8
- KT1 = 2
- KT2 = 3

Tong (Bytes) = 2*3*8 = 48 bytes.

V) Nhập, xuất mảng 2 chiều (ma trận):

Thường liên kết với 2 vòng for duyệt theo chỉ số để nhập dữ liệu cho từng phần tử, hoặc xuất dữ liệu của từng phần tử ra màn hình.

Ví dụ 6.6:

Hàm nhập ma trận a các số nguyên, có m hàng, n cột:

```
void NhapMT (int a[MAX][MAX], int m, int n)
{
	int i, j;
	for (i = 0; i < m ; i++) // hang i
		for (j = 0; j < n ; j++)  //cot j
		{
			cout<<"a["<<i<<"]["<<j<<"]= ";
			cin>>a[i][j];
		}
}
```

Ghi chú:

Trường hợp là ma trận vuông, khi đó m=n, ta chỉ cần dùng 1 đối n:

```
 void NhapMT (int a[MAX][MAX], int n)
{
	int i, j;
	for (i = 0; i < n ; i++) // hang i
		for (j = 0; j < n ; j++)  //cot j
		{
			cout<<"a["<<i<<"]["<<j<<"]= ";
			cin>>a[i][j];
		}
}
```

Ví dụ 6.7:

Hàm xuất ma trận a các số nguyên, có m hàng, n cột:

```
void XuatMT (int a[MAX][MAX], int m, int n)
{
	int i, j;
	for (i = 0; i < m ; i++) // hàng i
	{
		cout<<'\n'; //xuống hàng
		for (j = 0; j < n ; j++)  //các phần tử nằm trên hàng i (cột j)
			cout<<a[i][j]<<'\t'; //2 giá trị cách nhau 1 tab
	}
}
```

IV) Hàm và mảng 2 chiều:

- Hàm không thể trả về giá trị là một mảng 2 chiều.
- Đối của hàm là tên của mảng 2 chiều: Khi đó tham số thực truyền cho đối tương ứng cũng là tên của mảng 2 chiều, cùng kiểu và cùng kích thước với đối. Tên của tham số thực và tên của đối có thể trùng nhau hoặc khác nhau.

Đối	Tham số thực
Tên mảng 2 chiều	Tên mảng 2 chiều (cùng kiểu, kích thước với đối)

Ví dụ 6.8:

Các hàm nhập, xuất ma trận ở trên có thể sử dụng như sau:

```
int a[MAX][MAX];
```

- NhapMT(a,2,3);
- XuatMT(a,2,3);

6.1.4 Kiểu mảng

Dùng từ khoá typedef để định nghĩa, bằng cách trước định nghĩa mảng thông thường, ta đặt từ khoá typedef.

Ví dụ 6.9:

a) typedef float D*ay10*[10];// Day*10* là kiểu dữ liệu mảng 1 chiều kích thước 10
Day10 a,b; // a,b là 2 biến kiểu mảng 1 chiều có kích thước 10.

b) typedef float M*at10_20* [10][20];
// M*at10_20* là kiểu dữ liệu mảng 2 chiều kích thước 10x20
Mat10_20 c,d; // c,d là 2 biến kiểu mảng 2 chiều có kích thước 10x20

6.1.5 Khởi đầu cho các mảng

1. Sử dụng các hằng
2. Khi khởi đầu có thể không cần chỉ ra kích thước của mảng. Khi đó máy sẽ dành cho mảng 1 vùng nhớ đủ để thu nhận danh sách giá trị khởi đầu. Trong trường hợp là mảng 2 chiều thì kích thước chiều 2 phải chỉ ra:
3. Đối với mảng 2 chiều có thể khởi đầu theo cách số giá trị của mỗi hàng có thể khác nhau với điều kiện là:
    - Kích thước chiều 2 phải được chỉ ra.
    - Kích thước chiều 1 hoặc là không được chỉ ra, hoặc có chỉ ra thì giá trị phải lớn hơn kích thước thực sự của mảng

Ví dụ 6.10:

```
int a[6][3] = {
                {1,2},
                {3},
                {4,5,6},
                {0,3,1}
            };
// Thực sự đây là mảng a[3][3]
```

6.1.6 Các kỹ thuật xử lý cơ bản trên mảng

1. Kỹ thuật thử và sai

Cần xác định Kq khi biết Kq$\in\{a_1,..,a_n\}$. Ta dùng phép thử cho đến khi nào đúng:

- Cho Kq = a_1;
- Duyệt các phần tử còn lại để chính xác giá trị Kq.

Ví dụ 6.11:

Viết hàm tính giá trị lớn nhất của dãy a_0, .., a_{n-1}

```
int GT_Max (int a[MAX], int n)
{
	int GTLN, i;
	GTLN = a[0];
	for(i = 1; i < n; i++)
		if( GTLN < a[i])
			GTLN = a[i];
	return GTLN;
}
```

2. Kỹ thuật duyệt

- Toàn cục: duyệt tất cả các phần tử của tập hợp.
- Cục bộ: Chỉ trên một miền con của tập hợp có thể xét. Miền con này xác định từ các giá trị của hàm, hay là lập bảng.

Ví dụ 6.12:

Viết hàm trả về tổng các phần tử của ma trận vuông

```
//Duyệt toàn bộ: Cả mảng
int Tong(int a[MAX][MAX], int n)
{
	int i,j,S = 0;
	for(i = 0; i < n; i++)
		for(j = 0; j < n; j++)
			S += a[i][j];
	return S;
}
```

Ví dụ 6.13:

Viết hàm trả về tổng các số nguyên tố trong một dãy số nguyên

```
//Miền con xác định bằng giá trị của hàm.
int Tong_Nt(int a[MAX], int n)
{
	int i, S = 0;
	for( i = 0; i < n; i++)
		if ( Kt_Nt(a[i]))
			S += a[i];
	return S;
}
```

Ví dụ 6.14:

Sắp dãy n số nguyên theo yêu cầu:

- Các số âm ở đầu mảng và tăng
- Các số 0 giữa mảng.
- Các số dương cuối mảng và giảm.

Miền con xác định bằng cách lập bảng (a_i đứng trước a_j):

a[i] \ a[j]	< 0	0	>0
< 0	ĐK hoán đổi a[i] > a[j]	X	X
0	Hoán đổi	X	X
> 0	Hoán đổi	Hoán đổi	ĐK hoán đổi a[i]< a[j]

```
void Sap_Am_0_Duong(int a[MAX], int n)
{
	int i,j,MC,Tam;
	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)
		{
			Tam = a[i];
			a[i] = a[j];
			a[j] = Tam;
		}
	  }
}
```

3. Kỹ thuật kiểm tra tính đúng, sai:

Dạng 1 (bài toán AND):

- Đúng: nếu ∀i, a_i phải thỏa mãn.
- Sai: nếu ∃i, a_i không thỏa mãn.

Cách thực hiện như sau:

- Kq = 1; // Đúng
- Duyệt để tìm điều kiện gán Kq = 0; // Sai

Ví dụ 6.15:

Viết hàm kiểm tra một dãy các số có phải là dãy tăng.

```
int Kt_DTang(int a[MAX], int n)
{
	int i, Kq = 1;
	for(i = 0; i < n-1 && Kq; i++)
	  if( a[i] > a[i+1] )
		Kq = 0;
	return Kq;
}
```

Dạng 2 (bài toán OR):

- Đúng: nếu ∃i, a_i thỏa mãn.
- Sai: nếu ∀i, a_i không thỏa mãn.

Cách thực hiện như sau:

- Kq = 0; //Sai
- Duyệt để tìm điều kiện gán Kq = 1; // Đúng

Ví dụ 6.16:

Tìm x có trong mảng hay không? - Nếu không trả về 0, nếu có trả về 1

```
//Không có: x ≠ ai ;  ∀i;
//Có: x = ai ;  ∃i;
 int Tim_Pt(int a[MAX], int n, int x)
{
      int i, Kq = 0;
      for(i = 0; i < n ; i++)
            if( a[i] == x )
            {
                  Kq = 1;
                  break;
            }
      return Kq;
}
```

6.2 Xâu ký tự

6.2.1 Định nghĩa

Một xâu ký tự (chuỗi) là mảng một chiều các ký tự được kết thúc bởi ký tự NULL (\0).

Số lượng các ký tự khác NULL trong xâu gọi là chiều dài của xâu.

6.2.2 Khai báo

a. Xâu ký tự 8 bit:

```
char a[KT];
```

- a là tên, chỉ tên của xâu ký tự.
- KT là số nguyên dương, chỉ kích thước của xâu ký tự.
- a[i] là ký tự thứ i của a. Chỉ số đầu tiên luôn là 0.

Trong khi khai báo, ta phải khai báo xâu ký tự có chiều dài lớn hơn mảng được sử dụng 1 ký tự để đủ chỗ chứa ký tự NULL.

b. Xâu ký tự 16 bit:

```
wchar a[KT];
```

6.2.3 Kiểu xâu ký tự

Cách tạo là đặt từ khoá typedef trước khai báo xâu thông thường.

6.2.4 Các thao tác nhập xuất xâu ký tự

1. Xuất:

cout<< a;

Khi đó toán tử << của iostream sẽ in từng ký tự trong a cho đến khi gặp mã của NULL (\0).

2. Nhập:

cin>> a ;

Với a là một xâu ký tự không chứa ký tự tách (khoảng trắng, tab …), vì iostream xem các ký tự đó là ký tự tách chứ không phải dữ liệu.

Để nhập một xâu ký tự có chứa ký tự tách thì có thể dùng các cách sau:

a) Dùng hàm gets trong tệp tiêu đề <stdio.h> hoặc trong <string.h>:

gets(a);

b) Dùng hàm thành phần của iostream:

Cin.getline(a, so_ky_tu);

Hàm này đọc vào tối đa so_ky_tu – 1 ký tự, kể cả ký tự trắng.

Các xử lý thường gặp trên xâu ký tự như xác định chiều dài xâu, chép xâu s vào xâu t, nối 2 xâu,....

6.2.5 Khởi đầu cho xâu ký tự

Cũng theo quy tắc chung của mảng (1 chiều) đã giới thiệu ở trên. khởi đầu của mảng 1 chiều ký tự có thể là:

- Danh sách các hằng ký tự, cuối cùng là ký tự '\0'.
- Hoặc là một hằng xâu ký tự.

6.2.6 Hàm và xâu ký tự

Hàm không thể trả về một giá trị là xâu ký tự.

Đối của hàm có thể là tên xâu ký tự, khi đó tham số thực phải là tên xâu ký tự

Đối	Tham số thực
Xâu ký tự	Tên xâu ký tự

Ví dụ 6.17:

Viết hàm trả về chiều dài xâu ký tư

(Chiều dài xâu ký tự: là số lượng các ký tự trong xâu khác ký tự NULL)

```
int ChieuDai(char s[MAX])
{
    int l = 0;
    while (s[l] != NULL)
      l++;
    return l;
}
```

Ví dụ 6.18:

Viết hàm đảo ngược xâu ký tự s, kết quả lưu trử lại vào s.

Input: s

Output: s (đã đảo ngược)

```
void DaoNguoc(char s[MAX])
{
    int i, j, h;
```

```
    char Tam;
    h = ChieuDai(s);
    for(i = 0, j = h-1; i < j ; i++, j--)
    {
      Tam = s[i];
      s[i] = s[j] ;
      s[j] = Tam;
    }
}
```

6.2.7 Mảng các xâu ký tự

- Cách tiếp cận như là mảng 2 chiều các ký tự.
- Khai báo:

  char a[KT1][KT2];
- Nhập hoặc xuất mảng các xâu ký tự là nhập hoặc xuất từng xâu ký tự của mảng.
- Khởi đầu:
  Khởi đầu từng xâu ký tự bằng các hằng xâu ký tự.

Ví dụ 6.19:

```
    char a[2][80] ={
                    "Da Lat",
                    "Binh minh"
                  };
```

6.3 Kiểu cấu trúc

6.3.1 Khái niệm

Cấu trúc là tập hợp các biến, mảng (có thể khác kiểu dữ liệu) và được biểu thị bởi một tên duy nhất.

6.3.2 Khai báo

Trong các khai báo kiểu cấu trúc, luôn sử dung từ khóa **struct**.

a) Dạng 1: (Định nghĩa kiểu cấu trúc và khai báo biến cấu trúc riêng)

```
    struct KCT
    {
        //Khai báo các thành phần của nó
    }; // Có dấu chấm phảy
```

Trong đó:

- KCT: Là tên tự đặt, chỉ tên của kiểu dữ liệu cấu trúc.
- Mỗi thành phần cấu trúc ta gọi là trường, có dạng như định nghĩa thông thường các biến:

  KDL ten_bien;

Sau khi định nghĩa một kiểu dữ liệu có tên là KCT, ta có thể khai báo biến có kiểu là KCT, biến này gọi là biến cấu trúc (hay gọi đơn giản là cấu trúc):

  KCT Ten_cau_truc;

Ví dụ 6.20:

```
struct DATE
        {
                int  dd;
                int  mm;
                int  yyyy;
        };
```

Khai báo này mô tả một kiểu cấu trúc có tên là DATE, có 3 trường dữ liệu là: dd, mm, yyyy .

Sau đó ta có thể khai báo biến có kiểu cấu trúc DATE:

```
DATE  Ntn;
```

Ví dụ 6.21:

```
struct  PERSON
        {
                char Hoten[MAX];
                DATE Ntn;
                char Diachi[MAX];
                int SDT;
                double Luong;
        };
```

Khai báo này xác định một kiểu dữ liệu cấu trúc có tên là: PERSON, có các thành phần cấu trúc như trong định nghĩa. Từ đó ta có thể khai báo các biến cấu trúc có kiểu cấu trúc PERSON. Chẳng hạn:

```
PERSON  p1, p2;
```

b) Dạng 2:(Định nghĩa kiểu cấu trúc và khai báo biến cấu trúc đồng thời)

```
struct KCT
{
        //Khai báo các thành phần của nó
}Danh_sách_các_biến_cấu_trúc;
```

Ví dụ 6.22:

```
struct  PERSON
        {
                char Hoten[MAX];
                DATE Ntn;
                char Diachi[MAX];
                int SDT;
                double Luong;
        }p1, p2;
```

Với cách này, ta đã định nghĩa được một kiểu cấu trúc đặt tên là PERSON có các trường dữ liệu tương ứng, đồng thời khai báo 2 biến cấu trúc p1, p2 có kiểu PERSON.

c) Dạng 3:(khai báo biến cấu trúc không có tên của kiểu cấu trúc)

```
struct
{
        //Khai báo các thành phần của nó
}Danh_sách_các_biến_cấu_trúc;
struct
{
        int  ngay;
```

```
        int  thang;
        int  nam;
    } d1, d2;
```

Khai báo này xác định 2 biến cấu trúc là d1, d2 có các thành phần cấu trúc như khai báo.

6.3.3 Truy cập đến các thành phần của cấu trúc

Theo cú pháp:

```
Ten_cau_truc.tên_thành_phần;
```

Nếu thành phần lại là một cấu trúc, thì tương tự:

```
ten_cau_truc.ten_cau_truc.tên_thành_phần;
```

Ví dụ 6.23:

Với khai báo:

```
PERSON  p;
```

Ta có thể truy nhập vào các trường của p như sau:

```
p.Hoten
p.Ntn.dd
p.Ntn.mm
p.Ntn.yyyy
p.Luong
```

Ghi chú:

sau khi truy cập vào các thành phần dữ liệu, dữ liệu sẽ được xử lý theo yêu cầu bài toán.

Chẳng hạn,

- Nhập họ tên cho p: gets(p.Hoten);
- Nhập ngày sinh cho p: cin>>p.Ntn.dd;
- Xuất Địa chỉ của p: cout<<p.Diachi;

6.3.4 Các thao tác trên các cấu trúc:

1. Có thể thực hiện các phép toán gán, lấy địa chỉ trên các biến của thành phần cấu trúc như thực hiện trên các kiểu dữ liệu đã biết.
2. Nhập/Xuất dữ liệu cho một cấu trúc:
   Nhập, xuất từng thành phần dữ liệu.
3. Khởi đầu cho một cấu trúc:
   Có thể khởi đầu cho cấu trúc ngoài, cấu trúc tỉnh bằng cách viết vào sau khai báo của chúng một danh sách các giá trị cho các thành phần.
4. Phép gán cấu trúc:
   Hai cấu trúc cùng kiểu có thể gán cho nhau.

6.3.5 Định nghĩa kiểu cấu trúc bằng từ khóa typedef

Đặt trước một định nghĩa cấu trúc từ khóa typedef.

Dùng một trong 2 cách sau:

Cách 1:(Đặt tên kiểu cấu trúc)

```
typedef struct
{
    Các thành phần
```

}KCT;

Đặt kiểu cấu trúc (không tên) là KCT.

Cách 2: (Đổi tên kiểu cấu trúc)

typedef struct _KCT

{

Các thành phần

}KCT;

Đổi tên kiểu cấu trúc _KCT thành tên KCT

6.3.6 Lưu trử trong bộ nhớ

Chương trình cấp phát cho mỗi cấu trúc một vùng nhớ liên tục chứa đủ các thành phần dữ liệu của nó.

Ví dụ 6.24:

Với khai báo:

PERSON p;

Giả sử MAX = 20, chương trình cấp cho p một vùng nhớ liên tục có kích thước sau:

Byte	Trường dữ liệu
20	char Hoten[MAX]
12	DATE Ntn
20	char Diachi[MAX]
4	int SDT
8	double Luong
Tổng: 64 bytes	

6.4 Mảng cấu trúc

6.4.1 Cách tiếp cận

Mảng cấu trúc là mảng 1 chiều các phần tử có cùng 1 kiểu cấu trúc.

Ví dụ 6.25:

PERSON List[50];

Khai báo này xác định mảng 1 chiều, tên là List, kích thước 50, các phần tử là các biến cùng kiểu cấu trúc PERSON .

6.4.2 Các thao tác trên mảng cấu trúc

1. Khởi đầu:
   a. Khởi đầu các mảng cấu trúc ngoài, tĩnh. Chúng sẽ nhận giá trị 0 nếu không khởi đầu.
   b. Việc khởi đầu được thực hiện 1 lần khi dịch chương trình.
   c. Cách khởi đầu: Khởi đầu cho các thành phần của từng cấu trúc.

Ví dụ 6.26:

struct MONTH

{

```
        int number;
        char  name[12];
}Mang[12] = {
                {1, ”giêng”},
                {2, ”Hai”},
                 . . .
                {11, ”Mười một”},
                {12, “chạp”} // cuối: không có dấu phẩy
               };                        // Có dấu chấm phẩy
```

Khai báo trên xác định một mảng cấu trúc có tên là Mang, gồm 12 phần tử, mỗi phần tử là một cấu trúc kiểu MONTH, và danh sách 12 bộ khởi đầu cho 12 cấu trúc tương ứng, theo cú pháp trên.

2. Nhập, xuất:
   Nhập, xuất từng cấu trúc của mảng.
3. Phép gán:
   Có thể thực hiện:
   a. Gán biến cấu trúc cho phần tử mảng cấu trúc. (cùng kiểu)
   b. Gán phần tử mảng cấu trúc cho biến cấu trúc (cùng kiểu).
   c. Gán 2 phần tử của 1 mảng cấu trúc cho nhau.

6.4.3 Hàm và cấu trúc

Giá trị trả về của hàm có thể là một giá trị cấu trúc.
Đối của hàm có thể là

- Tên của một cấu trúc: Khi đó, tham số thực truyền cho đối tương ứng phải là một cấu trúc cùng kiểu.
- Tên của một mảng cấu trúc: Khi đó, tham số thực truyền cho đối tương ứng phải là một mảng cấu trúc cùng kiểu, cùng kích thước.

Đối	Tham số thực
Cấu trúc	Giá trị cấu trúc (cùng kiểu)
Tên của mảng cấu trúc	Tên của mảng cấu trúc (cùng kiểu)

Ví dụ 6.27:

Để quản lý nhân viên ta xây dựng cấu trúc gồm:

- **Mã nhân viên,**
- **Họ nhân viên,**
- **Tên nhân viên,**
- **Địa chỉ,**
- **Điện thoại,**
- **Mức lương,**
- **Phòng làm việc.**

Viết chương trình tổ chức menu thực hiện các chức năng sau:

- Xem danh sách nhân viên.
- Xem danh sách nhân viên thuộc một phòng.
- Sắp xếp danh sách nhân viên theo thứ tự giảm dần của mức lương.

Thực hiện:

Bước 1: Tạo Project với tên "QL_Nhanvien".

Bước 2: Tạo tập tin chương trình Nhanvien.cpp

Bước 3: Trong tập tin Nhanvien1.cpp, soạn code theo cấu trúc:

```
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <iomanip>
#include <string.h>

#define MAX 100
using namespace std;

//Dinh nghia kieu du lieu
struct NHANVIEN
{
	char MaNV[10];
	char HoNV[15];
	char TenNV[10];
	unsigned int Tuoi;
	char Dc[10];
	int Sdt;
	double Luong;
	char Phong[20];
};
//Khai bao nguyen mau ham
//Nhap,xuat
void Nhap ( NHANVIEN  Ds[MAX], int n);
void Xuat ( NHANVIEN  Ds[MAX], int n);
//Chuc nang
void DSPhong ( NHANVIEN  Ds[MAX], int n, char Phong[20]);
void DSLuong_Tang ( NHANVIEN  Ds[MAX], int n);
//To chuc menu
void XL_Menu(NHANVIEN Ds[MAX], int n, int Chon);
int ChonMenu();
void Menu();

void main()
{
	NHANVIEN Ds[MAX];
	int n, Chon;
```

```
	cout<<"\nNhap n = ";
	cin>>n;
	Nhap (Ds,n);
	do
	{
		Chon = ChonMenu();
		XL_Menu(Ds,n,Chon);
	}
	while(1);
}
void Menu()
{
	cout<<"\n	BANG MENU		";
	cout<<"\n1. Xem danh sach";
	cout<<"\n2. Xem danh sach theo phong";
	cout<<"\n3. Danh sach tang theo luong";
	cout<<"\n5. Thoat khoi chuong trinh!!!";
}
int ChonMenu()
{
	int Chon;
	for(;;)
	{
		Menu();
		cout<<"\nNhap Chon tu 1 -> 5: ";
		cin>>Chon;
		if (1 <= Chon && Chon <= 5)
			break;
	}
	return Chon;
}
void XL_Menu(NHANVIEN Ds[MAX], int n, int Chon)
{
	char Phong[20];
	switch(Chon)
	{
		case 1:
			cout<<"\n1. Xem danh sach";
			cout<<"\nDanh sach nhan vien:\n";
			Xuat(Ds, n);
			cout<<'\n';
			_getch();
			break;
		case 2:
			cout<<"\n2. Xem danh sach theo phong";
			cout<<"\nDanh sach nhan vien:\n";
			Xuat(Ds, n);
```

```
                    _flushall();
                    cout<<"\nXem phong nao: ";
                    gets(Phong);
                    cout<<"\nDanh sach nhan vien phong "<<Phong<<":\n";
                    DSPhong ( Ds, n,Phong);
                    cout<<'\n';
                    _getch();
                    break;
            case 3:
                    cout<<"\n3. Danh sach tang theo luong";
                    cout<<"\nDanh sach nhan vien:\n";
                    Xuat(Ds, n);
                    _flushall();
                    DSLuong_Tang (Ds,n);
                    cout<<"\nDanh sach nhan vien tang theo luong:\n";
                    Xuat(Ds, n);
                    cout<<'\n';
                    _getch();
                    break;
            case 5:
                    cout<<"\n5. Thoat khoi CT!\n";
                    exit(1);
        }
}
void Nhap ( NHANVIEN  Ds[MAX], int n)
{
        int i;
        for( i = 0; i < n; i++)
        {
                cout<<"\nNhap thong tin nguoi thu "<< i+1;
                cout<<"\nMa nhan vien: ";
                gets(Ds[i].MaNV);
                cout<<"\nHo nhan vien: ";
                gets(Ds[i].HoNV);
                cout<<"\nTen nhan vien: ";
                gets(Ds[i].TenNV);
                cout<<"\nTuoi: ";
                cin>>Ds[i].Tuoi;
                cout<<"\nDia chi: ";
                gets(Ds[i].Dc);
                cout<<"\nSo dien thoai: ";
                cin>>Ds[i].Sdt;
                cout<<"\nLuong: ";
                cin>>Ds[i].Luong;
                cout<<"\nPhong: ";
                gets(Ds[i].Phong);
        }
```

```
}
void Xuat ( NHANVIEN  Ds[MAX], int n)
{
	int i;
	for( i = 0; i < n; i++)
	{
		cout<<'\n';
		cout<<setiosflags(ios::left)
		<<setw(10)<< Ds[i].MaNV
		<<setw(15)<<Ds[i].HoNV
			<<setw(10)<<Ds[i].TenNV
			<<setw(4)<<Ds[i].Tuoi
			<<setw(10)<< Ds[i].Dc
			<<setw(10)<< Ds[i].Sdt
			<<setw(10)<< Ds[i].Luong
			<<setw(10)<< Ds[i].Phong ;
	}
}
//Xem danh sach cua mot phong
void DSPhong ( NHANVIEN  Ds[MAX], int n, char Phong[20])
{
	int i;
	for( i = 0; i < n; i++)
	{
		if(stricmp(Phong, Ds[i].Phong)==0)
		{
			cout<<'\n';
			cout<<setiosflags(ios::left)
				<<setw(10)<< Ds[i].MaNV
				<<setw(15)<<Ds[i].HoNV
				<<setw(10)<<Ds[i].TenNV
				<<setw(4)<<Ds[i].Tuoi
				<<setw(10)<< Ds[i].Dc
				<<setw(10)<< Ds[i].Sdt
				<<setw(10)<< Ds[i].Luong;
		}
	}
}
//Sap danh sach tang dan theo luong
void DSLuong_Tang ( NHANVIEN  Ds[MAX], int n)
{
	int i, j;
	NHANVIEN Tam;
	for( i = 0; i < n-1; i++)
		for(j=i+1; j < n; j++)
			if(Ds[i].Luong > Ds[j].Luong)
			{
```

```
                    Tam = Ds[i];
                    Ds[i] = Ds[j];
                    Ds[j] = Tam;
                }
}
```

6.5 union

union gồm nhiều thành phần cũng như cấu trúc, nhưng khác ở chỗ là các thành phần của union được cấp phát chung một vùng nhớ. Độ dài của union bằng độ dài của thành phần lớn nhất.

Định nghĩa kiểu union, khai báo biến union, mảng union, con trỏ union, cũng như cách truy nhập đến các thành phần union được thực hiện hoàn toàn tương tự như đối với cấu trúc.

Một cấu trúc có thể có thành phần kiểu union, ngược lại các thành phần của union lại có thể là cấu trúc.

Ví dụ 6.28:

```
union VAL
    {   unsigned int i;
        float v;
        unsigned char ch[2];
    };
val a,b,x[10];
```

Khai báo trên xác định một kiểu union có tên là VAL, có 3 thành phần là i có kiểu unsigned int, v có kiểu float, và ch có kiểu unsigned char, ba biến này cùng sử dụng chung một vùng nhớ.

Kích thước của VAL bằng kích thước của v và bằng 4, đó là kích thước lớn nhất trong 3 thành phần của Val . Tiếp theo là khai báo các biến union a,b và mảng unuion x có kiểu val.

6.6 Kiểu enum

Kiểu enum định nghĩa một tập hợp các hằng ký hiệu mà mỗi giá trị là một số nguyên. Việc định nghĩa này nhờ từ khóa enum.

6.6.1 Khai báo

Cú pháp khai báo như sau:

```
Enum TEN_ENUM
{
    TH1,
    TH2,
    ...
    THn_1,
    THn    // không có dấu phảy ;
};      // Có dấu chấm phảy ;
```

Trong đó, TEN_ENUM, THk là tên, chỉ tên của các hằng. Các THk mang giá trị tương ứng với thứ tự của nó trong danh sách, thứ tự đầu là 0.

TH1 có giá trị 0.

TH2, có giá trị 1.
...
THn, có giá trị n-1.

Sau đó có thể khai báo biến:

TEN_ENUM Bien;

Tất cả các phép toán trên số nguyên đều dùng được cho các giá trị kiểu enum.

6.6.2 Khởi đầu cho enum

- Gán Các hằng cho các tên hằng.
- Nếu có tên hằng nào không được khởi động tường minh, thì ngầm định nó có giá trị nguyên lớn hơn tên hằng kế trước 1 đơn vị. Một giá trị có thể được gán cho nhiều tên hằng.

BÀI TẬP

Cho a[0..n-1] là mảng các số nguyên, x là một số nguyên.
Cài đặt các hàm sau:

Bài 1: (Bài toán tìm kiếm.)
1. Tìm x có trong a ? Nếu có trả về chỉ số tương ứng (đầu tiên, cuối cùng). Nếu không trả về -1.
2. Tìm số nguyên tố trong a ? Nếu có trả về chỉ số tương ứng (đầu tiên, cuối cùng). Nếu không trả về -1.

Bài 2: (Bài toán Đếm.)
1. Dem : Đếm số lần xuất hiện của x trong a.
2. Dem_Am: Đếm Các số âm
3. Dem_Duong: Đếm Các số dương.
4. Dem_Nt : Đếm các số nguyên tố.
5. Đếm số lượng các đường chạy.
(Đường chạy: Dãy con có thứ tự dài nhất gồm những phần tử kế tiếp.)

Bài 3: (Kiểm tra tính đúng sai.)
Kiểm tra các phát biểu:
1. a không chứa 0.
2. a có thứ tự tăng.
3. a chứa ít nhất 3 phần tử liên tiếp trùng nhau..
4. a chỉ chứa 2 giá trị.
5. a chỉ chứa các giá trị từ 0 đến n-1.
6. Nếu a có chưa phần tử 0 thì phải chứa phần tử có giá trị 1.
7. Giả thiết a, b cùng có n phần tử . Kiểm tra a, b có phải là hoán vị của nhau.

Bài 4: (Bài toán tính Max.)
1. Max: Tính $\max(a_1,..,a_n)$.
2. Cs_Max : Tìm chỉ max : Trả về chỉ số đầu tiên đạt $\max(a_1,..,a_n)$.
3. Cs_Am_Max: Tìm chỉ số (đầu tiên) của số âm lớn nhất, nếu có. Nếu không, trả về 0.
4. Kc_Max: Khoảng cách lớn nhất giữa số x và các phần tử trong a.
5. Kc_Nt_Max: Trả về chỉ số của số nguyên tố có khoảng cách lớn nhất đến 1 phần tử trong a. (nếu có);
Nếu không trả về -1.

Bài 5: (Bài toán tính Min.)
1. Min: Tính $\min(a_1,..,a_n)$.
2. Cs_Min : Tìm chỉ min : Trả về chỉ số đầu tiên đạt $\min(a_1,..,a_n)$
3. Cs_Am_Min: Tìm chỉ số của số âm nhỏ nhất, nếu có. Nếu không, trả về 0.

Bài 6: (Bài toán tính Tổng.)
1. Tong: Tổng các phần tử trong mảng.
2. Tong_Nguyen_To : Tổng các số nguyên tố trong mảng.

3. Tong_Duy_Nhat: Tổng các giá trị chỉ xuất hiện 1 lần.
4. Tong_phan_Biet: Tổng các giá trị phân biệt.

Bài 7: (Bài toán sắp xếp)
1. Sap_Tang : Sắp a theo thứ tự tăng.
2. Sap_Duong_Tang :Sắp tăng các số dương, các số khác giữ nguyên vị trí.
3. Sap_0_Cuoi: Săp lại mảng a thỏa yêu cầu:
   - Các số 0 ở cuối mảng.
   - Các số còn lại ở đầu mảng và tăng.
4. Sap0_Am_Duong:
   - Các số 0 đầu mảng
   - Các số âm ở giữa mảng và có thứ tự giảm.
   - Các số dương cuối mảng và có thứ tự tăng.

Bài 8.

b là mảng 2 chiều kích thước m x n.
1. Tong: Tổng các phần tử của b.
2. Tong_Dong: Tổng các phần tử trên dòng i.
3. Tong_Cot: Tổng các phần tử trên cột j
4. Tong_Cheo1: Tổng các phần tử trên đường chéo ,(m=n)

Bài 9

Giả sử trong bảng điểm môn "Kỹ thuật lập trình" có n sinh viên. Điểm cho nguyên theo thang điểm 10. Viết hàm xuất ra màn hình bản thống kê số lượng sinh viên có số điểm $\geq k$; k = 1,..,10 .

Bài 10.

Một đề thi trắc nghiệm gồm n câu, mỗi câu có 5 khả năng chọn lựa, trong đó có duy nhất một khả năng đúng. Các khả năng được đánh nhãn bởi các ký tự là a, b, c, d, e. Với đáp án cho sẵn, viết hàm chấm điểm bài thi và xuất ra kết quả.

- Bài thi: là dãy các kết quả chọn trong mỗi câu của đề thi.
- Câu chọn đúng: Chọn phù hợp với đáp án.
- Chấm điểm: Điểm của bài bằng (Số câu đúng) / n .

Bài 11:

Viết một hàm trong C++ in ra màn hình các ký tự chỉ xuất hiện đúng một lần trong một xâu ký tự .

Bài 12:

Một từ là một dãy các ký tự khác khoảng trắng. Một câu khác rỗng gồm nhiều từ, hai từ được phân biệt bởi khoảng trắng. Câu rỗng là câu có số từ bằng 0.

Hãy viết một hàm trong C++ trả về số từ trong một câu.

Bài 13:

Viết các hàm:
- Trả về chiều dài của từ dài nhất trong câu.
- Trả về chiều dài của từ dài nhất trong câu mà chỉ chứa các ký tự giống nhau.

Bài 14:

Viết một chương trình C++ nhập vào một ma trận vuông cấp n, với n là số nguyên dương , các phần tử của ma trận là số nguyên , Xuất ra màn hình ma trận đã nhập, tính và in ra màn hình giá trị S – T , trong đó:

- $S = \sum_{i=1}^{n} h_i$; h_i là giá trị nhỏ nhất của hàng i ; $i \in \overline{1,n}$.
- T : Là tích các phần tử của ma trận, mà các phần tử này là các số dương và nguyên tố .

- Yêu cầu của chương trình là sử dụng các hàm:
  - Nhập ma trận ,
  - Xuất ma trận ,
  - Tính S,
  - Tính T.

Bài 15:

Cho dãy a gồm n số nguyên $a_1, a_2, \ldots, a_n$. Viết chương trình tìm và in ra màn hình dãy b , được xây dựng từ dãy a như sau:

$$b_{2k} = \min\{a_i : 2k \le i \le n\} \;; \quad \forall k \in \{1, \cdots, n/2\} \;.$$
$$b_{2k-1} = \max\{a_i : 2k-1 \le i \le n\}; \quad \forall k \in \{1, \cdots, n/2\} \;.$$

Yêu cầu: chương trình có sử dụng các hàm:

- Nhập một dãy số.
- Xuất ra màn hình một dãy số.
- Tính giá trị nhỏ nhất của một dãy số.
- Tính giá trị lớn nhất của một dãy số.

Bài 16:

Viết một chương trình nhập vào 2 đa thức, tính tổng và tích 2 đa thức và in ra màn hình các đa thức đã nhập, các đa thức tổng, tích.

Bài 17:

Viết chương trình thực hiện các thao tác trên các ma trận vuông cấp n. Yêu cầu của chương trình là:

- In ra màn hình menu có các chức năng sau:

1. In ra màn hình giá trị nhỏ nhất, lớn nhất trên mỗi cột của ma trận.
2. Hoán vị 2 hàng của ma trận - In ra ma trận kết quả.
3. Tìm và in ra ma trận chuyển vị của ma trận nhập.
4. Tính tổng các giá trị trên hàng i.
5. Tính tổng các giá trị trên cột j.
6. Xuất ra màn hình các phần tử nằm ở hàng i cột j thỏa mãn: đó là giá trị lớn nhất trên hàng i vừa là giá trị nhỏ nhất trên cột j.
7. Thóat .

- Nhập một ma trận vuông , muốn thực hiện thao tác nào thì chọn chức năng tương ứng của menu .

Bài 18:

Viết chương trình thực hiện các thao tác trên xâu ký tự. Yêu cầu của chương trình là:

- In ra màn hình menu có các chức năng sau:

1. Xác định chiều dài của xâu.
2. Ghép một ký tự vào cuối một xâu.
3. Đảo ngược một xâu - In ra xâu kết quả.
4. Đếm số lần xuất hiện của một ký tự cho trước trong 1 xâu ký tự.
5. Cắt ký tự cuối của xâu rồi ghép vào đầu xâu đó.
6. Xóa một ký tự cho trước ra khỏi xâu - In ra xâu kết quả.
7. Chèn xâu s vào xâu t vào vị trí thứ k của t, in ra xâu t.
8. Trả về một số từ xâu các ký số.
9. Trả về một xâu các ký số từ một số.
10. Thóat.

- Muốn thực hiện thao tác nào thì chọn chức năng tương ứng của menu.

Bài 19:

Viết chương trình thực hiện các thao tác trên số nguyên dương.Yêu cầu của chương trình là:

- In ra màn hình menu có các chức năng sau:

1. Xuất n số nguyên tố đầu tiên.
2. Xuất ra ước số nguyên tố lớn nhất của n.
3. Xuất ra ước số lẻ lớn nhất của n.
4. Đổi số nguyên dương sang dạng xâu ký tự.
5. Đối nội dung xâu ký tự sang dạng số.

- Muốn thực hiện thao tác nào thì chọn chức năng tương ứng của menu .

Bài 20:

Viết chương trình nhập hồ sơ các học sinh của lớp học gồm tên, tuổi, điểm trung bình.

- In ra danh sách lớp theo thứ tự tăng dần của tên (theo thứ tự từ điển).
- In ra danh sách các học sinh phải thi lại.
- In ra các học sinh hạng giỏi, khá , trung bình.

CHƯƠNG 7: CON TRỎ

7.1 Con trỏ

7.1.1 Định nghĩa biến con trỏ (gọi tắt là con trỏ):

Con trỏ là biến chứa địa chỉ của biến khác.

Vậy là con trỏ không chứa dữ liệu, chỉ chứa địa chỉ của dữ liệu.

Kích thước của biến con trỏ không phụ thuộc vào đối tượng mà nó trỏ tới là kiểu gì. Kích thước cố định của biến con trỏ là 2 byte dùng để lưu địa chỉ của biến. Khi nó đang lưu địa chỉ của biến nào, ta nói nó đang trỏ tới biến ấy.

Ta cần nhớ điểm cơ bản là: Con trỏ là biến, nên có thể nói đến giá trị của nó. Giá trị của con trỏ là địa chỉ của biến mà nó trỏ tới.

Vì có nhiều loại địa chỉ nên cũng có bấy nhiêu kiểu con trỏ tương ứng.

Nhắc lại rằng địa chỉ của biến là số thứ tự của byte đầu tiên trong một dãy các byte liên tiếp mà máy dành cho biến.

Để lấy địa chỉ của biến, ta dùng phép toán &.

- & là phép toán 1 ngôi trả về địa chỉ toán hạng của nó.
- &x trả về địa chỉ của biến x.

7.1.2. Khai báo

Cú pháp:

KDL *Ten_Con_Tro;

Trong đó:

- KDL là Kiểu dữ liệu của con trỏ, có thể là char, int, double, cấu trúc ...
- Ten_Con_Tro: là tên, chỉ tên của con trỏ.
- *: là phép toán con trỏ.

Ví dụ 7.1:

- int *px;
  Khai báo 1 biến con trỏ px kiểu int
  Trong đó:
  - ✓ px: là tên con trỏ.
  - ✓ *px : là dạng khai báo của con trỏ.
- float *c,*d;
  Khai báo biến con trỏ c, d kiểu float
- void *Ptr;
  Khai báo biến con trỏ Ptr không định kiểu để sau này trỏ về kiểu bất kỳ.

7.1.3 Các phép toán trên con trỏ và biểu thức

1) Phép toán 1 ngôi: Đó là phép toán " * " tác dụng lên con trỏ.

Phép toán một ngôi * coi toán hạng của nó là địa chỉ cần xét và thâm nhập tới địa chỉ đó để lấy ra nội dung.

Tức là: *px là nội dung cuả địa chỉ do px trỏ tới (nếu px trỏ tới x thì: *px == x).

2) Phép gán:

Có thể thực hiện phép gán:

- Các con trỏ cùng kiểu.
- Gán địa chỉ của biến cho tên con trỏ. Biến và con trỏ phải cùng kiểu.
  Cú pháp: **Ten_Con_Tro = &bien;**

Ví dụ 7.2:

```
int x, *px, q;
px = &x; // Lệnh 1
q = *px; // Lệnh 2
```

- Lệnh 1: Gán địa chỉ của biến x cho con trỏ px (hay px chứa địa chỉ của x)
- Lệnh 2: Gán nội dung của địa chỉ được con trỏ px trỏ tới (hay là giá trị biến x) cho q. Nói cách khác , q = = x .

Lưu ý:

Nếu con trỏ px trỏ tới x thì các cách viết x, *px là tương đương trong mọi ngữ cảnh .

Ví dụ 7.3:

```
double a, *x,*y;
int m, *n;
```

Có thể thực hiện: x = &a;

x = y;

không thực hiện được:

n = x;

x = &m;

Ghi chú: (về phép ép kiểu)

Vì kiểu địa chỉ của m là int, khác kiểu con trỏ x là kiểu double, nên không chấp nhận được phép gán x = &m .

Trong trường hợp này ta dùng phép ép kiểu:

x = (double*) &m

3) Phép tăng, giảm địa chỉ (Số học địa chỉ):

- Các phép toán tự tăng (++), tự giảm (--).

Tăng (trước, sau), giảm (trước, sau) một đơn vị là sizeof(KDL).

Ví dụ 7.4:

```
int  *p, *q,*r,*s;
    ++p;   // Tăng trước
    q--;   // Giảm sau
```

Khi đó:

- ✓ Địa chỉ ++p trỏ tới = địa chỉ hiện p đang trỏ + 4;
- ✓ Địa chỉ q-- trỏ tới = địa chỉ hiện q đang trỏ - 4;

- Phép cộng (+) hay trừ (-) con trỏ với số nguyên dùng cho kiểu mảng.

4) Nguyên tắc truy nhập bộ nhớ:

Con trỏ kiểu Type truy cập tới sizeof(Type) bytes.

5) Phép so sánh:

So sánh các con trỏ cùng kiểu.

6) Con trỏ trong biểu thức:

Trong biểu thức:

- Ta có thể sử dụng tên con trỏ, chẳng hạn gán địa chỉ của biến cho tên con trỏ
- Hoặc là dạng khai báo con trỏ.

7.1.4 Con trỏ kiểu void

Khai báo:

void *Ten_Con_Tro;

Đây là con trỏ đặc biệt không định kiểu trước, nó có thể nhận bất kỳ địa chỉ của kiểu nào.

Ví dụ 7.5:

```
void *pa, *pb;
float a ;
int b;
```

Các câu lệnh sau là hợp lệ:

```
pa = &a;
pb = &b;
```

7.1.5 Con trỏ NULL

Một con trỏ có thể nhận giá trị hằng đặc biệt: NULL

7.1.6 Cấp phát vùng nhớ, giải phóng vùng nhớ

1) Cấp phát vùng nhớ để lưu dữ liệu:

Để cấp phát vùng nhớ cho biến con trỏ trước khi sử dụng, trong C++ dùng hàm new. Cách viết như sau:

Ten_Con_Tro = new KDL;

Khi đó con trỏ sẽ trỏ tới địa chỉ đầu của vùng nhớ đã cấp phát cho nó.

b) Giải phóng vùng nhớ:

Vùng nhớ sau khi sử dụng xong có thể xóa bỏ bằng hàm delete để tiết kiệm bộ nhớ, và tăng tốc độ thực hiện của chương trình. Cú pháp như sau:

delete Ten_Con_Tro ;

Trong đó, vùng nhớ cần giải phóng tức là tên của biến con trỏ đang chiếm giữ vùng nhớ.

7.1.7 Kiểu con trỏ

Dùng từ khóa typedef:

```
Typedef   KDL *Ten_Kieu;
Tên_Kieu   a,b,c;
```

Ví dụ 7.6:

```
typedef  float *fl;     //fl là kiểu con trỏ float
fl  a,b,c;        // a, b, c là các con trỏ có kiểu fl, chính là các con trỏ float
```

7.1.8 Truyền tham số

1. Nhắc lại cách truyền bằng trị:

- Tham số của hàm luôn được truyền theo tham trị . Điều này có nghĩa là các giá trị thực (tham số thực) không bị thay đổi giá trị khi truyền cho các đối .
- Về cơ chế, hàm được gọi tới tạo ra bản sao riêng và tạm thời cho mỗi đối chứ không phải là địa chỉ của chúng, nên những thay đổi trong hàm không thể ghi lại được.

Ví dụ 7.8:

```
void hoanvi(int a,int b)
{
    int Tam;
```

```
        Tam = a;
        a = b;
        b = Tam;
}
```

Kết quả thực hiện chương trình:

- Nhập số x = 3 ; Nhập số y = 4
- Gọi hàm: hoanvi(x,y);
- x và y đầu: x = 3 ; y = 4
- x và y cuối: x = 3 ; y = 4

Nghĩa là không có sự thay đổi nào cả. Đó là vì các tham số thực (các giá trị thực của 2 biến x và y khi truyền cho tham số hình thức trong hoanvi(a,b) chỉ truyền giá trị chứ không truyền theo biến, tưc là truyển địa chỉ, do đó sự thay đổi của 2 số khi thực hiện hàm là không ghi lại được.

2. Truyền bằng biến

Truyền bằng biến cho các đối - về cơ chế - là truyền cả nội dung và địa chỉ của biến cho các đối, khi đó hàm có thể làm thay đổi tham số thực tương ứng.

Có 2 cách thực hiện: sử dụng con trỏ hoặc tham chiếu .

a. Dùng con trỏ:

Cách viết:

- Tham số hình thức của hàm: con trỏ,
- Tham số thực: địa chỉ của biến.

Ví dụ 7.9:

```
 void hoanvi(int *pa, int *pb)
{
        int tam;
        tam = *pa;
        *pa = *pb;
        *pb = tam;
}
```

Khi đó, gọi hàm như sau: hoanvi(&x,&y);

b) Dùng tham chiếu:

Cách viết:

- Tham số hình thức của hàm: Địa chỉ của biến,
- Tham số thực: Giá trị của biến.

Ví dụ7.10:

```
void hoanvi(int  &a,int &b)
{
        int tam;
        tam = a;
        a = b;
        b = tam;
}
```

Khi đó, gọi hàm như sau: hoanvi(x,y);

7.2 Con trỏ và mảng 1 chiều

7.2.1 Địa chỉ của phần tử đầu tiên và tên mảng

Với khai báo:

float a[10];

i:	0	1	2	3	4	5	6	7	8	9
a[i]:	.	.	.	.	.	.	.	.	.	.

Máy sẽ cấp cho a một vùng nhớ liên tiếp 40 bytes cho 10 phần tử kiểu float. Địa chỉ phần tử đầu tiên của mảng: &a[0]

Theo qui định của C/C++, tên mảng là a chứa địa chỉ phần tử đầu tiên của mảng, nên tên mảng là một hằng địa chỉ:

&a[0] tương đương với a (tên mảng)

Và trong mọi ngữ cảnh, C/C++ quy định cách viết:

&a[i] tương đương với a + i; a[i] tương đương với *(a + i)

7.2.2 Phép cộng (+) hay trừ (-) con trỏ với số nguyên cho mảng 1 chiều

(Con trỏ trỏ tới phần tử mảng)

Giả sử con trỏ px trỏ tới phần tử a[i], khi đó:

- px + k trỏ tới phần tử thứ k sau a[i], tức là a[i+k];
- px -k trỏ tới phần tử thứ k trước a[i], tức là a[i-k];
- *(px + i) tương đương với px[i];
- ++px trỏ tới phần tử tiếp theo là a[i+1] // Tăng trước
- px++ trỏ tới phần tử tiếp theo là a[i+1] // Tăng sau
- --px trỏ tới phần tử kế trước là a[i-1] // giảm trước
- px-- trỏ tới phần tử kế trước là a[i+1] // giảm sau

7.2.3 Cấp phát động cho mảng 1 chiều thông qua con trỏ

Ta có thể dùng con trỏ để cài đặt mảng 1 chiều (biến động).

1 Khai báo:

```
KDL *MD;
```

2. Cấp phát vùng nhớ:

```
MD = new KDL[MAX];     // Mảng có không quá MAX phần tử
```

3. Thu hồi vùng nhớ:

```
delete [ ]MD;
```

7.2.4 Đối của hàm là con trỏ

Khi đó, tham số thực có thể là:

- Địa chỉ của biến có kiểu tương ứng.
- Con trỏ có kiểu tương ứng.

- Tên của mảng một chiều có kiểu tương ứng

Đối	Tham số thực
Con trỏ	Con trỏ (cùng kiểu)
	Tên mảng 1 chiều (cùng kiểu)

Ví dụ 7.11:

Cho mảng a[0..n-1] có n số nguyên, x là số nguyên. Viết chương trình thực hiện chức năng sau:

Chen(a,n,x,k) ≡ chèn x vào a tại vị trí thứ k, kết quả trả về a. (đếm k từ 0)

Mảng cài đặt bằng con trỏ.

Thực hiện:

Bước 1: Tạo Project với tên "Md1chieu".

Bước 2: Tạo tập tin chương trình Md.cpp

Bước 3: Trong tập tin Md.cpp, soạn code theo cấu trúc:

```
#include <iostream>
using namespace std;
void Nhap(int *a, int n);
void Xuat(int *a, int n);
void Chen(int *a, int &n, int x, int k);

void main()
{
	int  *a, //Dùng con trỏ
		n, x, k;
	cout<<"\nNhap n = ";
	cin>>n;
	a = new int[n]; //cấp phát vùng nhớ đề a chứa đủ n giá trị kiểu int
	Nhap(a,n);
	Xuat(a,n);
	cout<<"\nNhap gia tri can chen x = ";
	cin>>x;
	do
	{
		cout<<"\nNhap vi tri can chen (0 <= k <= "<<n<<" )k = ";
		cin>>k;
		if(0 <= k && k <= n)
			break;
	}
	while(1);
	Chen(a, n,x,k);
	Xuat(a,n);
	delete []a; //Thu hồi vùng nhớ đã cấp phát cho a
}
```

```
//Cac ham nhap xuat
void Nhap(int *a, int n)
{
	for (int i = 0; i < n; i++)
	{
		cout<<"\na["<<i<<"] = ";
		cin>>*(a+i);
	}
}

void Xuat(int *a, int n)
{
	int i;
	cout<<"\n";
	for (i = 0; i < n; i++)
		cout<<*(a+i)<<'\t';
}

void Chen(int *a, int &n, int x, int k)
{
	int i;
	for(i = n-1; i >= k; i--)
		*(a+i+1) = *(a+i);
	*(a+k) = x;
	n = n+1;
	}
```

7.3 Con trỏ và xâu ký tự

7.3.1 Tiếp cận

Tương tự như mảng động 1 chiều, xâu ký tự có thể cài đặt bằng con trỏ. Ta có thể khai báo xâu ký tự như sau:

```
char *Chuoi;
```

7.3.2 Các thao tác trên con trỏ ký tự

1. Khởi tạo:

```
	Chuoi = NULL;	//Xâu rỗng
```

hoặc:

```
	Chuoi[0] = NULL;
```

2. Cấp phát vùng nhớ:

```
	Chuoi = new char[MAX];
```

3. Thu hồi vùng nhớ:

```
	delete [ ]Chuoi;
```

4. Phép gán:

- Gán 2 con trỏ ký tự cho nhau.

- Gán một hằng xâu ký tự cho con trỏ ký tự.
- Gán tên một xâu ký tự cho một con trỏ ký tự.

5. Nhập dữ liệu:
Có thể dùng:

- cin>>Chuoi; // Chuoi không có ký tự tách.
- gets(Chuoi);

6. Xuất dữ liệu:

```
cout<<Chuoi;
```

7. Tăng giảm địa chỉ cũng giống như mảng 1 chiều.

7.3.3 Đối của hàm là con trỏ ký tự

Tham số thực có thể là:

- Con trỏ ký tự.
- Xâu ký tự.

Đối	Tham số thực
Con trỏ ký tự	Con trỏ ký tự
	Tên xâu ký tự

Ví dụ 7.12:

Viết hàm chèn một ký tự Kt vào xâu ký tự a tại vị trí k.

```
void ChenKT(char *a, char Kt, int k)
{
	int i, l;
	l = strlen(a); //Hàm thư viện trả về chiều dài xâu a trong string.h
	if ( k > l || k < 0)
	{
		cout<<"\nVi tri chen khong hop le!";
		return;
	}
	else
	{
		for(i = l; i >= k; i--)
			*(a+i+1) = *(a+i);
		*(a+k) = Kt;
	}
}
```

Một cách sử dung, chẳng hạn chèn ký tự ‘e’ vào a tại vị trí 1:

```
ChenKT(a, ‘e’, 1);
```

7.4 Con trỏ và mảng 2 chiều

7.4.1 Phép cộng địa chỉ trong mảng 2 chiều

Xét khai báo:

```
int a[2][3];
```

Ta có mảng a gồm 6 phần tử được lưu trử kế tiếp trong bộ nhớ, và được xếp theo thứ tự dòng:

	hàng 1			hàng 2		
Phần tử	a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]
Địa chỉ	1	2	3	4	5	6

C++ quan niệm: Mảng 2 chiều là mảng 1 chiều của mảng 1 chiều. Nên a chính là mảng 1 chiều 2 dãy, mỗi dãy gồm 3 số nguyên. Khi đó kiểu địa chỉ của a là int[3], có kích thước: 4 bytes * 3 = 12 bytes.
a trỏ tới đầu hàng đầu tiên (hàng a[0], do đó trỏ tới a[0][0]).

7.4.2. Cài đặt mảng động 2 chiều bằng con trỏ

1. Khai báo:
   KDL *pa;
2. Cấp phát vùng nhớ để lưu trử dữ liệu:
   pa = new KDL[KT1*KT2];

Với khai báo trên, để cấp phát vùng nhớ cho pa ta viết:
   pa = new int[2*3];

3. Thu hồi vùng nhớ.
   delete [ ]pa;
4. Duyệt các phần tử của mảng:

Để duyệt các phần tử của mảng, theo cách đã biết là dựa vào chỉ số của các phần tử của mảng (pa[i][j]), ngoài ra có thể sử dụng con trỏ theo cách sau.
Tổng quát:
   Trong mảng a gồm m dòng và n cột thì:

pa + i*n + j trỏ tới phần tử a[i][j]

Mối liên hệ giữa i,j,t:
- t = i * n + j;
- i = t/n;
- j = t - i * n;

Các công thức trên được sử dụng để điều khiển con trỏ trỏ tới phần tử mảng cần truy xuất.

7.4.3 Đối của hàm là con trỏ

Tham số thực có thể là: con trỏ cùng kiểu, hoặc tên mảng 2 chiều (có ép kiểu)

Đối	Tham số thực
Con trỏ	Con trỏ cùng kiểu
	Tên mảng 2 chiều (ép kiểu)

Ví dụ 7.13

   Viết chương trình tính tích 2 ma trận vuông cấp n

Thực hiện:

Bước 1: Tạo Project với tên "Tich_MT".

Bước 2: Tạo tập tin chương trình TichMt.cpp

Bước 3: Trong tập tin TichMt.cpp, soạn code theo cấu trúc sau:

```
#include<iostream>
using namespace std;
void Nhap(int *a, int n, char Kt);
void Xuat(int *a, int n, char Kt);
void Tich_MT(int *a, int *b, int *c, int n);

void main()
{
	int *a, *b, *c, n;
	cout<<"\nNhap n = ";
	cin>>n;
	a = new int[n*n];
	b = new int[n*n];
	c = new int[n*n];

	Nhap(a,n,'a');
	Nhap(b,n,'b');
	Tich_MT(a,b,c, n);
	Xuat(a,n,'a');
	Xuat(b,n,'b');
	Xuat(c,n,'c');

	cout<<'\n';
	delete []a;
	delete []b;
	delete []c;
}
//Nhap ma tran Kt
void Nhap(int *a, int n, char Kt)
{
	int i, j;
	cout<<"\nNhap ma tran "<<Kt;
	for (i = 0; i < n; i++)
		for(j = 0; j < n; j++)
		{
			cout<<"\n"<<Kt<<"["<<i<<"]["<<j<<"]= ";
			cin>>*(a+i*n+j);
		}
}

//Xuat ma tran Kt
```

```
void Xuat(int *a, int n, char Kt)
{
	int i, j;
	cout<<"\nMa tran "<<Kt<<":\n";
	for (i = 0; i < n; i++)
	{
		cout<<'\n';
		for (j = 0; j < n; j++)
			cout<<*(a+i*n+j)<<'\t';
	}
}
//c = ab
void Tich_MT(int *a, int *b, int *c, int n)
{
	int i,j,k;
	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);
		}
}
```

7.5 Con trỏ và cấu trúc

7.5.1 Con trỏ cấu trúc và địa chỉ cấu trúc

- Con trỏ cấu trúc dùng để lưu địa chỉ của biến cấu trúc (cùng kiểu).
- Như mảng 1 chiều, tên của mảng cấu trúc được xem là hằng địa chỉ chứa địa chỉ của phần tử đầu tiên của mảng, nên con trỏ cấu trúc có thể chứa tên mảng cấu trúc (cùng kiểu).

7.5.2 Phép gán

Có thể gán:

- Địa chỉ của biến cấu trúc cho con trỏ cấu trúc (cùng kiểu).
- Tên mảng cấu trúc cho con trỏ cấu trúc (cùng kiểu).
- Hai con trỏ cùng kiểu cấu trúc cho nhau.

7.5.3 Truy nhập thông qua con trỏ

Theo cú pháp:

```
Ten_Con_Tro -> Ten_Truong;
```

7.5.4 Phép cộng, trừ con trỏ với số nguyên áp dụng cho mảng cấu trúc:

Như mảng 1 chiều.

7.5.5 Con trỏ và mảng cấu trúc

Giả sử con trỏ Ds trỏ tới đầu List, tức là có:
Ds = List; //gán địa chỉ List[0] cho Ds
Thì các cách viết sau là tương đương:

Địa chỉ: List + i,	&List[i]	Ds + i	&Ds[i]
Giá trị: *(List + i)	List[i]	*(Ds + i)	Ds[i]

Do đó, có thể truy nhập vào các trường bằng cách sau:

List[i].Tên_Truong
Ds[i].Tên_Truong

hoặc

(List + i) -> Tên_Truong
(Ds + i) -> Tên_Truong

Ví dụ 7.14:

Trong ví dụ 6.27, Hàm nhập danh sách mảng nhân viên Ds ta có thể viết lại như sau:

- Hàm Nhập dữ liệu cho nhân viên
- Sử dụng hàm trên để viết hàm nhập danh sách nhân viên.

```
void Nhap ( NHANVIEN  *p)
{
        cout<<"\nMa nhan vien: ";
        gets(p->MaNV);
        cout<<"\nHo nhan vien: ";
        gets(p->HoNV);
        cout<<"\nTen nhan vien: ";
        gets(p->.TenNV);
        cout<<"\nTuoi: ";
        cin>>p->Tuoi;
        cout<<"\nDia chi: ";
        gets(p->Dc);
        cout<<"\nSo dien thoai: ";
        cin>>p->Sdt;
        cout<<"\nLuong: ";
        cin>>p->Luong;
        cout<<"\nPhong: ";
        gets(p->Phong);
}
void Nhap ( NHANVIEN  Ds[MAX], int n)
{
        int i;
        for( i = 0; i < n; i++)
        {
                cout<<"\nNhap thong tin nguoi thu "<< i+1;
                Nhap ( &Ds[i]);
```

```
        }
}
```

7.6 Mảng con trỏ

7.6.1 Khái niệm

Mảng con trỏ là mảng mà mỗi phần tử của nó có thể chứa 1 địa chỉ nào đó.

7.6.2 Cú pháp khai báo

```
KDL *Ten_Con_Tro[KT];
```

- KDL có thể là char,int, float...
- KT: Kích thước của mảng.
- Mỗi phần tử của mảng là một con trỏ có kiểu là KDL.

7.7 Tìm hiểu thêm về hàm

Nhắc lại về: Đối – tham số thực – cách truyền.

Đối	Tham số thực	Ghi chú
Biến	Giá trị	Truyền bằng trị
Con trỏ	- Con trỏ - Địa chỉ của biến - Tên mảng 1chiều - Tên xâu ký tự - Tên mảng 2 chiều (ép kiểu)	Truyền bằng biến
Tham chiếu	Giá trị	Truyền bằng biến
Tên mảng	Tên mảng	Truyền bằng biến
Tên xâu ký tư.	Tên xâu ký tư.	Truyền bằng biến

7.7.1 Về đối của hàm

Các đối của hàm có thể chia thành 2 loại:

- Loại thứ nhất là các đối dùng để chứa các trị do các tham số thực nhập vào và ta gọi các đối này là các đối vào.
- Loại thứ hai là các đối dùng để chứa các kết quả do xử lý, tính toán và ta gọi các đối này là các đối ra.

  Ta sử dụng con trỏ (hay tham chiếu) cho các đối ra của hàm.

Ví dụ 7.15:

Viết hàm giải phương trình bậc 2. Đối chứa các hệ số là đối vào, đối chứa nghiệm là các đối ra.

Input: a, b, c

Output:

- 0; Nếu a = 0
- -1, nếu delta < 0
- 1; Nếu Delta > 0

```
int PTB2(double a, double b, double c, double *x1, double *x2)
{
        double Delta;
```

```
        int Kq;
        if (a==0)
                Kq = 0;
        else
        {
                Delta = b*b - 4*a*c;
                if (Delta < 0)
                        Kq = -1;
                else
                {
                        *x1 = (-b - sqrt(delta))/(2*a);
                        *x2 = (-b + sqrt(delta))/(2*a);
                        Kq =  1;
                }
        }
        return Kq;
}
```

7.7.2. Giá trị trả về của hàm là con trỏ

Giá trị trả về của hàm có thể là con trỏ kiểu char, int, double, cấu trúc... nhưng không thể là kiểu mảng.

Trong trường hợp này, cách viết hàm như sau (chỉ khác dòng tiêu đề.):

```
        KDL *Ten_Ham(Danh_Sach_Cac_Kieu_Va_Doi)
        {
                KDL *Kq;
                //...
                return Kq;
        }
```

Vì giá trị trả về của hàm là con trỏ, nên dựa vào cách cài đặt mảng bằng con trỏ, ta có thể xem giá trị trả về của hàm có thể là mảng động.

Ví dụ 7.16:

```
        Giá trị trả về của hàm là con trỏ.
 int *CopyArray(int *b,int n)
{
        int *a;
        for (int i = 0; i < n; i++)
                *(a + i) = *(b+i);
        return a;
}
```

Sử dụng:

```
int *a;
a = new int[n];
a = CopyArray(b,n);
```

7.7.3 Con trỏ trỏ đến dữ liệu không biến đổi

Khi một con trỏ trỏ đến một biến nào đó mà ta không muốn nội dung của biến này thay đổi, thì ta dùng từ khóa conts, cách viết hàm là trước kiểu dữ liệu của đối ta đặt từ khóa const. Chẳng hạn:

```
KDL Ten_Ham(const Kieu *p)
{
        //...
}
```

Ví dụ 7.17:

Viết hàm trả về số lần xuất hiện của ký tự KT trong xâu ký tự a:

```
int SLXH_KT(const char *a, char KT)
{
        int Dem = 0, i = 0;
        while (*(a+i) != NULL )
        {
                if( *(a+i) == KT)
                        dem++;
                i++;
        }
        return Dem;
}
```

7.7.4. Con trỏ hàm

Một con trỏ hàm lưu trử địa chỉ của hàm. Tương tự như mảng, tên hàm là địa chỉ của lệnh đầu tiên của hàm.

1. Khai báo.

KDL (*Ten)(Danh sach cac kieu);

2. Sử dụng con trỏ hàm:

Gán tên hàm cho con trỏ hàm, với yêu cầu kiểu hàm và kiểu con trỏ hàm phải tương thích. Sau phép gán, ta có thể dùng con trỏ hàm thay cho tên hàm.

3. Đối là con trỏ hàm.

Tham số thực là một tên hàm.

Nếu đối được khai báo:

double (*f)(double, int)

thì trong thân hàm có thể sử dụng các cách viết sau để xác định giá trị của hàm:

f(x,m) (f)(x,m) (*f)(x,m)

trong đó x có kiểu double, m có kiểu int.

4. Mảng con trỏ hàm

Khai báo:

KDL (*ten[KT])(Danh sach cac kieu);

BÀI TẬP

Các bài sau sử dụng mảng động

Bài 1:

1. Cs_Am_Max: Tìm chỉ số (đầu tiên) của số âm lớn nhất, nếu có. Nếu không, trả về -1.
2. Tong_Phan_Biet: Tổng các giá trị phân biệt.
2. Sap_Am_Tang: Sắp tăng các số âm, các số khác giữ nguyên vị trí.

Bài 2:

Viết một chương trình nhập vào một ma trận vuông cấp n, với n là số nguyên dương , các phần tử của ma trận là số nguyên , Xuất ra màn hình ma trận đã nhập, tính và in ra màn hình giá trị S – T , trong đó:

- $S = \prod_{i=1}^{n} h_i$, h_i là số âm nhỏ nhất của hàng i ; i $\in \overline{0, n-1}$
- $T = \sum_{j=1}^{n} v_j$,v_j Là giá trị lớn nhất của cột j ; j$\in \overline{0, n-1}$.

Yêu cầu của chương trình là sử dụng các hàm:

- Nhập ma trận ,
- Xuất ma trận ,
- Tính S,
- Tính T.

Bài 3:

Viết chương trình quản lý các khách hàng thuê bao điện thoại, trong đó sử dụng cấu trúc gồm

- Mã khách hàng,
- Họ tên,
- Địa chỉ,
- Số điện thoại.

Yêu cầu chương trình thực hiện các chức năng sau:

- Thêm một khách hàng mới
- Xóa một khách hàng
- Tìm kiếm địa chỉ khi biết số điện thoại

CHƯƠNG 8: THUẬT TOÁN ĐỆ QUY

8.1 Khái niệm đệ quy

1. Một khái niệm X gọi là được định nghĩa đệ quy (*recursion*) nếu trong định nghĩa của X có sử dụng ngay chính khái niệm X.

Ví dụ:

Giai thừa có thể định nghĩa đệ quy như sau:

- Giai thừa của 1 bằng 1.
- Giai thừa của n (n > 1) là tích của n với giai thừa n-1.

2. Một định nghĩa đệ quy bao gồm 2 thành phần:
   a. Thành phần dừng: không chứa khái niệm đang đinh nghĩa.
   b. Thành phần đệ quy: Có chứa khái niệm đang định nghĩa.

Trong ví dụ trên thì:

- Thành phần dừng: Giai thừa của 1 bằng 1.
- Thành phần đệ quy: Giai thừa của n (n > 1) là tích của n với giai thừa n-1.

3. Thuật toán đệ quy là thuật toán có chứa thao tác gọi đến nó.

Thuật toán đệ quy chứa các thao tác mà trong đó có chứa thao tác gọi lại thuật toán (gọi đệ quy).

4. Một hàm được gọi từ chính nó, thì đó cũng là một hình thức đệ quy.

Khi hàm gọi ngay chính nó, thì mỗi lần gọi, máy sẽ tạo ra các biến tự động mới hoàn toàn độc lập với các biến đã được tạo lập trong các lần gọi trước đó. Có bao nhiêu lần gọi hàm thì cũng có bấy nhiêu lần ra khỏi hàm,và mỗi lần ra khỏi hàm thì các biến tự động mất đi.

Sự tương ứng giữa các lần gọi hàm và các lần ra khỏi hàm được thực hiện theo thứ tự ngược lại, nghĩa là:

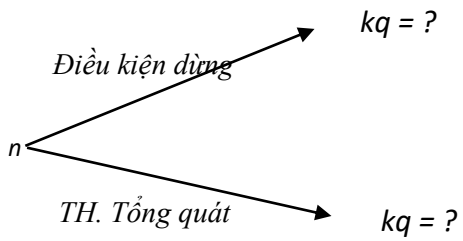
- Lần ra đầu tiên ứng với lần vào cuối cùng,
- Và lần ra khỏi hàm cuối cùng ứng với lần đầu tiên gọi hàm.

8.2. Cấu trúc của 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
}
```

Ghi chú :

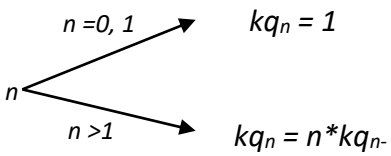
Để mô tả thuật toán đệ quy, ta có thể dùng một kỹ thuật, gọi là kỹ thuật vẽ cây, để liệt kê các trường hợp dừng, trường hợp đệ quy.



Ví dụ 8.1:

Viết hàm tính n! bằng đệ quy.

Cây liệt kê :




```
int Gtdq(int n)
{
	int kq;
	if (n == 1 || n ==0) //Điều kiện dừng
		kq = 1; //Xử lý đặc biệt
	else //Trường hợp tổng quát
		if(n > 1)
			kq = n*Gtdq(n-1);
	return kq;
}
```

Có thể viết cách khác như sau:

```
long Gtdq(int n)
{
	if (n == 1 || n ==0)
		return 1;
	if(n > 1)
		return(n*gtdq(n-1));
}
```

8.3 Cơ chế hoạt động hàm đệ quy

Xem lại ví dụ 8.1, và gọi hàm với tham số thực truyền vào là 3 (n= 3):

```
Gtdq(3);
```

Lần gọi thứ nhất:

Lần đầu tiên gọi tới hàm Gtdq(3) từ hàm main(), máy sẽ tạo ra biến tự động của hàm Gtdq(3). Biến tự động của hàm này là n. Ta gọi đối n được tạo ra ở lần gọi

thứ nhất là n thứ nhất. Giá trị thực là 3 được gán cho n thứ nhất. Lúc này biến n trong thân hàm được xem là n thứ nhất nên cũng có giá trị bằng 3, do đó câu lệnh:

```
if (n == 1 || n ==0)
```

là sai, nên sẽ bỏ qua câu lệnh:

```
return 1;
```

mà thực hiện câu lệnh:

```
return(n*Gtdq(n-1));
```

Theo câu lệnh này, máy sẽ tính biểu thức: n*Gtdq(n-1) // 3*Gtdq(2)

Để tính biểu thức trên, cần gọi tới chính hàm Gtdq.

Lần gọi thứ 2:

Lần gọi thứ 2 được thực hiện, máy sẽ tạo ra đối n mới, gọi là n thứ hai.

Giá trị n-1 được hiểu là n thứ nhất truyển cho n thứ hai.Vậy giá trị của n thứ hai là 2.

Bây giờ trong hàm , n được hiểu là n thứ hai. Câu lệnh if sẽ được bỏ qua vì điều kiện sai, nên máy tinh biểu thức n*Gtdq(n-1) // 2*Gtdq(1)

Lần gọi thứ 3:

Lần gọi thứ 3 được thực hiện, máy sẽ tạo ra tham số n mới, gọi là n thứ ba.

Giá trị n-1 được hiểu là n thứ hai truyển cho n thứ ba. Vậy giá trị của n thứ ba là 1.

Bây giờ trong hàm , n được hiểu là n thứ ba. điều kiện của câu lệnh if đúng nên máy thực hiện câu lệnh:

```
return 1;
```

Sau đó máy sẽ bắt đầu thực hiện lần lượt 3 lân ra khỏi hàm Gtdq:

Lần ra thứ nhất:

Lần ra thứ nhất ứng với lần vào thứ 3. Kết quả tham số n thứ ba được giải phóng, hàm cho giá trị Gtdq(1) = 1, máy sẽ trở về xét biểu thức: n*gtdq(n-1)

Theo câu lệnh return, máy sẽ thực hiện lần ra thứ hai.

Lần ra thứ hai:

Lần ra thứ hai ứng với lần vào thứ hai. Kết quả là tham số n thứ hai được giải phóng, hàm cho giá trị Gtdq(2) = 2*Gtdq(1) = 2. Máy trở về xét biểu thức: n*gtdq(2).

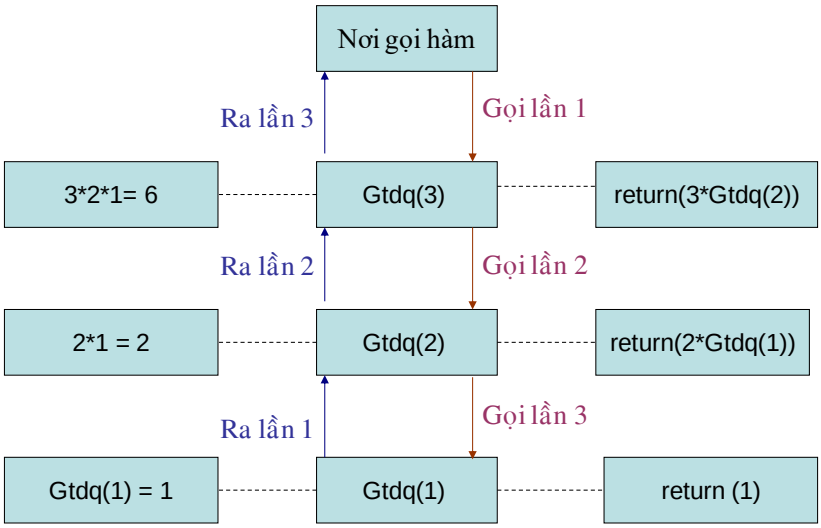
Theo câu lệnh return, máy sẽ thực hiện lần ra thứ ba.

Lần ra thứ ba:

Lần ra thứ ba ứng với lần vào thứ nhất để giải phóng n thứ nhất, trả về giá trị Gtdq(3) = 3 * Gtdq(2) = 6 cho hàm và trở về hàm main().

Nhận xét:

Cơ chế hoạt động vào ra của hàm khi gọi đệ quy là **VÀO SAU – RA TRƯỚC**



Ghi chú:

Nếu dùng đệ quy thì có thể sẽ tốn bộ nhớ và tốc độ thực hiện chương trình chậm so với cấu trúc lặp.

Tuy nhiên có những bài toán nếu dùng đệ quy thì cách giải dễ hiểu, rõ ràng hơn, thậm chí còn có những bài toán nếu không dùng đệ quy thì không giải được.

8.4 Phân loại các thuật toán đệ quy:

Căn cứ vào cách gọi đệ quy mà ta có các loại sau :

8.4.1. Đệ quy tuyến tính :

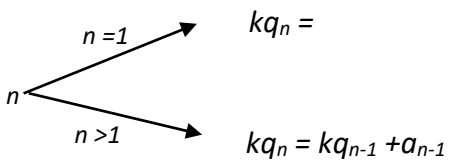
Trong mô tả thuật toán (hàm), thuật toán (hàm) chỉ có 1 lần gọi chính nó.

Thuật toán trong ví dụ 8.1 là đệ quy tuyên tính

- Trong ví dụ 8.1, thuật toán chỉ có một lần gọi chính nó : $kq_n = n*kq_{n-1}$

Ví dụ 8.2:

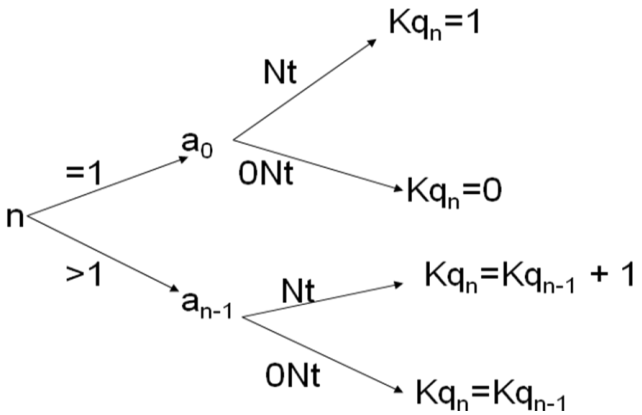
Tính tổng các phần tử trong dãy a[0,n-1].




```
int TinhTong(int  *a, int n)
{
        int kq;
        if ( n==1)
                kq = *a;
        else
                if (n >1)
                   kq = TinhTong(a,n-1) + *(a+n-1);
        return kq;
}
```

Ví dụ 8.3:

Đếm số lượng các số nguyên tố trong dãy a[0,n-1].




```
int Dem_Nt(int  *a, int n)
{
        int  Kq;
        if (n == 1)
                if (Nt(*a ) )
                        Kq = 1;
                else
                        Kq = 0;
        else
                if(n > 1)
                        if(Nt(*(a+n-1) ))
                                Kq =  Dem_Nt(a,n-1) +1;
                        else
                                Kq = Dem_Nt(a,n-1);
        return  Kq;
  }
```

8.4.2. Đệ quy nhị phân :

Trong mô tả thuật toán (hàm), thuật toán (hàm) có 2 lần gọi chính nó.

Ví dụ 8.4:

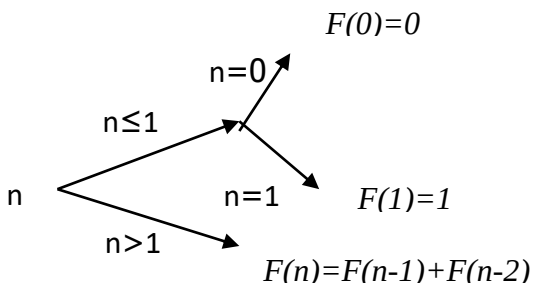
Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1, các phần tử sau đó được thiết lập theo quy tắc mỗi phần tử luôn bằng tổng hai phần tử trước nó cộng lại.

Công thức truy hồi của dãy Fibonacci là:

$$F(n)=\begin{cases}0; n=0\\ 1; n=1;\\ F(n-1)+F(n-2); n>1\end{cases}$$

Viết hàm xác định số Fibonacci thứ ***n***.

Cây liệt kê :



Cài đặt :

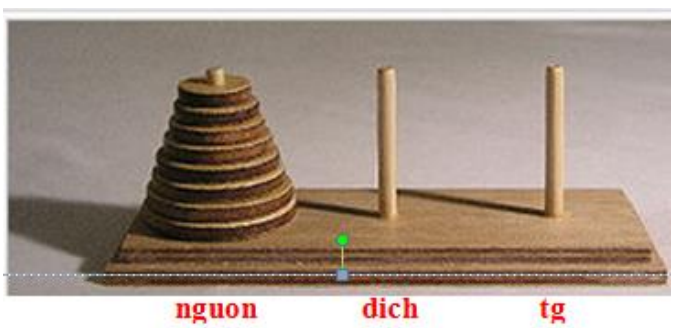
```
int Fib(int n)
{
	int kq;
	if(n == 0 )
		kq = 0;
	else
		if(n == 1 )
			kq = 1;
	else
		if ( n > 1)
			kq = Fib(n-1) + Fib(n-2);
	return kq;
}
```

Ví dụ 8.5 : Tháp Hà Nội

Có 3 chiếc cọc ký hiệu lần lượt là nguon, dich, tg va n chiếc đĩa kích thước khác nhau. Ban đầu đặt trên cọc nguon với thứ tự kích thước giảm dần.
Ta chuyển n đĩa từ cọc nguon sang cọc dich theo nguyên tắc :

- Mỗi lần chỉ chuyển được 1 đĩa.
- Mỗi đĩa có thể chuyển từ cọc này sang cọc khác
- Đĩa kích thước nhỏ phải đặt trên đĩa kích thước lớn hơn.

Mô tả và cài đặt thuận toán.

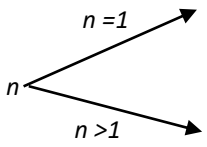


Phân tích thuật toán :

Ký hiệu THN(n, nguon, dich, tg) là các thao tác chuyển n đĩa từ cọc nguồn "nguon" sang cọc đích "dich" có sử dụng cọc trung gian "tg".

- Với n = 1, THN(1, nguon, dich, tg) : Thao tác chuyển 1 đĩa từ cột nguồn "nguon" sang cột đích "dich" lấy cột "tg" làm trung gian. Đây chính là điều kiện dừng, chuyển 1 đĩa từ cột nguồn "nguon" sang cột đích "dich".
- Với n > 1, THN(n, nguon, dich, tg) có thể tách thành dãy tuần tự 3 công việc như sau:
  - Chuyển (n -1) đĩa từ cột nguồn "nguon" sang cột đích là cột "tg" lấy cột "dich" làm trung gian: ThapHN(n - 1, nguon, tg, dich).
  - Chuyển 1 đĩa từ cột nguồn " nguon" sang cột đích "dich" lấy cột tg làm trung gian, ThapHN(1, nguon, dich, tg).
  - Chuyển (n -1) đĩa từ cột nguồn "tg" sang cột đích "dich" lấy cột "nguon" làm trung gian: ThapHN(n - 1, tg, dich, nguon).

Cây liệt kê có thể vẽ như sau :



Chuyển 1 đĩa từ nguon sang dich

ThapHN(n - 1, nguon, tg, dich);
ThapHN(1, nguon, dich, tg);
ThapHN(n - 1, tg, dich, nguon);

Cài đặt :

```
void ThapHN(int n, char nguon, char dich, char tg)
{
	if (n == 1)
		cout << "\nChuyen 1 dia tu coc " << nguon << " toi coc " << dich;
	else
	{
		ThapHN(n - 1, nguon, tg, dich);
		ThapHN(1, nguon, dich, tg);
		ThapHN(n - 1, tg, dich, nguon);
	}
}
```

8.4.3. Đệ quy phi tuyến:

Trong mô tả thuật toán, thuật toán (hàm) có lặp một số lần gọi chính nó.

Ví dụ 8.6 :

Xác định số hạng thứ n của dãy số :

$$x_n = \begin{cases} 1; n = 0 \\ n^2 x_0 + (n-1)^2 x_1 + (n-2)^2 x_2 + \cdots + (n-i)^2 x_i + \cdots + 2^2 x_{n-2} + 1^2 x_{n-1}; n > 0 \end{cases}$$

```
int Tinhxn (int n)
{
	int kq, i, t;
	if (n == 0)
```

```
                kq = 1;
        else
        {       t = 0;
                for ( i = 1; i <= n; i++)
                        t  +=  i*i*Tinhxn (n - i);
                kq = t;
        }

        return  kq;
}
```

8.4.4. Đệ quy hổ tương:

Đây là trường hợp 2 thuật toán đệ quy (hàm đệ quy) gọi lẫn nhau.

Ví dụ 8.7

Cho các dãy số nguyên (u_n), (v_n) xác định như sau :

$$u_n = \begin{cases} n; 0 \le n < 5 \\ u_{n-1} + v_{n-2}; n \ge 5 \end{cases}$$

$$v_n = \begin{cases} n-3; 0 \le n < 8 \\ u_{n-1} + v_{n-2}; n \ge 8 \end{cases}$$

Tính u_n, v_n .

```
int U( int n)
{
        int kq;
        if (0 <= n && n <5)
                kq = n;
        else
                if(n >=5)
                        kq = U(n-1) + V(n-2);
        return kq;

}

int V( int n)

{
        int kq;
        if (0 <= n && n <8)
                kq = n - 3;
        else
                if(n >=8)
                        kq = U(n-1) + V(n-2);
        return kq;
```

}

Lưu ý :

Trường hợp này cần phải khai báo nguyên mẫu các hàm U, V.

BÀI TẬP

(Các thuật toán được thiết kế bằng hình thức đệ quy)

Bài 1:

Tìm x có trong a ? Nếu có trả về chỉ số cuối cùng tương ứng. Nếu không có trả về -1.

Bài 2:

Viết hàm tính số lượng các đường chạy trong dãy.

(Đường chạy: dãy con tăng dần lớn nhất trong dãy)

Bài 3:

Viết hàm đếm số lượng từ trong một xâu ký tự

Bài 4:

Viết hàm tính tổng các số hạng nằm trên đường chéo chính của một ma trận vuông cấp n.

Bài 5:

Tính max(a_0,..,a_{n-1})

Bài 6.

Viết chương trình tùy chọn thực hiện các chức năng sau :

1. Tính tổng n sô lẻ dương đầu tiên.
2. Tính tích n sô lẻ dương đầu tiên.
3. Tính tổng : 1+(1.2)+(1.2.3) + ...+ (1.2...n)

Bài 7:

Tính ước chung lớn nhất của 2 số nguyên dương.

Bài 8:

Xuất xâu ký tự đảo ngược.

CHƯƠNG 9 : CÁC NGUYÊN LÝ LẬP TRÌNH CẤU TRÚC

9.1 Các lớp lưu trữ

Trong C++ chỉ rõ có 4 lớp lưu trữ các biến khác nhau: automatic, external, static và register. Chúng xác định bởi các từ khóa: auto, extern, static và register.

9.1.1 Biến cục bộ

- Các đặc trưng:

Vị trí khai báo	Cách khai báo	Địa chỉ trong bộ nhớ c/t.	Thời gian tồn tại	Phạm vi tác dụng	Khởi đầu
-Trong hàm hay khối lệnh. -Đối của hàm.	định nghĩa biến thông thường.	Vùng ngăn xếp.	Trong khoảng thời gian hàm hay khối lệnh hoạt động	Hàm hay khối lệnh chứa nó.	Không khởi đầu cho mảng tự động.

- Hoạt động của biến cục bộ như sau:

Khi khối lệnh hay hàm chứa các biến cục bộ hoạt động , chúng sẽ được cấp phát bộ nhớ và lưu trử trong ngăn xếp, khi chương trình ra khỏi khối lệnh hay hàm chứa chúng thì chúng mất đi. Vì tính chất tự động này, các biến khai báo trong khối lệnh hay trong 1 hàm còn gọi là biến tự động.

Ghi chú:

Các biến khai báo trong khối lệnh hay hàm , các đối của hàm đương nhiên là cục bộ, nên từ khóa này là không cân thiết và rất ít được dùng.

9.1.2 Biến ngoài và từ khoá extern

1. Biến ngoài (còn gọi là biến toàn cục)

- Các đặc trưng:

Vị trí khai báo	Cách khai báo	Địa chỉ trong bộ nhớ c/t.	Thời gian tồn tại	Phạm vi tác dụng	Khởi đầu
Ngoài tất cả các hàm.		Vùng cấp phát tĩnh.	Trong suốt thời gian chương trình chứa nó hoạt động.	Từ vị trí khai báo đến cuối tập tin chương trình.	Khởi đầu cho các mảng ngoài theo quy tắc khởi đầu.

- Mọi hàm đều có thể thâm nhập vào biến ngoài bằng cách tham trỏ tới nó theo tên.
- Biến ngoài không tự xuất hiện và tự biến đi, cho nên chúng còn giữ lại giá trị qua mỗi lầm gọi hàm. Nó cũng có thể sử dụng cho các tập tin khác (khi chương trình nhiều tập tin) nhờ khai báo extern .

2. Từ khoá extern:

Thêm từ khóa extern trước các định nghĩa dữ liệu thông thường bên ngoài các hàm, khi đó máy hiểu chúng là các biến ngoài , có các kiểu dữ liệu tương ứng, nhưng không cần cấp bộ nhớ cho nó.
chẳng hạn:

```
extern int n;
extern float x;
```

Ta cần phân biệt giữa khai báo một biến ngoài và định nghĩa nó:

- **Khai báo cho biết tính chất của biến(kiểu, kích thước...)**
- **Định nghĩa tạo ra việc cấp phát bộ nhớ cho biến.**

Từ khóa extern thường sử dụng trong các trường hợp:

a. Chương trình viết trong một tập tin, biến ngoài được tham trỏ tới trước khi được định nghĩa .

Ví dụ 9.1: // Minh họa extern

```
extern int a,b,c;
main()
{
        cout<<"\nGiá trị của a =  "<<a<<" ;  b =  "<<b<<" ; c  =  "<<c
              <<" của hàm main()";
        next();
}
int a 12,b=34,c=56;
next()
{
        cout<<\nGiá trị của a =  "<<a<<" b =  "<<b<<" c  = "<<c<<" của hàm next()";
}
```

Sau khi chạy chương trình cho kết quả là:

```
Giá trị của  a=12, b= 34, c  = 56
Giá trị của  a=12, b= 34, c  = 56
```

Hàm main() tham trỏ tới các biến ngoài a, b, c trước khi định nghĩa chúng, do Dòng lệnh int a 12,b=34,c=56; đặt sau hàm main(). Nên ta cần khai báo các biến ngoài a,b,c bằng dòng lệnh extern int a,b,c (chỉ là khai báo về tính chất biến và yêu cầu không cấp phát bộ nhớ.)

b) Chương trình gốc được viết trong nhiều tập tin và sử dụng nhiều biến ngoài trùng nhau. Các biến này gọi là các biên chung.

Khi đó các biến chung này chỉ định nghĩa trên tập tin gốc của chương trình, trên các tập tin khác thì thêm vào từ khóa extern để biết đó là các biến ngoài đã được định nghĩa rồi , không cần phải cấp phát bộ nhớ cho chúng và có thể thâm nhập được chúng.

Ghi chú:

Dù có khai báo extern nhưng vẫn phải có định nghĩa biến.

Ví dụ 9.2:

Trong file 1:

```
int sp = 0;
double val[MAXVAL];
```

Trong file 2:

```
extern int sp;
```

```
        extern double val[];
        double push() {...}
        double pop() {...}
        clear() {...}
```

Do khai báo extern trong file 2 nằm ở đầu và ở ngoài 3 hàm trên nên các biến được dùng cả trong 3 hàm.

3. Phạm vi:

a. Trường hợp biến ngoài và biến cục bộ trùng nhau:
   Ưu tiên cho biến cục bộ . Phạm vi tác dụng của biến cục bộ chỉ trong hàm hay khối lênh chứa nó. Bên ngoài phạm vi này là phậm vi tác dụng của biến toàn cục.
b. Toán tử phạm vi:
   Ký hiệu bởi :: (hai dấu hai chấm kế tiếp), được dùng để truy xuất một phần thử bị che bởi phạm vi tạm thời.

Ví dụ 9.3:

```
int a= 2;
int main()
{
        double a = 2.5;
        cout<<"\na cục bộ: "<<a;
        cout<<"\na ngoài : "<<::a; //Lấy a ngoài phạm vi này
        return 0;
}
```

Chương trình sẽ in ra:

```
a cục bộ: 2.5
a ngoài : 2
```

4. Hàm trả về một tham trỏ (tham chiếu).

Dạng hàm:

```
kdl   &ten_ham([Danh_sach_cac_kieu_va_doi])
{
        //Các câu lệnh
}
```

- Hàm sẽ trả về địa chỉ của một biến.
- Có thể gán giá trị cho tên hàm.

Ví dụ 9.4:

```
int &f();       //Khai báo nguyên mẫu
int x;          //Biến toàn cục
int main(void)
{
        f() = 100;
        cout<<"\nx = "<<x;
        return 0;
}
int &f()
{
        return x;
```

}

Câu lệnh return x trong hàm không trả về giá trị biên toàn cục x, nhưng trả về địa chỉ của x (dưới dạng tham trỏ). Nên trong hàm main() câu lệnh f() = 100; đưa giá trị 100 vào x vì f đã trả về một tham chiếu cho nó.

Ghi chú: Khi trả về một tham chiếu, cần lưu ý là biến được tham chiếu không ra ngoài phạm vi.

Chẳng hạn, trong hàm trên ta viết:

```
int &f()
{
        int x;
        return x;
}
```

Trong trường hợp này x là cục bộ và sẽ ra khỏi phạm vi khi hàm f() trả về. Điều này có nghĩa là tham chiếu được trả về bởi f() là vô dụng.

9.1.3 Lớp lưu trữ tĩnh - từ khoá static

Đây là cách lưu trữ tĩnh - trong hoặc ngoài - được xác định bằng từ khóa static.

1) Biến tỉnh trong

- Các đặc trưng:

Vị trí khai báo	Cách khai báo	Địa chỉ trong bộ nhớ c/t.	Thời gian tồn tại	Phạm vi tác dụng	Khởi đầu
Trong hàm.	Thêm từ khóa static trước định nghĩa biến thông thường.	Vùng cấp phát tĩnh.	Trong suốt thời gian chương trình chứa nó hoạt động.	Bên trong hàm chứa nó	Như mảng ngoài

Ví dụ 9.5:(về cách khai báo có từ khóa static.)

```
{
        static  int  n;
        . . .
}
```

Ghi chú:

Biến tĩnh trong khác với biến cục bộ ở điểm: Giá trị của biến tỉnh trong vẫn được lưu giữ khi ra khỏi hàm và giá trị này có thể sử dụng mỗi khi hàm được sử dụng trở lại (vẫn là biến cục bộ nhưng đã mất đi tính tự động).

Biến tự động và biến tĩnh trong (static) đối kháng nhau về phương diện khởi động.

Chẳng hạn:

```
static int n = 20;
```

Ví dụ 9.6:// Minh họa static

```
void In();
int main()
{
        int k ;
```

```
        clrscr();
        for(k = 1; k < 5; k++)
                In();
        _getch();
        return 0;
}
In()
{
        int k = 0;
        cout<<\ngiá trị của k = “<<k++;
}
```

Kết quả chương trình sẽ in 4 dòng:

giá trị của k = 0
giá trị của k = 0
giá trị của k = 0
giá trị của k = 0

Đó là vì:

- Biến k trong hàm main() khác biến k trong In() dù trùng tên (tính cục bộ).
- Gọi hàm In() 4 lần. Mỗi lần gọi, biến k lại được khởi động và lưu trên stack. Sau khi kết thúc hàm In(), biến này lại tự mất đi.Đó là tính tự động của k.

Nếu sửa lại hàm In() như sau:

```
void In()
 {
        static int k = 0;
        cout<<”\ngiá trị của k = "<<k++;
 }
```

Kết quả chương trình sẽ in 4 dòng:

giá trị của k = 0
giá trị của k = 1
giá trị của k = 2
giá trị của k = 3

Đó là vì biến k trong In() trở thành biến tỉnh (trong):

- Vẫn là cục bộ.
- Không còn tính tự động nữa, cụ thể là sau khi khởi động ở ngay lần đầu tiên khi gọi hàm In() , nó vẫn giữ nguyên giá trị khi ra khỏi hàm này.

2) Biến tỉnh ngoài:

- Các đặc trưng:

Vị trí khai báo	Cách khai báo	Địa chỉ trong bộ nhớ c/t.	Thời gian tồn tại	Phạm vi tác dụng	Khởi đầu
Ngoài tất cả các hàm.	Thêm từ khóa static trước định nghĩa biến	Vùng cấp phát tĩnh.	Trong suốt thời gian chương trình chứa	Từ vị trí khai báo đến cuối tập tin	Như mảng ngoài

	thông thường.		nó hoạt động.	chương trình, nhưng không mớ rộng sang tập tin khác.	

Ghi chú:
Phạm vi của nó không được mở rộng sang các tập tin khác bằng từ khóa extern như đối với biến ngoài. Thậm chí ta có thể dùng trùng tên với biến tỉnh ngoài trên các tập tin khác mà không ảnh hưởng lẫn nhau.
3) Cách che một biến ngoài:
Để che (vô hiệu hóa) một biến ngoài nào đó ta dùng từ khóa static.

Ví dụ 9.7: // Dùng static che biến ngoài.

```
int a = 12, b = 34, c=56;
main()
        {
                static int a,b;  // Che 2 biến  a,b
                cout<<\nDung static để che 2 biến ngoài  a,b ";
                cout<<"\nGiá trị của a,b,c trong hàm main(): \n"
                     <<"\na =  "<<a<<" ;  b =  <<b<<"  ; c  =  "<<c;
                next();
                gettch();
        }
next()
        {
                cout<<"\nGiá trị của a,b,c trong hàm next(): \n"
                     <<"\na =  "<<a<<" ;  b =  <<b<<"  ; c  =  "<<c;
        }
```

Kết quả chương trình:

- Giá trị của a = 0, b = 0, C++ = 56 trong hàm main()
- Giá trị của a = 12, b = 34, C++ = 56 trong hàm next()

Nhận xét:
Biến ngoài và biến tĩnh ngoài (static) đối kháng nhau trên phương diện giá trị.

9.1.4 Từ khóa register

Đây là lớp lưu trử cuối cùng gọi là register (thanh ghi).
Từ khóa register dùng khai báo các biến cục bộ và các đối kiểu int hoặc char theo

cách:

```
register int n;
register char ch;
```

- Các biến register được lưu trử trong các thanh ghi SI hoặc DI nếu có thể. Khi các thanh ghi này bận thì các biến register được lưu trử trong ngăn xếp như các biến cục bộ thông thường.
- Các biến register thường được sử dụng làm biến điều khiển để tăng tốc độ thực hiện của các vòng lặp.

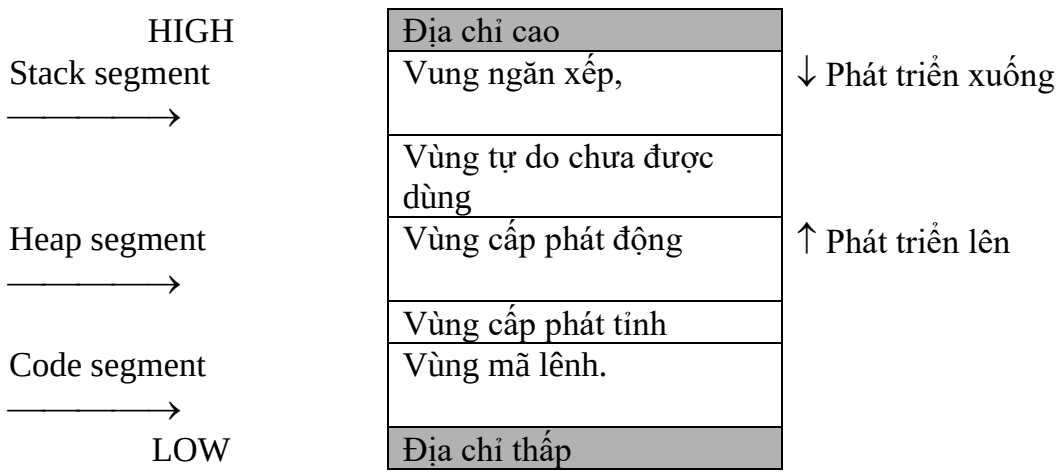
Ví dụ 9.8:

```
double pow(double x, register int n) //x^n
{
        double s = 1.;
        for ( ; n ; --n)
                s *= x;
        return s;
}
```

9.1.5 Bộ nhớ chương trình

Khi một chương trình được thực hiện thì hệ thống dành một vùng nhớ cho chương

trinh này theo sơ đồ:



- Vùng mã lệnh chứa các mã lệnh và hằng; Vùng cấp phát tỉnh chứa các đối tượng ngoài và tỉnh. Hai vùng nhớ này cố độ lớn cố định trong suốt thời gian thực hiện chương trinh.
- Vùng cấp phát động lưu trử các đối tượng (biến,cấu trúc, . . .)được cấp phát bằng hàm new. Vì vậy mỗi khi thực hiện hàm này thì vùng heap nở ra (lấn lên vùng tự do). Ngược lại khi thực hiện hàm delete (để giải phóng một vùng nhớ trên heap) thì vùng heap này giảm lại, vùng tự do lại phình ra.
- Vùng ngăn xếp dùng để lưu trử các đối tượng cục bộ. Mỗi khi chương trình thực hiện một khối lệnh hay hàm, thì các đối tượng khai báo trong khối hoặc hàm đố được lưu trử trong ngăn xếp và ngăn xếp phình ra (lấn xuống vùng tự do). Khi chương trình ra khỏi khối lượng hay hàm thì các đối tượng cục bộ đó bị lấy ra khỏi ngăn xếp, và ngăn xếp bị thu nhỏ lại, vùng tự do lại phình ra..

9.2 Bộ tiền xử lý trong C++

9.2.1 Phép thay thế - chỉ thị #define

Chỉ thị #define cho phép tạo ra các macro đơn giản và các macro thay thế theo tham số hình thức.

1. Chỉ thị #define các macro đơn giản:

a) Cách viết:

```
#define  tên   DAY_KY_TU // Không có dấu ; ở sau
```

Trong đó: DAY_KY_TU có thể là hằng, một tên hàm, một biểu thức, một đoạn chương trinh. . . Nếu là đoạn chương trình thì bắt đầu bởi { và kết thúc bởi }.

b) Tác dụng:

Thay thế tên bởi DAY_KY_TU cho phần còn lại của văn bản chương trình.

c) Vị trí: có thể đặt ở ngoài các hàm, đầu chương trình, hoặc trong hàm .

Ví dụ 2:

```
#define MAX 100    // Thay thế MAX bằng 100
#define ch 'a'                  // Thay thế ch bằng 'a'
#define  begin        {        // Thay thế begin  bằng {
#define  end   }      // Thay thế end  bằng }
#define dt  {cout<<"\nKết thúc"; return 0;}
#define  bt (Biểu_thức)
```

d) Định nghĩa lại:

Có thể định nghĩa lại giá trị thay thế của tên, và cũng với cú pháp trên. Ý nghĩa mới của tên có tac dụng từ vị trí định nghĩa lại đến cuối tập tin chương trình nếu không có sự định nghĩa lại nào khác. Nhưng trước khi định nghĩa lại, thì ta phải giải phóng nó bằng chỉ thị:

```
#undef  Ten
```

Ví dụ 9.9:

```
#define in cout<<
// . . .
int n= 200;
in(n);  // thay thế in bởi  cout
#define n 300// định nghĩa  n là 300
in(n);  // thay thế in là cout ; thay thế n là 300
#undef n //giải phóng n
#define n 100 // Định nghĩa lại n bằng 100
in(n);  // thay thế n là 100
```

2. Chỉ thị #define tạo các macro có tham số:

Khi sử dụng chỉ thị #define tạo các macro có tham số như hàm, thì sự thay thế phụ thuộc vào vào cách gọi tới macro.

Ví dụ 9.10:

```
#define max(A,B) (A)>(B)?(A):(B)// max(A,B) không có khỏang trắng
```

Khi đó dòng lệnh:

```
x= max(p+q,r+s);
```

được thay bằng câu lệnh:

```
x =  (p+q) > (r+s) ?(p+q): (r+s);
```

Như vậy nhờ các macro có tham số ta có thể xây dựng được các hàm đơn giản để sử dụng trong chương trình. Tham số của macro không ứng với với một kiểu dữ liệu nhất định, vì vậy macro ma x(A,B) chỉ phụ thuộc vào kiểu dữ liệu của các biểu thức thay thế cho A và B trong chương trình.

9.2.2 Phép bao hàm tập tin - chỉ thị #include

Chỉ thị #include được sử dụng để chỉ thị cho bộ tiền xử lý nhận nội dung của tập tin khác và đặt chèn vào tập tin chương trình nguồn đang xét.

a) Cách viết:

#include <[đường dẫn]tên_tập tin>

b) Tác dụng:

Trước khi dịch, trình biên dịch sẽ tìm đến tập tin theo đường dẫn ghi trong chỉ thị #include. Nếu tìm thấy thì nội dung của tập tin này được gọi ra và chèn vào tập tin nguồn đang xét tại đúng vị trí của #include. Nếu không tìm thấy, trình biên dịch sẽ thông báo lỗi:

unable to open include file '[đương_dẫn]tên_tập tin'

c) Ưng dụng:

Chỉ thị #include cho phép ghép các modun trên các tập tin khác nhau vào tập tin gốc để tạo thành một chương trình hoàn chỉnh. Chương trình này được dịch và thực hiện theo các qui tắc như đối với chương trình viết trên 1 tập tin.

Ghi chú:

Phép bao hàm tập tin có thể làm cho một tập tin được nối kết nhiều lần vào một chương trình nguồn, do đó nó sẽ được dịch nhiều lần nên có khả năng gây ra lỗi. Để làm cho các tập tin như vậy chỉ được dịch một lần, ta sử dụng chỉ thị biên dịch có điều kiện #ifndef.

Chẳng hạn ta có Tv.h

Để đảm bảo Tv.h chỉ được biên dịch một lần (trong khi include nhiều lần vào tập tin chương trình nguồn), tập tin Tv.h có thể tổ chức như sau:

//Cấu trúc tập tin Tv.h

```
#ifndef  _Tv.h_
#define _Tv.h_
    //Nội dung tập tin Tv.h
#endif
```

9.2.3 Tìm hiểu thêm về tổ chức chương trình bằng win32 console application

Một đề án thường được tổ chức bằng nhiều modun.

- Các tập tin thư viện *.h có thể lưu trử trong header files:
  - Định nghĩa các hằng
  - Định nghĩa các kiểu dữ liệu.
  - Định nghĩa các biến toàn cục.
  - Khai báo nguyên mẫu các hàm chức năng
  - //Có thể định nghĩa các hàm chức năng
- Các tập tin chương trình *.cpp có thể lưu trử trong Source files. Các tập tin *cpp sẽ chèn các tập tin thư viện cần thiết vào đầu tập tin:
  - Các tập tin *.cpp cài đặt các hàm chức năng, các hàm bổ trợ.

- Có một hàm *.cpp cài đặt hàm main(), có thể có các hàm tổ chức menu.

Ví dụ 9.11:

Thực hiện các thao tác trên dãy n số nguyên với các chức năng sau:

1. Trả về chỉ số của phần tử cuối cùng bằng x nếu có; trả về -1 nếu không có.
2. Tổng các số nguyên tố trong dãy
3. Trả về số đường chạy trong dãy

Thực hiện:

Bước 1: Tạo Project với tên "Vd_Tv".

Bước 2: Tạo các tập tin: Tv.cpp, Tv.h

Bước 3: Trong các tập tin Tv.h, soạn code theo cấu trúc:

```
#include<math.h>
//Khai bao nguyen mau
int Cscc(int *a, int n, int x);
int Tong_nT(int *a, int n);
int So_DC(int *a, int n);
int nt(int x);
//-------------------------------
int Cscc(int *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 = Cscc(a,n-1,x);
	return  Kq;
}

int Tong_nT(int *a, int n)
{
	int Kq;
	if ( n==1)
		if (nt(*a))
			Kq = *a;
		else
			Kq = 0;
	else
		if (n >1)
			if (nt(*(a+n-1)))
```

```
                        Kq = Tong_nT(a,n-1) + *(a+n-1);
                else
                        Kq = Tong_nT(a,n-1);
        return Kq;
}

int So_DC(int *a, int n)
{
        int  Kq;
        if (n == 1)
        Kq = 1;
        else
        if(n > 1)
                if(*(a+n-1)<*(a+n-2) )
                        Kq =  So_DC(a,n-1) +1;
                else
                        Kq = So_DC(a,n-1);
        return  Kq;
}
//Cac ham bo tro
int nt(int x)
{
        int Kq, i, m;
        double y;
        if(x < 2)
                Kq = 0;
        else
        {
                Kq = 1;
                y = x;
                m = (int)sqrt(y);
                i = 2;
                while (i <= m && Kq)
                {
                        if(x % i == 0)
                                Kq = 0;
                        i++;
                }
        }
        return Kq;
}
```

- Trong tập tin Tv.cpp:

```
#include<iostream>
#include “Tv.h”
#include<stdlib.h>
using namespace std;
```

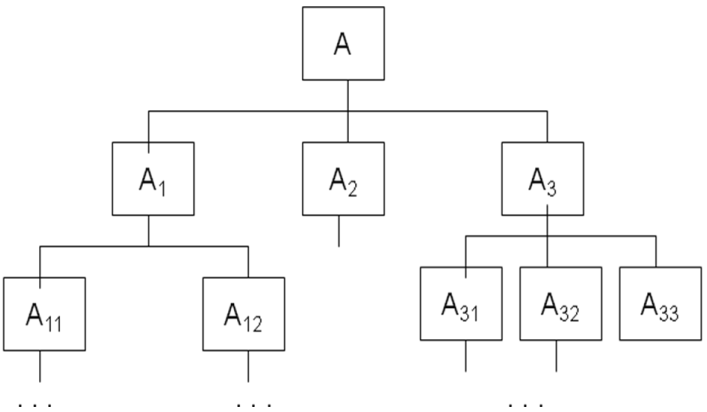
```
void Nhap(int *a, int n);
void Xuat(int *a, int n);
void XL_Menu(int *a, int n, int Chon);
void Menu();
int ChonMenu();
void main()
{
	int *a, n, Chon;
	cout<<"\nnhap n = ";
	cin>>n;
	a = new int[n];
	Nhap(a,n);
	do
	{
		Chon = ChonMenu();
		XL_Menu(a,n, Chon);
	}
	while(1);
	cout<<'\n';
	delete []a;
}

void Menu()
{
	cout<<"\n1. Chi so cua pt cuoi cung bang x";
	cout<<"\n2. Tong cac so nguyen to";
	cout<<"\n3. So duong chay";
	cout<<"\n4. Thoat khoi chuong trinh!!!";
}
int ChonMenu()
{
	int Chon;
	for(;;)
	{
		Menu();
		cout<<"\nnhap Chon tu 1 -> 4: ";
		cin>>Chon;
		if (1 <= Chon && Chon <= 4)
			break;
	}
	return Chon;
}
void XL_Menu(int *a, int n, int Chon)
{
	int Kq, x;
	switch(Chon)
	{
```

```
                case 1:
                        cout<<"\n1. Chi so cua pt cuoi cung bang x";
                        Xuat(a, n);
                        cout<<"\nnhap x = ";
                        cin>>x;
                        Kq = Cscc(a,n, x);
                        if (Kq == -1)
                                cout<<"\n"<<x<<" khong co trong a!";
                        else
                                cout<<"\nchi so cua pt cuoi cung == "<<x<<" la: "<<Kq;
                        break;
                case 2:
                        cout<<"\n2. Tong cac so nguyen to";
                        Xuat(a, n);
                        cout<<"\nTong cac so NT trong a: S =  "<<Tong_nT(a,n);
                        break;
                case 3:
                        cout<<"\n3. So duong chay";
                        Xuat(a, n);
                        cout<<"\nSo duong chay trong a: SDC =  "<<So_DC(a,n);
                        break;
                case 4:
                        cout<<"\n9. Thoat khoi CT!\n";
                        exit(1);
        }
}
//Cac ham nhap xuat
//void Nhap(int *a, int n)
//void Xuat(int *a, int n)
```

9.3 Các nguyên lý lập trình cấu trúc

9.3.1 Phân rã bài toán theo chức năng

Dựa vào các chức năng, các yếu tố cấu thành bài toán, ta phân rã bài toán thành các bài toán con. Đến lượt mình, các bài toán con lại phân rã thành các bài toán nhỏ hơn,... tiếp tục như vậy, cho đến khi gặp các bài toán đơn giản, có thể giải được và kiểm soát được tính đúng của thuật toán . Lời giải của bài toán đã cho sẽ được xác định từ các lời giải của các bài toán con



9.3.2 Phương pháp đi từ trên xuống (Top - Down method)

Đi từ cái chung đến cái riêng,từ kết luận đến cái đã biết,từ tổng thể đến đơn vị. Đây là phương pháp dùng rộng rãi trong quá trình thiết kế và cài đặt chương trình.

9.3.3 Phương pháp làm mịn dần

Phương pháp làm mịn dần gắn liền với quá trình phân rã và thiết kế từ trên xuống, nó chính xác dần thao tác và dữ liệu theo từng mức. Phương pháp này được đề xuất bởi K. Wirth

Ví dụ 9.12:

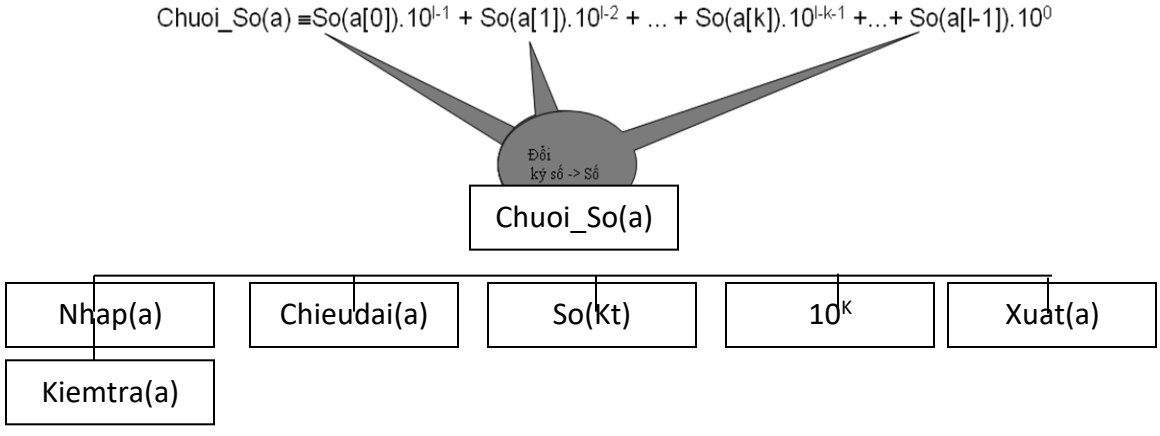
Viết chương trình đổi xâu ký số thành số

i	0	1	2	3	4					
a[i]	1	2	3	4	'\0'					

$$Chuoi\_So(a) \equiv So(a[0]).10^3 + So(a[1]).10^2 + So(a[2]).10^1 + So(a[3]).10^0$$
$$= 1x1000+2x100+3x10+4= 1234$$

i	0	1	2	3	4	...	l-1	l			
a[i]	•	•	•	•	•	...	•	'\0			

$$Chuoi\_So(a) \equiv So(a[0]).10^{l-1} + So(a[1]).10^{l-2} + ... + So(a[k]).10^{l-k-1} + ... + So(a[l-1]).10^0$$



```
int Chuoi_so(char a[MAX])
{
	int Kq = 0, i, l;
	l =  ChieuDai( a);
	for ( i = 0; i < l; i++)
	{
		Kq += So(a[i]) * _10Mu_k(l-1-i);
	}
	return Kq;
}
//------------------------------
//Tinh chieu dai
int ChieuDai(char a[MAX])
{
	int l = 0;
	while (a[l])
		l++;
	return l;
}
//------------------------------
//10 Mu k
int _10Mu_k(int k)
{
	int i, T =1;
	for ( i = 0; i < k; i++)
		T *= 10;
	return T;
}
//------------------------------
//Doi ky so thanh so
int So(char x)
{
	return (x - 48);
}
//------------------------------
//Kiem tra tinh hop le cua du lieu
//Hop le tra ve 1, nguoc lai tra ve 0.
int Kiemtra(char a[MAX])
{
	int Kq = 1, i;
	for (i = 0; (a[i] != NULL) && Kq ; i++)
		if (a[i] < '0' || a[i] > '9')
			Kq = 0;
	return Kq;
}
```

9.3.4 Phương pháp đi từ dưới lên (Bottum - up method)

Đi từ cái riêng đến cái chung, từ các đối tượng thành phần ở mức thấp lên các đối tượng ở mức cao, từ những đơn vị đã biết lắp đặt thành những đơn vị mới.

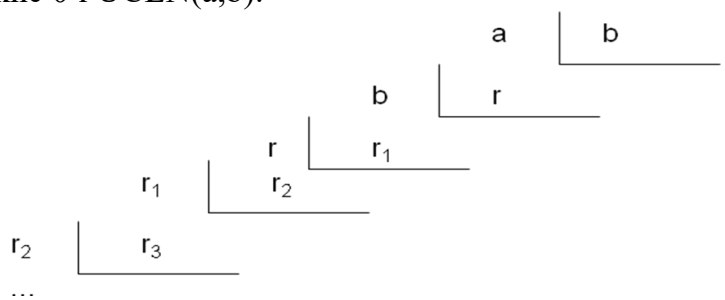
9.3.5 Nguyên lý bất biến

Vòng lặp phải bảo toàn trong thân của nó các quan hệ bất biến

Ví dụ 9.13:

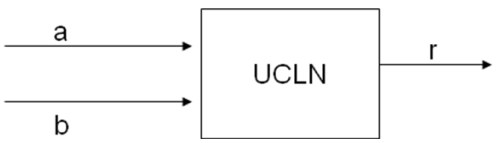
Tìm ước chung lớn nhất của 2 số nguyên dương a, b.

Sử dụng thuật chia Euclide: Muốn tìm UCLN của 2 số tự nhin a v b (giả sử a ≥ b), ta chia nguyn a cho b để tìm số dư r. Sau đĩ nếu r ≠ 0, ta lấy b chia r để tìm số dư mới. Phần dư sau cng khc 0 l UCLN(a,b).



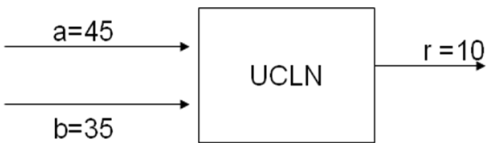
Để diễn đạt quá trình này trong một vòng lặp ta dùng nguyên lý bất biến sau đây:

Hãy tưởng tượng thao tác lấy UCLN là một hộp đen (một bộ biến đổi mà ta không quan tâm đến cấu trúc bên trong của nó). Bộ UCLN có 2 cửa vào được ghi là a và b và một cửa ra r. Với giả thiết UCLN chỉ nhận số bị chia qua cửa a, số chia qua cửa b để cho số dư qua cửa r.

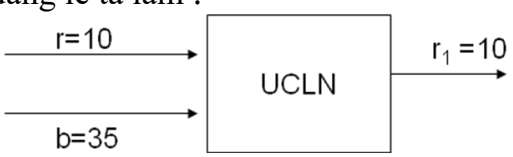


Minh họa:

- Lần lặp thứ nhất

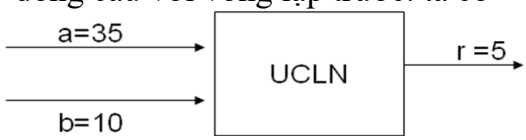


- Lần lặp thứ hai đáng lẽ ta làm :



và điều này mâu thuẫn với qui ước số bị chia ra cửa a, nên ta thấy ngay sự cân thiết thực hiện các phép gán: a =b; b = r;

và ta được vòng lặp này đồng cấu với vòng lặp trước. ta có



Thuật toán Euclide có thể mô tả như sau:

```
while ( b > 0)
{
        r = a%b;
        a = b;
        b = r;
}
return a;
```

9.3.6 Nguyên lý khung nhìn (View)

Cùng một đối tượng có thể nhìn nó theo nhiều cách khác nhau. Nhìn theo cách nào thì tổ chức thao tác theo cách đo

Ví dụ 14:

Tỉa các ký tự trùng nhau trong một xâu ký tự, chỉ để lại một.

Chẳng hạn:

	i	1	2	3	4	5	6	7	8	9	10
	a[i]	x	d	c	x	c	g	h	g	u	NULL
Tỉa	a[i]	x	d	c	g	h	u	NULL			

```
//Cách 1
Input  x;
Output  y;
        h = 0;
        for ( i = 0; x[i] != NULL; i++ )
        {
                j = 1;
                for (k = 0; k < h; k++ )
                        j = j && (x[i] != y[k]);
                if (j)
                y[h++] = x[i];
        }
        y[h] = NULL;
        x = y;
```

//Cách nhìn 2:

- Nhìn x vừa là input, vừa là output. Tức là ta tỉa tại chỗ xâu x chứ không dùng biến phụ y.
- Nhìn x là input, ta duyệt theo chỉ dẫn i. Nhìn x là output, ta duyệt theo chỉ dẫn h. Tức là ta tuân thủ theo quy định:
    - X[i] là ký tự đang xét.
    - Đoạn x[1..i-1] là đoạn nguồn đã duyệt.
    - Đoạn x[1..h] là đoạn đích, trong đó x[h] vừa được bổ sung vào cuối đích.

Đoạn trình trên có thể viết lại như sau:

```
        h= 0;
        for ( i = 0; x[i] != NULL; i++ )
        {
```

```
        j = 1;
        for (k = 0; k < h; k++ )
                j = j && (x[i] != x[k]);
        if (j)
        x[h++] = x[i];
}
x[h] = NULL;
```

CHƯƠNG 10: LẬP TRÌNH VỚI TẬP TIN

10.1 Mở đầu

Trong các chương trước, dữ liệu của chương trình được lưu trữ trong bộ nhớ chính nên sẽ mất đi khi chương trình kết thúc. Trong chương này ta tìm hiểu kỹ thuật lập trình trên tập tin để có thể lưu trữ dữ liệu chương trình vào bộ nhớ phụ nhằm sử dụng lại khi cần thiết.

Kỹ thuật lập trình trên tập tin bao thường gồm các bước:

- Mở tập tin để làm việc.
- Đọc dữ liệu từ tập tin vào biến bộ nhớ hay ghi dữ liệu từ biến bộ nhớ vào tập tin.
- Đóng tập tin.

Ta có thể xem mỗi tập tin như là một dãy các byte được lưu trữ theo một cách nào đó trong bộ nhớ phụ của máy tính. Khi một tập tin được mở, con trỏ tập tin hay là biến xác định vị trí đọc/ghi đặt tại vị trí đầu tập tin hoặc cuối tập tin. Mỗi thao tác đọc/ghi sẽ tác động lên vị trí hiện hành của con trỏ tập tin, và tự động di chuyển con trỏ tập tin đến vị trí kế tiếp, tức là dời qua một số byte nào đó tuỳ theo cách thức tổ chức tập tin.

10.1.1 Các loại tập tin

Có 2 loại chính:

- Tập tin văn bản ASCII.
- Tập tin nhị phân.

10.1.2 Tập tin văn bản

Đó là các tập tin có thể xem và sửa nội dung bằng các chương trình soạn thảo văn bản đơn giản như: các hệ soạn thảo chương trình nguồn của các ngôn ngữ lập trình, trình Notepad của Windows, NC, . . .

Một số loại tập tin này là các tập tin chương trình *.cpp, *.c, *.pas . . .

Mỗi tập tin văn bản ASCII gồm nhiều dòng. Hai dòng liên tiếp được phân cách bởi ký hiệu tách dòng, và kết thúc tập tin bằng ký hiệu kết thúc.

10.1.3 Tập tin nhị phân

Gồm tất cả các tập tin không có dạng văn bản.

Mỗi tập tin nhị phân có cấu trúc do phần mềm tạo ra quy định. Một số dạng tập tin nhị phân thường gặp như:

- Tập tin văn bản có định dạng như tập tin *.Doc của MS Word.
- Các tập tin *.EXE, *.COM của Dos hay Windows.
- Các tập tin hình ảnh như *.BMP, *GIF . .

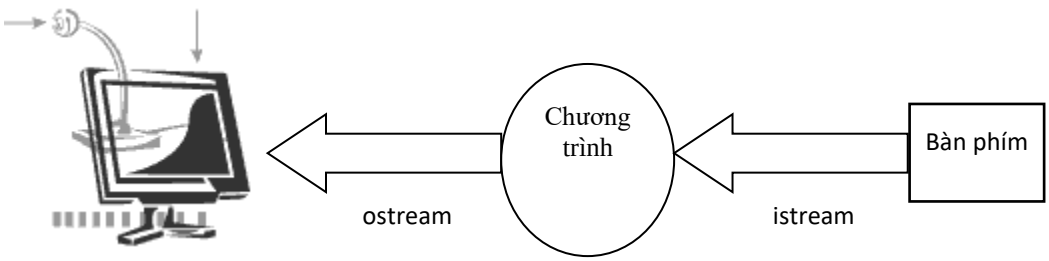
10.2 Nhập/xuất tập tin trong C++

10.2.1 Luồng

Với cách nhập / xuất theo kiểu tương tác, <iostream> tự động thiết lập sự nối kết giữa chương trình thực hiện trong bộ nhớ chính và các thiết bị I/O:

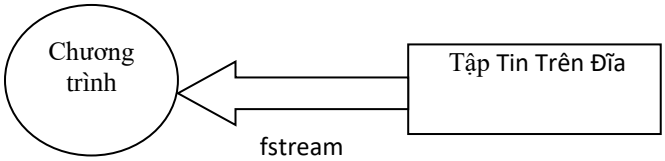
Một đối tượng istream mang tên là cin nối chương trình với bàn phím.

Một đối tượng ostream mang tên cout nối chương trình với màn hình.



Những luồng này (istream và ostream) được tự động tạo ra đối với những chương trình tương tác.

Tuy nhiên đối với một chương trình thực hiện nhập từ một tập tin văn bản hoặc xuất ra một tập tin văn bản chẳng hạn, thì một luồng phải được thiết lập giữa chương trình (trong bộ nhớ chính) và tập tin văn bản (lưu trữ trên đĩa).



Điều này có thể được thực hiện bằng cách dùng các lớp fstream, ifstream, ofstream. Các lớp này được khai báo trong tập tin tiêu đề <fstream >.

Trước khi một chương trình có thể đọc những giá trị từ một tập tin văn bản hoặc ghi những giá trị đến một tập tin văn bản, nó phải cấu tạo một luồng (đối tượng) fstream để hoạt động như là một sự nối giữa chương trình và tập tin.

Có ba loại luồng: nhập, xuất và nhập/xuất.

- Để tạo một luồng nhập (đọc nội dung tập tin) ta phải khai báo luồng đó thuộc ifstream.
- Để tạo một luồng xuất (ghi nội dung vào tập tin) ta phải khai báo luồng đó thuộc ofstream.
- Còn đối với các luồng làm cả các thao tác nhập/xuất (đọc/ghi) ta phải khai báo luồng đó thuộc fstream.

Việc tạo luồng có thể khai báo trong các dạng sau:

```
fstream Tên_Luong;   //Nhập /Xuất.
ifstream Tên_Luong;   //Nhập .
ofstream Tên_Luong;   //Xuất.
```

Ví dụ:

```
ifstream in;    // Mở để đọc – luồng nhập có tên là in
ofstream out;          // Mở để ghi  – luồng xuất có tên là out
fstream io;     // Mở để đọc/ghi  – luồng nhập/xuất có tên là io
```

10.2.2 Mở / Đóng tập tin

1. Mở tập tin

Sau khi tạo ra một luồng bằng cách sử dụng một khai báo thích hợp, ta có thể mở một tập tin gắn liền với luồng đó bằng hàm open().

Hàm này là thành phần của 3 luồng nói trên. Dạng tổng quát của hàm là:

```
void open (char *filename, int mode, int access)
```

Trong đó:

- filename: là tên của tập tin cần mở (có thể đi kèm với tên dường dẫn thư mục)
- mode: Các chế độ của tập tin được mở.
- access: Thuộc tính của tập tin được mở.

Mở tập tin để đọc /ghi /đọc ghi.

a. Các chế độ xác định bởi mode:

ios::in	Mở tập tin để đọc (nhập)
ios::out	Mở tập tin để ghi (xuất)
ios::binary	Mở tập tin ở chế độ nhị phân
ios::app	Kết xuất được thêm vào cuối tập tin.
ios::ate	Chuyển đến vị trí cuối tập tin sau khi tập tin được mở.
ios::nocreate	Làm cho việc gọi hàm open() thất bại nếu tập tin mở là tập tin mới (chưa có sẵn trên đĩa).
ios::noreplace	Làm cho việc gọi hàm open() thất bại nếu tập tin mở là tập tin đã có sẵn trên đĩa.
ios::trunc	Làm cho tập tin có sẵn trùng tên với tập tin mở sẽ bị xoá nội dung.

Ghi chú:

- Các chế độ trên có thể kết hợp lại bằng phép toán | (OR).
- Theo mặc định, tất cả các tập tin được mở để làm việc trong chế độ văn bản.

Trong chế độ văn bản, có một số trường hợp ký tự sẽ bị chuyển đổi. Chẳng hạn ký tự tạo dòng mới LF:

- Khi ghi, một ký tự LF (mã 10) chuyển thành 2 ký tự CR (mã 13) và LF.
- Khi đọc, hai ký tự liên tiếp CR và LF chỉ cho một ký tự LF.

Trong chế độ nhị phân, sẽ không có bất kỳ một trường hợp chuyển ký tự nào.

Bất kỳ một tập tin nào (văn bản hay dữ liệu) đều có thể hoạt động ở cả 2 chế độ văn bản và nhị phân, điểm khác nhau duy nhất của 2 chế độ này là việc chuyển đổi ký tự như đã nêu trên.

b. Các thuộc tính của đối access: (trong hệ điều hành DOS / WINDOWS)

Xác định cách truy xuất tập tin.

Thuộc tính	Ý nghĩa
0	Tập tin bình thường: Ghi / đọc
1	Tập tin chỉ đọc
2	Tập tin ẩn

4	Tập tin hệ thống
8	lưu trữ

Ghi chú:

- Các thuộc tính trên có thể kết hợp lại bằng phép toán | (OR).

c. Các giá trị ngầm định khi mở tập tin bằng hàm open().

- Giá trị mặc định của access là: 0 (ghi / đọc)
- Đối với luồng ifstream: giá trị mặc định của mode là ios:: in
- Đối với luồng ofstream: giá trị mặc định của mode là ios:: out

Trong trường hợp có hiệu lực về các giá trị mặc định, ta không cần ghi đầy đủ về chế độ của mode và thuộc tính access.

2. Các cách mở tập tin.

a. Mở để đọc:

```
fstream in; //Tạo luồng nhập, xuất
in.open("test", ios::in, 0);
```

Hoặc:

```
ifstream in; //Tạo luồng nhập
in.open("test");
```

b. Mở để ghi: Các cách mở sau là tương đương: Khi dùng ofstream

```
fstream out; //Tạo luồng nhập, xuất
out.open("test", ios::out, 0);
```

hoặc:

```
ofstream out; //Tạo luồng xuất
out.open("test");
```

c. Mở tập tin nhập/xuất: dùng fstream

```
fstream mystream;
mystream.open("test", ios:: in | ios:: out);//Kết hợp in,out
```

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”)

Việc chọn lựa cách viết nào là tùy phong cách lập trình của mỗi người.

3. 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

10.2.3 Các thao tác khác trên tập tin

1. 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.

```
if ( !mystream)
{
    cout<<”\nTập tin không mở được!”;
    exit(1);
}
```

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);
}
```

2. *Kiểm tra hết tập tin.*

Dùng hàm thành phần eof() của luồng ?fstream

$$?fstreamName.eof() = \begin{cases} 1; Hết\ dữ\ liệu \\ 0; Ngược\ lại \end{cases}$$

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
}
```

3. *Thao tác đọc dữ liệu từ tập tin, ghi dữ liệu vào tập tin.*

Trong C++ có tính năng dùng những toán tử để thực hiện những nhiệm vụ có chức năng tương tự.

Toán tử nhập >> mà ta đã dùng để nhập dữ liệu từ bàn phím dùng luồng istream bây giờ cũng có thể dùng để nhập dữ liệu từ tập tin thông qua luồng fstream.

Tương tự như vậy cho toán tử xuất.

Với các khai báo:

```
ifstream in(“test1”);
ofstream out(“test2”);
int sonhap;
```

- Đọc:

```
in>>sonhap;
```

- Ghi:

```
out<<sonhap;
```

4. *Đóng tập tin:*

Dùng hàm thành phần close()

Chúng ta đã thấy rằng việc khởi động một đối tượng fstream nhằm thiết lập đối tượng đó như là một kết nối giữa chương trình và tập tin. Một cách tổng quát fstream sẽ không kết nối nữa khi chương trình kết thúc (cũng giống như giá trị của biến mất đi khi chương trình sử dụng biến đó kết thúc). Tuy nhiên kỹ thuật lập trình tốt là chỉ ra sự không liên kết này một cách rõ ràng thông qua hàm thành phần close().

Tổng quát, hàm close() có thể được mô tả như sau:

mystream.close ();

Trong đó:

- Mystream là tên của đối tượng fstream mà nó được xem như là một liên kết đến tập tin.
- Hành vi:
  - Chương trình đang thực hiện và tập tin mang tên FileName không còn liên kết với nhau nữa, và Mystream trở thành không xác định.

10.2.4 Các ví dụ

Ví dụ 1:

Đọc các số nguyên từ bàn phím, dừng khi số đọc vào bằng THOAT, rồi ghi dữ liệu vào tập tin với định dạng 2 giá trị cách nhau 1 tab.

Viết hàm thực hiện thao tác trên và có kiểm tra lại bằng chương trình.

```
#include <iostream>
#include <fstream>
#include <conio.h>
#define THOAT -1
using namespace std;

void write_int(char *filename);

int main(void)
{
	char filename[80];
	system("cls");
	cout<<"Nhap ten file mo de ghi:";
	cin>>filename;
	write_int(filename);
	_getch();
```

```
	return 0;
}
void write_int(char *filename)
{
	ofstream out(filename);	//Mo de ghi
	if (!out)
	{
		cout<<"\nLoi mo file !";
		exit(1);
	}
	int n;
	for(;;)
	{
		cout<<"\nNhap so nguyen(go "<<THOAT<<" de dung): n = ";
		cin>>n;
		if (n == THOAT)
			break;
		out<<n;
		out<<'\t';
	}
	cout<<"\nghi xong du lieu vao tep "<<filename;
	out.close();
}
```

Ví dụ 2:

Giả sử tập tin văn bản \test1.txt có nội dung sau:

5

1 3 2 9 0

Trong đó:

- Hàng 1: chứa số lượng các phần tử của tập tin
- Hàng 2: Các giá trị của tập tin.

Viết hàm đọc các giá trị trong tập tin \test1.txt rồi lưu trữ vào mảng 1 chiều các số nguyên (hoặc xuất ra màn hình).

```
#include <iostream>
#include <fstream>
#include <conio.h>
#define MAX 100
using namespace std;
```

```
void File_Array(char *filename, int Arr[MAX], int &n);

int main()
{
	int n, Arr[MAX];
	char filename[80];
	system("cls");
	cout<<"Nhap ten file mo de doc:";
	cin>>filename;
	File_Array(filename,Arr,n);
	cout<<endl;
	for (int i = 0; i < n; i++)
		cout<<Arr[i]<<'\t';
	_getch();
	return 0;
}

void File_Array(char *filename, int Arr[MAX], int &n)
{
	ifstream in(filename);		//Mo de doc
	if (!in)
	{
		cout<<"\nLoi mo file !";
		exit(1);//return;
	}
	in>>n;
	for (int i = 0; i < n; i++)
	{
		in>>Arr[i];
		cout<<Arr[i]<<'\t';
	}
	in.close();
}
```

```
//Viết dạng khác
int File_Array(char *filename, int Arr[MAX], int &n);

int main()
{
	int n, kq, Arr[MAX];
	char filename[80];
	system("cls");
	cout<<"Nhap ten file mo de doc:";
	 cin>>filename;
	kq = File_Array(filename,Arr,n);
	if (kq)
		for (int i = 0; i < n; i++)
			cout<<Arr[i]<<'\t';
	_getch();
	return 0;
}

int File_Array(char *filename, int Arr[MAX], int &n)
{
	ifstream in(filename);		//Mo de doc
	if (!in)
		return 0; //Khong thanh cong
	in>>n;
	for (int i = 0; i < n; i++)
	{
		in>>Arr[i];
		cout<<Arr[i]<<'\t';
	}
	in.close();
	return 1; //thanh cong
}
```

Ví dụ 3:

Giả sử tập tin văn bản \test2.txt có nội dung là các số nguyên sau:

1 3 2 9 0

Viết hàm đọc các giá trị trong tập tin \test2.txt rồi lưu trữ vào mảng 1 chiều các số nguyên (hoặc xuất ra màn hình).

```
#include <iostream>
#include <fstream>
#include <conio.h>
#define MAX 100
using namespace std;
```

```
int File_Array1(char *filename, int arr[MAX], int &n);
int main()
{
	int n = 0, kq,  arr[MAX];
	char filename[80];
	system("cls");
	cout<<"Nhap ten file mo de doc:";cin>>filename;
	kq = File_Array1(filename,arr,n);
	if(kq)
	{
		cout<<"\nN="<<n;
		cout<<endl;
		for (int i = 0; i < n; i++)
			cout<<arr[i]<<'\t';
	}
	_getch();
	return 0;
}
int File_Array1(char *filename, int arr[MAX], int &n)
{
	ifstream in(filename);		//Mo de doc
	if (!in)
		return 0;
	n = 0;
	while (!in.eof())
	{
		in>>arr[n];
		cout<<arr[n]<<'\t';
		n++;
	}
	in.close();
	return 1;
}
```

Ví dụ 4:

Giả sử tập tin văn bản \test3.txt có nội dung sau:

3

1 2 3

4 5 6

7 8 9

Trong đó:

- Hàng 1: Chỉ cỡ của ma trận vuông.
- Các hàng sau là các phần tử của tập tin

Viết hàm đọc các giá trị trong tập tin \test3.txt rồi lưu trữ vào ma trận vuông cấp n.

```
#include <iostream>
#include <fstream>
#include <conio.h>
#define MAX 100
using namespace std;

void File_Mat(char *filename, int a[MAX][MAX], int &n);

int main()
{
	int n, a[MAX][MAX], i, j;
	char filename[80];
	system("cls");
	cout<<"Nhap ten file mo de doc:";cin>>filename;
	File_Mat(filename,a,n);
	cout<<"\nN="<<n;
	cout<<endl;
	for ( i = 0; i < n; i++)
	{
		cout<<"\n";
		for (j = 0; j < n; j++)
			cout<<a[i][j]<<'\t';
	}
	_getch();
	return 0;
}
void File_Mat(char *filename, int a[MAX][MAX], int &n)
{
	ifstream in(filename);		//Mo de doc
	if (!in)
	{
		cout<<"\nLoi mo file !";
		exit(1);
	}
```

```
        in>>n;
        int i, j;
        for ( i = 0; i < n; i++)
                for (j = 0; j < n; j++)
                        in>>a[i][j];
        in.close();
}
```

Ví dụ 5:

Viết hàm ghi dữ liệu của một ma trận vuông cấp n các số nguyên vào một tập tin.

Yêu cầu ghi theo định dạng như sau:

- Dòng 1: n
- 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
```

```
#include <iostream>
#include <fstream>
#include <conio.h>
#define MAX 100
using namespace std;

void Input_Mat(int a[MAX][MAX], int n);
void Mat_File(char *Filename, int a[MAX][MAX], int n);

int main()
{
        int n, a[MAX][MAX], i, j;
        char filename[80];
        system("cls");
        cout<<"\nNhap so phan tu ma tran:";
        cin>>n;
        Input_Mat(a, n);
        cout<<"Nhap ten file mo de luu:";cin>>filename;
```

```
	Mat_File(filename,a,n);
	_getch();
	return 0;
}
void Input_Mat(int a[MAX][MAX], int n)
{
	int i, j;
	for ( i = 0; i < n; i++)
	{
		for (j = 0; j < n; j++)
		{
			cout<<"a["<<i<<"]["<<j<<"]=";
			cin>>a[i][j];
		}
	}
}
void Mat_File(char *filename, int a[MAX][MAX], int n)
{
	ofstream out(filename);
	if (!out)
	{
		cout<<"\nLoi mo file !";
		exit(1);
	}
	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!";
}
```

Ví dụ 6:

Quản lý một nhân viên theo các thông tin sau:

- Mã số.
- Họ tên.

- Ngày tháng năm sinh.
- Địa chỉ thường trú.
- Lương.
- Số điện thoại.

Nhập thông tin của các nhân viên từ bàn phím, dừng khi mã số của nhân viên là 0.

Viết hàm ghi thông tin của các nhân viên vào một tập tin.

Yêu cầu ghi theo định dạng như sau:

1 Nguyễn A 01/01/2000 1,PĐTV 100000 811800

. . .

```
#include <iostream>
#include <fstream>
#include <conio.h>
#include<iomanip>
#define MAX 20
#define THOAT  0
using namespace std;

void write_struct(char *filename);

int main()
{

	char filename[80];
	system("cls");

	cout<<"Nhap ten file mo de luu:";cin>>filename;
	write_struct(filename);
	_getch();
	return 0;
}

//Ham nhap du lieu tu ban phim roi ghi vao tep
void write_struct(char *filename)
{
	ofstream out(filename);	//Mo de ghi
	if (!out)
	{
```

```
        cout<<"\nLoi mo file !";
        exit(1);//return;
}
int ms,ntn;
char *ten, *diachi;
long luong, sdt;
out<<setiosflags(ios::left);
for(;;)
{
        cout<<"\nNhap ma so  (go "<<THOAT<<" de dung): ms = ";
        cin>>ms;
        if (ms == THOAT)
                break;
        out<<setw(3)<<ms;
        cout<<"\nNhap ten:";
        flushall();
        ten=new char[MAX];
        gets(ten);
        out<<setw(20)<<ten;
        cout<<"\nNhap ngay sinh:";
        cin>>ntn;
        out<<setw(2)<<ntn;
        out<<'/';
        cout<<"\nThang: ";
        cin>>ntn;
        out<<setw(2)<<ntn;
        out<<'/';
        cout<<"\nNam: ";
        cin>>ntn;
        out<<setw(5)<<ntn;
        cout<<"\nNhap dia chi:";
        flushall();
        diachi=new char[MAX];
        gets(diachi);
        out<<setw(20)<<diachi;
        cout<<"\nNhap luong:";
        cin>>luong;
        out<<setw(8)<<luong;
        cout<<"\nNhap SDT:";
        cin>>sdt;
        out<<setw(8)<<sdt;
```

```
            out<<'\n';
        }
        cout<<"\nda ghi xong du lieu vao tep "<<filename;
        out.close();
}
```

Ví dụ 7:

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, Năm sinh, địa chỉ, Lương, số điện thoại.

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 (hoặc xuất ra màn hình).

```
#include <iostream>
#include <fstream>
#include <conio.h>
#include<iomanip>
#define MAX 20
#define THOAT  0
using namespace std;

struct date
{
        int ngay;
        int thang;
        int nam;
};
struct nhanvien
{
        int ms;
        char hoten[MAX];
        date ntn;
        char diachi[MAX];
        long luong;
        long sdt;
};
void xuat (nhanvien ds[MAX], int n);
int read_struct(char *filename,nhanvien ds[MAX]);
//**************************
int main(void)
{
```

```
	char filename[80];
	nhanvien ds[MAX];
	system("cls");
	cout<<"Nhap ten file mo de doc:";cin>>filename;
	int n = read_struct(filename,ds);
	xuat(ds,n);
	_getch();
	return 0;
}
int read_struct(char *filename,nhanvien ds[MAX])
{
	ifstream in(filename);
	if (!in)
		return 0;//khong thanh cong
	int 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;
		in>> ds[n].sdt;
		n++;
	}
	in.close();
	return  n; //thanh cong : trả về số lượng nhân viên
}
void xuat (nhanvien ds[MAX], int n)
{
	cout<<setiosflags(ios:: left);
	cout<<setw(3)<<"MS"
		<<setw(20)<<"Ho Ten"
		<<setw(11)<<"NTN Sinh"
		<<setw(20)<<"Dia chi"
		<<setw(8)<<"Luong"
		<<setw(8)<<"SDT";
	cout<<endl;
	for(int i = 0; i < n; i++)
```

```
        {
                cout<<setw(3)<<ds[i].ms
                        <<setw(20)<<ds[i].hoten
                        <<setw(2)<<ds[i].ntn.ngay<<'/'
                        <<setw(2)<<ds[i].ntn.thang<<'/'
                        <<setw(5)<<ds[i].ntn.nam
                        <<setw(20)<<ds[i].diachi
                        <<setw(8)<<ds[i].luong
                        <<setw(8)<<ds[i].sdt;
                cout<<endl;
        }
}
```

10.3 Nhập/Xuất nhị phân không định dạng

Nhập/xuất nhị phân cho phếp đọc/ghi từng khối dữ liệu (tính theo byte)

istream &get(char &ch)	Đọc một ký tự từ luồng và gán ký tự đó vào nội dung của ch. Hàm này trả về một tham chiếu đến luồng, nó là rỗng nếu hết tập tin
ostream &put(char ch)	Xuất nội dung của ch ra luồng và trả về một luồng
istream &read(unsigned char *buf, int num)	Đọc một số lượng num các byte từ luồng và ghi lên một vùng đệm được trỏ bởi buf.
ostream &write(const unsigned char *buf, int num)	Ghi một số lượng num các byte ở vùng đệm trỏ bởi buf đến luồng
int gcount()	Trả về tổng số ký tự đọc được ở lần đọc cuối ngay trước khi hết tập tin

Ghi chú:

Vì vùng đệm không được định nghĩa là một dãy ký tự nên ta sẽ dùng khai báo linh hoạt (char*) khi gọi hàm write() hoặc read().

Ví dụ 8:

```
#include <iostream>
```

```
#include <fstream>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
using namespace std;
int main()
{
	system("cls");
	// Mở để ghi
	ofstream out("test");
	if( !out)
	{
		cout<<"\n Khong mo duoc";
		return 0;
	}
	double num = 100.45;
	char st[] = "Van ban";
	out.write((char*)&num,sizeof(double));
	out.write(st,strlen(st));
	out.close();
	// Mở để đọc
	ifstream in("test");
	if( !in)
	{
		cout<<"\n Loi mo file";
		return 0;
	}
	system("cls");
	in.read((char*)&num,sizeof(double));
	cout<<num;
	cout<<endl;
	in.read(st,strlen(st));
	cout<<st;
	_getch();
	return 0;
}
```

10.4 Truy cập ngẫu nhiên

C++ sử dụng 2 con trỏ để quản lý tập tin là con trỏ get và con trỏ put. Con trỏ get trỏ đến vị trí sẽ được đưa vào nếu tập tin được thực hiện một tác vụ nhập nữa. Con trỏ put xác định ví trí của con trỏ mà tác vụ xuất tiếp theo sẽ đọc ra từ đó.

Mỗi khi thực hiện một thao tác nhập/xuất tập tin, con trỏ tương ứng sẽ được tự động tăng lên để chỉ vào vị trí kế tiếp.

Tuy nhiên hệ thống nhập/xuất của c++ cũng cung cấp nhiều hàm cho phép truy cập ngẫu nhiên các tập tin.

istream &seekg(streamoff offset, seek_dir origin) streamoff là một kiểu đã được định nghĩa trong iostream	Dời con trỏ get của tập tin một đoạn bằng offset byte từ vị trí được xác định bằng origin.
ostream &seekp(streamoff offset, seek_dir origin)	Dời con trỏ put của tập tin một đoạn bằng offset byte từ vị trí được xác định bằng origin.
Các giá trị liệt kê seek_dir ios:: beg - 0 ios:: cur - 1 ios:: end - 2	 - Tìm từ vị trí đầu tập tin. - Tìm từ vị trí hiện hành. - Tìm từ vị trí cuối tập tin.

streampos tellg()	Xác định vị trí hiện hành của các con trỏ get và put.
streampos tellg() streampos là một kiểu đã được định nghĩa trong iostream	

seekg(0,2)	Con trỏ get chỉ vị trí cuối tập tin
l = X.tellg() X: tên luồng	Số byte của tập tin

Ví dụ 9:

Nhập dữ liệu cho ma trận vuông cấp n các số nguyên từ bàn phím rồi ghi vào tập tin.Viết hàm thực hiện thao tác trên, có kiểm tra bằng chương trình.

```
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#define MAX 3
using namespace std;
void nhap (int a[MAX][MAX], int n);
//********************
int main()
{

	char tenfile[60];

	int a[MAX][MAX], n=3;
	//Mở để ghi
	system("cls");
	cout<<"Nhap ten file can mo(de doc):";
	cin>>tenfile;
	nhap(a,n);
	ofstream out(tenfile, ios_base::binary | ios_base::out);
	if( !out)
	{
		cout<<"\nLoi mo tep";
		return 0;
	}
	out.write((char *)&a, n*n*sizeof(int));
	out.close();
	cout<<"\nDa ghi du lieu vao tep "<<tenfile;
	_getch();
	return 0;
}
//********************
void nhap (int a[MAX][MAX], int n)
{
	for(int i=0; i<n; i++)
		for(int j=0; j<n; j++)
```

```
            {
                cout<<"a["<<i<<"]["<<j<<"]=";
                cin>>a[i][j];
            }
}
```

Ví dụ 10:

Đọc tập tin các số nguyên rồi lưu trữ vào một ma trận vuông.

```
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#define MAX 3
using namespace std;
void xuat (int a[MAX][MAX], int n);
//********************
int main(void)
{
    int in_array[MAX][MAX];
    char tenfile[60];
    int n;
    system("cls");
    cout<<"Nhap ten file can mo(de doc):";
    cin>>tenfile;
    fstream in(tenfile, ios_base::binary | ios_base::in);
    in.seekg(0,ios::end);
    int l = in.tellg();
    n = l/sizeof(int);
    n = (int)sqrt((double)n);
    cout<<"nn="<<n;
    in.seekg(0, ios::beg);
    in.read((char *)&in_array, n*n*sizeof(int));
    in.close();
    xuat(in_array, n);
    _getch();
    return 0;
}
//********************
void xuat (int a[MAX][MAX], int n)
{
```

```
        for(int i=0; i<n; i++)
        {
                cout<<"\n";
                for(int j=0; j<n; j++)
                        cout<<"\t"<<a[i][j];
        }
}
```

BÀI TẬP

Bài 1:

Viết hàm trả về số dòng trong nội dung của một tệp văn bản.

Bài 2.

Viết hàm nối 2 tệp.

Bài 3:

Hồ sơ các học sinh của lớp học gồm tên, tuổi, điểm trung bình. Viết chương trình với yêu cầu sau:

- Nhập thông tin n học sinh và lưu trữ vào tệp Hoc_Sinh
- Đọc thông tin từ tệp Hoc_Sinh:
    - In danh sách lớp theo thứ tự tăng dần của tên (theo thứ tự từ điển).
    - In ra danh sách các học sinh phải thi lại (điểm trung bình < 5).
    - In ra các học sinh hạng giỏi, (điểm trung bình > 9).

Bài 4:

Để quản lý số điện thoại, ta lưu trữ các thông tin về các thuê bao gồm: Hoten, Diachi, SoDienThoai.

Viết chương trình nhập thông tin n thuê bao và lưu trữ vào tệp Thue_Bao, và thực hiện các yêu cầu:

1. Thêm một số điện thoại mới.
2. Xóa 1 thuê bao.
3. Tìm kiếm số điện thoại theo Họ Tên.
4. Tìm Họ Tên khi biết số điện thoại.
5. Tìm địa chỉ khi biết số điiện thoại.
6. Kết thúc chương trình.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Tiến Huy & Trần Hạnh Nhi (1997), giáo trình "Kỹ thuật lập trình ", Trường Đại học khoa học tự nhiên t.p. Hồ Chí Minh, 1997

[2] Niklaus Wirth, N.Q.Cường dịch: *Cấu trúc dữ liệu + Giải thuật = Chương trình*, Nhà xuất bản ĐH &THCN, 1991.

[3] Joel Adams, Sanford Leestma & Larry Nyhoff (1995). C++ An Introduction to Computing. First Edition, Prentice-Hall, Inc.

[4] Patrick Henry Winston (1994). On To C++. First Edition, Addision-Wesley Publishing Company

[5] Trần Tuấn Minh (2002), giáo trình "Kỹ thuật lập trình", Trường Đại học Đà Lạt