

## MỤC LỤC

---

### Contents

<b>Bài 1. Tổng quan về lập trình và ngôn ngữ lập trình C.....</b>	<b>4</b>
I.Các khái niệm cơ bản.....	4
II.Các bước xây dựng chương trình.....	5
III.Biểu diễn thuật toán.....	7
IV.Cài đặt thuật toán bằng ngôn ngữ lập trình.....	7
<b>Bài 2 : CÁC KIỂU DỮ LIỆU TRONG C .....</b>	<b>9</b>
I.Các kiểu dữ liệu cơ sở .....	9
II. Biến, hằng , biểu thức và câu lệnh.....	9
III.Các lệnh nhập xuất.....	13
<b>BÀI 3 : CẤU TRÚC ĐIỀU KHIỂN.....</b>	<b>16</b>
I.Cấu trúc rẽ nhánh .....	16
II. Cấu trúc lặp.....	21
<b>Bài 4: Mảng .....</b>	<b>23</b>
I.Khái niệm .....	23
II.Khai báo.....	23
III.Truy xuất dữ liệu của mảng .....	25
<b>Bài 5: GIỚI THIỆU VỀ C++ .....</b>	<b>31</b>
I. Sự khác biệt giữa C++ và C.....	31
II. Nhập Xuất Trong C++.....	32
<b>Bài 6: Xử lý vào/ ra trên file văn bản ở C/C++.....</b>	<b>34</b>
I.Cách sử dụng với fstream. ....	34
II.Cách sử dụng với kiểu FILE .....	38
III.Chuyển hướng standard input và standard output ra file .....	42
<b>Bài 7: Xâu (Chuỗi) .....</b>	<b>43</b>
I.Khai báo chuỗi.....	43
II.Nhập chuỗi từ bàn phím.....	44
III.Xuất chuỗi lên màn hình.....	44
IV.Một số hàm xử lý chuỗi (trong string.h) .....	44

# CLB Lập Trình PTIT – ProPTIT

V.Bảng tóm tắt các hàm .....	49
<b>Bài 8: Độ quy .....</b>	<b>50</b>
I.Khái niệm: .....	50
II.Độ qui trong lập trình:.....	50
III.Quay lui . .....	54
<b>Bài 9: Cấu trúc STRUCT .....</b>	<b>60</b>
I.Mở đầu: .....	60
II.Xây dựng kiểu cấu trúc, khai báo: .....	61
<b>Bài 10: Con trỏ .....</b>	<b>64</b>
I. Khai báo con trỏ: .....	65
II. Dùng con trỏ để trỏ tới địa chỉ 1 biến: .....	65
III. Truy xuất: .....	65
IV. Một số phép toán trên con trỏ: .....	65
V. Tham chiếu và tham trị: .....	66
<b>Bài 11: Các phương pháp sắp xếp .....</b>	<b>67</b>
I. Sắp xếp nổi bọt .....	67
II. Sắp xếp chọn .....	68
III. Sắp xếp nhanh : .....	69
IV. Sort Algorithm .....	71
<b>Bài 12: String.....</b>	<b>74</b>
I. Kiểu chuỗi của C và hạn chế .....	74
II. Kiểu chuỗi string của C++ .....	74
III. Các phương thức tiện ích của string .....	74
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>83</b>

### LỜI NÓI ĐẦU

Thay mặt cho những người biên soạn tập tài liệu này, tôi có đôi lời muốn nói với các bạn.

Trước hết, những người soạn tài liệu này **KHÔNG** phải là GIÁO SU, TIẾN SĨ.... Hay có một trình độ học vấn uyên bác nào đó. Nhưng tất cả những kiến thức trong này đều được tích góp từ những nguồn đánh tin cậy từ “bồ đề google”..., đó là những phần kiến thức lý thuyết mà các THẦY GIÁO/ GIÁO SU/ TIẾN SĨ đã từng giảng dạy từ các trường đại học, nhưng đã được biến chuyển, hay chuyển thể sang cách học “Bình dân” nhất để những người mới bắt đầu không thấy lạ lẫm hay quá khó khăn để bắt đầu với một ngôn ngữ lập trình.

Những người biên soạn chúng tôi, đã từng là những người chưa biết gì về lập trình, và chúng tôi đã phải học, phải sai để có thể ngồi đây biên soạn ra tập tài liệu này. Là một người đi trước, nên chúng tôi sẽ biết được bạn sẽ sai đâu, sẽ vướng mắc ở đâu, vậy nên trong bộ tài liệu này, chúng tôi đã giảm tải những gì quá cao siêu hoặc uyên bác mà các nhà lập trình giỏi họ nói, để hạ mức của nó xuống, phổ thông giúp các bạn có thể dễ dàng tiếp cận hơn. Vì vậy tôi hi vọng đây là tập tài liệu hữu ích với các bạn.

Hơn thế nữa, chúng tôi cũng là Sinh viên như chính các bạn, vì vậy tầm hiểu biết không thể quá sâu rộng có thể giúp bạn giải đáp mọi thắc mắc được. Cũng vì vậy tập tài liệu này sẽ có đôi chỗ còn thiếu sót, mong các bạn sẽ cùng đóng góp, nghiên cứu trao đổi để chúng ta có một bộ tài liệu đầy đủ nhất có thể. Chúng tôi cần sự hợp tác của các bạn.

Chúc các bạn thành công!

# Bài 1. Tổng quan về lập trình và ngôn ngữ lập trình C

## I. Các khái niệm cơ bản

### 1. Lập trình máy tính

- Gọi tắt là lập trình (programming).
- Nghệ thuật cài đặt một hoặc nhiều thuật toán trừu tượng có liên quan với nhau bằng một ngôn ngữ lập trình để tạo ra một chương trình máy tính.

### 2. Thuật toán

- Là tập hợp (dãy) hữu hạn các chỉ thị (hành động) được định nghĩa rõ ràng nhằm giải quyết một bài toán cụ thể nào đó.

#### a. Ví dụ

Thuật toán giải PT bậc nhất:  $ax + b = 0$

(a, b là các số thực).

Input : a, b thuộc R

Output : đưa ra nghiệm x.

#### b. Các bước giải

- Nếu  $a = 0$ 
  - +  $b = 0$  thì phương trình có nghiệm bất kì.
  - +  $b \neq 0$  thì phương trình vô nghiệm.

Nếu  $a \neq 0$

Phương trình có nghiệm duy nhất  $x = -b/a$

### 3. Các tính chất của thuật toán

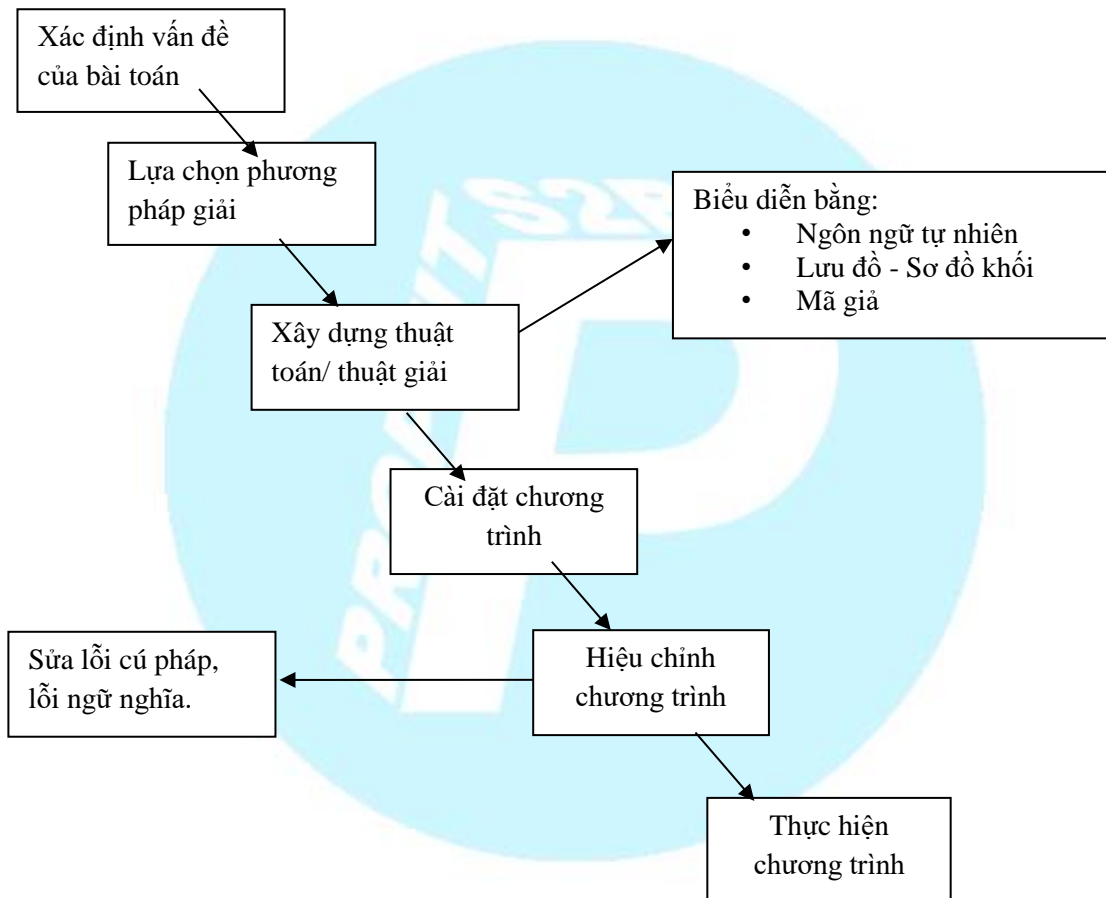
- Tính chính xác: quá trình tính toán hay các thao tác máy tính thực hiện là chính xác.
- Tính rõ ràng: các câu lệnh minh bạch được sắp xếp theo thứ tự nhất định.
- Tính khách quan: được viết bởi nhiều người trên máy tính nhưng kết quả phải như nhau.

## CLB Lập Trình PTIT – ProPTIT

- Tính phổ dụng: có thể áp dụng cho một lớp các bài toán có đầu vào tương tự nhau.
- Tính kết thúc: hữu hạn các bước tính toán.

## II. Các bước xây dựng chương trình

### 1. Xây dựng chương trình



## Ví dụ 2.1

Input : a, b thuộc R

Output : nghiệm phương trình  $ax + b = 0$

## Các bước giải

1. Nhập 2 số thực a và b.
2. Nếu  $a = 0$  thì
  - 2.1. Nếu  $b = 0$  thì
    - 2.1.1. Phương trình vô số nghiệm
    - 2.1.2. Kết thúc thuật toán.
  - 2.2. Ngược lại
    - 2.2.1. Phương trình vô nghiệm.
    - 2.2.2. Kết thúc thuật toán.
3. Ngược lại
  - 3.1. Phương trình có nghiệm.
  - 3.2. Giá trị của nghiệm đó là  $x = -b/a$
  - 3.3. Kết thúc thuật toán.

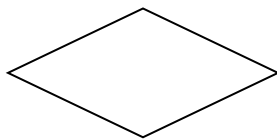
## 2.Sử dụng sơ đồ khối



Hình bình hành mô tả thao tác nhập/xuất dữ liệu.



Hình chữ nhật mô tả tác tính toán.

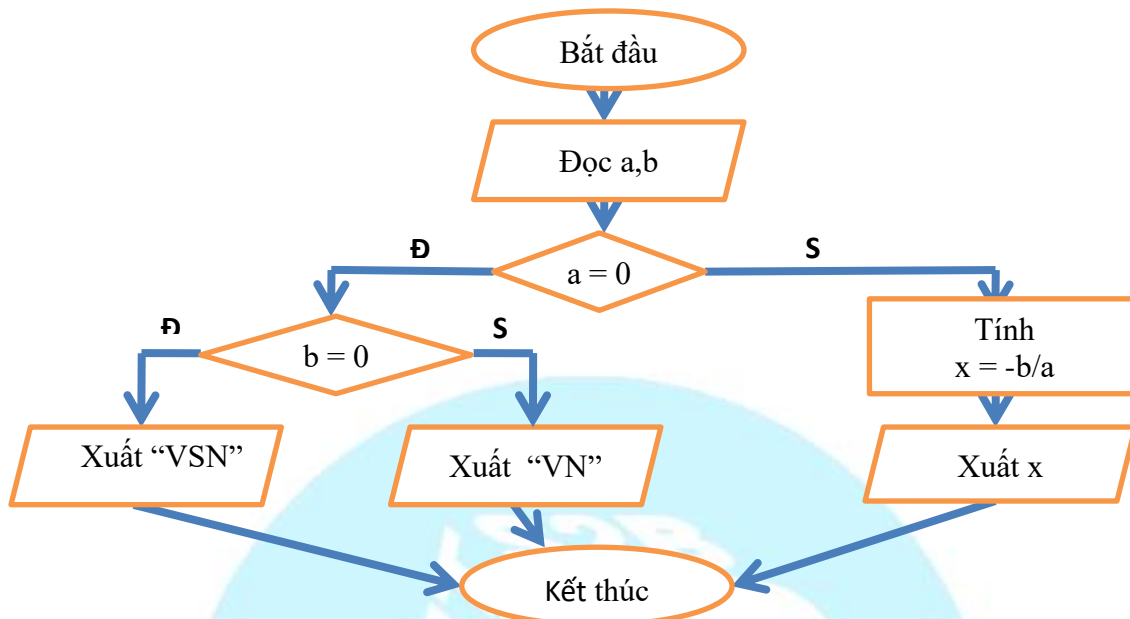


Hình thoi mô tả thao tác lựa chọn.



Mũi tên mô tả hướng thực hiện.

### III. Biểu diễn thuật toán



### IV. Cài đặt thuật toán bằng ngôn ngữ lập trình

Mỗi 1 ngôn ngữ có một cách biểu diễn bài toán riêng

Lấy ví dụ như biểu diễn cùng một bài toán trên ở Pascal và C là khác nhau

Cài đặt bằng Pascal

```
program giaipb1;
uses crt;
var a,b:real;
begin
  clrscr;
  writeln('Chương Trình Giai PT bac 1: ax+b=0');
  writeln('Nhập a và b'); readln(a,b);
  If (a=0) and (b<>0) then writeln('Phương trình vô nghiệm');
  If (a=0) and (b=0) then writeln('Phương trình có vô số nghiệm');
  If (a<>0) and (b<>0) then writeln('Phương trình có nghiệm x=', (-b)/a);
  readln;
end.
```

## Cài đặt bằng C

```
1  #include <stdio.h>
2  void main(){
3      float a, b;
4      printf("\nGiai phuong trinh bac nhat AX + B = 0");
5      printf("\nCho biet ba he so A B : ");
6      scanf("%f%f", &a, &b);
7      if (a == 0){
8          if (b != 0) printf("Phuong trinh vo nghiem");
9          else printf("Phuong trinh co nghiem khong xac dinh");
10     }
11     else printf("Dap so cua phuong trinh tren = %f", -b / a);
12     getch();
13 }
```





## Bài 2 : CÁC KIỂU DỮ LIỆU TRONG C

### I.Các kiểu dữ liệu cơ sở

Kiểu	Dung lượng xấp xỉ (đơn vị là bit)	Phạm vi
char	8	-128 tới 127
unsigned	8	0 tới 255
signed char	8	-128 tới 127
int	16	-32,768 tới 32,767
unsigned int	16	0 tới 65,535
signed int	16	Giống như kiểu int
short int	16	-128 tới 127
unsigned short int	16	0 tới 65, 535
signed short int	16	Giống như kiểu short int
long int	32	-2,147,483,648 tới 2,147,483,647
signed long int	32	Giống như kiểu long int
unsigned long int	32	0 tới 4,294,967,295
float	32	6 con số thập phân
double	64	10 con số thập phân
long double	128	10 con số thập phân

## II. Biến, hằng , biểu thức và câu lệnh

### 1.Biến

#### a. Ví dụ

```
int i, m, n, a, b c ; // Khai báo kiểu nguyên các biến i, m, n, a, b , c
```

# CLB Lập Trình PTIT – ProPTIT

char xau, chuoi ; // Khai báo kiểu kí tự cho biến xau, chuoi

## b. Cú pháp khai báo

<kí hiệu biến> <tên biến> ;

<kí hiệu biến> <tên biến 1> , <tên biến 2> , <tên biến 3> ;

## Chú ý

Phải khai báo biến trước khi sử dụng.

Trong C/ C++ có thể khai báo biến ở:

- + Ngoài hàm
- + Đầu hàm
- + Tham số hàm
- + Trong chương trình

## 2.Hằng

### a. Ví dụ:

```
#define PI 3.14 // không có dấu = hoặc dấu ;  
#define MAX 100
```

Cú pháp:

#define <tên hằng> <giá trị>

## 3.Biểu thức

### a. Khái niệm

- Tạo thành từ các toán tử (Operator) và các toán hạng (Operand).
- Toán tử tác động lên các giá trị của toán hạng và cho giá trị có kiểu nhất định.
- Toán tử: +, -, \*, /, %....
- Toán hạng: hằng, biến, lời gọi hàm...

Ví dụ

-  $2 + 3$ ,  $a \text{ div } 5$ ,  $(a + b) * 5$ , ...

### 3.1. Toán tử gán

Khái niệm

- Thường được sử dụng trong lập trình.

- Gán giá trị cho biến.

Cú pháp

$\langle \text{biến} \rangle = \langle \text{giá trị} \rangle;$

$\langle \text{biến} \rangle = \langle \text{biến} \rangle;$

$\langle \text{biến} \rangle = \langle \text{biểu thức} \rangle;$

- Có thể thực hiện liên tiếp phép gán.

c. Ví dụ:

```
1  #include <stdio.h>
2  main() {
3      int a,b,c,d,e,thuong;
4      a = 10;
5      b = a;
6      thuong = a / b;
7      a = b = c = d = e = 156;
8      e = 156;
9      d = e;
10     c = d;
11     b = c;
12     a = b;
13 }
```

### 3.2. Toán tử toán học

Toán tử 2 ngôi:

Có hai toán hạng trong biểu thức.

$+$ ,  $-$ ,  $*$ ,  $/$ ,  $\%$  (chia lấy phần dư)

$x = x + y \Leftrightarrow x += y;$

Ví dụ:

```
a = 1 + 2; b = 1 - 2; c = 1 * 2; d = 1 / 2;  
e = 1*1.0 / 2; f = float(1) / 2; g = float(1 / 2);  
h = 1 % 2;  
x = x * (2 + 3*5); ô x *= 2 + 3*5;
```

### 3.3.Toán tử quan hệ

#### a. Các toán tử quan hệ

- So sánh 2 biểu thức với nhau

- Cho ra kết quả 0 (hay false nếu sai) hoặc 1 (hay true nếu đúng)

(== , > , >= , < , <= , !=)

#### b.Ví dụ

```
s1 = (1 == 2);      s2 = (1 != 2);  
s3 = (1 > 2);       s4 = (1 >= 2);  
s5 = (1 < 2);       s6 = (1 <= 2);
```

## 4. Câu lệnh

Khái niệm:

Là một chỉ thị trực tiếp, hoàn chỉnh nhằm ra lệnh cho máy tính thực hiện một số tác vụ nhất định nào đó.

Trình biên dịch bỏ qua các khoảng trắng (hay tab hoặc xuống dòng) chen giữa lệnh.

Ví dụ:

```
5 a=2912;  
6 a =      2912;  
7 a  
8 =  
9 2912;
```

Phân loại:

Câu lệnh đơn: chỉ gồm một câu lệnh.

Câu lệnh phức (khối lệnh): gồm nhiều câu lệnh đơn được bao bởi { và }

Ví dụ:

## CLB Lập Trình PTIT – ProPTIT

```
a = 2912;           // Câu lệnh đơn

{                  // Câu lệnh phức/khối lệnh

    a = 2912;

    b = 1706;

}
```

### III. Các lệnh nhập xuất

Thư viện nhập xuất     `#include <stdio.h>` (standard input/output)

#### 1) Câu lệnh nhập

Cú pháp

`scanf(<chuỗi định dạng>[, <đs1>, <đs1>, ...]);`

<chuỗi định dạng> giống định dạng xuất nhưng chỉ có các đặc tả.

Các đối số là tên các biến sẽ chứa giá trị nhập và được đặt trước dấu &

Ví dụ cho a và b kiểu số nguyên

```
3   scanf ("%d", &a);           // Nhập giá trị cho biến a
4   scanf ("%d", &b);           // Nhập giá trị cho biến b
5   scanf ("%d%d", &a, &b);     // Nhập giá trị cho 2 biến a,b
```

Các câu lệnh sau đây sai

```
3   scanf ("%d", a);           /// Thiếu dấu &
4   scanf ("%d", &a, &b);      /// Thiếu %d cho biến b
5   scanf ("%f", &a);          /// a là biến kiểu số nguyên
6   scanf ("%9d", &a);         /// không được định dạng
7   scanf ("a = %d, b = %d", &a, &b); // thừa dấu nháy kép ở cuối
```

#### 2) Câu lệnh xuất

`printf(<chuỗi định dạng>[, <đs1>, <đs1>, ...]);`

<chuỗi định dạng> là cách trình bày thông tin xuất và được đặt trong cặp nháy kép “ ”.

+ Văn bản thường (literal text)

+ Ký tự điều khiển (escape sequence)

+ Đặc tả (conversion specifier)

### 3) Văn bản thường

Được xuất y hệt như lúc gõ trong chuỗi định dạng.

Ví dụ

+ Xuất chuỗi Hello World

```
printf("Hello "); printf("World");  
printf("Hello World");
```

+ Xuất chuỗi a + b

```
printf("a + b");
```

### 4) Ký tự điều khiển

Ký tự điều khiển (escape sequence)

-Gồm dấu \ và một ký tự như trong bảng sau:

Ký tự điều khiển	Ý nghĩa
\a	Tiếng chuông
\b	Lùi lại một bước
\n	Xuống dòng
\t	Dấu tab
\\	In dấu \
\?	In dấu ?
\"	In dấu “

Ví dụ

```
3 printf("\t"); printf("\n");  
4 printf("\t\n");
```

## 5) Đặc tả

- Gồm dấu % và một ký tự.
- Xác định kiểu của biến/giá trị muốn xuất.
- Các đối số chính là các biến/giá trị muốn xuất, được liệt kê theo thứ tự cách nhau dấu phẩy.

Đặc tả	Ý nghĩa	
%c	Ký tự	char
%d, %ld	Số nguyên có dấu	int, short, long
%f, %lf	Số thực	float, double
%s	Chuỗi ký tự	char[], char*
%u	Số nguyên không dấu	unsigned int/short/long

Ví dụ

```

1  #include <stdio.h>
2  main() {
3      int a = 10, b = 20;
4      printf("%d", a);           // -> Xuất ra 10
5      printf("%d", b);           // -> Xuất ra 20
6      printf("%d %d", a, b);     // -> Xuất ra 10 20
7      float x = 15.06;
8      printf("%f", x);           // -> Xuất ra 15.060000
9      printf("%f", 1.0/3);       // -> Xuất ra 0.333333
10 }
```

## BÀI 3 : CẤU TRÚC ĐIỀU KHIỂN


### I.Cấu trúc rẽ nhánh

#### 1)Lý thuyết

##### a)Câu lệnh rẽ nhánh thiếu – if

Câu lệnh if cho phép lựa chọn một trong hai nhánh tùy thuộc vào giá trị của biểu thức logic là đúng (true) hay sai (false).

Cấu trúc

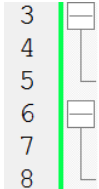
```
3 |  if (bt_logic) {  
4 |     khối_lệnh;  
5 | }
```

- Nếu bt\_logic cho giá trị đúng thì thực hiện khối lệnh (có thể là một hoặc nhiều câu lệnh) và thoát ra khỏi if.

Nếu bt\_logic cho giá trị sai thì thoát luôn ra khỏi if.

##### b)Câu lệnh rẽ nhánh đầy đủ

**if – else:** + Cú pháp:

```
3 |  if( bt_logic){  
4 |     Khối_lệnh 1;  
5 | }  
6 | else{  
7 |     Khối_lệnh 2;  
8 | }
```

+ Nếu bt\_logic có giá trị đúng thì thực hiện khối lệnh 1 và thoát khỏi if, ngược lại thực hiện khối lệnh 2 và thoát khỏi if.

**if ... else if:**

+ Cú pháp:



```

3  if(bt_logic1){
4      Khối lệnh 1;
5  }
6  else{
7      if(bt_logic2){
8          Khối lệnh 2;
9      }
10     else{
11         .....
12
13         if(bt_logic n-1){
14             Khối lệnh n-1;
15         }
16         else{
17             .....
18         }
19     }
20 }

```

+ Với cấu trúc rẽ nhánh đầu đủ, phải xác định định **else** hiện tại đang là của **if** nào, nếu trước **else** mà không có **if** thì đó là câu lệnh sai, chương trình không thể chạy được.

## Khi sử dụng biểu thức logic là các phép so sánh:

- So sánh bằng nhau sử dụng toán tử: ==

So sách lớn hơn: >

So sánh lớn hơn hoặc bằng:  $\geq$

So sánh nhỏ hơn:  $<$

So sánh nhỏ hơn hoặc bằng:  $\leq$

So sánh khác: !=

### c) Câu lệnh switch – case

+ Cú pháp:

```
3 switch (biểu thức){
4     case giá trị 1 : khối lệnh 1;
5     break;
6     case giá trị 2 : khối lệnh 2;
7     break;
8     ...
9     case giá trị n : khối lệnh n;
10    break;
11    default : khối lệnh;
12 }
```

+ Khi giá trị của **biểu thức** bằng **giá trị i** thì lệnh i sẽ được thực hiện. Nếu sau **lệnh i** không có lệnh **break** thì tiếp tục thực hiện các lệnh sau nó. Ngược lại thoát khỏi cấu trúc switch.

+ Nếu giá trị biểu thức không trùng với bất kỳ **giá trị i** nào thì lệnh tương ứng với từ khóa default sẽ được thực hiện.

**Chú ý : Khi sử dụng switch – case:**

- Không đặt dấu chấm phẩy sau câu lệnh **switch**.
- **Biểu thức** phải là có kết quả là **giá trị nguyên (char, int, long,...)**
- **Lệnh 1, 2...n** có thể gồm nhiều lệnh, nhưng không cần đặt trong cặp dấu { }

### Ví dụ

**VD1:** Viết chương trình kiểm tra một số là số chẵn hay số lẻ.

Ý tưởng: một số là số chẵn khi số đó chia hết cho 2, và ngược lại là số lẻ.

Code:

```
1 #include<stdio.h>
2 int main() {
3     int n;
4     scanf("%d",&n); // Nhập giá trị n
5     if(n%2==0) { // nếu n chia hết cho 2 ==> n là số chẵn
6         printf("Số %d là số chẵn",n);
7     }
8     else{ // ngược lại, n không chia hết cho 2 ==> n là số lẻ
9         printf("Số %d là số lẻ",n);
10    }
11    return 0;
12 }
```

**VD2:** Sử dụng switch – case để làm menu chọn một trong 3 bài là:1,2,3 Nếu không phải là một trong bài bài trên thì in ra bài được chọn không tồn tại trong menu.

Ý tưởng: áp dụng switch, nhập bài cần chọn n, switch sẽ so sánh giá trị n với các case, bằng với case nào thì chương trình sẽ chạy lệnh trong case đó, và thoát khỏi switch.

Code:

```
1  #include<stdio.h>
2  int main() {
3      int n;
4      scanf("%d",&n); // Nhập giá trị n- bài chọn
5      switch (n) {
6          case 1: {
7              printf("Bai duoc chon la bai 1");
8              break;
9          }
10         case 2: {
11             printf("Bai duoc chon la bai 2");
12             break;
13         }
14         case 3: {
15             printf("Bai duoc chon la bai 3");
16             break;
17         }
18         default :
19             printf("Bai duoc chon khong ton tai trong menu");
20             break;
21     }
22 }
```

### b) Bài tập

**Bài 1:** Nhập vào số nguyên n, kiểm tra số đó chẵn hay lẻ, âm hay dương và in kết quả kiểm tra ra màn hình.

**Bài 2:** Nhập vào hai số nguyên a, b. So sánh xem số nào lớn hơn, số nào nhỏ hơn hay hai số bằng nhau. In kết quả ra màn hình.

**Bài 3:** Hãy nhập 4 số thực a,b,c,d. Tìm giá trị lớn nhất của chúng và gán giá trị lớn nhất đó cho biến max. In giá trị max tìm được ra màn hình. Trong trường hợp 4 số bằng nhau thì in ra: không có số lớn nhất.

**Bài 4:** Giải và biện luận phương trình bậc nhất  $ax+b=0$

## CLB Lập Trình PTIT – ProPTIT

**Bài 5:** Giải và biện luận phương trình bậc 2  $ax^2 + bx + c = 0$

**Bài 6:** Nhập vào ba số  $a, b, c$  (là các số thực không âm). Kiểm tra xem đó có phải là ba cạnh của một tam giác hay không. Nếu có thì tam giác đó thuộc loại tam giác gì? (Thường, cân, vuông, vuông cân, hay đều).



## II. Cấu trúc lặp

### 1. Cấu trúc lặp do ... while.

Vòng lặp thực hiện khi điều kiện sai

a. Cú pháp:

```
3 | ☐ do{  
4 |     Khối lệnh;  
5 | }while (biểu thức);
```

b. VD:

```
1 | #include <stdio.h>  
2 | main() {  
3 |     int in;  
4 |     do{  
5 |         printf("Nhap vao password : ");  
6 |         scanf("%d", &in);  
7 |     }while (in != 12345);  
8 | }
```

### 2. Cấu trúc lặp while.

Vòng lặp thực hiện lặp lại trong khi biểu thức còn đúng.

a. Cú pháp:

```
3 | ☐ while (biểu thức) {  
4 |     Khối lệnh;  
5 | }
```

b.VD :

```
1 #include <stdio.h>
2 main() {
3     int i=0;
4     while (i++ < 3){
5         printf("%s", "Vi du su dung vong lap while.\n");
6     }
7 }
```

### 3.Cấu trúc lặp for

a. Cú pháp:

for(biểu thức 1;biểu thức 2;biểu thức 3) khối lệnh;

Trong đó:

Biểu thức 1: biểu thức khởi đầu.

Biểu thức 2: biểu thức điều kiện - điều kiện lặp.

Biểu thức 3: bước nhảy.

#### **Lưu ý:**

*Cả 3 biểu thức này đều là tùy chọn, chúng có thể vắng mặt trong Câu lệnh cụ thể nhưng các dấu chấm phẩy vẫn phải có.*

b.Ví dụ:

```
1 #include <stdio.h>
2 #include <conio.h>
3 main() {
4     for (int i=1; i<=3; i++) printf("PROPTIT"); // in 3 lan ra man hinh PROPTIT
5 }
```

## Bài 4: Mảng

### I. Khái niệm

#### 1. Mảng:

a. Khái niệm : Mảng là một tập hợp các phần tử có cùng một kiểu, gọi là kiểu phần tử. Kiểu phần tử có thể là có các kiểu bất kỳ: ký tự, số, chuỗi ký tự...; cũng có khi ta sử dụng kiểu mảng để làm kiểu phần tử cho một mảng (trong trường hợp này ta gọi là mảng của mảng hay mảng nhiều chiều).

#### b. Phân loại:

Mảng 1 chiều

Mảng nhiều chiều

#### c. Ứng dụng:

Mảng là kiểu dữ liệu được sử dụng rất thường xuyên. Chẳng hạn người ta cần quản lý một danh sách họ và tên của khoảng 100 sinh viên trong một lớp. Nhận thấy rằng mỗi họ và tên để lưu trữ ta cần 1 biến kiểu chuỗi, như vậy 100 họ và tên thì cần khai báo 100 biến kiểu chuỗi. Nếu khai báo như thế này thì đoạn khai báo cũng như các thao tác trên các họ tên sẽ rất dài dòng và rắc rối. Vì thế, kiểu dữ liệu mảng giúp ích ta trong trường hợp này; chỉ cần khai báo 1 biến, biến này có thể coi như là tương đương với 100 biến chuỗi ký tự; đó là 1 mảng mà các phần tử của nó là chuỗi ký tự. Hay như để lưu trữ các từ khóa của ngôn ngữ lập trình C, ta cũng dùng đến một mảng để lưu trữ chúng.

### II. Khai báo

#### Mảng 1 chiều

Nếu xét dưới góc độ toán học, mảng 1 chiều giống như một vector. Mỗi phần tử của mảng một chiều có giá trị không phải là một mảng khác.

Khai báo mảng với số phần tử xác định (khai báo tường minh)

**-Cú pháp:** < Kiểu> <Tên mảng> <[số phần tử]>;

Ý nghĩa:

+**Tên mảng:** đây là một cái tên đặt đúng theo quy tắc đặt tên của danh biểu. Tên này cũng mang ý nghĩa là tên biến mảng.

## CLB Lập Trình PTIT – ProPTIT

+ **Số phần tử**: là một hằng số nguyên, cho biết số lượng phần tử tối đa trong mảng là bao nhiêu (hay nói khác đi kích thước của mảng là gì).

+ **Kiểu**: mỗi phần tử của mảng có dữ liệu thuộc kiểu gì.

+ Ở đây, ta khai báo một biến mảng gồm có **số phần tử**, phần tử thứ nhất là **tên mảng** [0], phần tử cuối cùng là **tên mảng**[**số phần tử**-1]

Ví dụ:

```
int a[10]; /* Khai báo biến mảng tên a, phần tử thứ nhất là a[0], phần tử cuối cùng là a[9].*/
```

Khai báo mảng với số phần tử không xác định (khai báo không tường minh)

**Cú pháp: <Kiểu> <Tên mảng> [< >];**

Khi khai báo, không cho biết rõ số phần tử của mảng, kiểu khai báo này thường được áp dụng trong các trường hợp: vừa khai báo vừa gán giá trị, khai báo mảng là tham số hình thức của hàm.

Vừa khai báo vừa gán giá trị

Cú pháp:

**<Kiểu> <Tên mảng> [] = { Các giá trị cách nhau bởi dấu phẩy }**

Nếu vừa khai báo vừa gán giá trị thì mặc nhiên C sẽ hiểu số phần tử của mảng là số giá trị mà chúng ta gán cho mảng trong cặp dấu { }.

Chúng ta có thể sử dụng hàm **sizeof()** để lấy số phần tử của mảng như sau:

**Số phần tử = sizeof(tên mảng) / sizeof(kiểu);**

Ví dụ:

```
3 int dayso[]={66,65,69,68,67,70};
4 n=sizeof(dayso)/sizeof(int); /*Lấy số phần tử*/
```

### III.Khai báo mảng là tham số hình thức của hàm

Trong trường hợp này ta không cần chỉ định số phần tử của mảng là bao nhiêu.

**Ví dụ:**



```
int sapxep( int a[ ], int n){  
    ...  
}
```

### Mảng nhiều chiều

Mảng nhiều chiều là mảng có từ 2 chiều trở lên. Điều đó có nghĩa là mỗi phần tử của mảng là một mảng khác.

#### a. Khai báo mảng 2 chiều tường minh

- Cú pháp:

**<Kiểu> <Tên mảng> <[Số phần tử chiều 1]> <[Số phần tử chiều 2]>;**

- Ví dụ:

Người ta cần lưu trữ thông tin của một ma trận gồm các số thực. Lúc này ta có thể khai báo một mảng 2 chiều như sau:

```
3 float m[8][9]; /* Khai báo mảng 2 chiều có 8*9 phần tử là số thực*/
```

#### b. Khai báo mảng 2 chiều không tường minh

Để khai báo mảng 2 chiều không tường minh, ta vẫn phải chỉ ra số phần tử của chiều thứ hai (chiều cuối cùng).

- Cú pháp:

**<Kiểu> <Tên mảng> <[]> <[Số phần tử chiều 2]>;**

Cách khai báo này cũng được áp dụng trong trường hợp vừa khai báo, vừa gán trị hay đặt mảng 2 chiều là tham số hình thức của hàm.

## III. Truy xuất dữ liệu của mảng

### Mảng 1 chiều

- Mỗi phần tử của mảng được truy xuất thông qua **Tên biến mảng** theo sau là **chỉ số** nằm trong cặp **dấu ngoặc vuông [ ]**. Chẳng hạn a[0] là phần tử đầu tiên của mảng a được khai báo ở trên. Chỉ số của phần tử mảng là một biểu thức mà giá trị là kiểu số nguyên.

- Với cách truy xuất theo kiểu này, **Tên biến mảng[Chỉ số]** có thể coi như là một biến có kiểu dữ liệu là **kiểu** được chỉ ra trong khai báo biến mảng.

**Ví dụ 1:** Nhập số n và dãy các số nguyên a[0], a[1],..., a[n-1]. Sau đó in ra màn hình.

```
1  #include<conio.h>
2  #include<stdio.h>
3  int main() {
4      int n;
5      int a[100];
6      printf("Nhap so n: "); scanf("%d",&n);
7      for(int i = 0 ; i < n ; i++) {
8          printf("Nhap phan tu a[%d] = ", i);
9          scanf("%d", &a[i]);
10     }
11     printf("Mang vua nhap la: ");
12     for(int i = 0 ; i < n ; i++) {
13         printf("%d ",a[i]);
14     }
15     getch();
16     return 0;
17 }
```

**Ví dụ 2:** Nhập số n và dãy các số nguyên a[0], a[1],..., a[n-1]. Sau đó sắp xếp .

```

1  #include<stdio.h>
2  void sapxep(int a[],int n){
3      for(int i = 0 ; i < n ; i++){
4          for(int j = 0 ; j < n ; j++){
5              if( a[i] < a[j] ){
6                  int t=a[i];
7                  a[i]=a[j];
8                  a[j]=t;
9              }
10         }
11     }
12 }
13 void in(int a[],int n){
14     for(int i = 0 ; i < n ; i++){
15         printf("%d ",a[i]);
16     }
17 }
18 int main(){
19     int n;
20     int a[100];
21     printf("Nhap so n: "); scanf("%d",&n);
22     for(int i = 0 ; i < n ; i++){
23         printf("Nhap phan tu a[%d] = ", i);
24         scanf("%d", &a[i]);
25     }
26     sapxep(a,n);
27     printf("Mang sau khi sap xep la: ");
28     in(a,n);
29     return 0;
30 }

```

## Mảng nhiều chiều

-Ta có thể truy xuất một phần tử của mảng hai chiều bằng cách viết ra **tên mảng** theo sau là hai chỉ số đặt trong hai cặp dấu ngoặc vuông. Chẳng hạn ta viết `m[2][3]`.

-Với cách truy xuất theo cách này, **Tên mảng[Chỉ số 1][Chỉ số 2]** có thể coi là 1 biến có kiểu được chỉ ra trong khai báo biến mảng.

**Ví dụ 1:** Viết chương trình cho phép nhập 2 ma trận a, b có m dòng n cột, thực hiện phép toán cộng hai ma trận a,b và in ma trận kết quả lên màn hình.

```

1  #include<stdio.h>
2  void cong(int a[][10],int b[][10],int c[][10],int n,int m){
3      for(int i = 0 ; i < n ; i++){
4          for(int j = 0 ; j < m ; j++){
5              c[i][j] = a[i][j] + b[i][j];
6          }
7      }
8  }
9  void nhap(int a[][10],int n,int m){
10     for(int i = 0 ; i < n ; i++){
11         for(int j = 0 ; j < m ; j++){
12             printf("Nhap phan tu a[%d][%d] = ", i,j);
13             scanf("%d", &a[i][j]);
14         }
15     }
16 }
17 void in(int a[][10],int n,int m){
18     for(int i = 0 ; i < n ; i++){
19         for(int j = 0 ; j < m ; j++){
20             printf("%d ",a[i][j]);
21             printf("\n");
22         }
23     }
24 int main(){
25     int n,m;
26     int a[][10],b[][10],c[][10];
27     printf("Nhap so dong n: "); scanf("%d",&n);
28     printf("Nhap so cot m: "); scanf("%d",&m);
29     printf("Nhap ma tran A: \n");
30     nhap(a,n,m);
31     printf("Nhap ma tran B: \n");
32     nhap(b,n,m);
33     cong(a,b,c,n,m);
34     printf("Ma tran tong la: \n");
35     in(c,n,m);
36     return 0;
37 }

```

### Bài tập tự luyện:

**Bài 00.** Nhập số liệu cho dãy số nguyên  $a_0, a_1, \dots, a_{n-1}$  và một số  $x$  bất kỳ. Đếm số lần xuất hiện của số  $x$  trong dãy trên.

**Bài 01.** Nhập mảng  $(a, N)$  gồm các số nguyên dương. Nhập số  $X$ . Xác định vị trí của phần tử trên  $a$  có giá trị gần với  $X$  nhất

**Bài 02.** Nhập số liệu cho dãy số nguyên  $a_0, a_1, \dots, a_{n-1}$  và một giá trị thực  $x$ . Giả sử dãy  $a$  đã được sắp xếp theo thứ tự tăng dần. Hãy chèn giá trị  $x$  vào dãy  $a$  sao cho vẫn giữ được tính sắp xếp của mảng.

**Bài 03.** Nhập 2 mảng  $(a, N)$  và  $(b, M)$  và số nguyên  $p$  ( $0 \leq p < N$ ). Hãy chèn mảng  $b$  vào vị trí  $p$  của  $a$ . Ví dụ:  $(a, 4)$ : **5 3 6 7**;  $(b, 3)$ : **2 9 11**;  $p:1 \rightarrow a, 7$ : **5 2 9 11 3 6 7**

**Bài 04.** Nhập dãy  $n$  số ( $n \leq 1000$ ). Xác định đường chạy dài nhất, xuất lên màn hình vị trí phần tử đầu tiên và độ dài của đường chạy đó. Đường chạy là một dãy liên tiếp các phần tử không giảm của dãy ban đầu.

Ví dụ :        Nhập dãy 1 4 2 3 1 2 6 8 3 5 7

Đường chạy dài nhất là: 4 4

**Bài 05.** Nhập số liệu cho ma trận  $A$  kích thước  $m \times n$  có các phần tử là các số nguyên. Tìm các giá trị cực đại và cực tiểu của các phần tử và chỉ rõ vị trí của chúng trong ma trận

**Bài 06.** Viết chương trình tính tích 2 ma trận các số nguyên  $A$  cấp  $m \times n$  và  $B$  cấp  $n \times k$ .

**Bài 07.** Viết chương trình nhập vào vào ma trận  $A$  có  $n$  dòng,  $m$  cột, các phần tử là những số nguyên lớn hơn 0 và nhỏ hơn 100 được nhập vào từ bàn phím. In ra ma trận dưới dạng sắp xếp tăng dần trong đó phần tử ở góc trên bên trái sẽ nhỏ nhất, phần tử ở góc dưới bên phải sẽ lớn nhất.

**Bài 08.** Nhập số nguyên dương  $n$ . In ra ma trận xoáy ốc vuông cấp  $n$ .

Ví dụ với  $n = 4$

1   2   3   4

12 13 14 5

11 16 15 6

10 9 8 7

**Bài 09.** Nhập ma trận A là ma trận vuông cấp n. Thực hiện xoay ma trận một góc 90 độ theo chiều kim đồng hồ. Ví dụ:

1 2 3 4

5 6 7 8

9 10 11 12

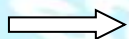
13 14 15 16

4 8 12 16

3 7 11 15

2 6 10 14

1 5 9 13



## Bài 5: GIỚI THIỆU VỀ C++

### I. Sự khác biệt giữa C++ và C

Hầu hết IT trên thế giới đều biết về 2 ngôn ngữ lập trình C, C++. Như chúng ta đã biết, C++ là ngôn ngữ ra đời sau ngôn ngữ C, thực chất nó mở rộng cho ngôn ngữ C nhằm tăng cường tính an toàn, cung cấp cho các lập trình viên nhiều lựa chọn hơn, đơn giản hóa lập trình ở mức cao hơn, và cung cấp một cách tiếp cận tốt hơn đối với những chương trình có quy mô lớn.

C++ cũng là ngôn ngữ lớn hơn với nhiều tính năng và phức tạp hơn so với C. Giữa C và C++ có rất nhiều khác biệt. Dưới đây là một số điểm khác biệt cơ bản giữa C và C++.

C	C++
Không phải ngôn ngữ hướng đối tượng.	Là một ngôn ngữ hướng đối tượng (gồm 4 khái niệm về hướng đối tượng)
Là một ngôn ngữ lập trình thủ tục.	Không phải là ngôn ngữ lập trình thủ tục.
Chỉ hỗ trợ các structure.	Hỗ trợ các lớp và đối tượng.
Không có biến tham chiếu, chỉ hỗ trợ con trỏ.	Hỗ trợ cả biến tham chiếu và con trỏ.
Sử dụng các hàm <i>scanf</i> và <i>printf</i> để nhập xuất.	Sử dụng các hàm <i>cin&gt;&gt;</i> và <i>cout&lt;&lt;</i> để nhập xuất.
Không thể khai báo hàm trong các structure.	Có thể khai báo hàm trong các structure.
Được xem là một ngôn ngữ lập trình cấp thấp.	Được xem là sự kết hợp giữa ngôn ngữ lập trình cấp thấp và cấp cao.

Không hỗ trợ các hàm inline, thay vào đó có thể sử dụng khai báo #define	Hỗ trợ các hàm inline.
Sử dụng phương pháp tiếp cận từ trên xuống (top-down).	Sử dụng phương pháp tiếp cận từ dưới lên (bottom-up).
Là ngôn ngữ lập trình hướng chức năng (function driven).	Là ngôn ngữ lập trình hướng đối tượng (Object driven).

Ngoài ra còn 1 số khác biệt khác như: sử dụng endl để xuống dòng thay vì \n như trong C. Toán tử '&' được sử dụng trong việc đa năng hóa (overload) như lấy địa chỉ của một đối tượng, thực hiện phép toán AND hoặc tạo 1 tham chiếu...

Còn về các cấu trúc lệnh, hàm, mảng... trong C++ được sử dụng tương tự như C.

## II. Nhập Xuất Trong C++

### 1. Nhập Dữ Liệu

-Để nhập dữ liệu từ bàn phím, trong C++ ta dùng đối tượng `cin>>`

Khác với `scanf` trong C, `cin>>` không cần phải sử dụng các toán tử như `%d`, `%c`, `%f...`

-Một số câu hỏi liên quan.

+ Giữa `scanf` và `cin>>` cái nào hoạt động nhanh hơn?

Trả lời: `scanf`, nhưng khác biệt là rất nhỏ.

+ Giữa `scanf` và `cin>>` cái nào tốt hơn?

Trả lời: Điều đó phụ thuộc vào sở thích cá nhân và những gì cần được thực hiện.

+ Nên sử dụng `scanf` hay `cin>>` ?

Trả lời: Bạn nên sử dụng theo cách mà bạn cảm thấy thoải mái nhất.

### 2. Xuất dữ liệu

-Để hiển thị thông tin hay xuất dữ liệu trong C++ người ta dùng đối tượng `cout<<`.

Khác với `printf`, để xuất dữ liệu ta không cần phải dùng các toán tử `%d %f %c`.



Ví dụ:

```
1  #include <iostream>
2  using namespace std;
3  main(){
4      int n;
5      float f;
6      cin>>n>>f;
7      cout<<n<<f;
8  }
```

-Chúng ta nên sử dụng iostream của C++ thay vì các hàm của C, bởi vì iostream an toàn hơn cho người dùng tự định nghĩa kiểu.



## Bài 6: Xử lý vào/ ra trên file văn bản ở C/C++

Trong C/C++, khi thao tác với 1 tệp dữ liệu, cần thực hiện theo thứ tự các bước sau :

1. Mở tệp tin <File>.
2. Thực hiện các thao tác đọc, ghi trên tệp tin đang mở.
3. Đóng tệp tin.

Trong C++ thì ta có 3 loại File stream cơ bản sau :

**ifstream** : Dùng cho file nhập vào. Loại này chỉ có thể được dùng để đọc dữ liệu vào file vào bộ nhớ mà thôi.

**ofstream** : Dùng cho file xuất ra. Loại này thì có thể dùng để tạo ra files và chép dữ liệu vào chúng.

**fstream** : Đây là kênh file.(File stream). Loại này thì có thể vừa tạo file, vừa ghi dữ liệu vào file và đọc dữ liệu từ file vào luôn.

C++ cung cấp cho bạn thư viện `fstream` `#include <fstream>` chứa các lớp, hàm phục vụ các thao tác này. Do vậy trước khi viết chương trình ta cần khai báo thư viện `fstream` ngay từ đầu chương trình.

### I.Cách sử dụng với `fstream`.

#### 1)Đối với file văn bản (\*.txt):

Để định nghĩa 1 đối tượng file ta dùng cú pháp sau : **`fstream dataFile`**

Ở đây **`dataFile`** chỉ là tên biến do người dùng đặt ra.

Để mở 1 file ta dùng cú pháp sau :

**`fstream <Tên biến tệp> (<Tên tệp>, <chế độ mở tệp>);`**

Hoặc:

**`fstream <Tên biến tệp>;`**

**`<Tên biến tệp>.open(<Tên tệp>, <chế độ mở tệp>);`**

```
1 #include<iostream>
2 #include<fstream>
3 using namespace std;
4 main() {
5     fstream dataFile;
6     dataFile.open("file.txt", ios::out);
7     // hoặc fstream fread ("E: \\data.txt", ios :: in);
8
9 }
10
```

Trong đó :

- **"file.txt"** là 1 chuỗi tên chứa tên file hoặc đường dẫn dẫn đến file
- **ios::out** là 1 flag của file và cái này cho ta biết chế độ nào mà chúng ta dùng để mở file.

Ở ví dụ trên thì tên file là **file.txt** còn flag file ở đây là **ios::out**. Điều này, cho C++ biết chúng ta mở file ở chế độ xuất ra. Chế độ xuất ra cho phép dữ liệu có thể được ghi vào file.

Trường hợp, mở file ở chế độ nhập vào, tức là cho phép dữ liệu được đọc vào từ file.

```
datafile.open("file.txt", ios::in);
```

### Các chế độ mở tệp tin <FILE>

ios :: in	Mở một tệp tin để đọc
ios :: out	Mở một tệp tin có sẵn để ghi
ios :: app	Mở một tệp tin có sẵn để thêm dữ liệu vào cuối tệp.
ios :: ate	Mở tệp tin và đặt con trỏ tệp tin vào cuối tệp.
ios :: trunc	Nếu tệp tin đã có sẵn thì dữ liệu của nó sẽ bị mất.
ios :: nocreate	Mở một tệp tin, tệp tin này bắt buộc phải tồn tại
ios :: noreplace	Chỉ mở tệp tin khi tệp tin chưa tồn tại.
ios :: binary	Mở một tệp tin ở chế độ nhị phân
ios :: text	Mở một tệp tin ở chế độ văn bản.

```
//Lưu ý: Bạn có thể mở tệp tin đồng thời 1 lúc ở nhiều chế độ.  
fstream myFile("data.txt", ios :: in | ios :: out | ios :: binary);  
// Mở tệp ở chế độ nhị phân.  
fstream myFile("data.txt", ios :: in | ios :: out | ios :: text);  
// Mở tệp ở chế độ văn bản.  
//Mặc định C++ đã có ios :: in | ios :: out rồi bạn có thể viết hay ko viết đều được.
```

Do đó, bạn có thể viết lại 2 ví dụ trên như sau :

```
fstream myFile("data.txt", ios :: binary); // Mở tệp ở chế độ nhị phân
```

```
fstream myFile("data.txt", ios :: text); // Mở tệp ở chế độ văn bản
```

**Lưu ý :** Một số trường hợp bạn ko thể mở 2 chế độ cùng nhau được.

**Ví dụ :** ios :: binary và ios :: text

## 2) Các phép xử lý vào/ ra trên file

### Ghi tệp văn bản bằng: "<<"

Các bước ghi dữ liệu vào tệp tin như sau:

#### - Mở file chỉ để đọc:

```
ofstream <Tên biến file> (<Tên file>, ios :: out);
```

Hoặc:

```
ofstream <Tên biến file> (<Tên file>); // Mặc định C++ sẽ thêm ios :: out
```

#### - Ghi dữ liệu vào file thao tác: "<<"

```
<Tên biến file> << <Dữ liệu cần ghi>;
```

#### - Đóng file

```
<Tên biến file>.close();
```

### Đọc tệp văn bản bằng: ">>"

Các bước ghi dữ liệu vào file như:

#### - Mở file chỉ để đọc:

## CLB Lập Trình PTIT – ProPTIT

ofstream <Tên biến file> (<Tên file>, ios :: in);

Hoặc:

ofstream <Tên biến file> (<Tên file>); // Mặc định C++ sẽ thêm

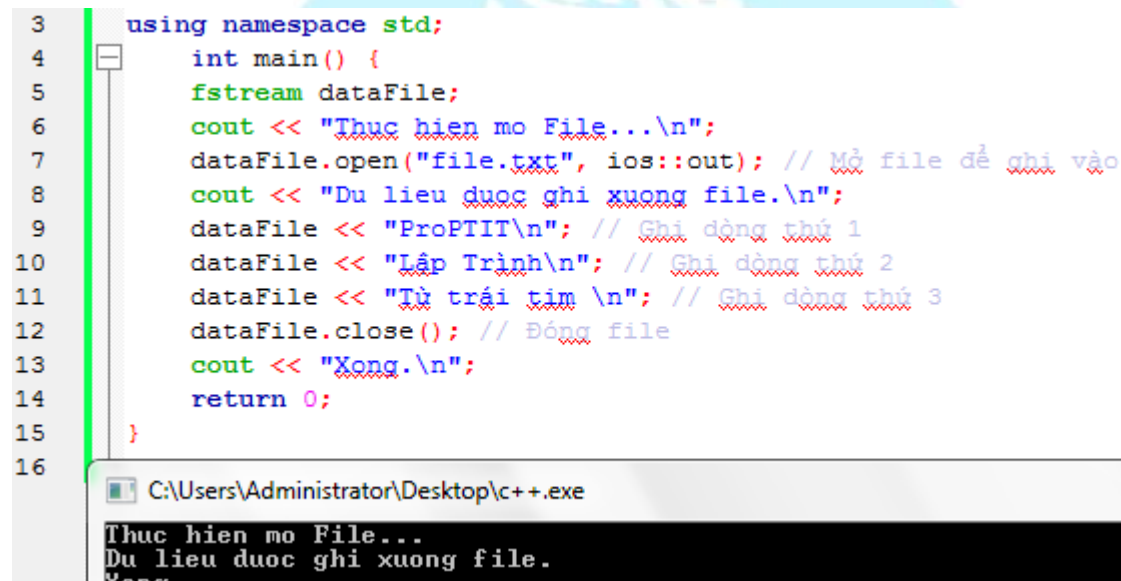
ios:: in

- **Đọc dữ liệu vào file thao tác: "<<"**

<Tên biến file> << <Dữ liệu cần đọc>;

- **Đóng tệp tin**

<Tên biến file>.close();



```
3 using namespace std;
4 int main() {
5     fstream dataFile;
6     cout << "Thực hiện mở File...\n";
7     dataFile.open("file.txt", ios::out); // Mở file để ghi vào
8     cout << "Dữ liệu được ghi xuống file.\n";
9     dataFile << "ProPTIT\n"; // Ghi dòng thứ 1
10    dataFile << "Lập Trình\n"; // Ghi dòng thứ 2
11    dataFile << "Từ trái tim \n"; // Ghi dòng thứ 3
12    dataFile.close(); // Đóng file
13    cout << "Xong.\n";
14    return 0;
15 }
16
```

C:\Users\Administrator\Desktop\c++.exe

Thực hiện mở File...  
Dữ liệu được ghi xuống file.  
Xong

ProPTIT

Lập Trình

Từ trái tim

***Quá trình đó thực sự như sau :***

ProPTIT\nLập Trình\nTừ trái tim\n<EOF>.

*Khi file lại được mở ra thì dữ liệu mới lại được gắn vào chỗ end-of-file đó.*

- **Xóa bộ nhớ đệm:**

cin.ignore();

### 3)Kiểm tra sự tồn tại của File trước khi mở hoặc mở file không được

Đôi khi chúng ta sẽ cần quyết định xem file có tồn tại trước khi chúng ta mở nó ra hay không và sau đây là 1 ví dụ:

```
4 int main() {  
5     fstream dataFile;  
6     dataFile.open("input.txt", ios::in);  
7     if(dataFile.fail()) {  
8         //Nếu file không tồn tại, thì tạo ra 1 file mới  
9  
10        dataFile.open("input.txt", ios::out);  
11        //...  
12    }else {  
13        dataFile.close();  
14    }  
15  
16 }
```

## II.Cách sử dụng với kiểu FILE .

### 1)Khái niệm

Trong các chương trình trước thì các dữ liệu đưa vào chương trình chỉ được tồn tại trong RAM, khi thoát chương trình thì tất cả dữ liệu đều bị mất. Để khắc phục tình trạng này Borland C cung cấp cho ta các hàm để lưu trữ và truy xuất tập tin, đó là kiểu **FILE** . Và ở đây ta chỉ đề cập đến 2 loại tập tin :

- Tập tin văn bản: là tập tin dùng để ghi các ký tự lên đĩa theo các dòng.
- Tập tin nhị phân: là tập tin dùng để ghi các cấu trúc dạng nhị phân (được mã hoá).

### 2)Thao tác với tập tin

Quá trình thao tác trên tập tin thông qua 4 bước:

Bước 1: Khai báo con trỏ trỏ đến tập tin.

Bước 2: Mở tập tin.

Bước 3: Các xử lý trên tập tin.

Bước 4: Đóng tập tin.

### 3) Khai báo

```
#include<stdio.h>
main() {
    // khai báo FILE *< tên biến >;
    FILE *f; // Khai báo biến con trỏ file f
}
```

### 4) Mở tập tin

fopen (< đường dẫn tên tập tin> , < kiểu truy nhập >);

```
1  #include<stdio.h>
2  main() {
3      // khai báo FILE *< tên biến >;
4      FILE *f; // Khai báo biến con trỏ file f
5      f = fopen ( "C:\\\\VD1.txt", "r" );
6
7  }
```

Các kiểu truy nhập tập tin thông dụng:

- t là kiểu truy nhập tập tin đối với dạng tập tin văn bản (text).
- b là kiểu truy nhập tập tin đối với dạng tập tin nhị phân (binary).
- r mở ra để đọc ( ready only).
- w mở ra để ghi (create / write).
- a mở ra để thêm vào (append).
- r+ mở ra để đọc và ghi (modify).

### 5) Các hàm đọc ghi nội dung tập tin

-Đọc tập tin:

+ fscanf(<FILE \*>, <định dạng>, <các tham biến>); Dữ liệu từ một tập tin theo định dạng

## CLB Lập Trình PTIT – ProPTIT

```
#include<stdio.h>
main(){
    // khai bao FILE *< tên biến >;
    FILE *f; // Khai bao bien con tro file f
    f = fopen ( "VD1.txt", "r" ); // chi can ten file neu file luu cung thu muc file code
    int n;
    fscanf(f,"%d",&n);
}
```

+ fgetc(<vùng nhớ>, <kích thước tối đa>, <FILE \*>); Đọc một chuỗi ký tự từ một tập tin với kích thước tối đa cho phép, hoặc gặp ký tự xuống dòng.

```
main(){
    // khai bao FILE *< tên biến >;
    FILE *f; // Khai bao bien con tro file f
    f = fopen ( "VD1.txt", "r" ); // chi can ten fil
    char s[100];
    fgetc(s,80,f);
    puts(s);
}
```

C:\Users\Administrator\Desktop\c.exe

LapTrinhPTIT

Process returned 0 (0x0) execution time : 0.013 s  
Press any key to continue.

+ getc(< FILE \* >); Đọc một ký tự từ tập tin đang mở.

```
main(){
    // khai bao FILE *< tên biến >;
    FILE *f; // Khai bao bien
    f = fopen ( "VD1.txt", "r" )
    char c;
    printf("%c", fgetc(f));
}
```

C:\Users\Administrator\Desktop\c.exe

A

Process returned 0 (0x0) execution time : 0.013 s  
Press any key to continue.



## CLB Lập Trình PTIT – ProPTIT

+Xóa bộ nhớ đệm:

`fflush(stdin);`

-Ghi tập tin:

+ `fprintf(<FILE *>, <định dạng>[, <các tham biến>]);` Ghi dữ liệu theo một định dạng nào đó vào tập tin. Ví dụ : `fprintf(f,"%d",x);`

```
2 main() {  
3     // khai báo FILE *< tên biến >;  
4     FILE *f; // Khai báo biến con trỏ file f  
5     f = fopen ( "VD1.txt", "w" ); // write  
6     int n=5;  
7     fprintf(f,"%d",n);  
8 }
```

+ `fputs(<chuỗi ký tự>, <FILE *>);` Ghi một chuỗi ký tự vào tập tin đang mở.

```
main() {  
    // khai báo FILE *< tên biến >;  
    FILE *f; // Khai báo biến con trỏ file f  
    f = fopen ( "VD1.txt", "w" ); // write  
    char s[50]="LapTrinhPTIT";  
    fputs(s,f);  
}
```

### 6)Đóng tập tin

Sau khi không còn làm việc với tập tin, để đảm bảo an toàn cho dữ liệu thì nhất thiết ta phải đóng tập tin lại.

`fclose ( < biến con trỏ tập tin > ) ;`

hoặc `fcloseall () ;`

Ví dụ : `fclose (f) ;`

### 7)Thao tác trên tập tin

- Xóa tập tin :

`remove ( < đường dẫn tập tin > );`

- Đổi tên tập tin :

rename ( < tên tập tin cũ >, < tên tập tin mới > );

### III.Chuyển hướng standard input và standard output ra file

Trong C++, để nhập xuất dữ liệu ta sử dụng các hàm cin, cout, scanf, printf... nhập liệu từ bàn phím (standard input) và xuất dữ liệu ra màn hình (standard output). Tuy nhiên nếu dữ liệu nhập vào khá lớn , và ta cần chạy chương trình nhiều lần để sửa lỗi, thì việc nhập dữ liệu từ bàn phím như vậy mất rất nhiều thời gian.

Một cách khắc phục tình trạng này là chuyển hướng standard input, thay vì nhập từ bàn phím thì chuyển thành nhập từ file. Chỉ cần nhập liệu vào file input một lần và sau đó có thể chạy chương trình nhiều lần mà không mất công nhập liệu lại. Sử dụng hàm **freopen**.

```
1  #include<iostream>
2  #include<fstream>
3  using namespace std;
4  main() {
5      freopen("test.in", "r", stdin);
6      freopen("test.out", "w", stdout);
7      int n;
8      cin>>n; // tự động vào ra file
9      cout<<n;
10 }
```

## Bài 7: Xâu (Chuỗi)

Chuỗi được xem như là một mảng 1 chiều gồm các phần tử có kiểu char như ký tự, con số và bất cứ ký tự đặc biệt như

+, -, \*, /, \$, #,...

Theo quy ước, một chuỗi sẽ được kết thúc bởi ký tự null ('\0' : kí tự rỗng).

Ví dụ chuỗi abcd sẽ có dạng 

a	b	c	d	\0
---	---	---	---	----

 như sau:

### I.Khai báo chuỗi

VD:

```
Untitled1.cpp
1  #include <conio.h>
2  #include <stdio.h>
3  #include <string.h>
4  main()
5  {
6      char vidu[50];
7      gets(vidu);
8      printf("%s",vidu);
9  }
```

Ví dụ ta nhập Hello, xâu vidu sẽ đc gán bằng chuỗi “Hello“

Kết quả in ra màn hình sẽ là “Hello”.

Chuỗi vidu[50] sẽ có tối đa là 49 kí tự do 1 phần tử cuối của chuỗi sẽ luôn là kí tự null (\0).

Khai báo con trỏ:

Khi ta cần nhập 1 chuỗi mà k biết độ dài của chuỗi, ta có thể dùng con trỏ.

char \*p; |

### Khởi tạo chuỗi

Ta có thể khởi tạo giá trị của xâu ngay từ khi khai báo:

```
2  main() {
3      char vidu[50] = "Hello World.";
4  }
```

Tuy nhiên chúng ta không được gán giá trị khi đã khai báo.

```
char vidu[50];
```

```
vidu = "Hello World."; // sai
```

### Nhập xuất chuỗi

#### II. Nhập chuỗi từ bàn phím

Để nhập một chuỗi ký tự từ bàn phím, ta sử dụng hàm gets()

**Cú pháp: gets(<Biến chuỗi>)**

Ví dụ:

```
1
2 main() {
3     char Ten[20];
4     gets(Ten);
5 }
```

Ta cũng có thể sử dụng hàm scanf() để nhập dữ liệu cho biến chuỗi, tuy nhiên lúc này ta chỉ có thể nhập được một chuỗi không có dấu khoảng trắng.

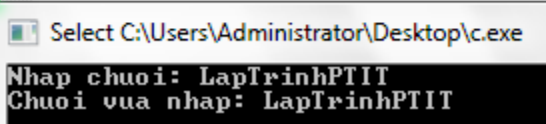
#### III. Xuất chuỗi lên màn hình

Để xuất một chuỗi (biểu thức chuỗi) lên màn hình, ta sử dụng hàm puts().

**Cú pháp: puts(<Biểu thức chuỗi>)**

Ví dụ: Nhập vào một chuỗi và hiển thị trên màn hình chuỗi vừa nhập.

```
4 main()
5 {
6     char Ten[12];
7     printf("Nhap chuoi: ");
8     gets(Ten);
9     printf("Chuoi vua nhap: ");
10    puts(Ten);
11 }
12
```



#### IV. Một số hàm xử lý chuỗi (trong string.h)

##### Cộng chuỗi - Hàm strcat()


**Cú pháp: char \*strcat(char \*des, const char \*source)**

## CLB Lập Trình PTIT – ProPTIT

Hàm này có tác dụng ghép chuỗi nguồn vào chuỗi đích.

*Ví dụ:* Nhập vào họ lót và tên của một người, sau đó in cả họ và tên của họ lên màn hình.

```
4 int main()
5 {
6     char HoLot[30], Ten[12];
7     printf("Nhap Ho Lot: "); gets(HoLot);
8     printf("Nhap Ten: "); gets(Ten);
9     strcat(HoLot, Ten); /* Ghen Ten vao HoLot*/
10    printf("Ho ten la: "); puts(HoLot);
11    return 0;
12 }
13
```

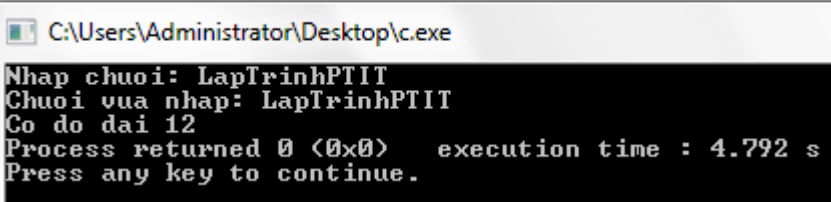


### Xác định độ dài chuỗi - Hàm strlen()

**Cú pháp:** int strlen(const char\* s)

*Ví dụ:* Sử dụng hàm strlen xác định độ dài một chuỗi nhập từ bàn phím.


```
char Chuoi[255];
int Dodai;
printf("Nhap chuoi: "); gets(Chuoi);
Dodai = strlen(Chuoi);
printf("Chuoi vua nhap: "); puts(Chuoi);
printf("Co do dai %d", Dodai);
return 0;
}
```



### Đổi một ký tự thường thành ký tự hoa - ngược lại

**Cú pháp:** char toupper(char c) // tolower()

```
5 int main() {
6     char c = 'h';
7     c=toupper(c); //doi thanh ky tu in hoa
8     printf("%c\n",c);
9     c=tolower(c); //doi thanh ky tu in hoa
10    printf("%c\n",c);
11 }
12
```



### Đổi chuỗi chữ thường thành chuỗi chữ hoa, hàmstrupr()

Hàmstrupr() được dùng để chuyển đổi chuỗi chữ thường thành chuỗi chữ hoa, kết quả trả về của hàm là một con trỏ chỉ đến địa chỉ chuỗi được chuyển đổi.

**Cú pháp:** char \*strupr(char \*s)

*Ví dụ:* Viết chương trình nhập vào một chuỗi ký tự từ bàn phím. Sau đó sử dụng hàmstrupr() để chuyển đổi chúng thành chuỗi chữ hoa.

```
4 main() {
5     char Chuoi[255],*s;
6     printf("Nhap chuoi: "); gets (Chuoi);
7     s=strupr (Chuoi) ;
8     printf("Chuoi chu hoa: "); puts (s);
9 }
```

### Đổi chuỗi chữ hoa thành chuỗi chữ thường, hàmstrlwr()

Muốn chuyển đổi chuỗi chữ hoa thành chuỗi toàn chữ thường, ta sử dụng hàmstrlwr(), các tham số của hàm tương tự như hàmstrupr()

**Cú pháp:** char \*strlwr(char \*s)

### Sao chép chuỗi, hàmstrcpy()

Hàm này được dùng để sao chép toàn bộ nội dung của chuỗi nguồn vào chuỗi đích.

**Cú pháp:** char \*strcpy(char \*Des, const char \*Source)

*Ví dụ:* Viết chương trình cho phép chép toàn bộ chuỗi nguồn vào chuỗi đích.

```
5  {  
6      char Chuoi[255], s[255];  
7      printf("Nhap chuoi: "); gets (Chuoi);  
8      strcpy(s, Chuoi);  
9      printf("Chuoi dich: "); puts(s);  
10 }
```

### Sao chép một phần chuỗi, hàm strncpy()

Hàm này cho phép chép n ký tự đầu tiên của chuỗi nguồn sang chuỗi đích.

**Cú pháp:** `char *strncpy(char *Des, const char *Source, size_t n)`

### Trích một phần chuỗi, hàm strchr()

Để trích một chuỗi con của một chuỗi ký tự bắt đầu từ một ký tự được chỉ định trong chuỗi cho đến hết chuỗi, ta sử dụng hàm strchr().

**Cú pháp :** `char *strchr(const char *str, int c)`

**Ghi chú:**

- Nếu ký tự đã chỉ định không có trong chuỗi, kết quả trả về là NULL.
- Kết quả trả về của hàm là một con trỏ, con trỏ này chỉ đến ký tự c được tìm thấy đầu tiên trong chuỗi str.

### Tìm kiếm nội dung chuỗi, hàm strstr()

Hàm strstr() được sử dụng để tìm kiếm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1.

**Cú pháp:** `char *strstr(const char *s1, const char *s2)`

Kết quả trả về của hàm là một con trỏ chỉ đến phần tử đầu tiên của chuỗi s1 có chứa chuỗi s2 hoặc giá trị NULL nếu chuỗi s2 không có trong chuỗi s1.

*Ví dụ:* Viết chương trình sử dụng hàm strstr() để lấy ra một phần của chuỗi gốc bắt đầu từ chuỗi “vien”.

```
5  {  
6      char Chuoi[255], *s;  
7      printf("Nhap chuoi: "); gets (Chuoi);  
8      s=strstr (Chuoi, "vien");  
9      printf("Chuoi trích ra: "); puts(s);  
10 }
```

```
F:\My Documents\Desktop\Untitled1.exe
Nhap chuoi: hoc vien bcvt
Chuoi trích ra: vien bcvt

-----
Process exited after 3.896 seconds with return value 0
Press any key to continue . . .
```

### So sánh chuỗi, hàm strcmp()

Để so sánh hai chuỗi theo từng ký tự trong bảng mã Ascii, ta có thể sử dụng hàm strcmp().

**Cú pháp: int strcmp(const char \*s1, const char \*s2)**

Hai chuỗi s1 và s2 được so sánh với nhau, kết quả trả về là một số nguyên (số này có được bằng cách lấy ký tự của s1 trừ ký tự của s2 tại vị trí đầu tiên xảy ra sự khác nhau).

- Nếu kết quả là số âm, chuỗi s1 nhỏ hơn chuỗi s2.
- Nếu kết quả là 0, hai chuỗi bằng nhau.
- Nếu kết quả là số dương, chuỗi s1 lớn hơn chuỗi s2.

### So sánh chuỗi, hàm stricmp()

Hàm này thực hiện việc so sánh trong n ký tự đầu tiên của 2 chuỗi s1 và s2, giữa chữ thường và chữ hoa không phân biệt.

**Cú pháp: int stricmp(const char \*s1, const char \*s2)**

Kết quả trả về tương tự như kết quả trả về của hàm strcmp()

### Đổi từ chuỗi ra số, hàm atoi(), atof(), atol() (trong stdlib.h)

Để chuyển đổi chuỗi ra số, ta sử dụng các hàm trên.

**Cú pháp : int atoi(const char \*s) : chuyển chuỗi thành số nguyên**

long atol(const char \*s) : chuyển chuỗi thành số nguyên dài

float atof(const char \*s) : chuyển chuỗi thành số thực



Nếu chuyển đổi không thành công, kết quả trả về của các hàm là 0.

### V. Bảng tóm tắt các hàm

**Các hàm kiểm tra ký tự.** (các hàm này trong thư viện **ctype.h**) Nếu đúng thì hàm cho giá trị khác 0. Nếu sai thì hàm cho giá trị bằng 0.

- `Int isalpha(int c)` : kiểm tra ký tự có là chữ cái không.
- `Int isdigit(int c)` : kiểm tra xem ký tự có là chữ số không.
- `Int islower(int c)`: kiểm tra ký tự có là chữ thường không.
- `Int isupper(int c)`: kiểm tra ký tự có là chữ hoa không.
- `Int ispace(int c)`: kiểm tra ký tự có là trống không (\n, dấu cách, \t).

**Các hàm xử lý chuỗi ký tự.** (các hàm này nằm trong thư viện **string.h**)

- `Int strlen(char *s)` trả về độ dài của chuỗi s;
- `Char *strupr(char *s)` đổi chữ thường trong chuỗi s sang chữ hoa.
- `Char *strlwr(char *s)` đổi chữ hoa sang chữ thường.
- `Char *strcat(char *s1, char *s2)` nối chuỗi s2 vào chuỗi s1;
- `Int strcmp(char *s1, char *s2)` cho giá trị âm nếu chuỗi s1 nhỏ hơn chuỗi s2. Và cho giá trị dương nếu chuỗi s1 lớn hơn chuỗi s2. Trả về giá trị bằng 0 nếu chuỗi s1 bằng chuỗi s2.
- `Int strcmpi(char *s1, char *s2)` so sánh 2 chuỗi nhưng không phân biệt chữ thường và chữ hoa.
- `Char *strcpy(char *s1, char *s2)` copy chuỗi s2 vào chuỗi s1.
- `Char *strncpy(char *s1, char *s2, int n)` sao chép n ký tự đầu của chuỗi s2 sang chuỗi s1
- `Char *strnset(char *s, int c, int n)` dùng để sao chép n lần ký tự c vào chuỗi s.
- `Char *strstr(char *s1, char *s2)` tìm sự xuất hiện của chuỗi s2 trong chuỗi s1. Nếu tìm thấy hàm cho địa chỉ của chuỗi con trong chuỗi s1. Trái lại cho NULL.
- `Char *strrev(char *s)` dùng đảo ngược chuỗi s. Nếu thành công hàm cho địa chỉ chuỗi đã đảo.
- **Cách xóa ký tự trong chuỗi.** Trong C không có hàm xóa nhưng ta có thể xóa bằng cách copy địa chỉ của ký tự sau đè lên địa chỉ của ký tự trước. VD ta có chuỗi ht = “Nguyen Van A”, muốn xóa ký tự “uy” ta làm như sau: **`strcpy(&ht[2], &ht[4]);`**

## Bài 8: Đệ quy

### I. Khái niệm:

Một đối tượng được gọi là *đệ qui* nếu nó được mô tả thông qua định nghĩa của chính nó. Nghĩa là, các đối tượng này được định nghĩa một cách *quy nạp* từ những khái niệm đơn giản nhất cùng dạng với nó. Trong toán học và tin học có rất nhiều đối tượng như thế.

Những bài toán *đệ qui* có thể được phân rã thành các bài toán nhỏ hơn, đơn giản hơn nhưng có cùng dạng với bài toán ban đầu. Những bài toán nhỏ lại được phân rã thành các bài toán nhỏ hơn. Cứ như vậy, việc phân rã chỉ dừng lại khi bài toán con đơn giản đến mức có thể suy ra ngay kết quả mà không cần phải phân rã nữa. Ta phải giải tất cả các bài toán con rồi kết hợp các kết quả đó lại để có được lời giải cho bài toán lớn ban đầu. Cách phân rã bài toán như vậy gọi là “chia để trị” (divide and conquer), là một dạng của đệ quy.

Ví dụ: Tính  $10!$

$10! = 10.9!$  và nhiệm vụ của ta là tính  $9!$

$9! = 9.8!$  và nhiệm vụ của ta là tính  $8!$

.....

$2! = 2.1!$  và nhiệm vụ của ta là tính  $1!$

$1!$  lại bằng 1, tính dc  $1!$  ta tính dc  $2! \Rightarrow 3! \Rightarrow 4! \Rightarrow \dots \Rightarrow 10!$

Phương pháp trên được gọi là phương pháp đệ qui!

### II. Đệ qui trong lập trình:

Để dễ hiểu, ta thử code để tính  $n!$

```

3 int gt (int n){
4     if (n <= 1) return 1;
5     else return ( n * gt(n - 1) );
6 }
7
```

Giả sử ta có 1 hàm là  $gt(n)$  để tính giai thừa của  $n$ .

Nếu  $n=0$  hoặc 1 (các TH đặc biệt) thì nó sẽ trả về giá trị cụ thể là 1.

Nếu  $n > 1$  thì nó sẽ trả về giá trị là  $n * gt(n-1)$ , để tính dc  $gt(n-1)$  nó lại gọi lại hàm giai thừa, và tính giai thừa của  $n-1$ , cứ thế đến khi có giá trị của  $1!$ , nó sẽ tính dc  $2!$  và sẽ tính dc  $n!$

## CLB Lập Trình PTIT – ProPTIT

Chu trình thực hiện với  $n = 3$ :

Có lệnh gọi hàm :  $x = \text{gt}(3)$ ;

Máy sẽ ghi nhớ:  $\text{gt}(3) = 3 * \text{gt}(2)$ ;

Máy đi tính  $\text{gt}(2)$ ;

Kế tiếp máy sẽ ghi nhớ:  $\text{gt}(2) = 2 * \text{gt}(1)$ ;

Theo như hàm thì  $\text{gt}(1) = 1$ ;

Máy quay ngược lại tính đc  $\text{gt}(2) = 2 * 1 = 2$ ;

$\text{gt}(3) = 3 * 2 = 6$  -----> kết quả cần tìm.

### 1) Xây dựng hàm đệ quy:

#### a) Xây dựng

+ Một công thức tính giá trị của hàm phụ thuộc vào chính hàm đó (quy nạp)

VD:  $5! = 5 * 4!$  (  $n! = n * (n-1)!$  ).

+ Điều kiện dừng lại không đệ quy tiếp

VD:  $n = 1$  thì trả về giá trị 1.

#### b) Ưu điểm và nhược điểm:

+ Ưu điểm: dễ xây dựng, code đơn giản dễ hiểu

+ Nhược điểm: tốn bộ nhớ, độ sâu đệ quy (Số lần gọi đệ quy) hữu hạn

#### c) Các ví dụ:

VD1: Nhập  $n$  và in ra màn hình  $n!$

```
3 | long long gt (int n){
4 |     if (n <= 1) return 1;
5 |     else return ( n * gt(n - 1) );
6 | }
7 |
```

VD2: Đếm số lượng chữ số của số nguyên dương n

```
3 void DaoNguoc(int n) {  
4     if(n != 0) {  
5         printf("%d", n%10);  
6         DaoNguoc(n/10);  
7     }  
8 }
```

VD3: Đổi số nguyên dương n sang hệ cơ số nhị phân

```
3 void NhiPhan(int n) {  
4     if(n!=0) {  
5         NhiPhan (n/2);  
6         printf("%d", n%2);  
7     }  
8 }
```

VD4: Tìm UCLN của a và b

```
3 int UCLN(int a, int b) {  
4     if(a==b) return a;  
5     else {  
6         if(a>b) a=a-b;  
7         else b=b-a;  
8         return UCLN(a, b);  
9     }  
10 }
```

VD5: Tính  $P(n)=1.3.5...(2n+1)$  với  $n \geq 0$

```

3  long Tich(int n){
4      if(n==0) return 1;
5      else{
6          return (2*n+1)*Tich(n-1);
7      }
8  }

```

## 2) Bài tập áp dụng:

Lập bảng  $C_n^k$  theo công thức truy hồi sau:

$$C_n^0 = C_n^n = 1$$

$$C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$$

Tam giác trên có dòng thứ  $n$  (bắt đầu từ  $n=0$ ) chứa  $n+1$  phần tử ( $k=0,1,\dots,n$ ) là các hệ số của nhị thức  $(a+b)^n$  và được gọi là tam giác Pascal.

### III. Quay lui .

**Thuật toán quay lui dùng để giải quyết các bài toán liệt kê các cấu hình. Mỗi cấu hình được xây dựng bằng cách xây dựng từng phần tử mỗi phần tử được chọn bằng cách thử tất cả các khả năng .**

*Giả thiết cấu hình cần liệt kê có dạng  $(x_1, x_2, \dots, x_n)$ . Khi đó thuật toán quay lui được thực hiện qua các bước sau:*

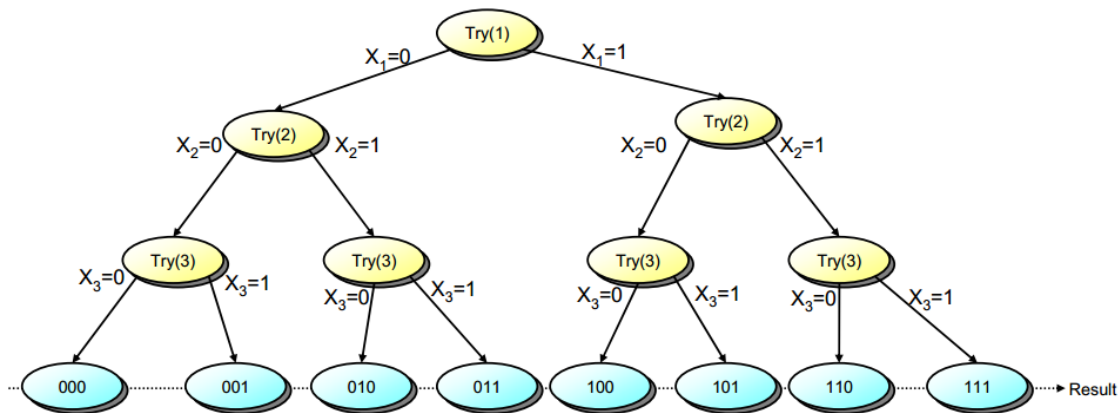
- 1) Xét tất cả các giá trị  $x_1$  có thể nhận, thử cho  $x_1$  nhận lần lượt các giá trị đó. Với mỗi giá trị thử cho  $x_1$  ta sẽ:
- 2) Xét tất cả các giá trị  $x_2$  có thể nhận, lại thử cho  $x_2$  nhận lần lượt các giá trị đó. Với mỗi giá trị thử gán cho  $x_2$  lại xét tiếp các khả năng chọn  $x_3$ ... cứ tiếp tục như vậy đến bước:
- n) Xét tất cả các giá trị  $x_n$  có thể nhận, thử cho  $x_n$  nhận lần lượt các giá trị đó, thông báo cấu hình tìm được  $(x_1, x_2, \dots, x_n)$ .

Ví dụ trong lập trình :

Vd 1:

Liệt kê các cấu hình sinh nhị phân có độ dài N.

$N = 3$ ;



Hình 1: Cây tìm kiếm quay lui trong bài toán liệt kê dãy nhị phân

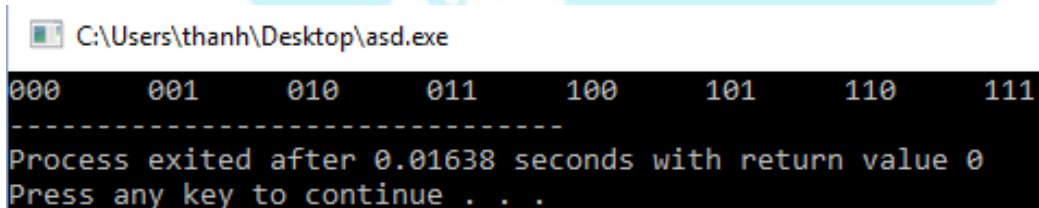
Code :

```

4  int a[100];
5  int n;
6  void sinh(int i)
7  {
8      for(int j=0;j<=1;j++){
9          a[i] = j; // phan tu o vi tri thu i = gia tri j
10         if(i==n-1){
11             // in ra cau hinh
12             for(int k=0;k<n;k++){
13                 cout<<a[k];
14             }
15             cout<<endl;
16         }
17         else sinh(i+1); // neu chu thoa man thi quay lui lai
18     }
19 }
20 main()
21 {
22     cin>>n;
23     sinh(0);
24 }

```

## Kết quả



```

C:\Users\thanh\Desktop\asd.exe
000    001    010    011    100    101    110    111
-----
Process exited after 0.01638 seconds with return value 0
Press any key to continue . . .

```

Chúng ta hiểu  $N = 3$  . thì 1 sinh nhị phân có 3 phần tử tương ứng với vị trí 0 , 1 , 2

Nên nếu  $i$  ở vị trí thứ 2 thì có nghĩa ta được 1 cấu hình sinh nhị phân , nếu mà chưa thỏa mãn thì theo lời gọi quay lui  $i+1$  ta gọi lại các hàm của nó và thực hiện .

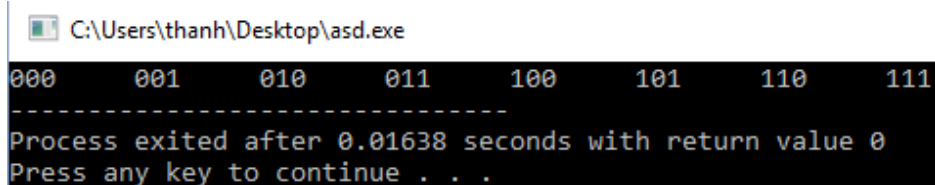
Đơn giản hơn

```

int a[100];
int n = 3;
void sinh()
{
    // khi loi gọi sinh (0);
    // ta có 1 vòng for
    for(int j=0;j<=1;j++){
        // tại đây i = 0 chưa tới vị trí để có 1 cấu hình
        // nên ta thực hiện lệnh else quay lại sinh( i+1 ) = 1
        // ta có thêm 1 vòng for
        for(int k=0;k<=1;k++){
            // tại đây i = 0 chưa tới vị trí để có 1 cấu hình
            // nên ta thực hiện lệnh else quay lại sinh( i+1 ) = 2
            // ta có thêm 1 vòng for
            for(int l=0;l<=1;l++){
                // tại đây khi i = 2 thỏa mãn để có được 1 cấu hình
                // nên ta thực hiện lệnh if (i==n-1)
                cout<<j<<k<<l<<endl;
                // trong code bên kia thì ta lưu
                // a[0] = j // a[1] = k // a[2] = l
                // rồi thực hiện thao tác in ra
            }
        }
    }
}
main()
{
    sinh();
}

```

## Kết quả



```

C:\Users\thanh\Desktop\asd.exe
000    001    010    011    100    101    110    111
-----
Process exited after 0.01638 seconds with return value 0
Press any key to continue . . .

```

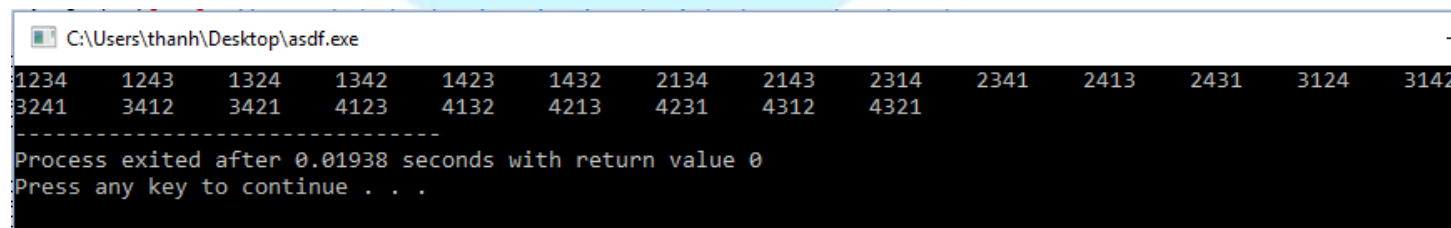


Lưu ý khi hết mỗi vòng for i lại giảm xuống và ta lại tiếp tục gọi vòng for khi i tăng lên .

**Vd2: liệt kê sinh hoán vị có độ dài n;**

```
1  #include<iostream>
2  using namespace std;
3  int n = 4;
4  int a[100];
5  bool check[100]; // mảng đánh dấu giá trị tại 0 thu j đã được sử dụng hay chưa
6  void sinhHoanVi(int i)
7  {
8      for(int j=1;j<=n;j++){
9          if(check[j]==false){ // nếu mà giá trị thu j chưa sử dụng thì ta sử dụng
10             a[i] = j;        // dùng j gán cho mảng a tại vị trí thu i
11             check[j] = true; // đánh dấu là đã sử dụng rồi
12             if(i==n-1){      // nếu mà tại vị trí cuối cùng thỏa mãn dk thì ta cho in ra giá trị
13                 for(int k=0;k<n;k++){
14                     cout<<a[k];
15                 }
16                 cout<<endl;
17             }
18             else sinhHoanVi(i+1); // nếu i chưa đạt tới giá trị cuối cùng thì ta tăng i lên
19             check[j] = false; // ta trả về giá trị thu j là chưa sử dụng để sử dụng lại
20         }
21     }
22 }
23 main()
24 {
25     sinhHoanVi(0);
26 }
```

**Kết quả :**



```
C:\Users\thanh\Desktop\asdf.exe
1234 1243 1324 1342 1423 1432 2134 2143 2314 2341 2413 2431 3124 3142
3241 3412 3421 4123 4132 4213 4231 4312 4321 1243 1324 1423 2134 2143
2314 2341 2413 2431 3124 3142 3412 3421 4123 4132 4213 4231 4312 4321
Process exited after 0.01938 seconds with return value 0
Press any key to continue . . .
```

**Bài toán:** Cho  $X = \{1, 2, 3, \dots, n\}$ . Hãy liệt kê tất cả các tập con  $k$  phần tử của  $X$  ( $k \leq n$ ).

**Giải:** Mỗi tập con của tập hợp  $X$  có thể biểu diễn bằng bộ có thứ tự gồm  $k$  thành phần  $a = (a_1 a_2 \dots a_k)$  thỏa mãn  $1 \leq a_1 \leq a_2 \leq \dots \leq a_k \leq n$ .

Trên tập các tập con  $k$  phần tử của  $X$  có thể xác định nhiều thứ tự khác nhau. Thứ tự dễ

## CLB Lập Trình PTIT – ProPTIT

nhìn thấy nhất là thứ tự từ điển được định nghĩa như sau:

Ta nói tập con  $a = a_1 a_2 \dots a_k$  đi trước tập con  $a' = a'_1 a'_2 \dots a'_k$  trong thứ tự từ điển và ký hiệu là  $a < a'$ , nếu tìm được chỉ số  $j$  ( $1 \leq j \leq k$ ) sao cho:  $a_1 = a'_1, a_2 = a'_2, \dots, a_{j-1} = a'_{j-1}, a_j < a'_j$ .

Chẳng hạn  $X = \{ 1, 2, 3, 4, 5 \}$ ,  $k = 3$ . Các tập con 3 phần tử của  $X$  được liệt kê theo thứ tự từ điển như sau:

1 2 3

1 2 4

1 2 5

1 3 4

1 3 5

1 4 5

2 3 4

2 3 5

2 4 5

3 4 5

Như vậy, tập con đầu tiên trong thứ tự từ điển là  $(1, 2, \dots, k)$  và tập con cuối cùng là  $(n-k+1, n-k+2, \dots, n)$ . Giả sử  $a = (a_1, a_2, \dots, a_k)$  là tập con hiện tại và chưa phải là cuối cùng, khi đó có thể chứng minh được rằng tập con kế tiếp trong thứ tự từ điển có thể được xây dựng bằng cách thực hiện các qui tắc biến đổi sau đối với tập con đang có.

Tìm từ bên phải dãy  $a_1, a_2, \dots, a_k$  phần tử  $a_i \neq n - k + i$

Thay  $a_i$  bởi  $a_{i+1}$ ,

Thay  $a_j$  bởi  $a_{i+j-i}$ , với  $j := i+1, i+2, \dots, k$

Chẳng hạn với  $n = 6, k = 4$ . Giả sử ta đang có tập con  $(1, 2, 5, 6)$ , cần xây dựng tập con kế tiếp nó trong thứ tự từ điển. Duyệt từ bên phải ta nhận được  $i = 2$ , thay  $a_2$  bởi  $a_{2+1} = 2 + 1 = 3$ .

Duyệt  $j$  từ  $i + 1 = 3$  cho đến  $k$ , ta thay thế  $a_3 = a_{2+3-2} = 3 + 3 - 2 = 4$ ,  $a_4 = a_{2+4-2} = 3 + 4 - 2 = 5$  ta nhận được tập con kế tiếp là  $(1, 3, 4, 5)$ .

Với qui tắc sinh như trên, chúng ta có thể mô tả bằng thuật toán sau:

```
1  #include<iostream>
2  using namespace std;
3  int n = 5;
4  int k = 3;
5  int a[100];
6
7  void sinh(int i)
8  {
9      for(int j=a[i-1]+1;j<=n-k+i;j++){
10         a[i] = j;
11         if(i==k){
12             for(int l=1;l<=k;l++){
13                 cout<<a[l];
14             }
15             cout<<endl;
16         }
17         else sinh(i+1);
18     }
19 }
20
21
22 main(){
23     sinh(1);
24 }
```

## Bài 9: Cấu trúc STRUCT

### I.Mở đầu:

Bạn hẳn là đã quen với các kiểu dữ liệu cơ bản như: **int**, **float**, **char** ....

Nhưng trên thực tế, có những lúc chúng ta cần những biến có thể lưu trữ được nhiều kiểu dữ liệu một lúc. VD: biến **sinhvien** lưu các tên, tuổi, sdt ... của một sinh viên...

Và cấu trúc **struct** trong ngôn ngữ lập trình C được tạo ra để giải quyết vấn đề đó.

VD:

```
#include <stdio.h>
#include <string.h>

// xây dựng 1 struct có tên là sinhvien
struct sinhvien{
    char ten[30]; // tên sinh viên
    char lop[30]; // lớp của sinh viên
    float diemtoan, diemli, diemhoa, diemtb; // điểm các môn của sinh viên
};

main()
{
    // sinhvien khai báo, giống như các kiểu dữ liệu cơ bản khác
    struct sinhvien sv1, sv2; // khai báo 2 biến có kiểu dữ liệu là sinhvien

    //Nhập thông tin sinh viên thứ nhất:
    printf("\nNhập thông tin sinh viên thứ nhất: \n");
    printf("\nNhập Họ tên: "); gets(sv1.ten); // truy cập đến các kiểu dữ liệu bên
    trong struct bằng dấu '.'
    printf("\nNhập lớp: "); gets(sv1.lop);
    printf("\nNhập điểm toán: "); scanf("%f",&sv1.diemtoan);
    printf("\nNhập điểm lí: "); scanf("%f",&sv1.diemli);
    printf("\nNhập điểm hoa: "); scanf("%f",&sv1.diemhoa);
    sv1.diemtb= (sv1.diemhoa +sv1.diemli +sv1.diemtoan) / 3; // tính điểm trung
    bình

    fflush(stdin);
    //Nhập thông tin sinh viên thứ 2:
    printf("\nNhập thông tin sinh viên thứ hai: \n");
    printf("\nNhập Họ tên: "); gets(sv2.ten); // truy cập đến các kiểu dữ liệu bên
```

trong struct bang dau ' '

```
printf("\nNhap lop: "); gets(sv2.lop);
printf("\nNhap diem toan: "); scanf("%f",&sv2.diemtoan);
printf("\nNhap diem li: "); scanf("%f",&sv2.diemli);
printf("\nNhap diem hoa: "); scanf("%f",&sv2.diemhoa);
sv2.diemtb= (sv2.diemhoa +sv2.diemli +sv2.diemtoan) / 3; // tinh diem trung
binh
//

//in thong tin 2 sinh vien:
printf("*****\n");
printf("%s \t %s \t %0.2f %0.2f %0.2f\n", sv1.ten, sv1.lop, sv1.diemtoan,
sv1.diemli, sv1.diemhoa);
printf("%s \t %s \t %0.2f %0.2f %0.2f\n", sv2.ten, sv2.lop, sv2.diemtoan,
sv2.diemli, sv2.diemhoa);
}
```

## II. Xây dựng kiểu cấu trúc, khai báo:

### 1) Xây dựng:

Để xây dựng 1 struct ta khai báo như sau:

```
struct ten_struct{
    kiểu dữ liệu cơ bản 1    tên biến 1;
    kiểu dữ liệu cơ bản 2    tên biến 2;
    .....
    kiểu dữ liệu cơ bản n    tên biến n;
};
```

Ngoài ra ta có thể tạo struct lồng trong struct

VD:

### 2) Khai báo:

Khi khai báo 1 biến có kiểu sinhvien như ở trên trong C thì bạn cần phải thêm từ '**struct**' đằng trước. VD: **struct** sinhvien sv1 là đúng, còn sinhvien sv1 thì lại sai. < trong C++ thì không cần thêm struct >

## CLB Lập Trình PTIT – ProPTIT

Khai báo mảng cấu trúc cũng giống như các kiểu dữ liệu cơ bản khác và thêm từ khóa 'struct' ở đầu:

VD: **struct** sinhvien dssv[100];

VD: struct sinhvien sv1, sv2 ;

### 3) Truy cập đến các thành phần:

Muốn truy cập đến các thành phần, ta thêm dấu '.' . VD: sv1.diemtoan , sv1.ten, sv1.ngaysinh.thang , .....

### 4) Gán các biến cấu trúc:

Khi gán sv1 = sv2 thì các dữ liệu của sv1 sẽ được gán với những dữ liệu tương ứng của sv2

Tức là: sv1.ten =sv2.ten;

sv1.lop = sv2.lop;

.....

### 5) Bài tập:

Viết chương trình thao tác với phân số:

Tạo 2 phân số.

Kiểm tra xem phân số tối giản chưa ? Rút gọn khi còn có thể.

Tính tổng, hiệu, tích, thương 2 phân số đó.

Viết chương trình thao tác với số phức, gồm các thao tác:

Tạo lập 2 số phức.

Tính tổng, hiệu, tích, thương 2 số phức đó.

Xây dựng chương trình quản lý sinh viên, sử dụng cấu trúc sau:

```
struct SV {char ten[30]; float diemtoan, diemly, tb;};
```

## CLB Lập Trình PTIT – ProPTIT

Nhập danh sách cho n sinh viên (n là số tự nhiên được nhập vào), chỉ nhập ten và diemtoan, diemly.

Tính giá trị trường  $tb = (toan + ly) / 2$ , sau đó sắp xếp lại danh sách theo tên và điểm trung bình tăng dần. Cho hiện kết quả trước và sau khi sắp xếp.

4.

Xây dựng chương trình quản lý học sinh. Mỗi học sinh quản lý các thông tin sau:

Họ tên, Năm sinh, Điểm trung bình, số thứ tự.

Với các chức năng sau:

- Nhập số liệu, mỗi lần có thể nhập m học sinh,  $m > 0$
- Xem danh sách: trên màn hình.
- Tìm kiếm :theo tên, theo năm sinh, theo tên và năm sinh.
- Sắp xếp: theo tên, theo điểm trung bình.
- Xóa thông tin 1 sinh viên khi biết số thứ tự của sinh viên đó.

## Bài 10: Con trỏ

Bình thường các bạn thường dùng các biến tĩnh hoặc là biến động.

int a, b, c[100]; thì a, b, c gọi là biến tĩnh.

Còn biến động như là lúc khai báo 1 mảng int \*a, sau khi nhập số phần tử là n ta mới cấp phát a = new int [n] thì a gọi là biến động.

Tuy nhiên thì biến động có nhiều ưu điểm hơn biến tĩnh:

### -Biến tĩnh:

- +Cấp phát ô nhớ dư, gây lãng phí ô nhớ.
- +Và ngược lại cũng có lúc cấp phát ô nhớ thiếu.

### - Biến động:

- +Kích thước, vùng nhớ ,địa chỉ vùng nhớ cấp phát có thể được thay đổi.
- +Phát sinh trong lúc chạy chương trình mà không phải ngay lúc đầu.

Tuy nhiên thì các biến động không có địa chỉ cụ thể nên ta không thể truy cập đến chúng được. Vì thế, người ta đã tạo ra biến **con trỏ**.

### -Ưu điểm:

- +Biến con trỏ không lưu dữ liệu mà lưu địa chỉ của ô chứa dữ liệu
- +Kích thước của biến con trỏ không phụ thuộc vào kiểu dữ liệu và luôn bằng 2 byte.

Ví dụ:

Chắc các bạn cũng đã quen với hàm hoán vị rồi phải k ? Và sau bài này thì các bạn biết tại sao lại có dấu & rồi đấy.

```
1  #include <iostream>
2  using namespace std;
3  void hoanVi(int &a, int &b){
4      int c=a;
5      a=b;
6      b=c;
7  }
```



## I. Khai báo con trỏ:

<kiểu dữ liệu> \*tên\_con\_trỏ;

Ví dụ:

int \*px => khai báo 1 con trỏ kiểu dữ liệu int;

## II. Dùng con trỏ để trỏ tới địa chỉ 1 biến:

Ta dùng: tên\_con\_trỏ = &tên\_biến;

Ví dụ:

```
1  #include <iostream>
2  using namespace std;
3  main(){
4      int a=5, *b;
5      b=&a;
6  }
```

Phép lấy & là phép lấy địa chỉ của biến

## III. Truy xuất:

- Vì con trỏ để lưu địa chỉ nên in ra con trỏ tức là in ra địa chỉ, muốn in ra giá trị của ô nhớ mà con trỏ chỉ tới ta phải thêm “\*” vào trước biến con trỏ.

Ví dụ:

```
1  #include <iostream>
2  using namespace std;
3  main(){
4      int a=5, *b;
5      b=&a;
6      cout<<b<<endl; // in ra địa chỉ của biến a
7      cout<<*b; // in ra giá trị của biến a
8  }
```

## IV. Một số phép toán trên con trỏ:

- Con trỏ cũng có phép gán con trỏ cho con trỏ, con trỏ cho địa chỉ của 1 biến tĩnh (như các ví dụ trên).
- Con trỏ có phép cộng

Ví dụ:

```
4  main(){
5      int a=5, *b;
6      b=&a;
7      b=b+10;
8  }
```

**V. Tham chiếu và tham trị:**

- Bình thường khi dùng hàm con chúng ta vẫn truyền tham trị tức là truyền tên biến . Nhưng khi muốn giá trị của những biến ta truyền vào thay đổi sau hàm con thì ta phải truyền vào địa chỉ các biến đó (truyền tham chiếu)

**Ví dụ:**

Truyền tham chiếu biến a và b	Truyền tham trị biến n
<pre>void hoan_vi(int &amp;a, int &amp;b){     int c = a;     a = b;     b = c; }</pre>	<pre>void nhap(int a[],int n){     for(int i=0; i&lt;n; i++) {         cin&gt;&gt;a[i];     } }</pre>

## Bài 11: Các phương pháp sắp xếp

Ta có 1 dãy số  $a_1, a_2, \dots, a_N$  được lưu trữ trong cấu trúc dữ liệu mảng. Sắp xếp dãy số  $a_1, a_2, \dots, a_N$  là thực hiện việc bố trí lại các phần tử sao cho hình thành được dãy mới  $a_{k1}, a_{k2}, \dots, a_{kN}$  có thứ tự (ví dụ thứ tự tăng) nghĩa là  $a_{ki} > a_{ki-1}$ .

Tùy theo yêu cầu bài toán và tính chất của dãy số, ta có các phương pháp sắp xếp khác nhau. Mình sẽ giới thiệu 3 phương pháp phổ thông nhất : sắp xếp nổi bọt, sắp xếp chọn và sắp xếp nhanh.

### I. Sắp xếp nổi bọt

#### 1. Ý tưởng

Xuất phát từ đầu dãy, so sánh 2 phần tử cạnh nhau để đưa phần tử nhỏ hơn lên trước, sau đó lại xét cặp tiếp theo cho đến khi tiến về đầu dãy. Nhờ vậy, ở lần xử lý thứ  $i$  sẽ tìm được phần tử ở vị trí đầu dãy là  $i$ .

#### 2. Giải thuật

-Bước 1:  $i=1$  // Lần xử lý đầu tiên

-Bước 2:  $j=N$  // Duyệt từ cuối dãy trở về vị trí

Trong khi  $j > i$  thực hiện:

Nếu  $a[j] < a[j-1]$ : hoán vị  $a[j]$  và  $a[j-1]$

$j=j-1$ ;

-Bước 3:  $i=i+1$  // Lần xử lý tiếp theo

Nếu  $i > N-1$  thì dừng

Ngược lại, lặp lại bước 2.

Nhận thấy : lần duyệt đầu tiên ta cần duyệt  $n$  phần tử, lần duyệt thứ 2 cần duyệt  $n-1$  phần tử, ... cứ duyệt như vậy cho đến lần duyệt cuối cùng là 1 phần tử. Vậy số lần duyệt của phương pháp này là  $n + (n-1) + (n-2) + \dots + 2 + 1 = n(n+1)/2$  tương đương với độ phức tạp  $O(n^2)$ .

#### 3. Ví dụ

Cho mảng  $a[9]=\{3,10,4,6,2,6,15,3,9,7\}$

	3	10	4	6	2	6	15	3	9	7
Bước 1	2	3	10	4	6	3	6	15	7	9
Bước 2		3	3	10	4	6	6	7	15	9
Bước 3			3	4	10	6	6	7	9	15
Bước 4				4	6	10	6	7	9	15
Bước 5					6	6	10	7	9	15
Bước 6						6	7	10	9	15
Bước 7							7	9	10	15
Bước 8								9	10	15
Bước 9									10	15
Kết quả	2	3	3	4	6	6	7	9	10	15

Code :

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

## II. Sắp xếp chọn

### 1. Ý tưởng

Chọn phần tử nhỏ nhất trong  $n$  phần tử ban đầu, đưa phần tử này về vị trí đúng là đầu tiên của dãy hiện hành. Sau đó không quan tâm đến nó nữa, xem dãy hiện hành chỉ còn  $n-1$  phần tử của dãy ban đầu, bắt đầu từ vị trí thứ 2. Lặp lại quá trình trên cho dãy hiện hành đến khi dãy hiện hành chỉ còn 1 phần tử.

### 2. Giải thuật

- Bước 1 : Đặt  $i=1, \min=1$  với  $\min$  là biến dùng để tìm vị trí phần tử nhỏ nhất trong dãy đang xét.
- Bước 2 : Tìm phần tử  $a[\min]$  nhỏ nhất trong dãy đang xét từ  $i$  đến  $n$ .
- Bước 3 : Hoán vị  $a[\min]$  và  $a[i]$ .
- Bước 4 : Nếu  $i \leq n-1$  thì  $i=i+1$  và quay lại bước 2. Nếu  $i=n$  thì dãy đã được sắp xếp.

Độ phức tạp :  $O(n^2)$  tương đương với phương pháp nổi bọt.

## 3. Ví dụ

Cho dãy  $a[9]=\{3,5,7,2,4,6,9,8,7,6\}$

	3	5	7	2	4	6	9	8	7	6
Bước 1	2	5	7	3	4	6	9	8	7	6
Bước 2		3	7	5	4	6	9	8	7	6
Bước 3			4	5	7	6	9	8	7	6
Bước 4				5	7	6	9	8	7	6
Bước 5					6	7	9	8	7	6
Bước 6						6	9	8	7	7
Bước 7							7	8	9	7
Bước 8								7	9	8
Bước 9									8	9
Kết quả	2	3	4	5	6	6	7	7	8	9

Code :

```
void selectionSort(int *a,int n)
{
    for(int i=0;i<n;i++)
    {
        int index=i;
        for(int j=i+1;j<n;j++)
        {
            if(a[j]< a[index]) index=j;
        }
        swap(a[i],a[index]);
    }
}
```

## III. Sắp xếp nhanh :

### 1. Ý tưởng

QuickSort chia mảng thành hai danh sách bằng cách so sánh từng phần tử của danh sách với một phần tử được chọn được gọi là phần tử chốt. Những phần tử nhỏ hơn hoặc bằng phần tử chốt được đưa về phía trước và nằm trong danh sách con thứ nhất, các phần tử lớn hơn chốt được đưa về phía sau và thuộc danh sách con thứ hai. Cứ tiếp tục chia như vậy tới khi các danh sách con đều có độ dài bằng

**-Có các cách chọn phần tử chốt như sau:**

- +Chọn phần tử đứng đầu hoặc đứng cuối làm phần tử chốt.
- +Chọn phần tử đứng giữa danh sách làm phần tử chốt.
- +Chọn phần tử trung vị trong 3 phần tử đứng đầu, đứng giữa và đứng cuối làm phần tử chốt.

+Chọn phần tử ngẫu nhiên làm phần tử chốt (Cách này hay được chọn vì tránh được các trường hợp đặc biệt)

### 2. Giải thuật :

-Bước 1: Nếu  $left \leq right$ , thực hiện bước 2. Ngược lại thì kết thúc.

-Bước 2: Chọn  $x = a_{(right - left)/2}$  là phần tử chốt,  $i = right$ ,  $j = left$ .

Phân hoạch dãy ban đầu  $a_{left} \dots a_{right}$  thành các đoạn :  $a_{left} \dots a_j$  (đoạn chứa những phần tử nhỏ hơn  $x$ ),  $a_{j+1} \dots a_{i-1}$  (đoạn chứa những phần tử bằng  $x$ ),  $a_i \dots a_{right}$  (đoạn chứa những phần tử lớn hơn  $x$ ).

-Bước 3: Sắp xếp đoạn 1:  $a_{left} \dots a_j$  : quay lại bước 1, với  $right = j$ .

-Bước 4: Sắp xếp đoạn 3:  $a_i \dots a_{right}$  : quay lại bước 1, với  $left = i$ .

**Nhận xét :** Việc chọn phần tử là phần tử chốt quyết định đến khả năng xử lý của phương pháp này.

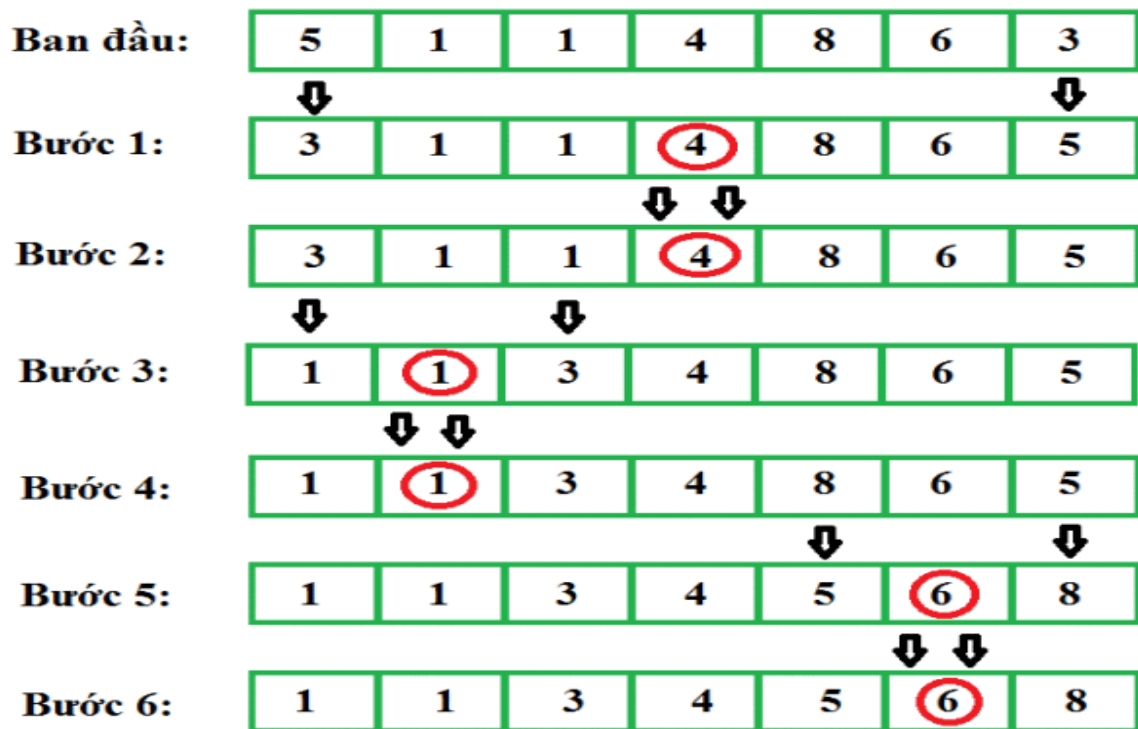
**-Trường hợp tốt nhất:** mỗi lần phân hoạch ta đều chọn được phần tử trung vị (phần tử lớn hơn hay bằng nửa số phần tử và nhỏ hơn hay bằng nửa số phần tử còn lại) làm mốc. Khi đó dãy được phân hoạch thành hai phần bằng nhau, và ta cần  $\log_2(n)$  lần phân hoạch thì sắp xếp xong. Ta cũng dễ nhận thấy trong mỗi lần phân hoạch ta cần duyệt qua  $n$  phần tử. Vậy độ phức tạp trong trường hợp tốt nhất thuộc  $O(n \log_2(n))$ .

**-Trường hợp xấu nhất:** mỗi lần phân hoạch ta chọn phải phần tử có giá trị cực đại hoặc cực tiểu làm mốc. Khi đó dãy bị phân hoạch thành hai phần không đều: một phần chỉ có một phần tử, phần còn lại có  $n-1$  phần tử. Do đó, ta cần tới  $n$  lần phân hoạch mới sắp xếp xong. Vậy độ phức tạp trong trường hợp xấu nhất thuộc  $O(n^2)$ .

**-Trường hợp trung bình :** Độ phức tạp  $O(n \log n)$ .

### 3. Ví dụ

Cho mảng  $a[7] = \{5, 1, 1, 4, 8, 6, 3\}$ .



Code :

```

void quickSort(int *a,int left,int right)
{
    int l=left, r=right, temp=a[(left+right)/2];
    while(l<=r)
    {
        while(a[l]<temp && l< right) l++;
        while(a[r]> temp && r> left) r--;
        if(l<=r)
        {
            swap(a[l++],a[r--]);
        }
    }
    if(left < r) quickSort(a, left, r);
    if(l < right) quickSort(a , l, right);
}

```

#### IV. Sort Algorithm

##### 1. Giới thiệu chung

Trong thực tế, cũng như trong khi giải quyết các vấn đề lớn. Sắp xếp rất quan trọng. Chúng ta có rất nhiều thuật toán sắp xếp như : Bubble Sort, Insert Sort,

Selection Sort,...Mỗi một thuật toán có sự phức tạp về thuật toán là khác nhau, đi kèm với nó là thời gian chạy, và ưu nhược điểm của nó.

Phần này chúng ta sẽ đi sâu vào phần Sort trong Algorithm

### 2. Cài đặt

Hàm sắp xếp sort được dùng chủ yếu trên 2 loại

- Sắp xếp tuần tự. Mặc định của sort sử dụng toán tử < (nhỏ hơn)
- Sắp xếp có điều kiện. Sử dụng khi toán tử < mặc định không sử dụng được (đối với struct) hoặc do yêu cầu sắp xếp của ta.

Dưới đây là các ví dụ về hàm sort trong algorithm

```
1 // sort algorithm example
2 #include <iostream>      // std::cout
3 #include <algorithm>     // std::sort
4 #include <vector>        // std::vector
5
6 bool myfunction (int i,int j) { return (i<j); }
7
8 struct myclass {
9     bool operator() (int i,int j) { return (i<j);}
10 } myobject;
11
12 int main () {
13     int myints[] = {32,71,12,45,26,80,53,33};
14     std::vector<int> myvector (myints, myints+8);           // 32 71 12 45 26 80 53 33
15
16     // using default comparison (operator <):
17     std::sort (myvector.begin(), myvector.begin()+4);       //(12 32 45 71)26 80 53 33
18
19     // using function as comp
20     std::sort (myvector.begin()+4, myvector.end(), myfunction); // 12 32 45 71(26 33 53 80)
21
22     // using object as comp
23     std::sort (myvector.begin(), myvector.end(), myobject);  //(12 26 32 33 45 53 71 80)
24
25     // print out content:
26     std::cout << "myvector contains:";
27     for (std::vector<int>::iterator it=myvector.begin(); it!=myvector.end(); ++it)
28         std::cout << ' ' << *it;
29     std::cout << '\n';
30
31     return 0;
32 }
```

Output:

```
myvector contains: 12 26 32 33 45 53 71 80
```



```

1.  /*
2.  sap xep struct */
3.  #include <iostream>
4.  #include <algorithm>
5.  using namespace std ;
6.
7.  struct diem { // dinh nghia 1 diem
8.      int x ;
9.      int y ;
10. };
11.
12. bool sort_x(diem a, diem b){ // ham so sanh
13.     return (a.x > b.x) || (a.x == b.x)&&(a.y < b.y) ;
14. }
15.
16. int main(){
17.     cout << "Nhap vao so diem : " ;
18.     int n ;
19.     cin >> n ;
20.     diem *a = new diem [n] ; // khai bao mang struct la diem
21.     cout << "Nhap vao toa do x, y cua diem" << endl ;
22.     for(int i = 0; i < n; i ++ ){
23.         cin >> a[i].x >> a[i].y ; //nhap diem
24.     }
25.     sort(a,a+n,sort_x) ; // ham sap xep thuc hien
26.     cout << "\nToa do diem trong danh sach " << endl ;
27.     for(int i = 0; i < n; i ++ ){ // In ra toa do
28.         cout << a[i].x << " " << a[i].y << endl ;
29.     }
30.     cout << endl;
31. }
32.

```

Input :	Output
5	Toa do diem trong danh sach
2 3	1 2
8 6	2 3
9 2	2 6
1 2	8 6
2 6	9 2

## Bài 12: String

### I. Kiểu chuỗi của C và hạn chế

Khi mới học C, chắc các bạn đều rất bối rối khi làm việc với xâu ký tự, việc sử dụng con trỏ lưu xâu ký tự rất phức tạp, dễ gây lỗi khiến nhiều người sử dụng cảm thấy khó tiếp cận và dễ phát sinh lỗi khi chạy chương trình.

Các chương trình C++ có thể sử dụng chuỗi theo cách thức cũ của *Ngôn ngữ C*: mảng các ký tự kết thúc bởi ký tự mã ASCII là 0 (ký tự '\0') cùng với các hàm thư viện khai báo trong <string.h>. Tuy nhiên nó nhiều bất tiện khi dùng theo cách thức này:

- Người lập trình phải chủ động kiểm soát bộ nhớ cấp phát cho chuỗi ký tự. Nói chung là phải am hiểu và rất thông thạo về kỹ thuật dùng bộ nhớ và con trỏ thì chương trình mới tránh được các lỗi về kỹ thuật.
- Không thể gán giá trị hay sử dụng phép toán + (ghép chuỗi) và các phép toán so sánh như: > (lớn hơn), < (nhỏ hơn)... mà phải gọi các hàm thư viện trong <string.h>;
- Nếu dùng kỹ thuật cấp phát động thì phải quản lý việc cấp thêm bộ nhớ khi chuỗi dẫn ra (chẳng hạn do ghép chuỗi) và phải hủy bộ nhớ (khi không dùng nữa) để tránh việc cạn kiệt bộ nhớ của máy tính trong trường hợp có nhiều chương trình hoạt động đồng thời. Đây cũng chính là những điểm khác nhau cơ bản giữa string trong C++ và char trong C.

### II. Kiểu chuỗi string của C++

Thư viện chuẩn STL (Standard Template Library) cung cấp kiểu string (xâu ký tự), giúp các bạn tránh khỏi hoàn toàn các phiền phức nêu trên.

```
#include <string>
using namespace std;
```

### III. Các phương thức tiện ích của string

Kiểu string của STL hỗ trợ các nhóm phương thức và phép toán tiện ích sau đây.

#### 1. Các phép toán và phương thức cơ bản

- Các toán tử +, += dùng để ghép hai chuỗi và cũng để ghép một ký tự vào chuỗi. Các phép so sánh theo thứ tự từ điển: == (bằng nhau), != (khác nhau), > (lớn hơn), >= (lớn hơn hay bằng), < (nhỏ hơn), <= (nhỏ hơn hay bằng);
  - Phương thức length( ) và phép lấy chỉ số [ ] để duyệt từng ký tự của chuỗi: nếu s là biến kiểu string thì s[i] là ký tự thứ i của s với  $0 \leq i < s.length()$ ;
  - Phép gán (=) dùng để gán biến kiểu string bằng một chuỗi, hoặc bằng string khác.
  - Có thể dùng toán tử "<<" với cout để xuất một chuỗi ra màn hình hoặc dùng toán tử ">>" với cin để nhập một chuỗi ký tự đến khi gặp một khoảng trống thì dừng.
- Vi dụ:

```
1  #include <iostream>
2  using namespace std;
3  main(){
4      string a="Hello ", b="Lop Hoc ProPTIT";
5      string c;
6      cout<<a+b<<endl;
7      cin>>c;
8      cout<<c;
9  }
```

*Một vấn đề thường nảy sinh trong các ứng dụng có sử dụng C-string: một C-String chưa khởi tạo cần được gán **NULL**.*

Tuy nhiên, rất nhiều hàm thư viện của C-String sẽ gặp sự cố trong thời gian chạy khi gặp đối tượng C-String là NULL. Chẳng hạn hàm `strlen(x)` khi `x` là NULL được một số trình biên dịch chấp nhận, nhưng với nhiều hiện thực khác của thư viện C-String, thì gặp lỗi trong thời gian chạy. `string` trong C++ không gặp vấn đề này, ta hoàn toàn có thể cho 1 chuỗi là rỗng mà không gặp bất cứ lỗi nào.

**String thực chất là một `vector<char>`** có bổ sung thêm một số phương thức và thuộc tính, do đó, nó có toàn bộ các tính chất của 1 vector, vd hàm `size()`, `push_back()`, toán tử `[]` ...

### **Phương thức Mô tả**

`v.size()` Số lượng phần tử

`v.empty()` Trả về 1 nếu chuỗi rỗng, 0 nếu ngược lại.

`v.max_size()` Trả về số lượng phần tử tối đa đã được cấp phát

`v1 == v2` Trả về 1 nếu hai chuỗi giống nhau

`v1 != v2` Trả về 1 nếu hai chuỗi khác nhau

`v.begin()` Trả về iterator đầu tiên của chuỗi

`v.end()` Trả về iterator lặp cuối cùng của chuỗi

`v.front()` Trả về tham chiếu đến phần tử đầu tiên của chuỗi

`v.back()` Trả về tham chiếu đến phần tử cuối cùng của chuỗi

`v1.swap(v2)` Hoán đổi 2 chuỗi với nhau (giống việc hoán đổi giá trị của 2 biến)

Ví dụ:

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main(){
5      string s1, s2;
6      cin>>s1>>s2;
7      //phuong thuc size() co y nghia nhu phuong thuc length()
8      cout<<s1.size()<<" "<<s2.size()<<endl;
9      //so sanh 2 string
10     if(s1==s2){
11         cout<<"s1 = s2"<<endl;
12     }else{
13         cout<<"s1 != s2"<<endl;
14     }
15     //in ra ki tu dau tien cua string s1
16     cout<<s1[0]<<endl;
17     //hoan vi 2 string s1 va s2
18     swap(s1,s2);
19 }

```

**Nhập một string:** `getline ( cin, string str);`

Để nhập một string ta có thể sử dụng **cin** tuy nhiên **cin** chỉ cho phép ta nhập 1 từ vậy để nhập một dòng văn bản ta sử dụng phương thức `getline(cin, string str);`

Ví dụ:

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main(){
5      string name;
6      cout<<"Nhap ten cua ban: ";
7      getline(cin,name);
8      cout<<"Xin chao: "<<name;
9  }
10

```

**Xóa bộ nhớ đệm**

Bạn hãy thử làm ví dụ sau để nhận ra vấn đề

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main(){
5      int age;
6      string name;
7      cout<<"Nhap tuoi: ";
8      cin>>age;
9      cout<<"Nhap ten cua ban: ";
10     getline(cin,name);
11     cout<<"Xin chao: "<<name;
12 }

```

Khi bạn chạy chương trình trên, chương trình chỉ cho phép chúng ta nhập tuổi rồi hiển thị kết quả mà không cho phép chúng ta nhập tên. Để khắc phục lỗi này ta phải xóa bộ đệm trước khi nhập string nếu trước đó ta có nhập một số như ví dụ trên. Để xóa bộ đệm ta dùng hàm `fflush(stdin)` hoặc `cin.ignore()`. Dưới đây là chương trình khắc phục lỗi của ví dụ trên

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main(){
5      int age;
6      string name;
7      cout<<"Nhập tuổi: ";
8      cin>>age;
9      //fflush(stdin);
10     cin.ignore();
11     cout<<"Nhập tên của bạn: ";
12     getline(cin,name);
13     cout<<"Xin chào: "<<name;
14 }
```

## 2. Các phương thức chèn, xóa, lấy chuỗi con

-Phương thức `substr(int pos, int nchar)`

Trích ra chuỗi con có nchar ký tự của một chuỗi cho trước tại vị trí pos.

Ví dụ:

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main(){
5      string s="Lop Hoc ProPTIT";
6      string str=s.substr(8,7);
7      cout<<str; // in ra ProPTIT
8 }
```

- Phương thức `insert( )`

Chèn thêm ký tự hay chuỗi vào một vị trí nào đó của chuỗi str cho trước.

Có nhiều cách dùng phương thức này:

`str.insert(int pos, char* s);` chèn s (mảng ký tự kết thúc '\0') vào vị trí pos của str;

`str.insert(int pos, string s);` chèn chuỗi s (kiểu string) vào vị trí pos của chuỗi str (hay sử dụng nhất);

`str.insert(int pos, int n, int ch);` chèn n lần ký tự ch vào vị trí pos của chuỗi str;

Ví dụ:

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main(){
5      string s="Lop Hoc ";
6      s.insert(8, "ProPTIT");
7      cout<<s; // in ra màn hình chu Lop Hoc ProPTIT
8  }
```

- Phương thức str.erase(int pos, int n)

Xóa n ký tự của chuỗi str kể từ vị trí pos; nếu không quy định giá trị n thì tất cả các ký tự của str từ vị trí pos trở đi sẽ bị xóa

Ví dụ:

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main(){
5      string s="Lop Hoc ProPTIT";
6      s.erase(7, 8);
7      cout<<s; // in ra màn hình chu Lop Hoc
8  }
```

### 3. Các So sánh

-Bạn có thể đơn giản là sử dụng những toán tử quan hệ ( ==, !=, <, <=, >= ) được định nghĩa sẵn. Tuy nhiên, nếu muốn so sánh một phần của một chuỗi thì sẽ cần sử dụng phương thức compare():

`int compare ( const string& str );`

`int compare ( const char* s );`

`int compare ( size_t pos1, size_t n1, const string& str );`

`int compare ( size_t pos1, size_t n1, const char* s );`

`int compare ( size_t pos1, size_t n1, const string& str, size_t pos2, size_t n2);`

`int compare ( size_t pos1, size_t n1, const char* s, size_t n2);`

Hàm trả về 0 khi hai chuỗi bằng nhau và lớn hơn hoặc nhỏ hơn 0 cho trường hợp khác.

Ví dụ:

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main (){
5      string str1 ("green apple");
6      string str2 ("red apple");
7      if (str1.compare(str2) != 0)
8          cout << str1 << " is not " << str2 << endl;
9      if (str1.compare(6,5,"apple") == 0)
10         cout << "still, " << str1 << " is an apple" << endl;
11         if (str2.compare(str2.size()-5,5,"apple") == 0)
12             cout << "and " << str2 << " is also an apple" << endl;
13             if (str1.compare(6,5,str2,4,5) == 0)
14                 cout << "therefore, both are apples" << endl;
15     }

```

#### 4. Các phương thức tìm kiếm và thay thế

- Phương thức find( )

Tìm kiếm xem một ký tự hay một chuỗi nào đó có xuất hiện trong một chuỗi str cho trước hay không. Có nhiều cách dùng phương thức này:  
 str.find(int ch, int pos = 0); tìm ký tự ch kể từ vị trí pos đến cuối chuỗi str  
 str.find(char \*s, int pos = 0); tìm s (mảng ký tự kết thúc '\0') kể từ vị trí pos đến cuối  
 str.find(string& s, int pos = 0); tìm chuỗi s kể từ vị trí pos đến cuối chuỗi.  
 Nếu không quy định giá trị pos thì hiểu mặc nhiên là 0; nếu tìm có thì phương thức trả về vị trí xuất hiện đầu tiên ngược lại trả về giá trị -1.

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main (){
5      string s="Lop Hoc ProPTIT";
6
7      int pos= s.find("ProPTIT");
8      cout<<pos; // in ra 8
9  }

```

-Hàm tìm kiếm ngược (rfind)

Tương tự như hàm find() nhưng hàm rfind() để tìm kiếm từ cuối lên đầu string

Ví dụ:



```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main (){
5      string s="Lop Hoc ProPTIT ProPTIT";
6
7      int pos= s.rfind("ProPTIT");
8      cout<<pos; // in ra 16
9  }
```

### - Phương thức replace( )

Thay thế một đoạn con trong chuỗi str cho trước (đoạn con kể từ một vị trí pos và đếm tới nchar ký tự về phía cuối chuỗi) bởi một chuỗi s nào đó, hoặc bởi n ký tự ch nào đó. Có nhiều cách dùng thứ tự tham số như sau:

```
str.replace(int pos, int nchar, char *s);
str.replace(int pos, int nchar, string s);
str.replace(int pos, int nchar, int n, int ch);
```

Ví dụ :

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  main (){
5      string s="Lop Hoc ProPTIT";
6      s.replace(0, 7, "CLB");
7      cout<<s; // in ra CLB ProPTIT
8  }
```

## 5. Tách chuỗi

Trong việc xử lý chuỗi ký tự, không thể thiếu được các thao tác tách chuỗi ký tự thành nhiều chuỗi ký tự con thông qua các ký tự ngăn cách. Các hàm này có sẵn trong các ngôn ngữ khác như Visual Basic, Java, hay thậm chí là trong <string.h> (không phải <string>) Với STL, các bạn có thể dễ dàng tự xây dựng một hàm với chức năng tương tự:



```

1  #include <iostream>
2  #include <vector>
3  #include <string>
4  using namespace std;
5  main(){
6      string S = "Xin chao tat ca cac ban";
7      int t=0, t2=0;
8      vector<string> split;
9      while(t< S.size()){
10         t2=S.find(" ", t);
11         if (t!=t2) split.push_back(S.substr(t, t2-t));
12         if(t2==-1) break;
13         else t=t2+1;
14     }
15     for (int i=0; i<split.size(); i++)
16         cout << split[i] << endl;
17 }

```

Output:

Xin  
chao  
tat  
ca  
cac  
ban

Đoạn chương trình sử dụng các kỹ thuật sau

- Phương thức find() dùng để tìm vị trí đầu tiên của ký\_tự\_tìm bắt đầu từ vị\_trí\_đầu. Hàm này trả về vị trí của ký tự tìm được (nếu tìm thấy) hoặc -1 (nếu không tìm thấy)
- Hàm lấy chuỗi con substr() để lấy ra chuỗi con
- Đoạn chương trình thực hiện tách các xâu ký tự kể cả trong trường hợp có nhiều ký tự space nằm liên tiếp nhau.

Một cách đơn giản hơn là bạn có thể gọi hàm strtok() trong string.h để làm việc này, nhưng không may là hàm này thao tác trên char\* chứ không phải string. Hàm thành viên c\_str() sẽ giúp bạn chuyển từ string thành dạng C-string:

```
const charT* c_str ( ) const;
```

Hàm này cũng tự động sinh ra ký tự **null** chèn vào cuối xâu.

Từ prototype ta cũng thấy được hàm trả về một hằng chuỗi, điều này đồng nghĩa với việc ta không thể thay đổi chuỗi trả về.

Gọi phương thức c\_str() ;

```
string s = "some_string";
```

```
cout << s.c_str() << endl;
```

```
cout << strlen(s.c_str()) << endl;
```

Sau đây là ví dụ bên trên được viết lại dùng hàm thành viên `c_str()` và các hàm trong `<string.h>`

```
1  #include <iostream>
2  #include <string>
3  #include <string.h>
4  using namespace std;
5  main(){
6      string S = "Xin chao tat ca cac ban";
7      char str[100], *p;
8      strcpy(str, S.c_str());
9      p=strtok(str, " ");
10     while(p!=NULL){
11         cout<<p<<endl;
12         p=strtok(NULL, " ");
13     }
14 }
```

Output:

Xin  
chao  
tat  
ca  
cac  
ban

# TÀI LIỆU THAM KHẢO

- [1].ĐẶNG BÌNH PHƯƠNG. *Ngôn ngữ lập trình C*.
- [2].PHẠM VĂN ÁT. *Kỹ thuật lập trình C cơ bản và nâng cao*. [ed.] 6. 2006.
- [3].PHAN THỊ HÀ. *Tin học cơ sở 2*. 2013.

