MỤC LỤC

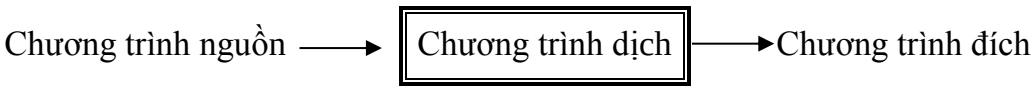
BÀI 1: KHÁI NIỆM LẬP TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH

Như đã biết, mọi bài toán có thuật toán đều có thể giải được trên máy tính điện tử. Khi giải bài toán trên máy tính điện tử, sau các bước xác định bài toán và xây dựng hoặc lựa chọn thuật toán khả thi là bước lập trình.

Lập trình *là sử dụng cấu trúc dữ liệu và các câu lệnh của ngôn ngữ lập trình cụ thể để mô tả dữ liệu và diễn đạt các thao tác của thuật toán*. Chương trình viết bằng ngôn ngữ lập trình bậc cao nói chung không phụ thuộc vào máy, nghĩa là một chương trình có thể thực hiện trên nhiều máy. Chương trình viết bằng ngôn ngữ máy có thể được nạp trực tiếp vào bộ nhớ và thực hiện ngay còn chương trình viết bằng ngôn ngữ lập trình bậc cao phải được chuyển đổi thành chương trình trên ngôn ngữ máy mới có thể thực hiện được.

Chương trình đặc biệt có chức năng chuyển đổi chương trình được viết bằng ngôn ngữ lập trình bậc cao thành chương trình thực hiện được trên máy tính cụ thể được gọi là ***chương trình dịch.***

Chương trình dịch nhận đầu vào là chương trình viết bằng ngôn ngữ lập trình bậc cao (chương trình nguồn) thực hiện chuyển đổi sang ngôn ngữ máy (chương trình đích).



Xét ví dụ, bạn chỉ biết tiếng Việt nhưng cần giới thiệu về trường của mình cho đoàn khách đến từ nước Mĩ, chỉ biết tiếng Anh. Có hai cách để bạn thực hiện điều này.

Cách thứ nhất: Bạn nói bằng tiếng Việt và người phiên dịch giúp bạn dịch sang tiếng Anh. Sau mỗi câu hoặc một vài câu giới thiệu trọn một ý, người phiên dịch dịch sang tiếng Anh cho đoàn khách. Sau đó, bạn lại giới thiệu tiếp và người phiên dịch lại dịch tiếp. Việc giới thiệu của bạn và việc dịch của người phiên dịch luân phiên cho đến khi bạn kết thúc nội dung giới thiệu của mình. Cách dịch trực tiếp như vậy được gọi là *thông dịch*.

Cách thứ hai: Bạn soạn nội dung giới thiệu của mình ra giấy, người phiên dịch dịch toàn bộ nội dung đó sang tiếng Anh rồi đọc hoặc trao văn bản đã dịch cho đoàn khách đọc. Như vậy, việc dịch được thực hiện sau khi nội dung giới thiệu đã hoàn tất. Hai công việc đó được thực hiện trong hai khoảng thời gian độc lập, tách biệt nhau. Cách dịch như vậy được gọi là *biên dịch*.

Sau khi kết thúc, với cách thứ nhất không có một văn bản nào để lưu trữ, còn với cách thứ hai có hai bản giới thiệu bằng tiếng Việt và bằng tiếng Anh có thể lưu trữ để dùng lại về sau.

Tương tự như vậy, chương trình dịch có hai loại là *thông dịch* và *biên dịch*.

a) Thông dịch

Thông dịch (interpreter) được thực hiện bằng cách lặp lại dãy các bước sau:

① Kiểm tra tính đúng đắn của câu lệnh tiếp theo trong chương trình nguồn;

② Chuyển đổi câu lệnh đó thành một hay nhiều câu lệnh tương ứng trong ngôn ngữ máy;

③ Thực hiện các câu lệnh vừa chuyển đổi được.

Như vậy, quá trình dịch và thực hiện các câu lệnh là luân phiên. Các chương trình thông dịch lần lượt dịch và thực hiện từng câu lệnh. Loại chương trình dịch này đặc biệt thích hợp cho môi trường đối thoại giữa người và hệ thống. Tuy nhiên, một câu lệnh nào đó phải thực hiện bao nhiêu lần thì nó phải được dịch bấy nhiêu lần.

Các ngôn ngữ khai thác hệ quản trị cơ sở dữ liệu, ngôn ngữ đối thoại với hệ điều hành,... đều sử dụng trình thông dịch.

b) Biên dịch

Biên dịch (compiler) được thực hiện qua hai bước:

① Duyệt, kiểm tra, phát hiện lỗi, kiểm tra tính đúng đắn của các câu lệnh trong chương trình nguồn;

② Dịch toàn bộ chương trình nguồn thành một chương trình đích có thể thực hiện trên máy và có thể lưu trữ để sử dụng lại khi cần thiết.

Như vậy, trong thông dịch, không có chương trình đích để lưu trữ, trong biên dịch cả chương trình nguồn và chương trình đích có thể lưu trữ lại để sử dụng về sau.

Thông thường, cùng với chương trình dịch còn có một số dịch vụ liên quan như biên soạn, lưu trữ, tìm kiếm, cho biết các kết quả trung gian,... Toàn bộ các dịch vụ trên tạo thành một môi trường làm việc trên một ngôn ngữ lập trình cụ thể. Ví dụ, Turbo Pascal 7.0, Free Pascal 1.2, Visual Pascal 2.1,... trên ngôn ngữ Pascal, Turbo C++, Visual C++,... trên ngôn ngữ C++.

Các môi trường lập trình khác nhau ở những dịch vụ mà nó cung cấp, đặc biệt là các dịch vụ nâng cấp, tăng cường các khả năng mới cho ngôn ngữ lập trình.

BÀI 2: CÁC THÀNH PHẦN CỦA NGÔN NGỮ LẬP TRÌNH

1. Các thành phần cơ bản

Mỗi ngôn ngữ lập trình thường có ba thành phần cơ bản là *bảng chữ cái*, *cú pháp* và *ngữ nghĩa.*

a) Bảng chữ cái là tập các kí tự được dùng để viết chương trình. Không được phép dùng bất kì kí tự nào ngoài các kí tự quy định trong bảng chữ cái.

Trong C++, bảng chữ cái bao gồm các kí tự:

- Các chữ cái thường và các chữ cái in hoa của bảng chữ cái tiếng Anh:

a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- 10 chữ số thập phân Ả Rập: 0 1 2 3 4 5 6 7 8 9
- Các kí tự đặc biệt:

+	-	*	/	=		<	>	[	]	.	"	, (dấu phẩy)
;	#	^	$	@	&	(	)	{	}	:	\	' (dấu nháy)
dấu cách (mã ASCII 32)								!				_ (dấu gạch dưới)

b) Cú pháp là bộ quy tắc để viết chương trình. Dựa vào chúng, người lập trình và chương trình dịch biết được tổ hợp nào của các kí tự trong bảng chữ cái là hợp lệ và tổ hợp nào là không hợp lệ. Nhờ đó, có thể mô tả chính xác thuật toán để máy thực hiện.

c) Ngữ nghĩa xác định ý nghĩa thao tác cần phải thực hiện, ứng với tổ hợp kí tự dựa vào ngữ cảnh của nó.

Ví dụ

Phần lớn các ngôn ngữ lập trình đều sử dụng dấu cộng (+) để chỉ phép cộng. Xét các biểu thức:

$$A + B \qquad (1)$$

$$I + J \qquad (2)$$

Giả thiết A, B là các đại lượng nhận giá trị thực và I, J là các đại lượng nhận giá trị nguyên. Khi đó dấu "+" trong biểu thức (1) được hiểu là cộng hai số thực, dấu "+" trong biểu thức (2) được hiểu là cộng hai số nguyên. Như vậy, ngữ nghĩa dấu "+" trong hai ngữ cảnh khác nhau là khác nhau.

Tóm lại, cú pháp cho biết cách viết một chương trình hợp lệ, còn ngữ nghĩa xác định ý nghĩa của các tổ hợp kí tự trong chương trình.

Các lỗi cú pháp được chương trình dịch phát hiện và thông báo cho người lập trình biết. Chỉ có các chương trình không còn lỗi cú pháp mới có thể được dịch sang ngôn ngữ máy.

Các lỗi ngữ nghĩa khó phát hiện hơn. Phần lớn các lỗi ngữ nghĩa chỉ được phát hiện khi thực hiện chương trình trên dữ liệu cụ thể.

2. Một số khái niệm

a. Tên

Mọi đối tượng trong chương trình đều phải được đặt tên theo quy tắc của ngôn ngữ lập trình và từng chương trình dịch cụ thể.

Trong C++ tên là một dãy liên tiếp *không quá 255 kí tự bao gồm chữ số, chữ cái hoặc dấu gạch dưới và bắt đầu bằng chữ cái hoặc dấu gạch dưới*. Trong chương trình dịch C++, tên có phân biệt chữ hoa, chữ thường.

Nhiều ngôn ngữ lập trình, trong đó có C++, phân biệt ba loại tên:

- Tên dành riêng;
- Tên chuẩn;
- Tên do người lập trình đặt.

Tên dành riêng

Một số tên được ngôn ngữ lập trình quy định dùng với *ý nghĩa xác định*, người lập trình không được sử dụng với ý nghĩa khác. Những tên này được gọi là *tên dành riêng* (còn được gọi là *từ khoá*).

Một số tên dành riêng trong C++: $main, include, if, while, void$.

Tên chuẩn

Một số tên được ngôn ngữ lập trình dùng với *ý nghĩa nào đó*. Những tên này được gọi là *tên chuẩn*. Tuy nhiên, người lập trình có thể khai báo và dùng chúng với ý nghĩa và mục đích khác.

Ý nghĩa của các tên chuẩn được quy định trong các *thư viện* của ngôn ngữ lập trình.

Ví dụ một số tên chuẩn trong C++: $cin\ \ cout\ \ getchar$

Tên do người lập trình đặt

Tên do người lập trình đặt được dùng với ý nghĩa riêng, xác định bằng cách khai báo trước khi sử dụng. Các tên này không được trùng với tên dành riêng.

Ví dụ tên do người lập trình đặt:

A1
DELTA
CT_Vidu

b. Hằng và biến

Hằng *là các đại lượng có giá trị không thay đổi trong quá trình thực hiện chương trình.*

Trong các ngôn ngữ lập trình thường có các hằng số học, hằng lôgic, hằng xâu.

- Hằng số học là các số nguyên hay số thực (dấu phẩy tĩnh hoặc dấu phẩy động), có dấu hoặc không dấu.
- Hằng lôgic là giá trị *đúng* hoặc *sai* tương ứng với *true* hoặc *false*.
- Hằng xâu là chuỗi kí tự trong bảng chữ cái. Khi viết, chuỗi kí tự này được đặt trong cặp dấu nháy kép.

Ví dụ

- Hằng số học: 2 0 -5 +18

1.5 -22.36 +3.14159 0.5

-2.236E01 1.0E-6

- Hằng lôgic: $true$ $false$
- Hằng xâu: $"Informatic"$ $"TIN\ HOC"$

Biến

Biến là đại lượng được đặt tên, dùng để lưu trữ giá trị và giá trị có thể được thay đổi trong quá trình thực hiện chương trình.

Tuỳ theo cách lưu trữ và xử lí, C++ phân biệt nhiều loại biến. Các biến dùng trong chương trình đều phải khai báo. Việc khai báo biến sẽ được trình bày ở các phần sau.

c. Chú thích

Có thể đặt các đoạn chú thích trong chương trình nguồn. Các chú thích này giúp cho người đọc chương trình nhận biết ngữ nghĩa của chương trình đó dễ hơn. Chú thích không ảnh hưởng đến nội dung chương trình nguồn và được chương trình dịch bỏ qua.

Trong Pascal các đoạn chú thích đặt giữa cặp dấu /* và */; hoặc sử dụng cặp dấu // đặt phía trước chú thích.

BÀI 3: CẤU TRÚC CHƯƠNG TRÌNH

Một chưng trình viết bằng C++ có cấu trúc gồm 2 phần:

Phần 1: Phần khai báo

Phần 2: Phân thân

Trong phần khai báo: ta (có thể) khai báo các thông tin:

+ Thư viện: Trong C++ có nhiều thư viện được chương trình cung cấp sẵn, thư viên chứa sẵn các hàm để người dùng sử dụng nhằm đơn giản hóa việc lập trình. Để khai báo thư viện ta dùng cú pháp:

$$\#include\ <\text{tên thư viện}>;$$

Trong đó $\#include$ là từ khóa; $<\text{tên thư viện}>$ thường là một số thư viện như $iostream;\ algorithm;\ string;$ ...Mỗi thư viện sẽ có một số hàm với chức năng khác nhau. Tuy nhiên để đơn giản ta chỉ sử dụng thư viện "tổng" là $<bits/stdc++.h>$

+ namespase: thường sử dụng là std

+ Hằng: các hằng được sử dụng trong chương trình

+ Biến: các biến được sử dụng trong chương trình

+ Hàm: các hàm được sử dụng trong chương trình

Phần thân: là hàm main() có cấu trúc

```
int main()
{
 [< dãy câu lệnh >]
}
```

Trong đó $[<\text{dãy câu lệnh}>]$ là khác nhau đối với mỗi chương trình. Trong $[<\text{dãy câu lệnh}>]$ có thể có ít nhất một câu lệnh $return\ 0;$

Để hiểu rõ hơn cấu trúc của một chương trình trong C++ ta xét ví dụ đơn giản sau:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int a,b;
4  int main()
5  {
6      cin>>a>>b;
7      cout<<a+b;
8      return 0;
9  }
```

+ ***Dòng 1:*** Khai báo thư viện.

+ ***Dòng 2:*** Khai báo sử dụng $namespace\ std$. Trong phạm vi kiến thức chương trình, ta sẽ không tìm hiểu về dòng này mà chỉ xem nó là một phần bắt buộc phải có khi viết chương trình.

+ ***Dòng 3:*** Khai báo hai biến a, b. Việc khai báo biến sẽ được nói kỹ hơn ở phần sau.

+ ***Dòng 4-9:*** thể hiện hàm $main()$, trong đó dòng 4 xem như là bắt buộc phải có trong tất cả chương trình, dấu { ở dòng 5 cho biết bắt đầu và dấu } ở dòng 9 cho biết kết thúc hàm $main()$. Các câu lệnh ở dòng 6, 7 sẽ được giải thích ở phần sau

BÀI 4: MỘT SỐ KIỂU DỮ LIỆU CHUẨN

1. Kiểu số nguyên

Kiểu dữ liệu	Kích thước	Miền giá trị
$short$	2 byte	-2^{15} đến $2^{15}-1$
int	4 byte	-2^{31} đến $2^{31}-1$
$long\ long$	8 byte	-2^{63} đến $2^{63}-1$

2. Kiểu số thực

Kiểu dữ liệu	Kích thước	Miền giá trị
$float$	4 byte	Số âm: -1.17549e-38 đến -3.40282e+38 Số dương: 1.17549e-38 đến 3.40282e+38
$double$	8 byte	Số âm: -2.22507e-308 đến - 1.79769e+308 Số dương: 2.22507e-308 đến 1.79769e+308

3. Kiểu ký tự

Kiểu ký tự trong C++ là kiểu dữ liệu chỉ lưu trữ được 1 ký tự trong bảng mã ASCII, ký tự này có thể là một chữ cái $a, b, c, \ldots x, y, z$, một chữ số $0, 1, 2, \ldots, 9$, một phép toán $+, -, *, /$ hay một ký tự bất kỳ khác $!, \&, \ldots$.

Biến kiểu ký tự được khai báo bằng từ khóa $char$ ($char$ là viết tắt của *character*).

Bản chất của kiểu ký tự là một số nguyên, ta có thể lấy giá trị số nguyên (mã ASCII trong hệ thập phân) của ký tự c bằng cách viết $(int)c$.

4. Kiểu logic

Kiểu logic hay còn gọi là kiểu luận lý được khai báo bằng từ khóa $bool$ chỉ nhận hai giá trị là $true$ hoặc $false$

Bản chất của kiểu $bool$ là kiểu số nguyên trong đó $true$ tương ứng với 1 còn $false$ tương ứng với 0.

Bài 5: KHAI BÁO BIẾN

Trong chương trình, biến cần được khai báo trước khi sử dụng. Tên biến dùng để xác lập quan hệ giữa biến với địa chỉ bộ nhớ nơi lưu trữ giá trị của biến.

Trong C++, biến có thể được khai báo ở nhiều vị trí khác nhau như: trong phần khai báo, trong một hàm, trong câu lệnh…Với mỗi vị trí khai báo khác nhau sẽ có những sự khác nhau nhất định. Trong nội dung bài này chỉ đề cập đến việc khai báo một biến đơn trong phần khai báo và trong hàm ***main***.

Để khai báo biến trong C++ ta dùng cú pháp:

<kiểu dữ liệu> <danh sách biến>;

Trong đó:

<kiểu dữ liệu> là một trong các kiểu dữ liệu trong C++ hoặc các kiểu dữ liệu người dùng tự định nghĩa.

<danh sách biến> là tên các biến do người dùng tự đặt. Nếu có nhiều hơn hai biến được khai báo thì giữa hai tên biến liên tiếp phải có dấu phẩy.

Ví dụ:

+ Khai báo biến a nhận giá trị nguyên trong đoạn $[-100,100]$:

$$short\ a;$$

+ Khai báo hai biến x, y nhận giá trị số thực:

$$double\ x, y;$$

Có thể gán giá trị cho biến ngay lúc khai báo. Ví dụ, để khai báo biến a có kiểu int rồi gán giá trị 10 cho a:

$$int\ a = 10;$$

Phạm vi sử dụng của biến:

+ Nếu biến được khai báo trong *<phần khai báo>* thì biến được sử dụng trong mọi ví trí của chương trình chính và chương trình con.

+ Nếu biến được khai báo trong một khối lệnh (trong { và }) thì biến chỉ được sử dụng trong khối lệnh đó tính từ vị trí khai báo.

Ví dụ:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int a=1;
    if(a>0) {
        int b=1;
      }
    b=2;
    return 0;
}
```

Với chương trình bên bạn sẽ nhận được thông báo lỗi ở câu lệnh $b = 2;$

Ta có thể thấy biến b đã được khai báo trước đó tuy nhiên phạm vi sử dụng của biến b chỉ nằm trong khối lệnh mà nó được khai báo, khi sử dụng b ở ngoài thì chương trình dịch sẽ xem như b chưa được khai báo.

Một số lưu ý khi khai báo biến:

1. Cần đặt tên biến sao cho gợi đến ý nghĩa của biến đó, điều này rất có lợi cho việc đọc, hiểu và sửa chương trình khi cần thiết.

2. Không nên đặt tên biến quá ngắn hay quá dài, dễ mắc lỗi khi viết nhiều lần tên biến.

3. Việc xác định kiểu dữ liệu cho biến cần phải lưu ý đến phạm vi giá trị của nó.

BÀI 6.1: PHÉP TOÁN

Trong ngôn ngữ lập trình C++ cũng như trong toán học và một số ngôn ngữ lập trình bậc cao khác đều các các phép toán để thực hiện các thao tác của thuật toán. Dưới đây ta xét một số phép toán trong C++:

1. Phép toán số học

Phép toán **cộng** ($+$), ví dụ $16 + 3$ cho kết quả là *19*

Phép toán **trừ** ($-$), ví dụ $16 - 3$ cho kết quả là *13*

Phép toán **nhân** ($*$), ví dụ $16 * 3$ cho kết quả là *48*

Phép toán **chia** ($/$), ví dụ $16/3$

Nếu a hoặc b là hai số ***có kiểu thực*** thì kết quả của a/b là số thực, ví dụ $16/3$ cho kết quả là 5.33333

Nếu a và b là hai số ***có kiểu nguyên*** thì kết quả của a/b là ***phần nguyên***, ví dụ $16/3$ cho kết quả là *5*.

Phép toán **chia lấy dư** ($\%$), ví dụ $16 \% 3$ cho kết quả là 1.

2. Phép toán so sánh

Phép toán nhỏ hơn ($<$), nhỏ hơn hoặc bằng ($<=$),

lớn hơn ($>$), lớn hơn hoặc bằng ($>=$),

bằng ($==$), khác ($!=$)

Lưu ý: Kết quả của các phép toán so sánh cho giá trị logic, ví dụ:

$1 < 2$ cho kết quả là $true$

$100 == 200$ cho kết quả là $false$

$43 != 54$ cho kết quả là $true$

3. Phép toán logic

Phép toán phủ định ($!$), ví dụ:

$!\,true$ cho kết quả là $false$

$!\,false$ cho kết quả là $true$

Phép toán và (and), ví dụ:	Phép toán hoặc (or), ví dụ:
$false$ ***and*** $false$ cho kết quả $false$ $false$ ***and*** $true$ cho kết quả $false$ $true$ ***and*** $false$ cho kết quả $false$ $true$ ***and*** $true$ cho kết quả ***true***	$false$ ***or*** $false$ cho kết quả $false$ $false$ ***or*** $true$ cho kết quả ***true*** $true$ ***or*** $false$ cho kết quả ***true*** $true$ ***or*** $true$ cho kết quả ***true***

4. Phép toán tự tăng, tự giảm

Phép toán tự tăng ($++$) được đặt trước hoặc sau một biến dùng để tăng giá trị của biến lên một đơn vị. Ví dụ nếu ban đầu a có giá trị bằng 10 thì $++a$ hoặc $a++$ sẽ **tăng** giá trị của biến a lên 11.

Phép toán tự giảm ($--$) được đặt trước hoặc sau một biến dùng để giảm giá trị của biến đó xuông một đơn vị. Ví dụ nếu ban đầu a có giá trị bằng 10 thì $--a$ hoặc $a--$ sẽ **giảm** giá trị của biến a xuống 9.

BÀI 6.2: BIỂU THỨC

1. Biểu thức số học

Biểu thức số học là một biến kiểu số hoặc một hằng kiểu số hoặc các biến kiểu số, hằng kiểu số và các hàm số học liên kết với nhau bởi một số hữu hạn các phép toán số học, các dấu ngoặc tròn (và) tạo thành một biểu thức có dạng tương tự như cách viết trong toán học với những quy tắc sau:

- Chỉ dùng cặp ngoặc tròn để xác định trình tự thực hiện các phép toán trong trường hợp cần thiết;
- Viết lần lượt từ trái qua phải;
- Không được bỏ qua dấu nhân ($*$) trong tích

Các phép toán được thực hiện theo thứ tự:

- Thực hiện các phép toán trong ngoặc trước;
- Trong dãy các phép toán không chứa ngoặc thì thực hiện từ trái sang phải, theo thứ tự các phép toán nhân ($*$), chia ($/$), chia lấy dư ($\%$) được thực hiện trước và các phép toán cộng ($+$), trừ ($-$) được thực hiện sau.

Một số hàm số học thường dùng:

Hàm	Trong toán học	Trong C++
Trị tuyệt đối	$\|x\|$	abs(x)
Căn bậc hai	$\sqrt{x}$	sqrt(x)
Lũy thừa	x^y	pow(x, y)

Ví dụ 1: Giả sử các biến trong trường hợp sau là **số thực**, các biểu thức trong toán học được chuyển sang biểu thức trong C++

Biểu thức trong toán học	Biểu thức trong C++
$5a + 6b$	$5 * a + 6 * b$
$\frac{xy}{z}$	$x * y/z$
$\frac{1}{ab}$	$1/(a * b)$
$\|a - b\|$	$abs(a - b)$
$\frac{x+y}{x-\frac{1}{2}} - \frac{x-z}{xy}$	$(x + y)/(x - 1/2) - (x - y)/(x * y)$
$\frac{-b-\sqrt{b^2-4ac}}{2a}$	$(-b - sqrt(b * b - 4 * a * c))/(2 * a)$

Ví dụ 2: Giả sử các biến trong trường hợp sau là **số nguyên**, các biểu thức trong toán học được chuyển sang biểu thức trong C++

Biểu thức trong toán học	Biểu thức trong C++
$\frac{xy}{z}$	(double)x * y/z
$\frac{1}{ab}$	(double)1/(a * b)

2. Biểu thức so sánh

Hai biểu thức cùng kiểu liên kết với nhau bởi phép toán so sánh cho ta một biểu thức so sánh.

Biểu thức so sánh có dạng:

<biểu thức 1> <phép toán so sánh> <biểu thức 2>

Trong đó *<biểu thức 1>* và *<biểu thức 2>* là hai biểu thức cùng kiểu.

Biểu thức so sánh được thực hiện theo trình tự:

1. Tính giá trị của biểu thức
2. Thực hiện phép toán so sánh

Giá trị của biểu thức so sánh là giá trị logic (true hoặc false)

Ví dụ 1: Điều kiện để số nguyên dương a có giá trị nhỏ hơn hoặc bằng 100:

a <= 100

Ví dụ 2: Điều kiện để điểm M có tọa độ (x, y) thuộc hình tròn tâm I(a, b), bán kính R là:

(x − a) * (x − a) + (y − b) * (y − b) <= R * R

3. Biểu thức logic

Biểu thức logic đơn giản là biến logic hoặc hằng logic.

Biểu thức logic là biểu thức logic đơn giản, các biểu thức quan hệ liên kết với nhau bởi phép toán logic.

Giá trị của biểu thức logic là giá trị logic.

Ví dụ 1: Điều kiện $5 \le x \le 10$ được chuyển qua biểu thức logic trong C++:

5 <= x and x <= 10

Ví dụ 2: Giả sử m và n là hai biến nguyên, để xác định m và n đồng thời chia hết cho 3 hay không được viết bằng biểu thức logic trong C++:

(m % 3 == 0 and n % 3 == 0) or (m % 3 ! = 0 and n % 3 ! = 0)

Lưu ý: Dấu ngoặc đơn dùng để xác định thứ thự ưu tiên của phép toán, trong ví dụ 1 không cần dùng đến vì thứ tự ưu tiên của phép toán and thấp hơn thứ tự ưu tiên của phép toán <=. Hay nói cách khác phép toán <= được thực hiện trước phép toán and

BÀI 6.3: CÂU LỆNH GÁN

Trong ngôn ngữ lập trình C++ ta có thể thay đổi giá trị của một biến bằng cách sử dụng toán tử tự tăng/giảm hoặc gán giá trị cua một biểu thức cho biến.

Cú pháp của câu lệnh gán:

<tên biến> = <biểu thức>;

Ví dụ:

```
b = a + 12;
a = a + b;
```

Lưu ý: *<tên biến>* và *<biểu thức>* phải cùng kiểu dữ liệu

Mở rộng:

Với câu lệnh gán `a = a + b;` ta có thể viết lại gọn hơn `a+= b;`

Tương tự:

a = a − *(biểu thức);*	→	a−= *(biểu thức);*
a = a + *(biểu thức);*	→	a+= *(biểu thức);*
a = a / *(biểu thức);*	→	a/= *(biểu thức);*
a = a ∗ *(biểu thức);*	→	a ∗= *(biểu thức);*
a = a % *(biểu thức);*	→	a% = *(biểu thức);*

Bài 7: NHẬP, XUẤT DỮ LIỆU

Khi lập trình để giải một bài toán trong Tin học, ta cần biết Input, Output của bài toán là gì, hay nói cách khác ta cần biết nhập vào dữ liệu nào, chương trình xuất ra dữ liệu nào.

Bài toán ví dụ: Viết chương trình nhập từ bàn phím hai số nguyên dương n, m rồi in ra màn hình tổng bình phương của hai số.

Input và output của bài toán là:

+ Input: n, m

+ Output: $(n + m)^2$

Như vậy khi viết chương trình, người lập trình cần phải có lệnh để nhập hai số nguyên dương n, m từ bàn phím và in kết quả $(n + m)^2$ ra màn hình.

C++ cũng như các ngôn ngữ lập trình khác đều có các câu lệnh dùng để nhập và xuất dữ liệu.

1. Xuất dữ liệu ra màn hình

Để xuất dữ liệu ra màn hình ta dùng cú pháp:

cout <<biểu thức 1<<biểu thức 2<<….<<biểu thức n;

Trong đó:

cout là từ khóa

Biểu thức 1, biểu thức 2, …., biểu thức n là các biểu thức trong C++

Ví dụ 1: Để in giá trị số nguyên dương n ra màn hình, ta viết:

$$cout << n;$$

Ví dụ 2: Để in giá trị hai số nguyên dương n, m ra màn hình, giữa hai số cách nhau một ký tự trắng, ta viết:

$$cout << n << \ \ << m;$$

Ví dụ 3: Để in giá trị hai số nguyên dương n, m ra màn hình trên hai dòng, ta viết:

$$cout << n << endl << m;$$

Trong đó: *endl* là ký tự xuống dòng, có thể thay thế *endl* bằng $'\backslash n'$

$$cout << n << '\backslash n' << m;$$

Cách khác: Dùng hai lệnh *cout*

$$cout << n << endl;$$
$$cout << m;$$

2. Nhập dữ liệu từ bàn phím

Để nhập dữ liệu từ bàn phím, ta dùng cú pháp

cin >>tên biến 1>>tên biến 2>>….>>tên biến n;

Trong đó:

+ *cin* là từ khóa;

+ *tên biến 1, tên biến 2,…,tên biến n* là các biến được người dùng khai báo từ trước.

Ví dụ 1: Để nhập giá trị cho biến n, ta viết:

$$cin >> n;$$

Ví dụ 2: Để nhập giá trị cho hai biến n, m, ta viết:

$$cin >> n >> m;$$

Lưu ý: Khi thực hiện đến câu lệnh *cin*, chương trình sẽ dừng lại để người dùng nhập giá trị cho các biến, giữa các giá trị cách nhau bằng **dấu cách (ký tự trắng)** hoặc dấu **enter (dấu xuống dòng)**.

Ví dụ: khi gặp câu lệnh $cin >> n >> m >> k;$ chương trình sẽ dừng lại và người dùng có thể nhập giá trị 10, 4, 7 lần lượt cho ba biến n, m, k bằng cách nhấn bàn phím:

10 dấu cách 4 dấu cách 7 enter

Hoặc:

10 enter 4 enter 7 enter

3. Một số ví dụ

Ví dụ 1: Viết chương trình nhập từ bàn phím số nguyên dương a. Hãy tăng giá trị của a lên 1 đơn vị rồi in giá trị của a ra màn hình.

Chương trình:

```
#include <bits/stdc++.h>
using namespace std;
int a;
int main()
{
      cin>>a;
      a++;
      cout<<a;
      return 0;
}
```

Ví dụ 2: Viết chương trình nhập từ bàn phím hai số nguyên dương n, m $(1 \le n, m \le 100)$, in ra màn hình các giá trị $n + m; n - m; n * m;$ mỗi giá trị trên 1 dòng.

Chương trình:

```
#include <bits/stdc++.h>
using namespace std;
int n,m;
int main()
{
      cin>>n>>m;
      cout<<n+m<<endl;
      cout<<n-m<<endl;
      cout<<n*m;
      return 0;
}
```

Ví dụ 3: Viết chương trình nhập từ bàn phím một ký tự chữ cái rồi in ra màn hình giá trị mã ASCII của ký tự đó.

Chương trình:

```
#include <bits/stdc++.h>
```

```
using namespace std;
char c;
int main()
{
      cin>>c;
      cout<<(int)c;
      return 0;
}
```

Ví dụ 4: Viết chương trình nhập từ bàn phím ba số nguyên a, b, c rồi in ra trung bình cộng của ba số.

Chương trình:

```
#include <bits/stdc++.h>
using namespace std;
int a, b, c;
int main()
{
      cin>>a>>b>>c;
      double tbc=(double)(a+b+c)/3;
      cout<<tbc;
      return 0;
}
```

CÂU HỎI TRẮC NGHIỆM – CƠ BẢN

Câu 1. Kiểu dữ liệu số nguyên là:

A. $int, short, long\ long$ **B.** $int, float, long\ long$

C. $float, long\ long, double$ **D.** $int, long\ long, double$

Câu 2. $double$ thuộc kiểu dữ liệu?

A. Số nguyên **B.** Số thực **C.** Ký tự **D.** Logic

Câu 3. Chương trình cấp phát bao nhiêu bộ nhớ cho một biến được khai báo kiểu int?

A. 1 byte **B.** 2 byte C. 3 byte **D.** 4 byte

Câu 4. Để lấy mã ASCII của ký tự c ta viết

A. $ascii(c)$ **B.** $(int)c$ **C.** $c - 65$ **D.** $c + 65$

Giải thích: Phương án B còn gọi là ép kiểu, nghĩa là chuyển đổi kiểu dữ liệu của giá trị trong biến sang kiểu dữ liệu khác, ơ đây ta chuyển đổi kiểu dữ liệu của giá trị trong c từ ký tự sang số. Lưu ý rằng trong C++ bản chất của ký tự cũng là số, đó là mã ASCII của ký tự trong hệ thập phân.

Câu 5. Cú pháp khai báo biến là:

A. *<kiểu dữ liệu> <danh sách biến>;* **B.** *<danh sách biến> <kiểu dữ liệu>;*

C. $int\ a;$ **D.** *<kiểu dữ liệu> a,b,c;*

Câu 6. Để khai báo một biến nhận giá trị số nguyên trong đoạn $[-10^6, 10^6]$ ta viết:

A. $int\ a;$ **B.** $double\ a;$ **C.** $a\ int;$ **D.** *a double;*

Giải thích: Câu A và B đều có thể là đáp án, tuy nhiên đây là đoạn số nguyên nên ta dùng kiểu int

Câu 7. Dựa vào câu lệnh gán $a = (double)x;$ hãy cho biết cần phải khai báo biến a có kiểu dữ liệu nào là hợp lý nhất?

A. int **B.** $double$ **C.** $float$ **D.** $long\ long$

Giải thích: a phải có kiểu số thực $double$ nên đáp án đúng là B

Câu 8. Trong các khai báo dưới đây, khai báo nào hợp lệ

A. $int\ a\ b;$ **B.** $int\ a, b;$ **C.** $int\ a = 10; b = 20;$ **D.** $int\ a + b;$

Câu 9. Để khai báo và gán giá trị bằng 10 cho biến a ta viết

A. $int\ a, a = 10;$ **B.** $int\ a = 10;$ **C.** $int\ a\ (10);$ **D.** $int\ 10 = a;$

Câu 10. Hãy cho biết kết qủa của biểu thức $20/3$

A. 6 **B.** 6.(6) **C.** 2 **D.** 6.66666

Giải thích: Phép chia a/b trong C++ trả về giá trị là phần nguyên nếu a và b là số nguyên, do vậy kết quả của $20/3$ là 6

Câu 11. Hãy cho biết kết quả của biểu thức $20\%3$

A. 6 **B.** 6.(6) **C.** 2 **D.** 6.66666

Giải thích: Phép chia $a\%b$ trong C++ trả về giá trị là phần dư nếu a và b là số nguyên, do vậy kết quả của $20\%3$ là 2

Câu 12. Hãy cho biết kết quả của biểu thức $(double)20/3$

A. 6 **B.** 6.(6) **C.** 2 **D.** 6.66666

Giải thích: Phép chia a/b trong C++ trả về giá trị là phần nguyên nếu a và b là số nguyên, trả về số thực nếu tử hoặc mẫu số là số thực. Ở đây tử số đã được ép kiểu sang số thực nên kết quả là 6.66666

Câu 13. Phép toán so sánh bằng được ký hiệu

A. $=$ **B.** $equal$ **C.** $!=$ **D.** $==$

Câu 14. Phép toán so sánh khác được ký hiệu

A. $other$ **B.** $==$ **C.** $!=$ **D.** $<>$

Câu 15. Hãy cho biết kết quả của biểu thức $1\ or\ 0\ and\ 1$

A. 1 **B.** 0 **C.** $false$ **D.** Biểu thức không hợp lệ

Câu 16. Để tăng giá trị của biến a lên 1 đơn vị, ta viết:

A. $a++;$ **B.** $a = a + 1;$ **C.** $a+= 1;$ **D.** Tất cả đều đúng

Câu 17. Để giảm giá trị của biến a xuống 1 đơn vị, ta viết:

A. $a--;$ **B.** $--a;$ **C.** $a-= 1;$ **D.** Tất cả đều đúng

Câu 18. Hàm $sqrt(x)$ có ý nghĩa:

A. $|x|$ **B.** x^2 **C.** $\sqrt{x}$ **D.** Tất cả đều sai

Câu 19. Giả sử a, b là hai số nguyên, biểu thức toán học $\frac{1}{ab}$ được biểu diễn

A. $1/a/b$ **B.** $1/(a*b)$ **C.** $(double)1/a/b$ **D.** $1/a*b$

Câu 20. Giả sử a, b là hai số thực, biểu thức toán học $\frac{1}{ab}$ được biểu diễn

A. $\frac{1}{a*b}$ **B.** $1/(a*b)$ **C.** $(double)1/a*b$ **D.** $1/a*b$

Câu 21. Hãy cho biết kết quả của biểu thức $1/a$ trong trường hợp a là một số nguyên và $a > 1$

A. 1 **B.** 0 **C.** 0.01 **D.** 0.0001

Câu 22. Điều kiện để điểm M có tọa độ (x, y) thuộc đường tròn tâm $O(a, b)$ bán kính r là

A. $MO \le r$

B. $(x-a)*(x-a)+(y-b)*(y-b) <= r*r$

C. $(x-a)(x-a)+(y-b)(y-b) <= r^2$

D. $(x-a)*(x-a)+(y-b)*(y-b) \le r*r$

Câu 23. Điều kiện để kiểm tra số nguyên dương a là một số chẵn:

A. $a\%2 == 0$ **B.** $a\%2 = 0$ **C.** $a/2 == 0$ **D.** $a/2 = 0$

Câu 24. Điều kiện $a \le b \le c$ được viết

A. $a <= b\ and\ b <= c$ **B.** $a <= b\ and\ a <= c$

C. $a <= b\ or\ b <= c$ **D.** $a <= b\ or\ a <= c$

Câu 25. Trong trường hợp nào biểu thức $(int)sqrt(a)*(int)sqrt(a) == a$ cho giá trị 1

A. Trong mọi trường hợp **B.** a là số nguyên tố

C. a là số nguyên dương **D.** a là số chính phương

Câu 26. Biểu thức $(a > b\ or\ b > a)$ cho giá trị 0 trong trường hợp nào sau đây:

A. $a = 4$ và $b = 4$ **B.** $a = 5$ và $b = 4$

C. $a = 4$ và $b = 5$ **D.** Trong mọi trường hợp

Câu 27. Biểu thức $(a >= b\ and\ b >= a)$ cho giá trị 1 trong trường hợp nào sau đây:

A. $a = 4$ và $b = 4$ **B.** $a = 5$ và $b = 4$

C. $a = 4$ và $b = 5$ **D.** Trong mọi trường hợp

Câu 28. Cú pháp câu lệnh gán:

A. *<tên biến> = <biểu thức>;* **B.** *<tên biến> == <biểu thức>;*
C. *<biểu thức> == <tên biến>;* **D.** *<biểu thức> = <tên biến>;*

Câu 29. Để gán giá trị biểu thức $a + x$ cho biến a ta viết
A. $a += x;$ **B.** $a + +x;$ **C.** $a = x;$ **D.** $x = x + a;$

Câu 30. Cú pháp nhập dữ liệu từ bàn phím
A. *cin<<biến 1<<biến 2<<...<<biến n;*
B. *cout<<biến 1<<biến 2<<...<<biến n;*
C. *cin>>biểu thức 1>>biểu thức 2>>...>>biểu thức n;*
D. *cin>>biến 1>>biến 2>>...>>biến n;*

Câu 31. Cú pháp xuất dữ liệu màn hình
A. *cout<<biểu thức 1<<biểu thức 2<<...<<biểu thức n;*
B. *cout<<biến 1<<biến 2<<...<<biến n;*
C. *cout>>biểu thức 1>>biểu thức 2>>...>>biểu thức n;*
D. *cin>>biến 1>>biến 2>>...>>biến n;*

Câu 32. Để in giá trị hai biến a, b ra màn hình, mỗi số trên một dòng, ta viết
A. $cin >> a >> endl >> b;$ **B.** $cout >> a >> endl >> b;$
C. $cout << a << " " << b;$ **D.** $cout << a << endl << b;$

Câu 33. Để nhập giá trị hai biến a, b ta viết
A. $cin >> a >> b;$ **B.** $cin << a << b;$
C. $cin >> a, b;$ **D.** $cout << a << b;$

Câu 34. Hãy cho biết kết quả in ra màn hình khi thực hiện câu lệnh $cout << a * a;$ trong đó biến a có kiểu dữ liệu int và có giá trị 1000000
A. 1000000 **B.** 1000000000000
C. Có thể là một số nguyên âm **D.** Lỗi cú pháp

Câu 35. Hãy cho biết kết quả in ra màn hình sau khi thực hiện xong các câu lệnh

```
int a=123,s=0;
s+=a%10; a/=10;
s+=a%10; a/=10;
s+=a; cout<<s;
```

A. 6 **B.** 9 **C.** 0 **D.** Kết quả khác

Câu 36. Hãy cho biết kết quả in ra màn hình sau khi thực hiện xong câu lệnh

$$cout << (1 == 1);$$

A. 1 **B.** true **C.** 0 **D.** Kết quả khác

Bài tập cơ bản

Xem tại: http://lqdonkh.xyz/viewtag/basic

Bài 8: CÂU LỆNH RẼ NHÁNH

1. Rẽ nhánh là gì?

Trong thực tế chúng ta thường bắt gặp một hành động nào đó chỉ được thực hiện khi một điều kiện nào đó được thỏa mãn. Ví dụ:

Câu 1: **Nếu** *trời mưa* **thì** *Nam ở nhà học bài*

Câu 2: **Nếu** *trời không mưa* **thì** *Nam đi đá bóng*

Ta có các nhận xét như sau:

+ Trong câu 1: Hành động là *"ở nhà học bài"*; điều kiện là *"trời mưa"*. Hành động *"ở nhà học bài"* chỉ được thực hiện khi điều kiện *"trời mưa"* đúng.

+ Trong câu 2: Hành động là *"đi đá bóng"*; điều kiện là *"trời không mưa"*. Hành động *"đi đá bóng"* chỉ được thực hiện điều kiện *"trời không mưa"* đúng.

+ Điều kiện "*trời mưa"* ở câu 1 và điều kiện *"trời không mưa"* ở câu 2 là phủ định của nhau. Do vậy ta có thể viết lại cả hai câu trên như sau:

Câu 3: **Nếu** *trời mưa* **thì** *Nam ở nhà học bài* **ngược lại** *Nam đi đá bóng*

+ Trong câu 1 và câu 2 đều có cấu trúc chung:

Cấu trúc 1: **Nếu** *<điều kiện>* **thì** *<hành động>;*

+ Trong câu 3 có cấu trúc:

Cấu trúc 2: **Nếu** *<điều kiện>* **thì** *<hành động 1>* **ngược lại** *<hành động 2>*;

Hai cấu trúc ở trên được gọi là cấu trúc rẽ nhánh, đây là cấu trúc được sử dụng phổ biến trong ngôn ngữ lập trình.

Ta xét một ví dụ khác qua bài toán: hãy cho biết số nguyên dương n là số chẵn hay số lẻ

Sử dụng tiếng Việt ta có thể viết như sau:

Nếu *n chia hết cho* 2 **thì** *n là số chẵn;*

Nếu *n không chia hết cho 2* **thì** *n là số lẻ;*

Hoặc có thể viết:

Nếu *n chia hết cho 2* **thì** *n* là số chẵn; **ngược lại** *n là số lẻ;*

Sử dụng ngôn ngữ lập trình C++ ta có thể viết như sau:

$$\boldsymbol{if}(n\ \%\ 2\ ==\ 0)\ cout << \text{“}n\ la\ so\ chan\text{”};$$

$$\boldsymbol{if}(n\ \%\ 2\ !=\ 0)\ cout << \text{“}n\ la\ so\ le\text{”};$$

Ta cũng có thể viết lại:

$$\boldsymbol{if}(n\ \%\ 2\ ==\ 0)\ cout << \text{“}n\ la\ so\ chan\text{”};$$

$$\boldsymbol{else}\ cout << \text{“}n\ la\ so\ le\text{”};$$

Để hiểu rõ cấu trúc được viết trong các câu lệnh ở trên, ta sẽ tìm hiểu về cấu trúc câu lệnh if trong C++

2. Câu lệnh if

Cú pháp:

Dạng 1: **if***(<điều kiện>) <câu lệnh>;*

Dạng 2: **if** *(<điều kiện>) <câu lệnh 1>;* **else** *<câu lệnh 2>;*

Trong đó:

if, **else**: từ khóa

<điều kiện>: là biểu thức logic (trả về giá trị logic)

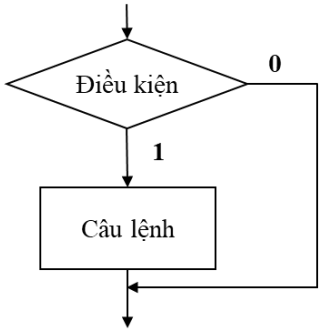
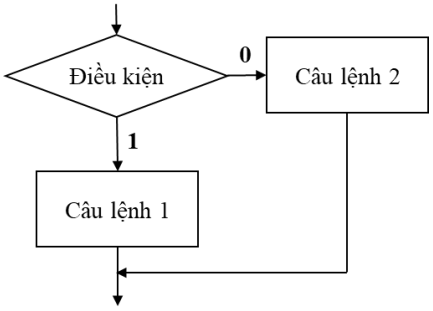
<câu lệnh>, <câu lệnh 1>, <câu lệnh 2>: **một** câu lệnh trong C++

Chú ý:

if, **else** là hai từ khóa được viết in thường

<điều kiện> luôn được đặt trong cặp ngoặc (và)

Hoạt động: được biểu diễn bằng sơ đồ khối như sau

Dạng 1	**Dạng 2**
	
<điều kiện> sẽ được tính giá trị trước; nếu *<điều kiện>* **đúng** thì *<câu lệnh>* được thực hiện, nếu *<điều kiện>* **sai** thì <câu lệnh> không được thực hiện.	*<điều kiện>* sẽ được tính giá trị trước; nếu *<điều kiện>* **đúng** thì chỉ có *<câu lệnh 1>* được thực hiện, nếu *<điều kiện>* **sai** thì chỉ có *<câu lệnh 2>* được thực hiện.

Ví dụ 1: Viết chương trình nhập từ bàn phím số nguyên n ($|n| \le 10^9$). Hãy cho biết n là số âm hay số dương hay số 0.

Cách 1: *Sử dụng 3 câu lệnh dạng 1*

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin>>n;
    if(n>0) cout<<"So duong";
    if(n<0) cout<<"So am";
    if(n==0) cout<<"So 0";
    return 0;
}
```

Cách 2: *Sử dụng câu lệnh dạng 2*

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin>>n;
    if(n>0) cout<<"So duong";
      else
        if(n<0) cout<<"So am";
           else cout<<"So 0";
      return 0;
}
```

Ví dụ 2: Viết chương trình nhập từ bàn phím hai số nguyên a, b ($|a|, |b| \le 10^9$); hãy in ra màn hình giá trị lớn nhất trong hai số.

Sau khi nhập hai số *a, b*, ý tưởng cơ bản để giải bài toán này là: thực hiện so sánh giá trị hai số, *nếu* $a > b$ *thì in ra màn hình số a, nếu* $a \leq b$ *thì in ra màn hình số b;* Từ ý tưởng này ta có thể sử dụng 2 câu lệnh *if* dạng 1 để giải quyết như sau:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
      int a,b;
      cin>>a>>b;
      if(a>b) cout<<a;
      if(a<=b) cout<<b;
      return 0;
}
```

Cũng có thể sử dụng 1 câu lệnh *if* dạng 2 nếu thực hiện theo ý tưởng: *nếu* $a > b$ *thì in ra màn hình số a, ngược lại in ra màn hình số b;* Chương trình:

```
using namespace std;
int main()
{
      int a,b;
      cin>>a>>b;
      if(a>b) cout<<a;
            else cout<<b;
      return 0;
}
```

Ngoài ra, còn có cách chỉ cần sử dụng 1 câu lệnh *if* dạng thiếu: *Lúc đầu xem như số a là giá trị lớn nhất (gán* maxx = a*) sau đó so sánh maxx với* b*, nếu* b > maxx *thì gán lại* maxx = b*. Cuối cùng in ra màn hình số* maxx. Chương trình:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int main()
4  {
5        int a,b;
6        cin>>a>>b;
7        int maxx=a;
8        if(b>maxx) maxx=b;
9        cout<<maxx;
10       return 0;
11 }
```

Nhận xét: sau khi thực hiện xong hai câu lệnh ở dòng 7 và 8 thì maxx chính là giá trị lớn nhất của hai số a, b; nếu trong đề bài yêu cầu tìm giá trị lớn nhất của ba số a, b, c thì lúc này chỉ cần so sánh giá trị của maxx và c, nếu c > maxx thì gán lại maxx = c; kết quả chính là *maxx*.

Chương trình sau đây sẽ in ra mà hình giá trị lớn nhất trong 3 số nguyên a, b, c bằng cách thay đổi chương trình ở ví dụ 2:

```
#include <bits/stdc++.h>
```

```
using namespace std;
int main()
{
      int a,b,c;
      cin>>a>>b>>c;
      int maxx=a;
      if(b>maxx) maxx=b;
      if(c>maxx) maxx=c;
      cout<<maxx;
      return 0;
}
```

3. Câu lệnh ghép

Xét bài toán: Viết chương trình nhập từ bàn phím hai số nguyên a, b $(|a|, |b| \leq 10^9)$; hãy tăng gấp đôi giá trị mỗi số a, b nếu $a > b$; ngược lại hoán đổi giá trị của chúng.

Theo đề bài ta có thể áp dụng câu lệnh rẽ nhánh dạng 2:

Nếu $a > b$ **thì** thực hiện hai câu lệnh $a = a * 2;$ và $b = b * 2$

ngược lại thực hiện 3 câu lệnh $int\ tg = a; a = b; b = tg;$

Vấn đề: kết quả chương trình sẽ bị sai nếu viết

```
if(a>b) a=a*2;b=b*2;
  else
    int tg=a;a=b;b=tg;
```

Lúc này chỉ có câu lệnh $a = a * 2;$ phụ thuộc vào điều kiện $a > b;$ hay nói cách khác câu lệnh if chỉ bao gồm đoạn $if(a > b)\ a = a * 2;$ còn câu lệnh $b = b * 2;$ sẽ được thực hiện sau khi câu lệnh if thực hiện xong.

Nếu muốn cả hai câu lệnh $a = a * 2; b = b * 2;$ đều phụ thuộc vào điều kiện $a > b$ thì ta phải ghép chúng thành 1 câu lệnh, để làm điều này ta sẽ dùng câu lệnh ghép:

```
{
      <câu lệnh 1>;
      <câu lệnh 2>;
      ...
      <câu lệnh n>;
}
```

Đoạn chương trình trên sẽ được viết thành:

```
if(a>b)
    {
      a=a*2;
      b=b*2;
    }
    else
      {
         int tg=a;
```

```
            a=b;
            b=tg;
        }
```

Chương trình đầy đủ ở bài tập Basic05

3. Bài tập

Xem tại: http://lqdonkh.xyz/contest/if

BÀI 9: CẤU TRÚC LẶP

1. Lặp

Có thể hiểu, *lặp là một hành động được thực hiện nhiều lần*. Chúng ta dễ dàng bắt gặp những hành động như vậy trong cuộc sống, ví dụ:

+ Đi học mỗi ngày. Với mỗi học sinh, hành động "*đi học*" được thực hiện hằng ngày, có thể ngày thứ 7 và chủ nhật vẫn đi học (thêm).

+ Chép một câu thơ nào đó 100 lần. Hành động "*chép một câu thơ*" được lặp đi lặp lại đến 100 lần.

Trong lập trình ta có thể bắt gặp các bài tập như:

+ In ra màn hình 100 số 1. Câu lệnh `cout << 1 << " ";` được thực hiện lặp lại 100 lần

+ In ra màn hình các số từ 1 đến 100. Câu lệnh `cout << i << " ";` được thực hiện 100 lần, tuy nhiên trong mỗi lần thì giá trị của *i* được thay đổi từ 1 đến 100.

Như vậy trong lập trình ta có thể hiểu *lặp là **một** câu lệnh được thực hiện nhiều lần.*

Trong tài liệu này sẽ trình bày hai câu lệnh `for` và `while` để giải quyết vấn đề: "*một câu lệnh được thực hiện nhiều lần*".

2. Câu lệnh `for`

Cú pháp:

***for**(<khởi tạo>;<điều kiện>;<thay đổi>) <câu lệnh>;*

Trong đó:

+ ***for*** là từ khóa

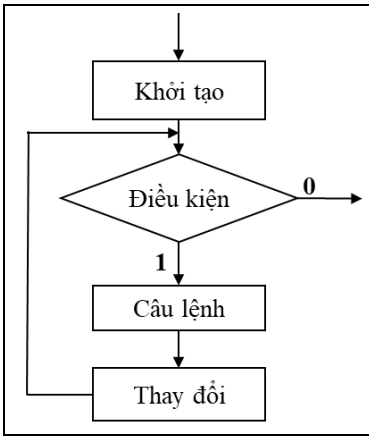
+ *<khởi tạo>:* thường là một câu lệnh gán, dùng để xác định giá trị ban đầu cho các biến.

+ *<điều kiện>:* là một biểu thức logic.

+ *<thay đổi>:* thường là câu lệnh gán, dùng để thay đổi giá trị của biến trong *<khởi tạo>*.

+ *<Câu lệnh>* là một câu lệnh trong C++

Hoạt động: Thể hiện qua sơ đồ khối dưới đây



+ Lệnh *<khởi tạo>* được thực hiện đầu tiên và chỉ duy nhất một lần.

+ Nếu *<điều kiện>* **sai** thì câu lệnh for kết thúc, các câu lệnh khác trong chương trình được thực hiện; nếu *<điều kiện>* ***đúng*** thì *<câu lệnh>* được thực hiện.

+ *<thay đổi>* sẽ được thực hiện ngay sau *<Câu lệnh>;* khi thực hiện *<thay đổi>* xong, chương trình sẽ quay lại để kiểm tra *<điều kiện>*.

Ví dụ:

Vd1: Để in ra màn hình 100 lần số 1, ta viết câu lệnh `for` như sau:

```
for( int i=1 ; i<=100 ;  i++ ) cout<<1<<" ";
```

Từ khóa	*Khởi tạo*	*Điều kiện*	*Thay đổi*	*Câu lệnh*

Vd2: Để in ra màn hình giá trị các số từ 1 đến 100, ta viết câu lệnh for như sau:

```
for(int i=1;i<=100;i++) cout<<i<<" ";
```

Trong ví dụ này chỉ khác ở ví dụ trước ở *<câu lệnh>*, thay vì in ra màn hình số 1 ở vị dụ này in ra giá trị của i, để ý rằng ban đầu i được khởi tạo bằng 1, sau khi in giá trị của i thì i được tăng lên 1, quá trình này lặp đi lặp lại đến khi điều kiện $i <= 100$ cho kết quả sai.

Vd3: Để in ra màn hình giá trị các số từ 100 về 1, ta viết câu lệnh for như sau:

```
for(int i=100;i>=1;i--) cout<<i<<" ";
```

Câu lệnh cout << i << " "; vẫn đực giữ nguyên như ở ***Vd2***, tuy nhiên các thành phần khác đều có sự thay đổi để đảm bảo giá trị i giảm từ 100 về 1.

+ *<Khởi tạo>:* ban đầu $i = 100$

+ *<Thay đổi>:* $i--$; sau mỗi lần lặp, giá trị của i sẽ giảm đi 1 đơn vị

+ *<điều kiện>:* $i >= 1$; Giá trị của i sẽ giảm dần nên đến một lúc nào đó $i < 1$ thì dừng thực hiện câu lệnh for.

Vd4: Để in ra màn hình giá trị các số **chẵn** từ 1 đến 100, ta viết câu lệnh for như sau:

Cách 1: Kết hợp với câu lệnh if; Xét các số từ 1 đến 100, nếu số nào chia hết cho 2 in ra màn hình

```
for(int i=1;i<=100;i++)
    if(i%2==0) cout<<i<<" ";
```

Cách 2: Chỉ cần xét các số chẵn:

+ *khởi tạo:* $i = 2$ + *điều kiện:* $i <= 100$ + *Thay đổi:* $i = i + 2$;	`for(int i=2;i<=100;i=i+2) cout<<i<<" ";`

Thông qua các ví dụ ở trên, ta thấy câu lệnh *for* trong C++ khá linh động, có thể thay đổi giá trị của các biến trong phần *<thay đổi>* một cách tùy ý để phù hợp với các bài toán khác nhau. Trong C++, câu lệnh *for* có thể thay thế hoàn toàn *while* mà ta sẽ xét ở phần tiếp theo, tuy nhiên trong phạm vi kiến thức của học sinh THPT ta chỉ cần hiểu và vận dụng được câu lệnh *for* để thay đổi giá trị **tăng** từ *a* đến *b* hoặc **giảm** từ *b* về *a*.

3. Câu lệnh *while*

Cú pháp:

while *(<điều kiện>) <câu lệnh>;*

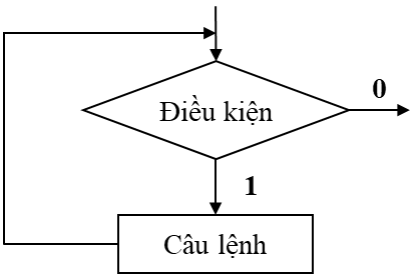
Trong đó:

+ *while*: từ khóa

+ *<điều kiện>*: là một biểu thức logic

+ *<câu lệnh>:* là một câu lệnh trong C++

Hoạt động: được thể hiện thông qua sơ đồ khối



+ *<điều kiện>* được kiểm tra trước, nếu ***đúng*** thì thực hiện *<câu lệnh>*, nếu **sai** thì kết thúc câu lệnh *while*.

+ sau khi *<câu lệnh>* được thực hiện, chương trình sẽ quay lại kiểm tra *<điều kiện>*

=> Trong khi *<điều kiện>* cho giá trị ***đúng*** thì thực hiện *<câu lệnh>*.

Ví dụ:

Vd1: Đoạn chương trình sau đây sẽ in ra màn hình các số nguyên từ 1 đến 100, giữa hai số có ít nhất một ký tự trắng.

```
int i=1;
while(i<=100){
  cout<<i<<" ";
  i++;
}
```

Vd2: Đoạn chương trình sau đây tính tổng các chữ số của số nguyên dương n.

Ý tưởng: Liên tục chia lấy dư cho 10 để lấy được hàng đơn vị của n và chia lấy nguyên cho 10 để xóa hàng đơn vị của n. Quá trình kết thúc khi $n = 0$

```
int s=0;
while(n>0){
  s+=n%10;
  n/=10;
}
cout<<s;
```

Vd3: Đoạn chương trình sau đây tìm ước chung lớn nhất của hai số nguyên dương n, m

Cách 1: Xét các số i từ 1 đến $min(n, m)$, nếu n và m cùng chia hết cho i thì lưu lại giá trị i làm ước chung. Do i tăng nên số được lưu lại cuối cùng là ước chung lớn nhất của hai số n, m.

```
int i=1,uc;
while(i<=min(n,m))
{
     if(n%i==0 and m%i==0) uc=i;
     i++;
}
cout<<uc;
```

Cách 2: Sử dụng thuật toán Euclid

Dạng phép trừ	***Dạng phép chia***

`while(n!=m)`
`  if(n>m) n-=m;`
`  else m-=n;`
`cout<<n;`
	`while(m!=0){`
	`  int r=n%m;`
	`  n=m;`
	`  m=r;`
	`}`
	`cout<<n;`

4. Một số bài tập ví dụ

a. Dãy số

Cho hai số nguyên a và b.

Yêu cầu: Hãy đưa ra các số nguyên nằm trên đoạn $[a, b]$ chia hết cho 2 hoặc chia cho 3 dư 1.

Dữ liệu vào: Hai số nguyên a và b;

Giới hạn: $1 \leq a \leq b \leq 10^6$;

Kết quả: Đưa ra màn hình các số tìm được theo trình tự tăng dần của giá trị, các số cách nhau một dấu cách. Dữ liệu vào luôn đảm bảo có kết quả

Ví dụ:

Input	Output
5 15	6 7 8 10 12 13 14

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
	int a,b;
	cin>>a>>b;
	for(int i=a;i<=b;i++)
		if(i%2==0 or i%3==1) cout<<i<<" ";
	return 0;
}
```

b. Dãy bình phương giảm dần

Cho hai số nguyên dương a và b.

Yêu cầu: Đưa ra theo thứ tự giá trị bình phương giảm dần của các số nguyên dương k thỏa mãn điều kiện $min\ (a, b) \leq k \leq max\ (a, b)$;

Dữ liệu vào: Hai số nguyên a, b;

Giới hạn: $1 \leq a, b \leq 10^6$;

Kết quả: Đưa ra màn hình các số tìm được theo trình tự giảm dần của các giá trị;

Ví dụ:

Input	Output
5 8	64 49 36 25

Ý tưởng:

Gọi $maxx$ là giá trị lớn nhất trong hai số a, b

Gọi $minx$ là giá trị nhỏ nhất trong hai số a, b

Duyệt các số i giảm dần từ $maxx$ về $minx$ rồi in ra giá trị $i * i$

Chương trình tham khảo:

```
1	#include <bits/stdc++.h>
```

```
2   using namespace std;
3   int main()
4   {
5         int a,b;
6         cin>>a>>b;
7         int maxx=max(a,b);
8         int minx=min(a,b);
9         for(int i=maxx;i>=minx;i--)
10              cout<<i*i<<" ";
11        return 0;
12  }
```

Giải thích:

+ Dòng **7**: dùng để tìm giá trị lớn nhất của hai số a, b. Ở đây không sử dụng câu lệnh if mà sử dụng hàm chuẩn $max()$ có sẵn trong C++.

+ Dòng **8**: dùng để tìm giá trị nhỏ nhất của hai số a, b. Tương tự như dòng 7, ở đây dùng hàm chuẩn $min()$ có sẵn trong C++.

c. Số đối xứng

Cho số nguyên dương n, hãy cho biết n có phải là số đối xứng hay không? Biết rằng số đối xứng là số có nếu đọc từ trái qua phải cũng có giá trị như đọc từ phải qua trái.

lưu ý: Không sử dụng kiểu xâu

Dữ liệu vào: Số nguyên dương n

Giới hạn: $1 \leq n \leq 10^{18}$

Kết quả: In số 0 nếu n không đối xứng, ngược lại in 1

Input	Output
12321	1

Input	Output
123312	0

Ý tưởng:

Với số nguyên dương n ta tạo số m là số đảo của n.

Ví dụ $n = 1234$ thì số $m = 4321$ (viết theo thứ tự từ phải sang trái các chữ số trong n)

Sau đó so sánh, nếu $n = m$ thì n là số đối xứng, ngược lại n không phải là số đối xứng.

Để tạo số m ta làm như sau:

Bước 1: gán $m = 0$

Bước 2: Với mỗi chữ số trong n (tính từ hàng đơn vị) ta đưa vào m.

Chương trình tham khảo:

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   long long n;
4   int main ()
5   {
6       cin >> n;
7       long long m=0,k=n;
```

```
8       while (n!=0)
9       {
10          m=m*10 + n%10;
11          n=n/10;
12      }
13      if (m==k) cout << 1;  else  cout << 0;
14      return 0;
15  }
```

Giải thích:

+ Dòng **10**: $n\%10$ là chữ số hàng đơn vị của n. Dòng này thể hiện việc "đưa chữ số trong n vào m" như đã trình bày trong ý tưởng.

+ Dòng **13**: So sánh m và k, trong đó k đã được gán bằng n ở dòng **7**, do vậy việc so sánh này thực chất là so sánh m với n để đưa ra kết quả. Cần phải lưu lại giá trị của n ban đầu vào biến k vì giá trị của n lúc này (ở dòng **13**) đã là 0.

d. Kiểm tra tính nguyên tố

Một số nguyên dương n được gọi là số nguyên tố nếu n có chỉ có hai ước số là 1 và chính nó. Ví dụ 7 là số nguyên tố vì 7 chỉ có 2 ước số là 1 và 7. Số 9 không phải là số nguyên tố vì nó có nhiều hơn hai ước số.

Yêu cầu: Hãy cho biết n có phải là số nguyên tố hay không?

Dữ liệu vào: Số nguyên dương n

Giới hạn: $1 \le n \le 10^{12}$

Kết quả: ghi 1 nếu n là số nguyên tố, ngược lại ghi 0

Input	Output
7	1

Input	Output
10	0

Ý tưởng:

Dựa vào định nghĩa có thể thấy nếu n là số nguyên tố thì n không có ước số nào nằm trong đoạn $[2, n-1]$, có thể giới hạn lại đoạn này là $[2, n/2]$ vì n chắc chắn không chia hết cho bất kỳ số nào thuộc đoạn $[n/2+1, n-1]$. Ví dụ với $n = 20$ thì n không chia hết cho các số $11, 12, .., 19$.

Ta có thể giới hạn lại đoạn trên là $[2, \sqrt{n}]$ vì:

Giả sử $n = a * b$ $(2 \le a \le b < n)$ thì chắc chắn xảy ra trường hợp $2 \le a \le \sqrt{n} \le b < n$. Do vậy chỉ cần kiểm tra a có phải là ước của n hay không chứ không cần kiểm tra b.

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    long long n;
    int k=1;
    cin>>n;
    if(n<=1) k=0; else
    for(int x=2;x<=sqrt(n);x++)
```

```
        if(n%x==0) k=0;
    cout<<k;
    return 0;
}
```

e. Phân tích thành thừa số nguyên tố

Cho số nguyên dương n. Hãy phân tích n thành tích các số nguyên tố.

Dữ liệu vào: Số nguyên dương n

Giới hạn: $1 < n \leq 10^6$

Kết quả:

Dãy gồm k số nguyên tố $a_1, a_2, \dots, a_k$ sao cho $a_1 < a_2 < \cdots < a_k$ và $a_1.a_2 \dots .a_k = n$

Ví dụ:

Input	Output
100	2 2 5 5

Ý tưởng

Ta sẽ xét các số tự nhiên i từ 2 trở lên. Với mỗi giá trị của i,

+ nếu i là ước cả n thì:

- in ra màn hình giá trị i

- gán $n = n/i$

Việc này được thực hiện lặp lại cho đến khi n không chia hết cho i

+ nếu i không phải là ước của n thì $i++$

Vấn đề:

+ Vì sao trong ý tưởng trên không cần kiểm tra số i là số nguyên tố? Vì nếu $n\%i == 0$ thì việc in i và $n = n/i$ được thực hiện nhiều lần cho đến khi i không còn là ước của n, điều này có nghĩa là tất cả các bội của i cũng không phải là ước của n hay nói cách khác i chính là số nguyên tố nếu $n\%i == 0$

+ i sẽ tăng lên đến bao nhiêu, dễ thấy i không vượt quá n, tuy nhiên ta chỉ cần cho i tăng đến $\sqrt{n}$ vì nếu trong đoạn $[2, \sqrt{n}]$ mà không có giá trị của i nào là ước của n thì trong đoạn $[\sqrt{n}, n-1]$ cũng không có giá trị nào của i là ước của n. Lúc này n là số nguyên tố (nếu $n > 1$).

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin>>n;
    for(int i=2;i*i<=n;i++)
        while(n%i==0)
            {
                cout<<i<<" ";
```

```
                    n=n/i;
                }
        if(n>1) cout<<n;
        return 0;
}
```

f. Basic10_số hoàn thiện

Số hoàn thiện (hay còn gọi là số hoàn chỉnh, số hoàn hảo hoặc số hoàn thành) là một số nguyên dương mà tổng các ước nguyên dương chính thức của nó (số nguyên dương bị nó chia hết ngoại trừ nó) bằng chính nó.

Cho số nguyên dương n. Hãy cho biết n có phải là số hoàn thiện hay không?

Dữ liệu vào: Số nguyên dương n

Giới hạn: $1 \leq n \leq 10^{12}$

Kết quả: Ghi số 1 nếu n là số hoàn thiện, ngược lại ghi 0

Input	Output
6	1

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
      long long n;
      long long d=0;
      int i;
      cin>>n;
      for(i=1;i*i<n;i++)
            if(n%i==0) d+=i+n/i;
      if(i*i==n) d+=n;
      cout<<(d==2*n);
      return 0;
}
```

5. Bài tập

Xem tại: http://lqdonkh.xyz/contest/loop

Bài 10: HÀM

1. Khái niệm

Hàm là một loại đối tượng được sử dụng nhiều trong các ngôn ngữ lập trình. Trong C++ chúng ta đã sử dụng hàm trong các bài học trước đây, ví dụ như hàm $abs(x)$ dùng để lấy giá trị tuyệt đối của số x, hàm $sqrt(x)$ dùng để lấy căn bậc 2 của x. Trong C++ còn có rất nhiều hàm khác, các hàm này đều có đặc điểm chung là được ngôn ngữ lập trình xây dựng sẵn, người dùng chỉ việc lấy ra dùng mà không cần quan tâm đến nó hoạt động như thế nào, chỉ cần quan tâm đến các tham số và kết quả của nó. Ví dụ với hàm $sqrt(x)$ thì người dùng chỉ cần biết x là gì và $sqrt(x)$ dùng để làm gì.

Ngoài các hàm C++ đã xây dựng sẵn, người dùng có thể tạo ra các hàm mới phù hợp với mục đích khác nhau để giải các bài toán khác nhau.

Bài toán ví dụ: Tính tổng $S = a^m + b^n + c^p + d^q$; trong đó a, b, c, d, m, n, p, q là các số nguyên dương có giá trị không vượt quá 10.

Xác định bài toán:

+ Input: a, b, c, d, m, n, p, q
+ Output: tổng S

Cách 1: Viết 4 đoạn chương trình để tính $s_1 = a^m$; $s_2 = b^n$; $s_3 = c^p$; $s_4 = d^q$; sau đó tính tổng $s_1 + s_2 + s_3 + s_4$. Chương trình tham khảo cho cách này như sau:

```
#include <bits/stdc++.h>
using namespace std;
int a,b,c,d,m,n,p,q;
long long s;
int main()
{
      cin>>a>>b>>c>>d>>m>>n>>p>>q;
      long long s1=1;
      for(int i=1;i<=m;i++) s1=s1*a;
      long long s2=1;
      for(int i=1;i<=n;i++) s2=s2*b;
      long long s3=1;
      for(int i=1;i<=p;i++) s3=s3*c;
      long long s4=1;
      for(int i=1;i<=q;i++) s4=s4*d;
      s=s1+s2+s3+s4;
      cout<<s;
      return 0;
}
```

Nhận xét: Với cách này ta phải viết lại 4 lần các câu lệnh tương tự nhau để tính 4 lũy thừa, điều này không cần thiết.

Cách 2: Sử dụng hàm

Trong cách này ta sẽ viết một lần duy nhất để tính lũy thừa.

Giả sử bài toán lũy thừa cần giải là x^y

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int a,b,c,d,m,n,p,q;
4  long long luythua(int x, int y)
5  {
6       long long lt=1;
7       for(int i=1;i<=y;i++) lt=lt*x;
8       return lt;
9  }
10 int main()
11 {
12      cin>>a>>b>>c>>d>>m>>n>>p>>q;
13      cout<<luythua(a,m)+luythua(b,n)+luythua(c,p)+luythua(d,q);
14      return 0;
15 }
```

Trong chương trình trên, các dòng từ **4** đến **9** là hàm dùng để giải quyết bài toán x^y. Sau khi đã xây dựng xong hàm tính x^y ta chỉ cần gọi lại nó (dòng **13**) như các hàm có sẵn của C++.

Như vậy, *hàm là một dãy lệnh mô tả một số thao tác nhất định và có thể được thực hiện (được gọi) từ nhiều vị trí trong chương trình.*

2. Lợi ích khi sử dụng hàm

+ Tránh được việc phải viết lặp đi lặp lại một dãy lệnh nào đó. Trong bài toán trên, dãy lệnh tính lũy thừa được viết lại 4 lần ở cách 1, trong khi dùng hàm chỉ cần viết 1 lần.

+ Có thể chia nhỏ công việc cho nhiều người (nhóm người) thực hiện xây dựng các hàm khác nhau để giải một bài toán lớn.

+ Phục vụ cho quá trình trừu tượng hóa: Một người lập trình có thể sử dụng lại các hàm có sẵn mà không cần quan tâm đến các thao tác này được cài đặt như thế nào.

+ Thuận tiện cho việc phát triển, nâng cấp chương trình

3. Phân loại và cấu trúc của hàm

a. Phân loại hàm

Trong C++ có hai loại hàm:

Loại 1: Hàm trả về giá trị, ví dụ hàm $abs(x), sqrt(x)$

Loại 2: Hàm không trả về giá trị.

b. Cấu trúc của hàm

Hàm trong C++ có cấu trúc như sau:

```
<kiểu dữ liệu> <tên hàm>([<danh sách tham số>])
{
        [<dãy câu lệnh>]
}
```

Trong đó:

<kiểu dữ liệu>: là kiểu dữ liệu của hàm, *<kiểu dữ liệu>* có thể là int, double, char, bool .... nếu hàm thuộc loại **không** trả về giá trị thì *<kiểu dữ liệu>* là **void.**

<tên hàm>: là tên do người dùng tự đặt theo quy tắc đặt tên của C++.

<danh sách tham số>: Mỗi hàm thường dùng để giải một bài toán nào đó, mỗi bài toán thường có input và output. *Danh sách tham số* để thể hiện input hoặc output của bài toán mà hàm giải quyết. Có thể xem lại ví dụ ở trên, hàm $luythua(x, y)$ dùng để giải quyết bài toán x^y nên x, y được dùng làm tham số.

Lưu ý: trong danh sách tham số, mỗi tham số có cấu trúc khai báo tương tự như biến nên có kiểu dữ liệu phía trước (xem lại)

<dãy câu lệnh>: là các câu lệnh trong C++. Trong *<dãy câu lệnh>* có thể có một hoặc nhiều lệnh dạng:

return *[<biểu thức>];*

Ý nghĩa của lệnh này là trả về giá trị cho hàm rồi kết thúc việc thực hiện hàm. Trong loại hàm không trả về giá trị thì không có *<biểu thức>*. Ví dụ trong hàm luythua(x, y) có lệnh return lt; dùng để trả về giá trị lũy thừa x^y cho hàm

4. Ví dụ về cách viết và sử dụng hàm

Trong phần này sẽ có một số khái niệm mới như: *tham số hình thức*, *tham số thực sự*, *tham biến*, *tham trị*,...Những khái niệm này sẽ được giải thích thông qua các ví dụ dưới đây.

Ví dụ 1: Vẽ hình chữ nhật bằng dấu sao (*) như hình (8×5)

Chương trình tham khảo:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  void vehcn()
4  {
5       cout<<"********"<<endl;
6       cout<<"*      *"<<endl;
7       cout<<"*      *"<<endl;
8       cout<<"*      *"<<endl;
9       cout<<"********"<<endl;
10 }
11 int main()
12 {
13      vehcn();
14      return 0;
15
}
```

Giải thích:

+ Hàm vehcn() (từ dòng **3** đến **10**) dùng để vẽ hình chữ nhật theo yêu cầu bài toán. Hàm này **không** có giá trị trả về nên *kiểu dữ liệu* của hàm là **void**. Hàm này không có danh sách tham số.

+ Dòng **13**: hàm vehcn() sau khi xây dựng được gọi lại nhưng không có danh sách tham số. Cách gọi lại hàm đã xây dựng tương tự như gọi lại hàm có sẵn của chương trình.

Ví dụ 2: In ra màn hình hình chữ nhật rỗng có các cạnh bằng dấu sao (*), trong đó hai cạnh liên tiếp lần lượt có a và b dấu sao (*)

Chương trình tham khảo:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  void vehcn(int a, int b)
4  {
5        for(int i=1;i<=a;i++) cout<<"*";
6        cout<<endl;
7        for(int i=2;i<b;i++)
8        {
9              cout<<"*";
10             for(int j=2;j<a;j++) cout<<" ";
11             cout<<"*"<<endl;
12       }
13       for(int i=1;i<=a;i++) cout<<"*";
14 }
15 int main()
16 {
17       int a,b;
18       cin>>a>>b;
19       vehcn(a,b);
20       return 0;
21
22 }
```

Kết quả khi nhập $a = 10$ và $b = 8$

```
10 8
**********
*        *
*        *
*        *
*        *
*        *
*        *
**********
```

Giải thích:

+ Hàm $vehcn(int\ a, int\ b)$ (từ dòng **3** đến **14**) dùng để vẽ hình chữ nhật theo yêu cầu bài toán. Trong đó a, b là hai tham số có kiểu nguyên (int), các tham số hình thức được khai báo như ở dòng **3** được gọi là *tham số hình thức*. Hàm này **không** có giá trị trả về nên *kiểu dữ liệu* của hàm là ***void***

+ Dòng **19**: hàm $vehcn(a, b)$ sau khi xây dựng được gọi lại với hai tham số truyền vào là a, b. Cách gọi lại hàm đã xây dựng tương tự như gọi lại hàm có sẵn của chương trình. Các tham số được sử dụng trong lời gọi hàm được gọi là *tham số thực sự*.

+ Dòng **5**: Dùng để in ra a dấu * của cạnh trên cùng

+ Dòng **13**: Dùng để in ra a dấu * của cạnh dưới cùng

+ Dòng **7-12**: Dùng để in ra hai cạnh hai bên. Để làm điều này cần in $b - 2$ dòng, mỗi dòng in dấu * đầu tiên, tiếp đến là $a - 2$ dấu cách, cuối dùng là một dấu $*$.

Ví dụ 3: Tìm ước chung lớn nhất của hai số nguyên dương a, b

Chương trình tham khảo:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int ucln(int a, int b)
4  {
5       while(a%b!=0)
6       {
7            int r=a%b;
8            a=b;
9            b=r;
10      }
11      return b;
12 }
13 int main()
14 {
15    int a,b;
16    cin>>a>>b;
17    cout<<"UC max = "<<ucln(a,b)<<endl;
18    cout<<a<<" "<<b;
19 }
```

Kết quả khi nhập $a = 12$ và $b = 18$

```
12 18
UC max = 6
12 18
```

Giải thích:

+ Hàm ucln(int a, int b) (từ dòng **3** đến **12**) dùng để tìm ước chung lớn nhất của hai số nguyên a, b. Trong đó a, b là hai *tham số hình thức* có kiểu nguyên (int). Hàm này có giá trị trả về nên kiểu dữ liệu của hàm là int (ước chung lớn nhất của hai số nguyên là một số nguyên), cuối hàm (dòng **11**) có câu lệnh return b; dùng để trả về giá trị b chính là ước chung lớn nhất của hai số nguyên a, b ban đầu (lưu ý giá trị b này đã được xử lý thông qua các câu lệnh ở dòng **5** đến **10**)

+ Dòng **17**: hàm ucln(a, b) sau khi xây dựng được gọi lại với hai *tham số thực sự* truyền vào là a, b.

+ Dòng **18**: Dùng để in ra màn hình giá trị của hai số a, b sau khi thực hiện tìm ước chung lớn nhất của chúng. Nhận thấy rằng, trong hàm ucln(int a, int b) (từ dòng **3** đến **12**) có các câu lệnh làm thay đổi giá trị của hai số a, b tuy nhiên ***sau khi thoát khỏi hàm giá trị của hai số này không thay đổi***. Hai tham số hình thức a, b (ở dòng **3**) được gọi là *tham trị*

Ví dụ 4: Hoán đổi giá trị hai số nguyên a, b

Chương trình tham khảo:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  void hoandoi(int &a, int &b)
4  {
5       int tg=a;
6       a=b;
7       b=tg;
8  }
9  int main()
10 {
11      int a,b;
```

```
12      cin>>a>>b;
13      hoandoi(a,b);
14      cout<<a<<" "<<b;
15  }
```

Giải thích:

+ Hàm hoandoi(int &a, int &b) (từ dòng **3** đến **8**) dùng để hoán đổi giá trị hai số a, b. Điểm khác biệt trong hàm này là hai tham số hình thức a, b có ký tự & ở phía trước. Với cách khai báo như vậy ta gọi a, b là các ***tham biến***, giá trị của tham biến có thể thay đổi sau khi thực hiện lời gọi hàm.

+ Dòng **13**: hàm hoandoi(a, b) sau khi xây dựng được gọi lại với hai *tham số thực sự* truyền vào là a, b.

+ Dòng **14**: in ra màn hình giá trị của hai số a, b sau khi thực hiện lời gọi hàm. Việc in ra màn hình giúp ta biết được giá trị của a, b có thực sự thay đổi hay không.

BÀI 11: KIỂU MẢNG

1. Bài toán

Xét bài toán sau: Viết chương trình nhập vào nhiệt độ trung bình của 7 ngày trong tuần. Hãy cho biết:

1. Nhiệt độ trung bình của cả tuần?

2. Số ngày có nhiệt độ lớn hơn nhiệt độ trung bình của cả tuần?

Hướng dẫn:

+ Nhiệt độ trung bình của cả tuần: tổng nhiệt độ của 7 ngày chia cho 7

+ Xét lần lượt nhiệt độ của các ngày, nếu ngày nào có nhiệt độ lớn hơn nhiệt độ trung bình của cả tuần thì tăng biến đếm.

Chương trình:

Sử dụng các kiến thức ở bài trước ta có thể viết chương trình như sau:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
      double t1,t2,t3,t4,t5,t6,t7,ttb;
      int d=0;
      cin>>t1>>t2>>t3>>t4>>t5>>t6>>t7;
      ttb=(t1+t2+t3+t4+t5+t6+t7)/7;
      if(t1>ttb) d++;
      if(t2>ttb) d++;
      if(t3>ttb) d++;
      if(t4>ttb) d++;
      if(t5>ttb) d++;
      if(t6>ttb) d++;
      if(t7>ttb) d++;
      cout<<ttb<<endl<<d;
      return 0;
}
```

Nhận xét:

+ Chương trình khai báo 7 biến khác nhau để lưu trữ nhiệt độ trung bình của 7 ngày.

+ Chương trình dùng 7 câu lệnh *if* để kiểm tra nhiệt đô của 7 ngày có lớn hơn nhiệt độ trung bình hay không.

Từ đó nảy sinh vấn đề: nếu số lượng ngày cần xét lớn (ví dụ 100 ngày, 10000 ngày…) thì số lượng biến đơn cần khai báo rất lớn, số lượng câu lệnh *if* cần dùng rất nhiều. Điều này làm cho chương trình dài, dễ xảy ra sai sót, đặc biệt là số lượng ngày cần phải biết trước nên chương trình không giải quyết được với những trường hợp số ngày khác nhau.

Để giải quyết các vấn đề ở trên, ta sử dụng **kiểu mảng** để quản lý các biến đơn lưu trữ nhiệt độ một cách dễ dàng để thực hiện các thao tác như: nhập, xuất, tính tổng, duyệt qua toàn bộ các biến…mà không cần nhiều câu lệnh riêng rẽ.

2. Kiểu mảng một chiều

Mảng một chiều là dãy hữu hạn các phần tử cùng kiểu.

Ví dụ:

+ Dãy số thực lưu trữ nhiệt độ của 7 ngày trong tuần, nhiệt độ của mỗi ngày là một phần tử có kiểu *double*.

+ Dãy số nguyên lưu trữ sĩ số các lớp của một trường học, sĩ số của mỗi lớp học là một phần tử có kiểu *int*.

Để sử dụng kiểu mảng một chiều cần xác định được:

+ Tên kiểu mảng một chiều;

+ Số lượng phần tử;

+ Kiểu dữ liệu của các phần tử;

+ Cách khai báo biến;

+ Cách tham chiếu đến một phần tử.

a. Khai báo mảng một chiều

Cú pháp:

<kiểu dữ liệu> <tên biến mảng>[<số lượng phần tử tối đa>];

Trong đó:

<kiểu dữ liệu>: là kiểu dữ liệu của các phần tử trong mảng

<tên biến mảng>: Do người dùng tự đặt theo quy tắc đặt tên của C++

<số lượng phần tử tối đa>: Số lượng lớn nhất các phần tử mảng có thể lưu trữ.

Ví dụ:

+ Khai báo mảng để lưu trữ nhiệt độ của 7 ngày trong tuần

double tdo[7];

Nhiệt độ của mỗi ngày là kiểu số thực nên *<kiểu dữ liệu>* là *double*; *tdo* là tên biến mảng, có thể sử dụng các tên khác như *nhietdo, ndo, ...*; Cần khai báo để lưu trữ được nhiệt độ của 7 ngày trong tuần nên mảng cần có 7 phần tử.

+ Khai báo mảng để lưu trữ sĩ số của các lớp trong một trường học có tối đa 50 lớp.

int siso[50];

Sĩ số của một lớp học là số nguyên nên có thể dùng kiểu *int* để làm *<kiểu dữ liệu>* của các phần tử trong mảng, *siso* là tên biến mảng; trường có tối đa 50 lớp nên mảng cần có 50 phần tử để lưu trữ sĩ số của lớp học, mỗi phần tử là sĩ số của một lớp.

b. Cách tham chiếu đến mảng một chiều

Tham chiếu đến một phần tử trong mảng có thể hiểu là "gọi" đến một phần tử nào đó. Mục đích của việc tham chiếu là sử dụng một phần tử vào các công việc như: nhập, xuất, câu lệnh gán…Về cơ bản ta có thể hiểu mỗi phần tử của mảng là một biến đơn.

Để tham chiếu đến một phần tử của mảng ta dùng ***tên biến mảng, tiếp theo là chỉ số của phần tử được đặt trong cặp ngoặc [và]***.

Lưu ý: chỉ số (hay còn gọi là vị trí) của các phần tử trong mảng được đánh số thứ tự từ trái sang phải bắt đầu từ 0.

Ví dụ: Giả sử ta có mảng $songuyen$ có 5 phần tử có giá trị các phần tử và chỉ số tương ứng như sau:

Chỉ số	0	1	2	3	4
Giá trị	4	6	3	8	10

Để tham chiếu đến phần tử có chỉ số 0, 2, 4 trong mảng $songuyen$, ta viết lần lượt:

$songuyen[0]$

$songuyen[2]$

$songuyen[4]$

3. Một số ví dụ

Ví dụ 1:

Chương trình sau đây thực hiện khai báo mảng để lưu trữ dãy số nguyên có tối đa 1000 số, sau đó nhập dữ liệu cho dãy rồi in dãy số ra màn hình:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int a[1000];
4  int n;
5  int main()
6  {
7    cin>>n;
8    for(int i=0;i<n;i++) cin>>a[i];
9    for(int i=0;i<n;i++) cout<<a[i]<<" ";
10   return 0;
11 }
```

```
5
4 3 7 1 2
4 3 7 1 2
```

Giải thích:

+ Dòng 3: khai báo mảng có tối đa 1000 phần tử, mỗi phần tử có kiểu dữ liệu int;

+ Dòng 4: khai báo số nguyên n là số lượng phần tử thực tế của mảng được người dùng nhập từ bàn phím ở dòng 7, việc nhập n phải được thực hiện trước khi nhập giá trị cho dãy số;

+ Dòng 8: Nhập giá trị cho dãy số bằng cách nhập giá trị cho từng phần tử, các phần tử trong mảng có chỉ số từ 0 đến $n-1$ nên giá trị của biến i trong câu lệnh for cũng thay đổi từ 0 đến $n-1$.

+ Dòng 9: In giá trị của dãy số ra màn hình bằng cách in giá trị của từng phần tử trong mảng.

Ví dụ 2:

Chương trình sau đây nhập vào nhiệt độ của n ($n \leq 1000$) ngày. Sau đó tính và in ra:

1. Nhiệt độ trung bình của n ngày;
2. Số ngày có nhiệt độ lớn hơn nhiệt độ trung bình của n ngày.

```
#include <bits/stdc++.h>
using namespace std;
double a[1000];
```

```
int n;
int main()
{
      cin>>n;
      for(int i=0;i<n;i++) cin>>a[i];
      double ttb=0;
      for(int i=0;i<n;i++) ttb+=a[i];
      ttb/=n;
      int d=0;
      for(int i=0;i<n;i++)
            if(a[i]>ttb) d++;
      cout<<ttb<<endl<<d;
      return 0;
}
```

Ví dụ 3:

Chương trình sau đây nhập vào số nguyên dương n ($n \leq 1000$) và dãy số nguyên $a_1, a_2, \ldots, a_n$ sau đó in ra màn hình vị trí của phần tử có giá trị lớn nhất trong dãy số, nếu có nhiều phần tử có cùng giá trị lớn nhất thì in ra vị trí lớn nhất tìm được.

Ý tưởng:

+ Khai báo mảng một chiều a (có vị trí các phần tử từ 0 đến $n-1$)

Giả sử ban đầu phần tử đầu tiên trong dãy có giá trị lớn nhất, gán $vt = 0$; Lần lượt so sánh các phần tử tiếp theo (từ vị trí 1 đến $n-1$) với phần tử ở vị trí vt, nếu $a[i] \geq a[vt]$ thì $vt = i$.

Kết quả là $vt + 1$

Chương trình tham khảo:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int a[1000];
4  int n;
5  int main()
6  {
7        cin>>n;
8        for(int i=0;i<n;i++) cin>>a[i];
9        int vt=0;
10       for(int i=0;i<n;i++)
11             if(a[i]>=a[vt]) vt=i;
12       cout<<vt+1;
13       return 0;
14 }
```

Question and Answer:

Q1: Tại sao ở dòng 10 lại ghi $vt + 1$ chứ không phải vt?

A1: Vì dãy số theo yêu cầu đề bài bắt đầu từ 1, còn mảng lưu trữ dãy số bắt đầu từ 0.

Q2: Có thể bỏ dấu $=$ ở điều kiện $a[i] >= a[vt]$ (dòng 9) được không? vì sao?

A2: Không thể vì nếu bỏ dấu $=$ thì vt là vị trí nhỏ nhất của các phần tử có cùng giá trị lớn nhất (sai yêu cầu bài toán).

Q3: Nếu muốn in ra màn hình giá trị lớn nhất của dãy số thì chương trình trên cần sửa lại như thế nào?

A3: Chỉ cần sửa lại ở dòng 10 là $cout << a[vt];$

Ví dụ 4:

Sắp xếp dãy số nguyên có tối đa 1000 số theo thứ tự tăng dần bằng thuật toán trao đổi.

Ý tưởng:

Với mỗi cặp số bất kỳ trong dãy, nếu số đứng trước có giá trị lớn hơn số đứng sau thì hoán đổi giá trị của chúng.

Cụ thể: Xét mọi cặp số a_i, a_j $(i < j)$ trong dãy, nếu $a_i > a_j$ thì hoán đổi giá trị hai số a_i, a_j với nhau.

Chương trình tham khảo:

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   int a[1000];
4   int n;
5   int main()
6   {
7     cin>>n;
8     for(int i=0;i<n;i++) cin>>a[i];
9     for(int i=0;i<n-1;i++)
10       for(int j=i+1;j<n;j++)
11         if(a[i]>a[j])
12            {
13              int tg=a[i];
14              a[i]=a[j];
15              a[j]=tg;
16            }
17    for(int i=0;i<n;i++) cout<<a[i]<<" ";
18    return 0;
19  }
```

Giải thích:

+ Dòng 9+10: Sử dụng 2 vòng lặp for lồng nhau để xác định 1 cặp số trong dãy, ở dòng 10 ta thấy $j = i + 1$ để đảm bảo điều kiện $j > i$ nghĩa là $a[j]$ luôn là số đứng sau $a[i]$.

+ Dòng 13+14+15: dùng để hoán đổi giá trị hai số $a[i]$ và $a[j]$ bằng cách dùng biến trung gian tg. Có thể thay 3 dòng lệnh này bằng hàm $swap(a[i], a[j]);$

Ví dụ 5: Chèn phần tử

Cho số nguyên dương n, k $(n \leq 1000; k \leq 10^9)$ và dãy số nguyên dương $a_1, a_2, \ldots, a_{n-1}, a_n$ đã được sắp xếp tăng dần. Hãy chèn số nguyên k vào dãy sao cho dãy số vẫn giữ nguyên thứ tự sắp xếp.

Ví dụ:

Input	Output
5 8 3 4 6 7 9	3 4 6 7 **8** 9

Ý tưởng:

Thực hiện theo các bước sau:

Bước 1: Tìm vị trí cần chèn, giả sử vị trí tìm được là vt

Bước 2: Đẩy các số có vị trí $n, n-1, n-2, \dots, vt+1$ ra phía sau một vị trí: $a_i = a_{i-1}$

Bước 3: Gán $a_{vt} = k$

Chương trình tham khảo:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int a[1001];
4  int n,k;
5  int main()
6  {
7        cin>>n>>k;
8        for(int i=1;i<=n;i++) cin>>a[i];
9        int vt=1;
10       while(a[vt]<k and vt<=n) vt++;
11       for(int i=n;i>=vt;i--) a[i+1]=a[i];
12       a[vt]=k;
13       for(int i=1;i<=n+1;i++) cout<<a[i]<<" ";
14       return 0;
15 }
```

Giải thích:

+ Dòng 9, 10: Thể hiện Bước 1; sau khi chương trình thực hiện xong 2 dòng này thì vt chính là vị trí cần chèn.

+ Dòng 11: Thể hiện Bước 2; ở đây biến i là vị trí của các phần tử cần được dịch qua phải một bước, ta cần dịch phần tử cuối cùng trước nên i giảm từ n về $vt+1$

+ Dòng 13: Thể hiện Bước 3;

4. Tạo dãy số ngẫu nhiên

a. Hàm $rand()$

Hàm $rand()$ có ý nghĩa là tạo một số nguyên ngẫu nhiên trong đoạn $[0, RAND\_MAX]$, trong đó $RAND\_MAX$ là một hằng số có sẵn có giá trị phụ thuộc vào chương trình dịch (trong tài liệu này sử dụng chương trình dịch có $RAND\_MAX = 32767$)

+ Để tạo một số nguyên ngẫu nhiên có giá trị thuộc đoạn $[0, x-1]$ ta viết: $rand()\%x$

Ví dụ: Để tạo một số nguyên ngẫu nhiên có giá trị thuộc đoạn $[0,300]$ ta viết $rand()\%301$

+ Để tạo một số nguyên ngẫu nhiên có giá trị thuộc đoạn $[-x, x]$ ta viết:

$$rand()\%(x+1) - rand()\%(x+1)$$

Ví dụ: Để tạo một số nguyên ngẫu nhiên có giá trị thuộc đoạn $[-300,300]$ ta viết:

$$rand()\%301 - rand()\%301$$

+ Để tạo một dãy số $a_1, a_2, \dots, a_{n-1}, a_n$ trong đó mỗi số a_i $(i = 1 \dots n)$ có giá trị ngẫu nhiên trong đoạn $[-300,300]$ ta viết:

```
for(int i=1;i<=n;i++)
      a[i]=rand()%301-rand()%301;
```

b. Bài toán

Viết chương trình nhập từ bàn phím số nguyên dương n, k $(n \leq 1000)$ rồi tạo dãy số nguyên $a_1, a_2, \dots, a_{n-1}, a_n$ sao cho mỗi số a_i $(i = 1 \dots n)$ có giá trị thuộc đoạn $[-300,300]$. In ra màn hình dãy số vừa tạo và tổng các số là bội của số nguyên k.

Nhận xét:

Trong bài này chỉ nhập hai số nguyên dương là n và k còn dãy số không được nhập từ bàn phím mà được tạo một cách ngẫu nhiên. Do đó ta phải sử dụng đến hàm $rand()$;

Chương trình tham khảo:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int main()
4  {
5        srand(time(0));
6        int n,k;
7        int a[1001];
8        cout<<"Nhap n, k: ";
9        cin>>n>>k;
10       cout<<"Day so ngau nhien: ";
11       for(int i=1;i<=n;i++)
12             a[i]=rand()%301-rand()%301;
13       for(int i=1;i<=n;i++) cout<<a[i]<<" ";
14       cout<<endl;
15       int sumk=0;
16       for(int i=1;i<=n;i++)
17             if(a[i]%k==0) sumk+=a[i];
18       cout<<"Tong cac so la boi cua "<<k<<": "<<sumk;
19       return 0;
20 }
```

Giải thích:

+ Dòng 5: $srand(time(0));$ dùng để khởi tạo bộ số ngẫu nhiên trước khi dùng hàm $rand()$; Nếu không có $srand(time(0));$ thì chương trình sẽ cho ra kết quả giống nhau trong các lần thực hiện chương trình.

5. Bài tập

Xem tại: http://lqdonkh.xyz/contest/arr1d

BÀI 12: KIỂU XÂU

Trong bài 2, chúng ta đã đề cập đến kiểu xâu thông qua giới thiệu về hằng xâu, ví dụ:

`"Tin hoc"  "phong chong dich Covid 19"  "mang khau trang"`

Xâu là dãy các ký tự trong bộ mã ASCII, mỗi ký tự được gọi là một phần tử của xâu.

Số lượng ký tự trong xâu được gọi là độ dài của xâu.

Ví dụ:

Xâu `"Tin hoc"` có 7 ký tự nên độ dài của xâu là 7;

Xâu `"phong chong dich Covid 19"` có 25 ký tự nên độ dài của xâu là 25;

Xâu có độ dài 0 được gọi là xâu rỗng, ký hiệu `""`

Có thể xem xâu là mảng một chiều mà mỗi phần tử là một ký tự, các phần tử của xâu được đánh số thứ tự bắt đầu từ 0.

Ví dụ:

Số thứ tự của các ký tự trong xâu `"Tin hoc"` là:

0	1	2	3	4	5	6
T	i	n		h	o	c

Số thứ tự của các ký tự trong xâu `"chuc mung nam moi"` là:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
c	h	u	c		m	u	n	g		n	a	m		m	o	i

Tương tự mảng, *tham chiếu* đến một phần tử trong xâu được xác định bởi tên biến xâu và chỉ số được đặt trong cặp ngoặc [và];

Ví dụ:

Giả sử biến *monhoc* đang lưu trữ giá trị xâu `"Tin hoc"` thì:

monhoc[0] cho ký tự 'T' *monhoc*[2] cho ký tự 'n'

monhoc[4] cho ký tự 'h' *monhoc*[5] cho ký tự 'o'

Trong C++, để làm việc với xâu ta sử dụng thư viện string.

#include < string >

Sau đây là trình bày về cách khai báo kiểu dữ liệu xâu và các thao tác trên xâu

1. Khai báo xâu

Để khai báo biến xâu, ta sử dụng tên dành riêng *string*.

Cú pháp: *string <tên biến xâu>;*

<tên biến xâu> do người dùng tự đặt theo quy tắc đặt tên trong C++.

Ví dụ: Khai báo biến xâu để lưu trữ họ và tên của một người

string hoten;

2. Nhập xâu

Để nhập một xâu từ bàn phím ta dùng cấu trúc:

getline(cin,<tên biến xâu>);

Ví dụ: để nhập giá trị cho xâu s từ bàn phím ta viết: $getline(cin, s);$

3. Phép ghép xâu

Phép ghép xâu được dùng để ghép 2 hay nhiều xâu thành một xâu.

Ký hiệu: dấu cộng (+)

Ví dụ:

Phép ghép xâu

```
"chuc mung" + " nam moi"
```

cho kết quả quả là `"chuc mung nam moi"`

Cho xâu $s = "aaa"$ và $t = "bbb"$

Nếu thực hiện câu lệnh $s = s + t;$ thì giá trị của xâu s là $"aaabbb"$

Nếu thực hiện câu lệnh $s = t + s;$ thì giá trị của xâu s là $"bbbaaa"$

4. Các phép so sánh xâu

Các phép so sánh xâu: bằng $(==)$, lớn hơn hoặc bằng $(>=)$, nhỏ hơn hoặc bằng $(<=)$, lớn hơn $(>)$, nhỏ hơn $(<)$, khác $(!=)$

Quy tắc so sánh xâu:

Q1: Hai xâu bằng nhau nếu chúng giống nhau hoàn toàn

Ví dụ: $"Tin\ hoc" == "Tin\ hoc"$

Q2: Nếu xâu A là đoạn đầu của xâu B thì A nhỏ hơn B

Ví dụ: $"May\ tinh" < "May\ tinh\ cua\ toi"$

Q3: Xâu A nhỏ hơn xâu B nếu ký tự đầu tiên khác nhau giữa chúng kể từ trái sang trong xâu A có mã ASCII nhỏ hơn xâu B

Ví dụ: $"quoc\ gia" < "quoc\ su"$

Ví dụ: $"quoc\ Gia" < "quoc\ gia"$

5. Một số phương thức cơ bản với xâu

Trong C++ xâu là một đối tượng. Các hàm được xây dựng cho một đối tượng được gọi là phương thức. Một phương thức thường được sử dụng theo cú pháp

<tên biến>.<tên phương thức([danh sách tham số])>

a. Phương thức $\boldsymbol{length}()$

$s.length()$ cho giá trị là độ dài của xâu s

Ví dụ: Với $s = "Tin\ hoc";$ thì $cout << s.length();$ cho kết quả in ra màn hình là 7

b. Phương thức $\boldsymbol{substr}()$

$s.substr(vt, n)$ tạo xâu con gồm n ký tự liên tiếp bắt đầu từ vị trí vt của xâu s.

Ví dụ: Cho $s = "chuc\ mung\ nam\ moi";$

t=s.substr(2,7); sẽ cho xâu t có giá trị $t = "uc\ mung";$

c. Phương thức $find()$

$s.find(t)$ cho biết vị trí xuất hiện đầu tiên của xâu t trong xâu s, nếu t không xuất hiện trong s thì cho kết quả là -1.

Ví dụ: Cho $s = "cabcdbc";$

$s.find("bc");$ cho giá trị là 2.

$(int)s.find("abx");$ cho giá trị là -1.

d. Phương thức $erase()$

$s.erase(vt, n)$ thực hiện xóa n ký tự bắt đầu từ vị trí vt của xâu s.

Ví dụ: Cho $s = "cabcdbc";$ sau khi thực hiện $s.erase(2,3);$ thì giá trị của $s = "cabc"$

e. Phương thức $insert()$

s.insert(vt, t) thực hiện chèn xâu t vào vị trí vt của xâu s.

Ví dụ: Cho $s = "cabcdbc";$

Sau khi thực hiện $s.insert(2, "123");$ thì giá trị của $s = "ca123bcdbc";$

5. Một số ví dụ

Ví dụ 1:

Chương trình sau đây nhập vào họ tên của hai người vào hai biến xâu rồi đưa ra màn hình xâu dài hơn, nếu bằng nhau thì đưa ra xâu nhập sau.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
     string ht1,ht2;
     getline(cin,ht1);
     getline(cin,ht2);
     if(ht1.length()>ht2.length()) cout<<ht1;
     else cout<<ht2;
     return 0;
}
```

Ví dụ 2:

Chương trình sau đây nhập vào hai xâu rồi kiểm tra ký tự đầu tiên của xâu thứ nhất có trùng với ký tự cuối cùng của xâu thứ 2 hai không?

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
     string s,t;
     getline(cin,s);
     getline(cin,t);
```

```
    int last=t.length()-1;
    if(s[0]==t[last]) cout<<"trung nhau";
    else cout<<"khac nhau";
    return 0;
}
```

Ví dụ 3:

Chương trình sau nhập một xâu từ bàn phím rồi đưa ra màn hình xâu đó theo thứ tự ngược lại.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
     string s;
     getline(cin,s);
     int last=s.length()-1;
     for(int i=last;i>=0;i--) cout<<s[i];
     return 0;
}
```

Ví dụ 4:

Chương trình sau nhập một xâu từ bàn phím rồi đưa ra màn hình xâu đó sau khi loại bỏ hết các dấu cách (nếu có)

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
     string s;
     getline(cin,s);
     while((int)s.find(" ")!=-1)
     {
          int vt=(int)s.find(" ");
          s.erase(vt,1);
     }
     cout<<s;
     return 0;
}
```

Ví dụ 5:

Chương trình sau nhập xâu s_1 từ bàn phím, tạo xâu s_2 bằng cách loại bỏ các chữ số có trong s_1 (giữ nguyên thứ tự xuất hiện của chúng) rồi đưa kết quả ra màn hình.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
     string s1,s2="";
     getline(cin,s1);
     int last=s1.length()-1;
```

```
      for(int i=0;i<=last;i++)
            if(s1[i]>'9' or s1[i]<'0') s2=s2+s1[i];
      cout<<s2;
      return 0;
}
```

BÀI TẬP LẬP TRÌNH – KIỂU XÂU

1. Đếm số từ

Hãy viết chương trình nhập vào một xâu rồi in ra màn hình số lượng từ trong xâu. Biết rằng một dãy các ký tự liên tiếp không chứa ký tự trắng được gọi là một từ.

Xâu nhập vào chỉ có ký tự chữ cái, chữ số, ký tự trắng; không có trường hợp ký tự trắng đứng đầu xâu, cuối xâu hoặc hai ký tự trắng đứng liền nhau.

Ý tưởng:

Đếm số dấu cách trong xâu, số từ là số lượng dấu cách + 1

Chương trình tham khảo:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int main()
4  {
5        string s;
6        getline(cin,s);
7        int sokytutrang=0;
8        int last=s.length()-1;
9        for(int i=0;i<=last;i++)
10             if(s[i]==' ') sokytutrang++;
11       if(s.length()!=0) sokytutrang++;
12       cout<<sokytutrang;
13       return 0;
14 }
```

Giải thích:

+ Dòng 9, 10 dùng để đếm số lượng ký tự trắng trong xâu s như đã nêu ở ý tưởng.

+ Dòng 11: nếu xâu s rỗng (có độ dài 0) thì số từ trong xâu là 0, do vậy dòng này kiểm tra nếu xâu **khác** rỗng thì tăng $sokytutrang$ lên 1 (chính là số từ trong xâu)

2. Xâu đối xứng

Một xâu được gọi là xâu đối xứng nếu đọc xâu đó từ trái sang phải cũng như đọc từ phải sang trái. Ví dụ các xâu sau là xâu đối xứng: "12321" "$anna$" "$abccba$". Hãy lập trình nhập từ bàn phím một xâu rồi cho biết xâu đó có phải là xâu đối xứng hay không?

Ý tưởng:

Tạo xâu t là xâu đảo của s, sau đó thực hiện so sánh hai xâu s và t, nếu bằng nhau thì s là xâu đối xứng, ngược lại s không phải là xâu đối xứng.

Chương trình tham khảo:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int main()
4  {
5        string s;
```

```
6       getline(cin,s);
7       int last=s.length()-1;
8       string t="";
9       for(int i=0;i<=last;i++)
10           t=s[i]+t;
11      if(s==t) cout<<"Doi xung"; else
12       cout<<"Khong doi xung";
13      return 0;
14  }
```

Giải thích:

+ Dòng **9, 10** dùng để tạo xâu t là xâu đảo của s, xâu t lúc đầu gán bằng rỗng (dòng **8**). Xâu t được tạo bằng cách lấy từng ký tự của xâu s (từ trái sang phải) để ghép vào đầu xâu t.

+ Dòng **11** dùng để so sánh hai xâu

3. Thay thế xâu

Viết chương trình nhập từ bàn phím xâu s rồi in ra màn hình xâu s sau khi thay thế tất cả các cụm ký tự "anh" bằng "em".

Ví dụ: nhập từ bàn phím xâu $s =$ "$anh\ ay\ la\ anh\ cua\ toi$"

Xâu s sau khi được thay thế là: $s =$ "$em\ ay\ la\ em\ cua\ toi$"

Ý tưởng:

Thay thế xâu "anh" bằng xâu "em" đồng nghĩa với việc xóa xâu "anh" sau đó chèn xâu "em" vào vị trí cần xóa. Do vậy trong bài này cần dùng đến hai phương thức *erase()* và *insert()* trong xâu, ngoài ra hai phương thức này cần xác định vị trí nên ta sử dụng thêm phương thức *find()*. Việc thay thế được thực hiện nhiều lần nên sử dụng câu lệnh lặp *while*.

Chương trình tham khảo:

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   int main()
4   {
5       string s;
6       getline(cin,s);
7       while((int)s.find("anh")!=-1)
8       {
9           int vt=(int)s.find("anh");
10          s.erase(vt,3);
11          s.insert(vt,"em");
12      }
13      cout<<s;
14      return 0;
15  }
```

Lưu ý:

+ Khi thực hiện phương thức *find()* trong xâu cần ép kiểu về *int* để tránh trường hợp không trả về -1 khi không tìm thấy xâu.

+ Trước khi thực hiện phương thức $erase()$ (dòng 10) và $insert()$ (dòng 11) cần lưu lại vị trí tìm được của chuỗi "anh" như ở dòng 9

4. Chuẩn hóa xâu

Nhập từ bàn phím xâu s, hãy in ra màn hình xâu s sau khi thực hiện chuẩn hóa xâu theo các quy tắc:

+ Xóa các ký tự trắng đầu xâu;

+ Xóa các ký tự trắng cuối xâu;

+ Nếu có hai ký tự trắng liên tiếp trong xâu thì xóa đi 1 ký tự trắng.

Ý tưởng:

Sử dụng phương thức $erase()$ để xóa các ký tự trắng theo yêu cầu đề bài. Việc cần làm là xóa ký tự trắng ở vị trí nào và bao nhiêu ký tự cần xóa:

+ Xóa các ký tự trắng đầu xâu: Vị trí: 0; số lượng ký tự cần xóa: 1

+ Xóa các ký tự trắng cuối xâu: Vị trí: $s.lenth() - 1$; số lượng ký tự cần xóa: 1

+ Nếu có hai ký tự trắng liên tiếp trong xâu thì xóa đi 1 ký tự trắng: Vị trí:$s.find("\sqcup\sqcup")$; số lượng ký tự cần xóa: 1;

Lưu ý: $\sqcup$ thay cho ký hiệu ký tự trắng

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
	string s;
	getline(cin,s);
	while(s[0]==' ') s.erase(0,1);
	while(s[s.length()-1]==' ') s.erase(s.length()-1,1);
	while((int)s.find("  ")!=-1) s.erase((int)s.find("  "),1);
	cout<<s;
	return 0;
}
```

5. Dãy ngoặc đúng

Xét xâu s chỉ bao gồm các ký tự *ngoặc mở* '(' và *ngoặc đóng* ')'. Xâu s xác định một cách đặt ngoặc đúng nếu thỏa mãn các điều kiện:

+ Số *ngoặc mở* bằng số *ngoặc đóng*;

+ Nếu duyệt từ trái sang phải, số lượng *ngoặc mở* luôn lớn hơn hoặc bằng số lượng số *ngoặc đóng*.

Ví dụ xâu "(((()(())))" là một cách đặt ngoặc đúng, còn xâu "(()())(()))" là một cách đặt ngoặc sai.

Hãy viết chương trình nhập vào xâu s và kiểm tra xâu s có phải là một cách đặt ngoặc đúng hay không?

Ý tưởng:

Ban đầu xem như dãy ngoặc nhập vào là một dãy ngoặc đúng, sau đó tìm các trường hợp để đưa ra kết luận dãy ngoặc sai.

Ta sẽ xét lần lượt các ký tự từ đâu đến cuối xâu s đồng thời đếm số lượng ký tự *ngoặc mở* và *ngoặc đóng*, nếu tại một vị trí trong xâu s mà số lượng *ngoặc mở* nhiều hơn số lượng *ngoặc đóng* thì xâu s là một dãy ngoặc sai.

Khi xét hết xâu s mà số lượng *ngoặc mở* khác số lượng *ngoặc đóng* thì xâu s là một dãy ngoặc sai.

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
	string s;
	getline(cin,s);
	int n_mo=0,n_dong=0;
	int last=s.length()-1;
	string check="Day ngoac dung";
	for(int i=0;i<=last;i++)
	{
		if(s[i]=='(') n_mo++;
			else n_dong++;
		if(n_mo<n_dong) check="Day ngoac sai";
	}
	if(n_mo!=n_dong) check="Day ngoac sai";
	cout<<check;
	return 0;
}
```

6. Giá trị của xâu

Giá trị của xâu s chính bằng tổng giá trị của các ký tự số có trong xâu s, ví dụ $s = "1a2b"$ thì giá trị của xâu s là $1 + 2 = 3$.

Viết chương trình nhập vào xâu s rồi in ra màn hình giá trị của nó.

Ý tưởng:

Xét lần lượt từng ký tự trong xâu s, nếu ký tự đang xét là số thì cộng dồn giá trị của ký tự đó vào tổng.

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
	string s;
	getline(cin,s);
```

```
 7        int tong=0;
 8        int last=s.length()-1;
 9        for(int i=0;i<=last;i++)
10             if(s[i]<='9' and s[i]>='0')
11                  tong+=(int)s[i]-48;
12        cout<<tong;
13        return 0;
14   }
15
```

Giải thích:

+ Dòng 10: dùng để kiểm tra một ký tự có phải là số hay không;

+ Dòng 11: xác định giá trị của một ký tự số bằng cách ép kiểu ký tự về số nguyên sau đó trừ đi 48 là mã ASCII của số 0

7. Mã Xê Da

Để giữ bí mật người ta phải mã hóa các thông tin trước khi cần truyền đi hoặc lưu trữ. Một trong những cách được mã hóa sớm nhất được sử dụng rộng rãi trong thời cổ đại là cách mã hóa do Xê Da đề xuất: Trong thông điệp người ta thay mỗi chữ cái bằng các chữ cái đứng sau nó k vị trí trong bảng chữ cái. Việc tìm kiếm và thay thế được tiến hành vòng tròn trong bảng chữ cái. Nếu bảng chữ cái có n chữ thì sau chữ cái thứ $n-1$ là chữ cái thứ n, sau chữ cái thứ n là chữ cái thứ nhất…Cách mã hóa này được gọi là mã Xê Da. Các ký tự ngoài bảng chữ cái được giữ nguyên.

Ví dụ bảng chữ cái tiếng Anh có 26 chữ cái. Nếu $k = 2$ có nghĩa là $'a'$ được thay thế bằng $'c'$, $'b'$ được thay thế bằng $'d'$,…,$'y'$ được thay thế bằng $'a'$ và $'z'$ được thay thế bằng $'b'$. Các ký tự in hoa sẽ được thay thế bằng ký tự in hoa tương ứng, ví dụ "$TIN\ HOC$" sẽ được thay thế bằng "$VKP\ JQE$".

Hãy lập trình nhập vào xâu s và số nguyên dương k $(1 < k \leq 26)$ và in ra màn hình xâu mã hóa theo quy tắc mã Xê Da

Ý tưởng:

Với mỗi ký tự chữ cái trong xâu nhập vào ta sẽ thay thế ký tự đó bằng một ký tự khác theo quy tắc của mã Xê Da.

Trong C++ cho phép thực hiện cộng một ký tự với một số (vì bản chất của một ký tự là một số – mã ASCII của ký tự đó), vấn đề còn lại là cần xác định $s[i] + k$ là ký tự nào trong bảng chữ cái.

Cần tách riêng trường hợp ký tự in hoa và ký tự in thường để xử lý, cách xử ký tương tự nhau nên trong phần tiếp theo chỉ xét trường hợp ký tự in thường:

Giả sử thứ tự các ký tự trong bảng chữ cái là 0, 1, 2,…24, 25; giá trị $s[i] + k$ sẽ được đưa về số thứ tự trong bảng chữ cái bằng cách $s[i] + k - 97$ (97 là mã ASCII của ký tự $'a'$). Trong trường hợp $s[i] + k - 97 \leq 25$ thì kết quả là ký tự tương ứng với số thứ tự, ngược lại nếu $s[i] + k - 97 > 25$ (lớn hơn số thứ tự bảng chữ cái) thì cần phải quay về đầu bảng chữ cái bằng cách $s[i] + k - 97 - 26$;

Việc còn lại là đưa số thứ tự trong bảng chữ cái thành ký tự bằng cách ép kiểu số thứ tự (+ thêm 97) thành ký tự.

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
	int k;
	string s;
	getline(cin,s);
	cin>>k;
	int last=s.length()-1;
	for(int i=0;i<=last;i++)
		if(s[i]>='A' and s[i]<='Z')
		{
			int sott=s[i]-65+k;
			if(sott>25) sott-=26;
			s[i]=(char)(sott+65);
		} else
		if(s[i]>='a' and s[i]<='z')
		{
			int sott=s[i]-97+k;
			if(sott>25) sott-=26;
			s[i]=(char)(sott+97);
		}
	cout<<s;
	return 0;
}
```

BÀI 13: KIỂU DỮ LIỆU TỆP

1. Vai trò kiểu tệp

Tất cả các dữ liệu có các kiểu dữ liệu đã xét đều được lưu trữ ở bộ nhớ trong (RAM) và do đó dữ liệu sẽ bị mất khi tắt máy. Với một số bài toán, dữ liệu cần được lưu trữ để xử lí nhiều lần và với khối lượng lớn cần có kiểu dữ liệu tệp (file).

Kiểu dữ liệu tệp có những đặc điểm sau:

- Được lưu trữ lâu dài ở bộ nhớ ngoài (đĩa từ, CD,…) và không bị mất khi tắt nguồn điện;
- Lượng thông tin lưu trữ trên tệp có thể rất lớn và chỉ phụ thuộc vào dung lượng đĩa.

2. Phân loại tệp và thao tác với tệp

Xét theo cách tổ chức dữ liệu, có thể phân tệp thành hai loại:

+ *Tệp văn bản* là tệp mà dữ liệu được ghi dưới dạng các kí tự theo mã ASCII. Trong tệp văn bản, dãy kí tự kết thúc bởi nhóm kí tự xuống dòng hay kí tự kết thúc tệp tạo thành một dòng.

Các dữ liệu dạng văn bản như sách, tài liệu, bài học, giáo án, các chương trình nguồn viết bằng ngôn ngữ bậc cao,... thường được lưu trữ dưới dạng tệp văn bản.

+ *Tệp có cấu trúc* là tệp chứa dữ liệu được tổ chức theo một cách thức nhất định. Dữ liệu ảnh, âm thanh,... thường được lưu trữ dưới dạng tệp có cấu trúc.

Xét theo cách thức truy cập, có thể phân tệp thành hai loại:

+ *Tệp truy cập tuần tự* cho phép truy cập đến một dữ liệu nào đó trong tệp chỉ bằng cách bắt đầu từ đầu tệp và đi qua lần lượt tất cả các dữ liệu trước nó.

+ *Tệp truy cập trực tiếp* cho phép tham chiếu đến dữ liệu cần truy cập bằng cách xác định *trực tiếp* vị trí (thường là số hiệu) của dữ liệu đó.

Khác với mảng, số lượng phần tử của tệp không xác định trước.

Hai thao tác cơ bản đối với tệp là ghi dữ liệu vào tệp và đọc dữ liệu từ tệp.

Thao tác đọc/ghi với tệp được thực hiện với từng phần tử của tệp.

Để có thể thao tác với kiểu dữ liệu tệp, người lập trình cần tìm hiểu cách thức mà ngôn ngữ lập trình cung cấp cách:

+ Khai báo biến tệp;

+ Mở tệp;

+ Đọc/ghi dữ liệu;

+ Đóng tệp.

3. Thao tác với tệp

Ngôn ngữ lập trình cung cấp nhiều cách để làm việc với tệp, mỗi các có những ưu điểm khác nhau, tuy nhiên trong phạm vi tài liệu này chỉ giới thiệu cách đơn giản nhất là dùng $freopen()$

Lưu ý: trong phạm vi THPT chỉ xét đến loại tệp văn bản (truy cập tuần tự).

Thực hiện thao tác **mở tệp để đọc**, ta dùng cấu trúc:

$$freopen(<tên\ tệp>,"r",stdin);$$

Thực hiện thao tác **mở tệp để ghi**, ta dùng cấu trúc:

$freopen(<tên tệp>,"w",stdout);$

Trong đó:

+ $freopen$ là từ khóa

+ $<tên tệp>$ là một xâu, thường là hằng xâu, cho biết tên của tệp đang được xử lý

+ $"r"$ là viết tắt của read, chỉ thị đọc dữ liệu

+ "w" là viết tắt của write, chỉ thị ghi dữ liệu

+ $stdin$ là viết tắt của ***standard input***, nghĩa là thiết bị nhập chuẩn, ở đây là bàn phím.

+ $stdout$ là viết tắt của ***standard output***, nghĩa là thiết bị xuất chuẩn, ở đây là màn hình

Như vậy có thể hiểu lệnh $freopen(...)$ dùng để thông báo cho chương trình biết sẽ thay đổi thiết bị nhập/xuất chuẩn (bàn phím/màn hình) bằng tệp với tên được khai báo trong lệnh.

Sau khi dùng lệnh $freopen(<tên tệp>,"r",stdin);$ ta thực hiện các lệnh nhập dữ liệu bình thường (lệnh cin hoặc $getline$)

Sau khi dùng lệnh $freopen(<tên tệp>,"w",stdout);$ ta thực hiện các lệnh xuất dữ liệu bình thường (lệnh $cout$)

4. Một số ví dụ

Lưu ý: *Tất cả các tệp được mô tả đều có đặc điểm chung: các số trên 1 dòng được cách nhau bằng ít nhất một ký tự trắng.*

Ví dụ 1:

Cho tệp văn bản *maxvalue.inp* chứa 2 số nguyên a, b trên 1 dòng. Hãy tìm giá trị lớn nhất trong 2 số a, b rồi ghi kết quả vào tệp *maxvalue.out*

Ví dụ:

maxvalue.inp	maxvalue.out
3 5	5

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
      freopen("maxvalue.inp","r",stdin);
      freopen("maxvalue.out","w",stdout);
      int a,b;
      cin>>a>>b;
      int maxx=a;
      if(maxx<b) maxx=b;
      cout<<maxx;
      return 0;
}
```

Lưu ý:

+ Tệp *maxvalue.inp* cần phải có trên máy tính và nằm cùng thư mục với tệp chương trình nguồn (*.cpp). Trong tệp có chứa hai số nguyên;

+ Nếu bỏ dòng 5, 6 trong chương trình thì đây là chương trình tìm giá trị lớn nhất của hai số nguyên (tương tự như ví dụ 2, mục câu lệnh *if*)

+ Khi có dòng 5 thì *cin* sẽ đọc dữ liệu từ tệp *maxvalue.inp*

+ Khi có dòng 5 thì *cout* sẽ ghi vào tệp *maxvalue.out*

+ Kết quả không được ghi ra màn hình do vậy để xem kết quả người dùng cần mở tệp *maxvalue.out*

Ví dụ 2:

Cho tệp *dayso.inp* có cấu trúc:

+ Dòng đầu ghi số nguyên n $(1 \le n \le 1000)$

+ Dòng thứ hai ghi lần lượt các số nguyên $a_1, a_2, \dots, a_n$

Hãy tính và ghi vào tệp *dayso.out* các kết quả sau:

1. Trung bình cộng của dãy số $a_1, a_2, \dots, a_n$;

2. Các phần tử trong dãy có giá trị không nhỏ hơn trung bình cộng dãy số

Ví dụ:

Dayso.inp	Dayso.inp
5 1 3 2 6 5	3.4 6 5

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int a[1000];
int n;
int main()
{
      freopen("dayso.inp","r",stdin);
      freopen("dayso.out","w",stdout);
      cin>>n;
      for(int i=0;i<n;i++) cin>>a[i];
      double tbc=0;
      for(int i=0;i<n;i++) tbc+=a[i];
      tbc/=n;
      cout<<tbc<<endl;
      for(int i=0;i<n;i++)
            if(a[i]>=tbc) cout<<a[i]<<" ";
      return 0;
}
```

Thông thường trong các bài tập liên quan đến tệp đều cho trước cấu trúc cụ thể (cho biết trước bao nhiêu số trong tệp) như ví dụ này để ngươi dùng dễ dàng đọc dữ liệu từ tệp. Tuy nhiên cũng có những trường hợp tệp không có cấu trúc rõ ràng, xét ví dụ 3:

Ví dụ 3:

Cho tệp *songuyen.inp* gồm nhiều số (có không quá 10^6 số, mỗi số có giá trị tuyệt đối không quá 1000), hãy tính tổng các số trong tệp *songuyen.inp* rồi ghi kết quả vào tệp *songuyen.out*

Vấn đề ở bài toán này là không biết trước trong tệp có bao nhiêu số nguyên do vậy không thể sử dụng câu lệnh lặp như ví dụ 2. Mặc dù vậy ý tưởng của bài này vẫn là: "*đọc một số nguyên cho đến khi kết túc tệp*".

Chương trình tham khảo:

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   int main()
4   {
5         freopen("songuyen.inp","r",stdin);
6         freopen("songuyen.out","w",stdout);
7         int n,s=0;
8         while(cin>>n) s+=n;
9         cout<<s;
10        return 0;
11  }
```

Giải thích: Để ý ở dòng 8, *cin >> n* được đặt làm điều kiện của câu lệnh *while*, điều này vẫn đúng cú pháp vì *cin >> n* sẽ trả về *true* nếu xác định được một giá trị cho *n* ngược lại trả về *fasle*