AND : &&

OR : ||

NOT : !

Toán tử so sánh : > , < , <= , >= , == , !=

Break ; lệnh này để ngắt chương trình .  
boolalpha : lệnh này để thay đổi cách xuất dữ liệu của kiểu bool thành true và false thay vì 1 và 0

HÀM:

1.Abs( ) : giá trị tuyệt đối

2.Pow (a,b) : tính a^b ( a là cơ số , b là số mũ ) và trả về kết quả là số double

3.Sqrt (n) : tính căn bậc hai của một số và trả về kết quả là số double

4.Round (n) : làm tròn số thành số nguyên

Lưu ý : cho một hàm int a(int n) và int b = 100 nếu gán b cho n thì thay đổi n , b sẽ không bị thay đổi

Còn nếu int a(int &n) và int b = 100 nếu gán b cho n thì thay đổi n giá trị của b sẽ thay đổi theo giá trị của n

Các hàm gọi lẫn nhau ví dụ như hàm a gọi hàm b thì hàm b phải được xây dựng trước hàm a

Còn nếu như muốn xây dựng lộn xộn thì phải khai báo nguyên mẫu hàm

VD hàm a gọi hàm b thì phải khai báo nguyên mẫu hàm b trước hàm a

Int b(int)

Int a(int)

MỤC 1,2,3,4 : phải dùng thư viện cmath

5.Setprecision(n) : làm tròn số đến chữ số thập phân mong muốn ( nhưng phải dùng thư viện iomanip)

Lệnh #define dùng để định nghĩa một dữ liệu nào đó

Chú ý #define luythua(a) a\*a vd như luythua(3+3) thì luythua(3+3) = 3+3 \* 3+3 chứ ko phải (3+3) \* (3+3 )

Các lệnh cơ bản của git

git init

git clone

git pull

git add và git add .

git commit

git push

git log

git log --decorate --graph –o

MẢNG

Cú pháp của mảng : tên mảng[ số lượng phần tử có trong mảng ] = {…}

Để truy cập vào phần tử trong mảng thì có cú pháp là tên mảng[index=chỉ số]

Chỉ số của mảng bắt đầu từ 0

Cách duyệt các phần tử trong mảng có thể dùng vòng lặp for() hoặc dùng for (each) vd như ta có mảng a[n] thì duyệt bằng for (int x : a) { cout << x }

Lưu ý for (each) chỉ dùng được khi số lượng phần tử trong mảng bằng đúng với số lượng phần tử mà ng dùng nhập từ bản phím