* Implicit type conversion: Defind:
* Ép kiểu ngầm đinh là compiler nó tự chuyển đổi các kiểu dữ liệu sao cho phù hợp để có thể compile được. Thường chỉ áp dụng với các kiểu dữ liệu cơ bản: số thực, số nguyên, luận lý.
* Các lưu ý khi ép kiểu ngầm định:

1. Cẩn thận bị tràn dữ liệu. Vd:

Int N=200;

Char C=N;

Cout<<C;

* Số được in ra màn hình sẽ không xác định vì kiểu char(-128, 127). Nên kết quả sẽ sai.

1. Cẩn thận khi ép kiểu lớn vào kiểu nhỏ(sẽ xảy ra sai số):

Float f=0,123456789 (đây là kiểu double);

Cout<<f;

* Số được in ra màn hình: 0,123456791, nó sẽ sai số từ số thứ 7 trở đi, vì ở kiểu float chỉ chính xác ở 7 chữ số đầu.

1. Khi ép số thực vào số nguyên sẽ dẫn đến mất phần thực.

Float f=4,5;

Int N=f;

Cout<<N;

* Màn hình sẽ in ra: 4, Phần thực 0.5 bị mất.
* Khi ép với nhiều type:

1. Nếu có trong đó int là kiểu dữ liệu lớn nhất thì compiler sẽ chuyển tất cả sang int. (promotions)

VD: Char c=4;

Short t=5;

Byte b= 6;

Cout<<typeid(b+c+s).name()<<” “<<b+c+s;

* Kết quả sẽ ra: int 15 => chuyển sang kiểu int.

1. Còn nếu có kiểu dữ liệu lớn hơn int thì nó sẽ chuyển sang kiểu dữ liệu được ưu tiên lớn nhất (type có miền rộng nhất).

Bảng ưu tiên(increase L-R): int, unsigned int, long, unsigned long, long long, unsigned long long, float, double, long double.(cứ miền nào lớn hơn thì thứ tự ưu tiên cao hơn, và Z<R).

VD: char C=10;

Int N= 1000;

Long L=574;

Cout<<typeid(C+N+L).Name()<<” “<<C+N+L;

* Ra: Long 1584, vì Long có thứ tự ưu tiên cao hơn (miền lớn hơn).
* Syntax:

-) Cout<<typeid(Số cần biết kiểu(có thể sử dụng biểu thức)).name(), syntax này có thể chỉ là syntax bản thiếu.

Implicit type conversion:

* Chủ yếu ở cú pháp:
* Syntax: (data type)variable. VD:

Int N, M;

Float S=M/N;=> 1.

Explicit: C:

Float S=(float )M/N or S=M/N.

Notes: chỉ khi chạy lên(run on máy tính) mới biết có sai k.

* Std::Cout<<static\_cast<data type>(variable).

Notes: Máy tự quét lỗi.