

Let's solve it

12

Type Casting (Data Type casting)  
Conversion of interpretation w.r. to  
memory bytes.

Implicit (Internally happens by  
itself)

↳ downgrade	↳ upgrade
4 bytes → 2 bytes	2 bytes → 4 bytes
data loss (truncation)	15 data → 15.00 data

Explicit (on purpose specially done)

Typecasting

15

|

10

Integer  
Arithmetic

$$\begin{array}{r} 15 \\ 10 \overline{) 15} \\ \underline{10} \\ 5 \end{array}$$

int a = 15;

int b = 10;

a / b.00X

a / (float) b

15 / 10.0

15 / (float) 10

Explicit  
Typecast

You could always do

```
int a = 5;  
float b = 100;
```

Here, b is  
float through.  
which might be  
~~more than~~  
required.

a / b ; ~~Float Arithmetic~~  
mixed mode

↑ float / b float

Overall arithmetic will  
be float arithmetic.

In programming  
there is a rule  
"No more, No less"

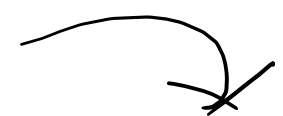
---

Your program can be future ready.

i.e. define PI 3.14159

# Application of Explicit Type Casting.

Imagine you are given  
memory bytes.

void to  (char)  
You can read one char

or

one int (int)

or

one double as per

(double)

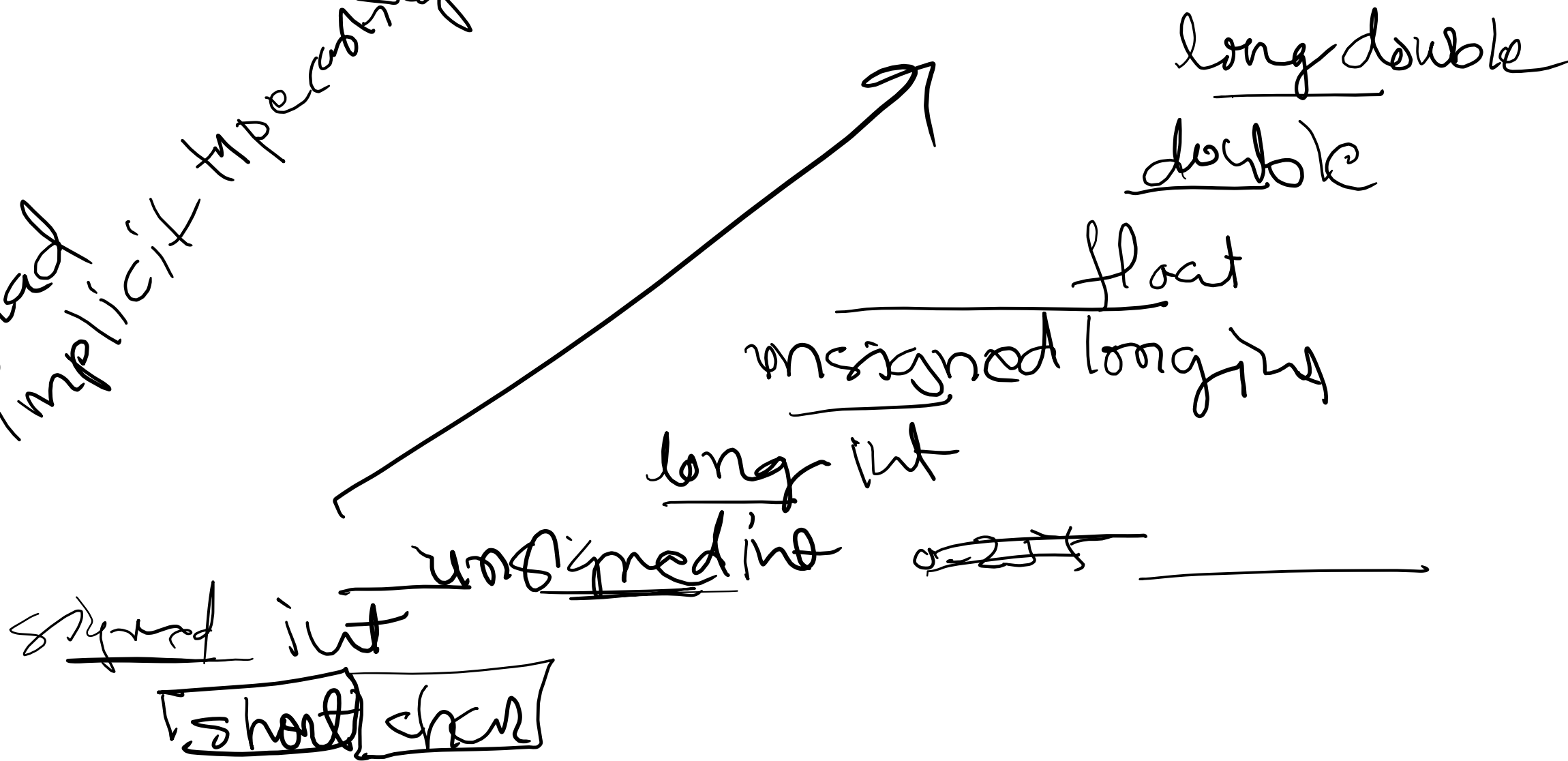
your choice.

How will you do this?

Explicit Typecasting

# Conversion hierarchy

upgrad  
of implicit typecasting



```
int a;  
float b = 10.50;
```

```
a = b; // implicit downgrade  
a = (int) b; // explicit downgrade  
           (not required)
```



$$\text{Sum} = \sum_{i=1}^n \left( \frac{1}{i} \right)^*$$

Develop a C program to find sum.

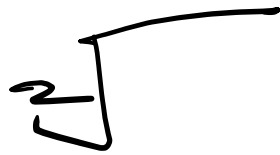
\* Integer Vs float arithmetic

1 + 0 + 0 + 0 + 0 \_\_\_\_\_ ?

# Math functions

#include <math.h>

sqrt



square root

man 3 sqrt

$2^{1/2}$

pow(2, 1/2) ?

gcc -lm -o c

gcc -o c

-lm

          \*

towards  
the end  
of command

Managing Input & output operations

I/O

character }  
integer }  
real / }  
string ( sess 2)

— Formatted I/O

# Character Test Functions

Used for testing what type of character the data is

i.e — letter/alpha

— upper  
— lower

alphanumeric  
alphabet or  
numeric

— digit  
0 — 9

— space

— punctuation

— printable  
or not

#include <ctype.h>

Application of char test functions

→ password requirements/rules/  
policies for security.

Formatted Input/output

`scanf("%d",`

↑  
width

`printf("%d", n);`

