1.3b Type Conversion

Type Conversion

- When we write expressions involved data that involves two different data types , such as multiplying an integer and floating – point number, we need to perform a type conversion….

- Two type of conversions mainly

  - Implicit type conversion

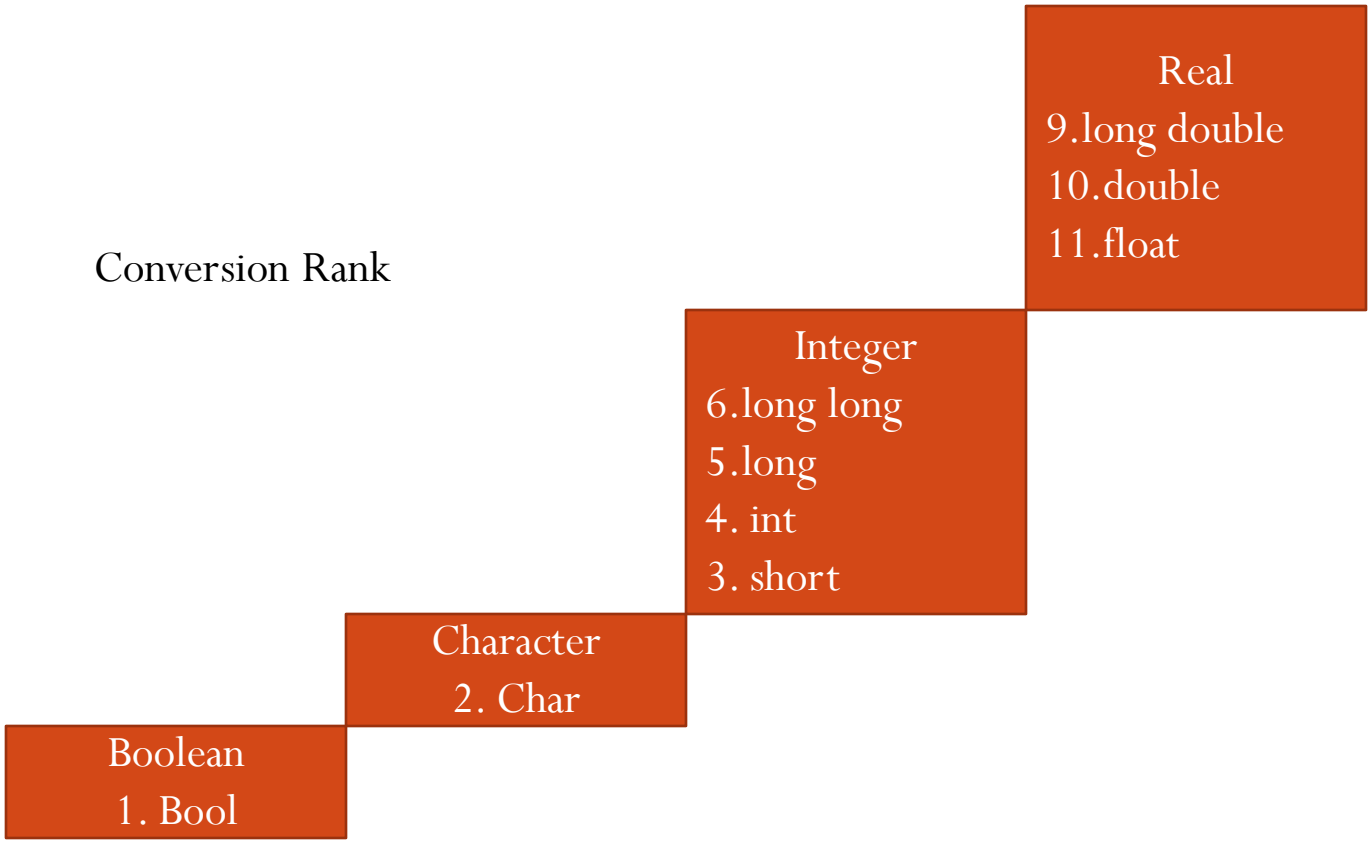
  - Explicit type conversion

Implicit Type Conversion

- When the types of the two operands in a binary expression are different, C automatically converts one type to another which is known as Implicit Type Conversion.
- Some of the simple conversions are :
  - Conversion Rank
  - Conversions in Assignment Expressions
  - Promotion
  - Demotion
  - Conversion in other Binary Expressions

Conversion Rank

- In C , we assign a rank to the integral and floating point arithmetic types



Conversions in Assignment Expressions

- A simple assignment involves an assignment operator and two operands
- Depending on the difference in the rank, C tries to either promote or demote the right expression to make it the same rank as the left variable.
- Promotion occurs if the right expression has lower rank
- Demotion occurs if the right expression has a higher rank

Promotion

- The rank of the left expression is elevated to the rank of the left variable
- The value of the expression is the value of the right expression after the promotion

bool b = true;	
char c = 'A';	
int i = 1234;	
long double d = 3458.0004	
c = b;	//Value of c is SOH
i = c;	//Value of i is 65
d = b;	//Value of d is 1.0
d = i;	//Value of d is 1234.0

Demotion

- If the size of the variable at the left side can accommodate the value of the expression, there is no problem
- An integral or real value can be assigned to a Boolean Type.
- If the value of the expression on the right is zero, false(0) is stored. If the result is not zero, either positive or negative, true (1) is stored.
- When an integer or a real is assigned to a variable of type character, the least significant byte of the number is converted to a character and stored.
- When a real is stored in an integer, the fraction part is dropped
- If the integral part is larger than the maximum value that can be stored, the results are invalid and unpredictable
- When to store a long double in a variable of type float, the results are valid if the value fits or invalid if it is too large

Example for demotion

bool b = false;	
char c =‘A’;	
short s = 78;	
int j = INT_MAX;	
int k = 65;	
…	
b = c ;	//Value of b is 1 (true)
s = j;	// value of s is unpredictable
c = k + 1;	// demotion: value of c is ‘B’

Conversion with other Binary Expressions

1. The operand with the highest rank is determined using the highest ranking mentioned above
2. The low-ranked operand is promoted to the rank defined in step 1. After the promotion, both expressions have the same rank
3. The operation is performed with the expression value having the type of the promoted rank

bool b = true;	
char c ='A';	
int i = 3650;	
short s = 78;	
long double d= 3458.0004;	
b + c ;	//b promoted ; result is 'B' ('A'+1)
i * s;	// the result is an int
d * c;	//result is long double

Explicit Type Conversion

- It is not done by compiler, instead the data from one to another is converted using explicit type conversion
- Explicit type conversion uses the unary cast operator
- To cast data from one type to another, we specify the new type in parentheses before the value we want converted
- To convert an integer a to a float: (float) a
- The value stored is still of type integer, but the value of the expression is promoted to float
- *average = (float) totalscores / numscores;*
- //there is an explicit conversion of totalscores to float and then an implicit conversion of numscores so that it will match
- *(float) (a/10)* will give the result *0.0*, so need to be written as

(float) a / 10

Example :Cast operator

- #include <stdio.h>

```
main()
{
 int sum = 17,
count = 5;
double mean;
 mean = (double) sum / count;
printf("Value of mean : %f\n", mean );
}
```

When the above code is compiled and executed, it produces the following result −Value of mean : 3.400000

Thankyou!!!!