Type Casting in C

In *computer science*, **type conversion** or **typecasting** refers to changing an entity of one *datatype* into another. There are two types of conversion: implicit and explicit. The term for implicit type conversion is **coercion**. Explicit type conversion in some specific way is known as **casting**.

Implicit type conversion

Implicit type conversion, also known as **coercion**, is an automatic type conversion by the **compiler**.

For example:

```
double  d;
long    l;
int     i;
if (d > i)      d = i;
if (i > l)      l = i;
if (d == l)     d *= 2;
```

is legal in a *C language* program. Although *d*, *l* and *i* belong to the different datatypes, they will be automatically converted into the same datatype each time a comparison or assignment is executed.

```
int x;
for(x=97; x<=122; x++)
{
    printf("%c", x);   /*Implicit casting from int to char thanks to %c*/
}
```

Explicit type conversion

There are several kinds of explicit conversion.

- checked

  before the conversion is performed a runtime check is done to see if the destination type can actually hold the source value. If not an error condition is raised.

- unchecked

  no check is performed and when the destination type can not hold the source value the result is undefined.

- bit pattern

  The data is not interpreted at all and just the raw bit pattern is copied.

Each programming language has its own rules on how types can be converted. In general, both objects and fundamental data types can be *converted*.

```
int x;
for(x=97; x<=122; x++)
{
    printf("%c", (char)x);   /*Explicit casting from int to char*/
}
```

in C

A **cast**, or *explicit type conversion*, is special programming instruction which specifies what data type to treat a variable as (or an intermediate calculation result) in a given expression.

Casting will ignore "extra" information (but never adds information to the type being casted). The C **cast** is either "unchecked" or "bit pattern".

As an example with fundamental data types, a fixed-point float could be cast as an integer, where the data beyond the decimal (or binary) point is ignored. Alternatively, an integer could be cast as a float if, for example, a function call required a floating point type (but, as noted, no information is really added - 1 would become 1.0000000).

C-style casting

This style of casting is used in C and Java. It follows the form:

```
(type)expression
```

Example Program

```
// An example of implicit conversion
#include<stdio.h>
int main()
{
    int x = 10;    // integer x
    char y = 'a';  // character c

    // y implicitly converted to int. ASCII
    // value of 'a' is 97
    x = x + y;

    // x is implicitly converted to float
    float z = x + 1.0;

    printf("x = %d, z = %f", x, z);
    return 0;
}
```