Introduction to C++ (Continues)

Typecasting:

Type conversion is used to avoid data loss in C++. There are two types of typecasting- Implicit typecasting and explicit typecasting.

Implicit typecasting:

- Here the compiler automatically converts one datatype to another datatype.
- In this all datatypes in an expression are upgraded to the datatype of the variable with the largest datatype.

```
bool -> char -> short int -> int ->

unsigned int -> long -> unsigned ->

long long -> float -> double -> long double
```

- One disadvantages of this type of conversion is that the sign (Due to the implicit conversion to unsigned) may be lost and overflow may occur.

Example:

```
// An example of implicit conversion

#include <iostream>
using namespace std;

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;

    cout << "x = " << x << endl
         << "y = " << y << endl
```

```
                << "z = " << z << endl;

            return 0;
        }
```

Explicit Typecasting:

In this user can explicitly convert the datatype of a variable to another datatype.

Example:

```
// An example of Explicit conversion

#include <iostream>
using namespace std;

int main ()
{
      int y=1001, j=365, n;
      n = (y-1) *j; /* The results in wrong answer since integer
                       range is exceeded on multiplication */
      n = (y-1) *(long)j; // C syntax (also supported by C++)
      n = (y-1) *long(j); // C++ syntax
      return 0;
}
```

Scope Resolution Operator:

Scope Resolution Operator is used to resolve the conflict between the local and the global variable.

Example:

```
// An example of Explicit conversion
#include <iostream>
using namespace std;
int I = 10;

int main ()
{
      int I=20;
      cout<<endl<<i<<"\t"<::i<<endl;
      {
            int I=30;
            cout<<endl<<i<<"\t"<::i<<endl;

      }
      cout<<endl<<i<<"\t"<::i<<endl;
      return 0
}
```