C++ CASTING OPERATORS

A cast is a special operator that forces one data type to be converted into another. As an operator, a cast is unary and has the same precedence as any other unary operator.

The most general cast supported by most of the C++ compilers is as follows:

```
(type) expression
```

Where type is the desired data type. There are other casting operators supported by C++, they are listed below:

- **const_cast<type>** ***expr*****:** The const_cast operator is used to explicitly override const and/or volatile in a cast. The target type must be the same as the source type except for the alteration of its const or volatile attributes. This type of casting manipulates the const attribute of the passed object, either to be set or removed.

- **dynamic_cast<type>** ***expr*****:** The dynamic_cast performs a runtime cast that verifies the validity of the cast. If the cast cannot be made, the cast fails and the expression evaluates to null. A dynamic_cast performs casts on polymorphic types and can cast a A* pointer into a B* pointer only if the object being pointed to actually is a B object.

- **reinterpret_cast<type>** ***expr*****:** The reinterpret_cast operator changes a pointer to any other type of pointer. It also allows casting from pointer to an integer type and vice versa.

- **static_cast<type>** ***expr*****:** The static_cast operator performs a nonpolymorphic cast. For example, it can be used to cast a base class pointer into a derived class pointer.

All of the above-mentioned casting operators will be used while working with classes and objects. For now, try the following example to understand a simple cast operators available in C++. Copy and paste the following C++ program in test.cpp file and compile and run this program.

```
#include <iostream>
using namespace std;

main()
{
   double a = 21.09399;
   float b = 10.20;
   int c ;

   c = (int) a;
   cout << "Line 1 - Value of (int)a is :" << c << endl ;

   c = (int) b;
   cout << "Line 2 - Value of (int)b is  :" << c << endl ;

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Line 1 - Value of (int)a is :21
Line 2 - Value of (int)b is  :10
```