A class declared inside a function becomes local to that function and is called Local Class in C++. For example, in the following program, Test is a local class in fun().

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

void fun()
{
      class Test
      {

      };
}

int main()
{
    return 0;
}
```

Following are some interesting facts about local classes.
1) *A local class type name can only be used in the enclosing function.* For example, in the following program, declarations of t and tp are valid in fun(), but invalid in main().

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

void fun()
{

      class Test
      {

      };

      Test t;
      Test *tp;
}

int main()
{
    Test t;
    Test *tp;
    return 0;
}
```

2) *All the methods of Local classes must be defined inside the class only.* For example, program 1 works fine and program 2 fails in compilation.

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

void fun()
{
    class Test
    {
    public:


       void method() {
          cout << "Local Class method() called";
       }
    };

    Test t;
    t.method();
}

int main()
{
    fun();
    return 0;
}
```

Output:

```
Local Class method() called
```

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

void fun()
{
    class Test
    {
    public:
        void method();
    };


    void Test::method()
    {
        cout << "Local Class method()";
    }
```

```
}

int main()
{
    return 0;
}
```

Output:

```
Compiler Error:
 In function 'void fun()':
 error: a function-definition is not allowed here before '{' token
```

3) *A Local class cannot contain static data members. It may contain static functions though.* For example, program 1 fails in compilation, but program 2 works fine.

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

void fun()
{
      class Test
      {
         static int i;
      };
}

int main()
{
    return 0;
}
```

```
Compiler Error:
 In function 'void fun()':
 error: local class 'class fun()::Test' shall not have static data member 'int
fun()::Test::i'
```

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

void fun()
{
    class Test
    {
    public:
        static void method()
        {
            cout << "Local Class method() called";
```

```
        }
    };

    Test::method();
}

int main()
{
    fun();
    return 0;
}
```

Output:

```
Local Class method() called
```

4) *Member methods of local class can only access static and enum variables of the enclosing function. Non-static variables of the enclosing function are not accessible inside local classes.* For example, the program 1 compiles and runs fine. But, program 2 fails in compilation.

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

void fun()
{
      static int x;
      enum {i = 1, j = 2};


      class Test
      {
        public:
          void method() {
              cout << "x = " << x << endl;
              cout << "i = " << i << endl;
          }
      };

      Test t;
      t.method();
}

int main()
{
    fun();
    return 0;
}
```

Output:

```
x = 0
i = 1
```

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

void fun()
{
      int x;


      class Test
      {
        public:
          void method() {
              cout << "x = " << x << endl;
          }
      };

      Test t;
      t.method();
}

int main()
{
    fun();
    return 0;
}
```

Output:

```
In member function 'void fun()::Test::method()':
error: use of 'auto' variable from containing function
```

5) *Local classes can access global types, variables and functions. Also, local classes can access other local classes of same function.*. For example, following program works fine.

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

int x;

void fun()
{
```

```
        class Test1 {
        public:
           Test1() { cout << "Test1::Test1()" << endl; }
        };


        class Test2
        {

             Test1 t1;
        public:
            void method() {

                cout << "x = " << x << endl;
            }
        };

        Test2 t;
        t.method();
}

int main()
{
    fun();
    return 0;
}
```

Output:

```
Test1::Test1()
x = 0
```