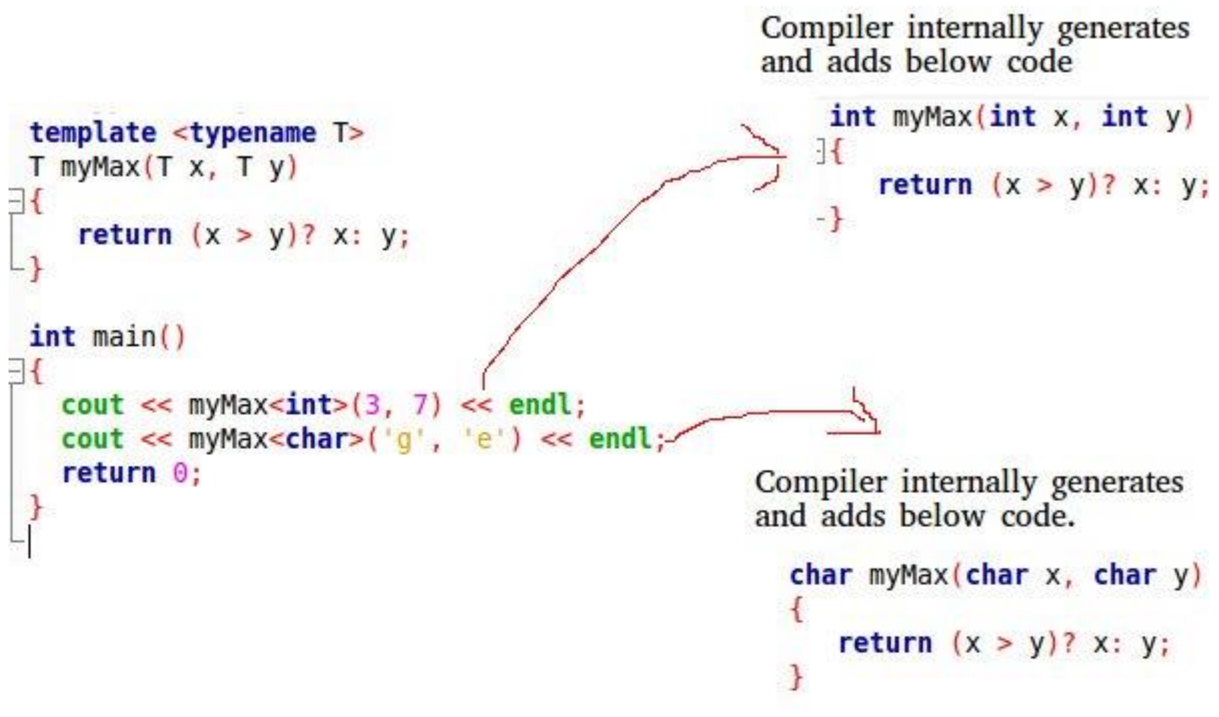
Function Templates

Function Templates We write a generic function that can be used for different data types. Examples of function templates are sort(), max(), min(), printArray().

Function Template:

- Generic functions use the concept of a function template. Generic functions define a set of operations that can be applied to the various types of data.
- The type of the data that the function will operate on depends on the type of the data passed as a parameter.
- For example, Quick sorting algorithm is implemented using a generic function, it can be implemented to an array of integers or array of floats.
- A Generic function is created by using the keyword template. The template defines what function will do.


Compiler internally generates and adds below code
Compiler internally generates and adds below code.

```
template <typename T>
T myMax(T x, T y)
{
    return (x > y)? x: y;
}

int main()
{
  cout << myMax<int>(3, 7) << endl;
  cout << myMax<char>('g', 'e') << endl;
  return 0;
}
```

```
int myMax(int x, int y)
{
    return (x > y)? x: y;
}
```

```
char myMax(char x, char y)
{
    return (x > y)? x: y;
}
```

- **Template** is a simple yet very powerful tool in C++.
- The simple idea is to pass data type as a parameter so that we don't need to write the same code for different data types.
- For example, a software company may need to sort() for different data types.
- Rather than writing and maintaining multiple codes, we can write one sort() and pass data type as a parameter.
  C++ adds two new keywords to support templates: 'template' and 'typename'.
- The second keyword can always be replaced by the keyword 'class'.

#include <iostream>

```
using namespace std;

// One function works for all data types.  This would work
// even for user defined types if operator '>' is overloaded
template <typename T> T myMax(T x, T y)
{
   return (x > y) ? x : y;
}

int main()
{
   cout << myMax<int>(3, 7) << endl; // Call myMax for int
   cout << myMax<double>(3.0, 7.0)
        << endl; // call myMax for double
   cout << myMax<char>('g', 'e')
        << endl; // call myMax for char

   return 0;
}
```

Output

```
7
7
g
```