C++- Templates

Introduction

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

There is a single definition of each container, such as vector, but we can define many different kinds of vectors for example, vector <int> or vector <string>.

Function Template

The general form of a template function definition is shown here:

```
template <class type> ret-type func-name(parameter list) {
   // body of function
}
```

Here, type is a placeholder name for a data type used by the function. This name can be used within the function definition.

Class Template

Just as we can define function templates, we can also define class templates. The general form of a generic class declaration is shown here:

```
template <class type> class class-name {
    .
    .
    .
}
```

Here, type is the placeholder type name, which will be specified when a class is instantiated. You can define more than one generic data type by using a comma-separated list.