

Section 19 : Template Functions and Classes

15 April 2022 16:49

© Rajat Kumar

<https://www.linkedin.com/in/imRajat/>

<https://github.com/im-Rajat>

Section 19 : Template Functions and Classes

Templates :

- Templates are used for generic programming.
- Generalization is based on the data type.

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

maximum(10, 15);
// (12.5, 9.5);
```

- We can also use multiple data types.

```
template<class intT, class doubleR>
void add(T x, R y)
{
    cout << x + y;
}

add(10, 12.9);
```

- Template on Class :

```
template<class T>
class stack
{
private:
    T s[10];
    int top;
public:
    void push(T x);
    T pop();
};
```

- Functions of Class Stack :

```

template<class T>
void Stack<T>::push(T x)
{
    =
}

template<class T>
T Stack<T>::pop()
{
    =
}
}

```

- For class we have to write template. For every function, when we are implementing outside using scope resolution, we must use a template.
- For creating object of class stack, we need to mention the data type :
 - Stack<int> s;
 - Stack<float> s2;
- Whenever a new body start {...}, We have to mention that template again.
 - Template<class T>
- Whenever we use class name, we have to write <T>.
 - void Stack<T>::push(T x) {...}
- Template is a very powerful feature for the collection framework.
- **Template parameter** : It can be used to pass a type as argument.
- Both class & typename keyword can be used in template.
- Validity of template parameters is inside that block only.