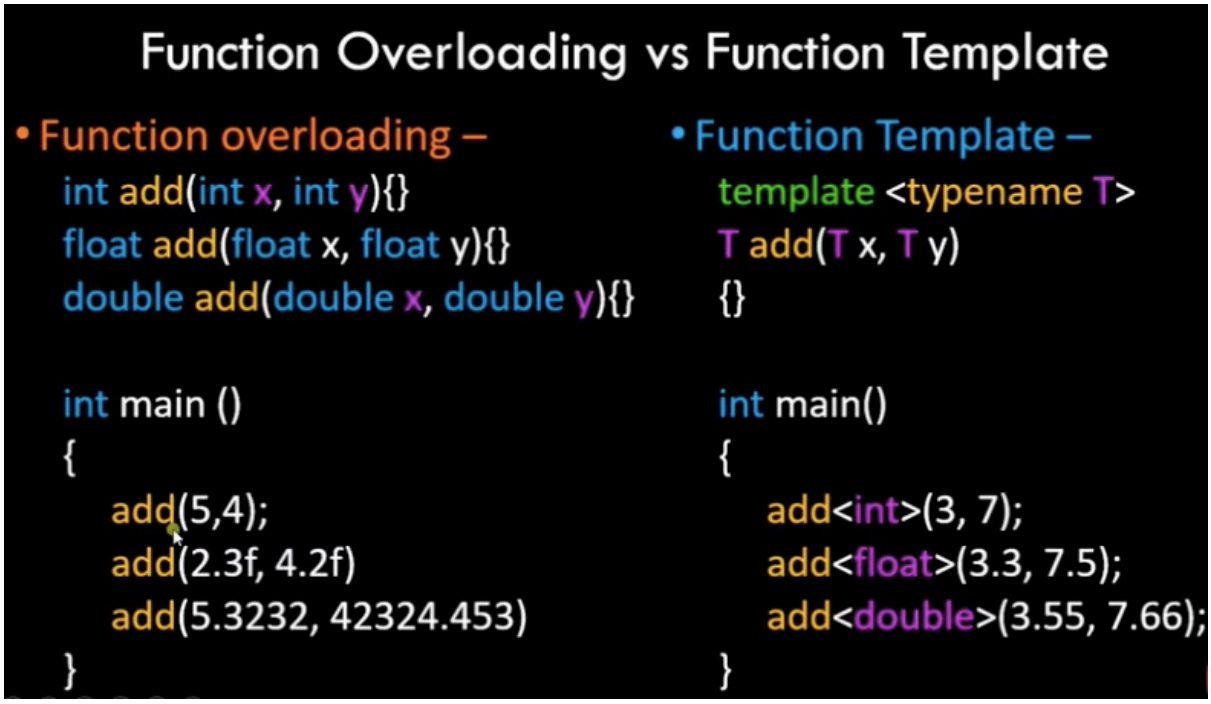
Templates in C++

- Template is simple and yet very powerful tool in C++
- 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.
- 2 Types –
  - Function Template
  - Class Template

Function Templates in C++

- Function templates are special functions that can operate with generic types.
- This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type.
- The simple idea is to pass data type as a parameter so that we don't need to write same code for different data types.
- We write a generic function that can be used for different data types.




```
#include<iostream>
using namespace std;
/*
int add(int x, int y)
{
        return(x+y);
}
float add(float x, float y)
{
        return(x+y);
}
double add(double x, double y)
{
        return(x+y);
}
*/

template <typename T>
T add(T x, T y)
{
        return(x+y);
}

int main()
{

cout<<add<int>(5,4)<<endl;
cout<<add<float>(2.3f,4.2f)<<endl;
cout<<add<double>(5.3223,443.222)<<endl;
return 0;
}
```

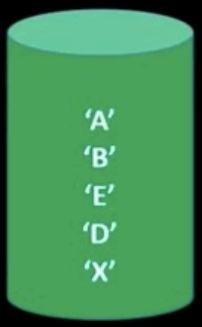
```
template <typename T, typename U>
U add(T x, U y)
{
        return(x+y);
}
```

Class Templates in C++

- Sometimes, you need a class implementation that is same for all classes, only the data types used are different.
- Normally, you would need to create a different class for each data type OR create different member variables and functions within a single class.
- In Class Templates We write a CLASS that can be used for different data types.

Example


```
class Stack
{
    public:
    int arr[5]
    private:
    push();
    pop();
}
```




```
class Stack
{
    public:
    char arr[5]
    private:
    push();
    pop();
}
```

```
#include<iostream>
using namespace std;
template <typename T, typename U>  // T is a datatype
class Weight
{
        private:
                T kg;
                U grams;
        public:
                void setData(T x, U y)
                {
                        kg = x;
                        grams = y;
                }
                T getData()
                {
                        return kg;
                }
                U getGramData()
                {
                        return grams;
                }
};

int main()
{
Weight <int,double>obj1;
obj1.setData(5,0.53);

cout<<"Value is "<<obj1.getData()<<endl;
cout<<"Value is "<<obj1.getGramData()<<endl;

/*
Weight <double>obj2;
obj2.setData(544.322);
cout<<"Value is "<<obj2.getData()<<endl;
*/

return 0;
}
```