Function Template & Class Template

Templates

- Templates are designed to create generic functionality.
- Template can be applied to class and function
  - Function Template (generic function)
  - Class Type (Generic UDT)
- Function template: A pattern for creating definitions of functions that differ only in the type of data they manipulate

Template for swapping

- Main purpose of function template is to reuse functionality for different data types.

```
#include<iostream>
using namespace std;
template<class T>
  void swap(T &x, T &y)
  { T temp = x; x = y;
    y = temp;
  }
int main()
{ int a=10, b=20;
 swap(a,b);
 cout<<"\n"<<a<<" "<<b;
float p=10.10, k=20.20;
swap(p,k);
 cout<<"\n"<<p<<" "<<k;
return 0; }
```

Summary of Function Template

- No actual code is generated until the function named in the template is called
- A function template uses no memory
- When passing a class object to a function template, ensure that all operators referred to in the template are defined or overloaded in the class definition
- Function templates can be overloaded – need different parameter lists
- Like regular functions, function templates must be defined before being called

Class Templates

- It is possible to define templates for classes.
- Unlike functions, a class template is instantiated by supplying the type at object definition/declaration

```
#include<iostream>
using namespace std;
 template <class T>
  class Joiner
  {
  public:
     T Combine(T x, T y)
        {return x + y;}
  };
```

```
int main()
{
Joiner<int> ij;
Joiner<float> cj;
cout<<"\n"<<ij.Combine(10,10);
cout<<"\n"<<cj.Combine(10.10,20.20);
return 0;
}
```

Class Templates and Inheritance

- Templates can be combined with inheritance
- Template class can be inherited from a template class
- Templates are widely used un writing data structure and generic functionality
- STL library of C++ internally uses class template

Thank You

Use me as much as possible!!!!

……. Templates