Templates

CS 110C

Anita Rathi

Topics

- Template Definition
- Template Functions
- Template Class

Templates in C++

- A function or class may be written more than once for different types of parameters or data members.
- Templates allow a generic function or generic class to be defined.
- Templates pass data type as a parameter so that you don't need to write the same code for different data types.
- Two keywords are used to define templates: *'template'* and *'typename'*.
- The *typename* keyword can be replaced by the keyword 'class'.

Function Templates in C++

- Generic functions are defined to be used for different data types.
- Sample Code – Templatefunc.cpp

```
template <typename T >
T large(T x, T y, T z){
  if (x>=y && x>=z)
    return x;
  else if (y>=x && y>=z)
    return y;
  else
    return z;
}

int main(){
  int a = 45,  b = 67, c = 50;
  double l = 5.7,  m = 6.8, n = 5.1;
  cout<<"Largest of integers = "<< large(a,b,c)<<endl;
  cout<<"Largest of doubles = "<< large(l,m,n)<<endl;
}
```

```
Largest of integers = 67
Largest of doubles = 6.8
```

Class Templates

- Templates enable the programmer to separate the functionality of an implementation from the type of data used in the class.
- //Sample Code **Box.cpp**

```
template <class ItemType>
class Box{
  private:
    ItemType item;
  public:
    void setItem(const ItemType& x){
      item = x;
    }
    ItemType getItem() const{
      return item;
    }
};

int main(){
  Box<long> longBox;
  Box<string> stringBox;

  longBox.setItem(769);
  stringBox.setItem("California");
  cout<<"Item of longBox = "<< longBox.getItem()<<endl;
  cout<<"Item of longBox = "<< stringBox.getItem()<<endl;
}
```

```
Item of longBox = 769
Item of longBox = California
```