Practical Lecture : Templates Day 2

Quick Recap

Let’s take a quick recap of previous lecture –

- Introduction to templates
- Function template
- class template

Today’s

Today we are going to cover –

- Inheritance in template class(single level)

Let’s Get Started-

Inheritance

Syntax: simple inheritance example without template class

```
Class baseClass
{
      Datamembers;
      member functions;
}

Class derivedClass: public baseClass
{
      Datamembers;
      member functions;
}
```

Inheritance

Syntax: simple inheritance example with template class

```
<template class T>
Class baseClass
{
      T Datamembers;
      member functions;
}
<template class T>
Class derivedClass: public baseClass<T>
{
      T Datamembers;
      member functions;
}
```

Inheritance example with template class

```
#include <iostream>
using namespace std;
template <class T>        //before baseclass definition, provide the
template <class Type>
class BaseClass {
 protected:
   T x;
  public:
    void setdata(T a)
    {
        x=a;
    }
    void display ()
    {
        cout<<"x ="<< x <<endl;
     }
```

Inheritance example with template class

```
template <class  T>     //before baseclass definition, provide the template
<class Type>
class Derived :public BaseClass<T>   //Here <writing <Type> at the end is
madatatory
{
    T z;
  public:
    void setZ( T b)
    {
        z=b;
    }
    void display ()
    {
      BaseClass<T>::display(); //whenever you access base class member,
mention <type>
      cout<<" z= "<<z;
```

Inheritance example with template class

```
int main () {
  Derived <int> D; //while creating objects, mentioning <type> is
important as it tells how many bytes to allocate for objects
  D.setdata(10);
  D.setZ(5);
  D.display();
  return 0;
}
```

Output:

```
x=10
z=5
```

Inheritance example with template class

```
int main () {
  Derived <int> D;
  D.setdata(10);
  D.setZ(5);
  D.display();
  return 0;
}
```

Output:

```
x=10
z=5
```

Inheritance with constructor

Observe how the constructor calls are made when working with templates. Note the text in bold letters.

```
#include<iostream>
using namespace std;

template<class t>
class base {
protected:
    t a;
public:
    base(t aa){
        a = aa;
        cout<<"base "<<a<<endl;
    }
};
```

Inheritance with constructor

```
template <class t>
class derived: public base<t>
{
    public:
        derived(t a): base<t>(a)
 {
        }
        //Here is the method in derived class
    void sampleMethod() {
        cout<<"In sample Method"<<endl;
    }
};
```

Note: Everytime you are referring to base class , you have to use base_class_name<type>
e.g. here in this case we always use base<t>

Inheritance with constructor

```
int main() {
    derived<int> q(1);
    // calling the methods
    q.sampleMethod();
}
```

Output:

```
base 1
In sample Method
```

Practice question

Which of the following is incorrect about in template inheritance?

1. The correct way of accessing base class members are using baseclassname<type>
2. You can access the base class using normal inheritance method
3. While creating objects of derived class, mention <datatype>, else complier will report an error
4. <template class Type> is mandatory before every class declaration.

Practice question

Which of the following is incorrect about in template inheritance?

1. The correct way of accessing base class members are using baseclassname<type>
2. You can access the base class using normal inheritance method
3. While creating objects of derived class, mention <datatype>, else complier will report an error
4. <template class Type> is mandatory before every class declaration.

Any
Questions ??

Thank You!

See you guys in next class.