**UNIT-6 TEMPLATE**

*Template* is one of the features added to C++ recently, it is a new concept which enable us to define generic classes and functions and thus provides support for generic programming. Generic programming is an approach where generic types are used as parameters in algorithms so that they work for a variety of suitable data types and structures.

A template can be used to create a family of classes or functions, for example. A class template for an **array** class would enable us to create arrays of various data types such as **int, array and float array** similarly, we can define a template for a function, say  **func1()** , that would help us create various versions of **func1()** for adding **int, float and double** type values.

Template can be considered as a kind of macro, when an object of a specific type is defined for actual use, the template definition for that class is substituted with the required data types at the time of actual use of the class or function, the temples are sometimes called *parameterized classes or functions.*

The general format of a class template is

tempalte<class T>

class classname

{

//class member specification with anonuymous type T

//wherever approprate

};

The class template definition is very similar to an ordinary class definition except the prefix **template<class T>** and the use of type T. this prefix tell s the compiler that we are going to declare a template and use as a type name in the declaration.

A class created from a class template is called a **template class.** The syntax for defining an object of a template class is

className<type> objectName(arglist);

this process of creating a specific class from a class template is called *instantiation.* The compiler will perform the error analysis only when an instantiation takes place. It is therefore, advisable to create and debug an ordinary class before converting it into template.

**#include<iostream>**

**#include<iostream.h>**

**#include<conio.h>**

**template<class T>**

**class ctemp**

**{**

**T \*v;**

**public:**

**int size;**

**int i;**

**ctemp()**

**{**

**size=3;**

**v=new T[size];**

**}**

**void get()**

**{**

**int i;**

**for(i=0;i<size;i++)**

**{**

**cout<<"\n enter numbers";**

**cin>>v[i];**

**}**

**}**

**void display()**

**{**

**for(i=0;i<3;i++)**

**{**

**cout<<v[i]<<endl;**

**}**

**}**

**};**

**int main()**

**{**

**ctemp<int>v1;**

**v1.get();**

**v1.display();**

**getch();**

**return 0;**

**}**

**CLASS TEMPLATE WITH MULTIPLE PARAMETERS**

**We can use more than one generic data type in a class template . they are declared as a comma separated list within the template specification as shown below.**

template <class T1, Class T2>

class classname

{

…………

…………

………..

};

For example

#include<iostream.h>

template <class T1, class T2>

class templatedemo

{

T1 a;

T2 b;

public:

templatedemo(T1 x, T2 y)

{

a=x;

b=y;

}

void show()

{

cout<<" a is "<<a;

cout<<"\n b is "<<b;

}};

void main()

{

templatedemo <float, int> ob1(10.20,20);

templatedemo <int,char> ob2(45,'z');

ob1.show();

ob2.show();

getch();

}

**Template function:** like template class, we can also define templates function that could be used to create a family of function with different argument types. The general format of a function templates is

Template < class T>

Return type function name(arguments of type T)

{

}

For example, we a program input two numbers and swap each other by using template function.

#include<iostream.h>

#include<conio.h>

template <class T>

void swap(T &x, T &y)

{

T temp;

temp= x;

x=y;

y=temp;

}

void main()

{

int a,b;

cout<<"enter two numbers ";

cin>>a>>b;

swap(a,b);

cout<<"swaped numbers are";

cout<<"\n a is "<<a;

cout<<"\n b is "<<b;

getch();

}

**Write a program input 10 numbers and sort them in ascending order**

**by using template function.**

#include<iostream.h>

#include<conio.h>

template<class T>

void sort(T a[], int size)

{

T temp;

int i,j;

for(i=0; i<size-1; i++)

{

for (j=i+1; j<size; j++)

{

if(a[i]>a[j])

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}}}

cout<<"sorted numbers are";

for(i=0;i<size;i++)

{

cout<<a[i];

}}

void main()

{

int num[10];

int size=10;

int i;

cout<<"\n enter 10 numbers";

for(i=0;i<size;i++)

{

cin>>num[i];

}

sort(num,size);

getch();

}