**What is ‘Template Class’ in C++**:

**“Template Class”** is an important feature of C++ which enables the coder to write **generic** functions or classes. In a **generic function or class**, the type of data (i.e: int, float, double, etc.) upon which the function or class operates is specified as a parameter.

**Why ‘Template Class’**?

By creating a templated class/ function, you can define the nature of your algorithm to be independent of any kind of data types.

Once you have written a templated code, your compiler will automatically generate the correct code for the type of data that is actually used when you execute the function.

**Format for writing a ‘Template Class’ in C++**

Remember the simple **DynamicArray** class we discussed in our **Lab-01** where we created a simple C++ class to create a dynamically allocated array for only holding integer type of values. If we convert that simple class into a templated class, then that class object will be able to hold any valid type of numeric values (int, float, double). Now, the format for writing a template function in C++ (in the source .cpp file) is as follows:

**template <class** ItemType**>**

***return-type*** Class\_Name**<**ItemType**>::**functionName(parameters)

{

// your code goes here

}

Now, if we convert the header file of that DynamicArray class to a templated version, it will be like as given below:

**dynamicarray.h**

**#ifndef DYNAMICARRAY\_H\_INCLUDED**

**#define DYNAMICARRAY\_H\_INCLUDED**

template <class ItemType**>**

class DynamicArray**{**

private:

ItemType\* data;

public:

DynamicArray(int);

~DynamicArray();

void insertItem(int, ItemType);

ItemType getItem(int);

**};**

**#endif**

If we convert the cpp file of that DynamicArray class to a templated version, it will be like as given below:

**dynamicarray.cpp**

**#include “dynamicarray.h”**

template <class ItemType>

DynamicArray<ItemType>::DynamicArray(int size)

{

data = new ItemType[size];

}

template <class ItemType>

void DynamicArray<ItemType>::insertItem(int index, ItemType item)

{

data[index] = item;

}

template <class ItemType>

ItemType DynamicArray<ItemType>::getItem(int index)

{

return data[index];

}

template <class ItemType>

DynamicArray<ItemType>::~DynamicArray()

{

delete[] data;

}

**Creating and using template class objects in the driver (main.cpp) file**:

**main.cpp**

**#include “dynamicarray.cpp”**

**#include <iostream>**

using namespace std;

**int main()**

**{**

int defaultSize = 3;

**// Creating and using a DynamicArray object**

**// dealing with integer type of data**

DynamicArray<int> intArray(defaultSize);

for (int index=0,data=10;index<3; index++, data += 10)

{

intArray.insertItem(index,data);

}

int temp;

cout<< “Integer Values: ”;

for(int index=0;index<3;index++)

{

temp = intArray.getItem(index);

cout<< temp<< “ ”;

}

cout<<endl;

**// Creating and using a DynamicArray object   
// dealing with char type of data**

DynamicArray<char> charArray(defaultSize);

for(int index=0, value = ‘A’; index<3; index++, value++)

{

charArray.insertItem(index,value);

}

char tempChar;

cout<< “Character type Values: ”;

for(int index=0; index<3; index++)

{

tempChar = charArray.getItem(index);

cout<< tempChar<<“ ”;

}

cout<<endl;

return 0;

**}**

**---------------------------------------**