Templates

Templates are powerful features of C++ which allows you to write generic programs. In simple terms, you can create a single function or a class to work with different data types using templates.

Templates are often used in larger codebase for the purpose of code reusability and flexibility of the programs.

The concept of templates can be used in two different ways:

- Function Templates
- Class Templates

Function Templates

A function template works in a similar to a normal function, with one key difference.

A single function template can work with different data types at once but, a single normal function can only work with one set of data types.

Normally, if you need to perform identical operations on two or more types of data, you use function overloading to create two functions with the required function declaration.

However, a better approach would be to use function templates because you can perform the same task writing less and maintainable code.

How to declare a function template?

A function template starts with the keyword template followed by template parameter/s inside < > which is followed by function declaration.

```
template <class T>
T someFunction(T arg)
{
   ... .. ...
}
```

In the above code, T is a template argument that accepts different data types (int, float), and class is a keyword.

You can also use keyword typename instead of class in the above example.

When, an argument of a data type is passed to someFunction( ), compiler generates a new version of someFunction() for the given data type.

Example 1: Function Template to find the largest number

Program to display largest among two numbers using function templates.

// If two characters are passed to function template, character with larger ASCII value is displayed.

```cpp
#include <iostream>

using namespace std;

// template function

template <class T>

T Large(T n1, T n2)

{

        return (n1 > n2) ? n1 : n2;

}


int main()

{

        int i1, i2;

        float f1, f2;

        char c1, c2;


        cout << "Enter two integers:\n";

        cin >> i1 >> i2;

        cout << Large(i1, i2) <<" is larger." << endl;


        cout << "\nEnter two floating-point numbers:\n";
```

```cpp
        cin >> f1 >> f2;
        cout << Large(f1, f2) <<" is larger." << endl;


        cout << "\nEnter two characters:\n";
        cin >> c1 >> c2;
        cout << Large(c1, c2) << " has larger ASCII value.";


        return 0;
}
```

Output

Enter two integers:

5

10

10 is larger.

Enter two floating-point numbers:

12.4

10.2

12.4 is larger.

Enter two characters:

z

Z

z has larger ASCII value.

In the above program, a function template Large() is defined that accepts two arguments n1 and n2 of data type T. T signifies that argument can be of any data type.

Large() function returns the largest among the two arguments using a simple conditional operation.

Inside the main() function, variables of three different data types: int, float and char are declared. The variables are then passed to the Large() function template as normal functions.

During run-time, when an integer is passed to the template function, compiler knows it has to generate a Large() function to accept the int arguments and does so.

Similarly, when floating-point data and char data are passed, it knows the argument data types and generates the Large() function accordingly.

This way, using only a single function template replaced three identical normal functions and made your code maintainable.

Class Templates

Like function templates, you can also create class templates for generic class operations.

Sometimes, you need a class implementation that is same for all classes, only the data types used are different.

Normally, you would need to create a different class for each data type OR create different member variables and functions within a single class.This will unnecessarily bloat your code base and will be hard to maintain, as a change is one class/function should be performed on all classes/functions.

However, class templates make it easy to reuse the same code for all data types.

How to declare a class template?

```
template <class T>
class className
{
  ... .. ...
public:
  T var;
  T someOperation(T arg);
  ... .. ...
};
```

In the above declaration, T is the template argument which is a placeholder for the data type used.

Inside the class body, a member variable var and a member function someOperation() are both of type T.

How to create a class template object?

To create a class template object, you need to define the data type inside a < > when creation.

className<dataType> classObject;

For example:

className<int> classObject;

className<float> classObject;

className<string> classObject;

Example 3: Simple calculator using Class template

Program to add, subtract, multiply and divide two numbers using class template


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

template <class T>
class Calculator
{
private:
        T num1, num2;

public:
        Calculator(T n1, T n2)
        {
```

```cpp
        num1 = n1;

        num2 = n2;

    }


    void displayResult()

    {

        cout << "Numbers are: " << num1 << " and " << num2 << "." << endl;

        cout << "Addition is: " << add() << endl;

        cout << "Subtraction is: " << subtract() << endl;

        cout << "Product is: " << multiply() << endl;

        cout << "Division is: " << divide() << endl;

    }


    T add() { return num1 + num2; }


    T subtract() { return num1 - num2; }


    T multiply() { return num1 * num2; }


    T divide() { return num1 / num2; }
};


int main()
{

    Calculator<int> intCalc(2, 1);

    Calculator<float> floatCalc(2.4, 1.2);
```

```
cout << "Int results:" << endl;

intCalc.displayResult();


cout << endl << "Float results:" << endl;

floatCalc.displayResult();


return 0;
}
```

Output

Int results:

Numbers are: 2 and 1.

Addition is: 3

Subtraction is: 1

Product is: 2

Division is: 2

Float results:

Numbers are: 2.4 and 1.2.

Addition is: 3.6

Subtraction is: 1.2

Product is: 2.88

Division is: 2

In the above program, a class template Calculator is declared.The class contains two private members of type T: num1 & num2, and a constructor to initalize the members.

It also contains public member functions to calculate the addition, subtraction, multiplication and division of the numbers which return the value of data type defined by the user. Likewise, a function displayResult() to display the final output to the screen.

In the main() function, two different Calculator objects intCalc and floatCalc are created for data types: int and float respectively. The values are initialized using the constructor.

Notice we use <int> and <float> while creating the objects. These tell the compiler the data type used for the class creation.

This creates a class definition each for int and float, which are then used accordingly.

Then, displayResult() of both objects is called which performs the Calculator operations and displays the output.

Questions:

1. Write a program that uses function templates to swap numbers.

2. Write a program that uses class templates to search for a particular number in an array. It should display the number and the position at which the number is found. (The array size can be of your choice).