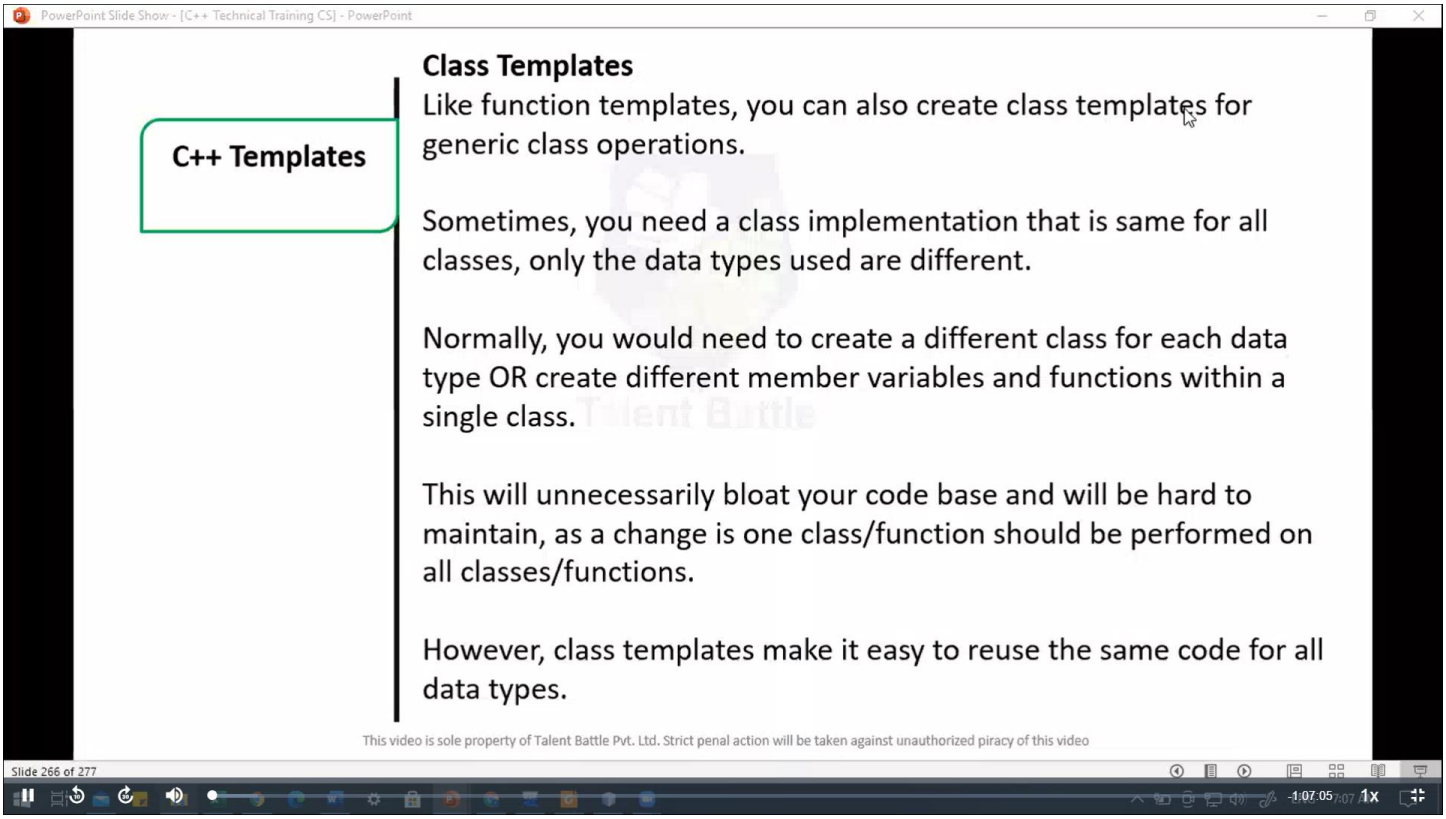
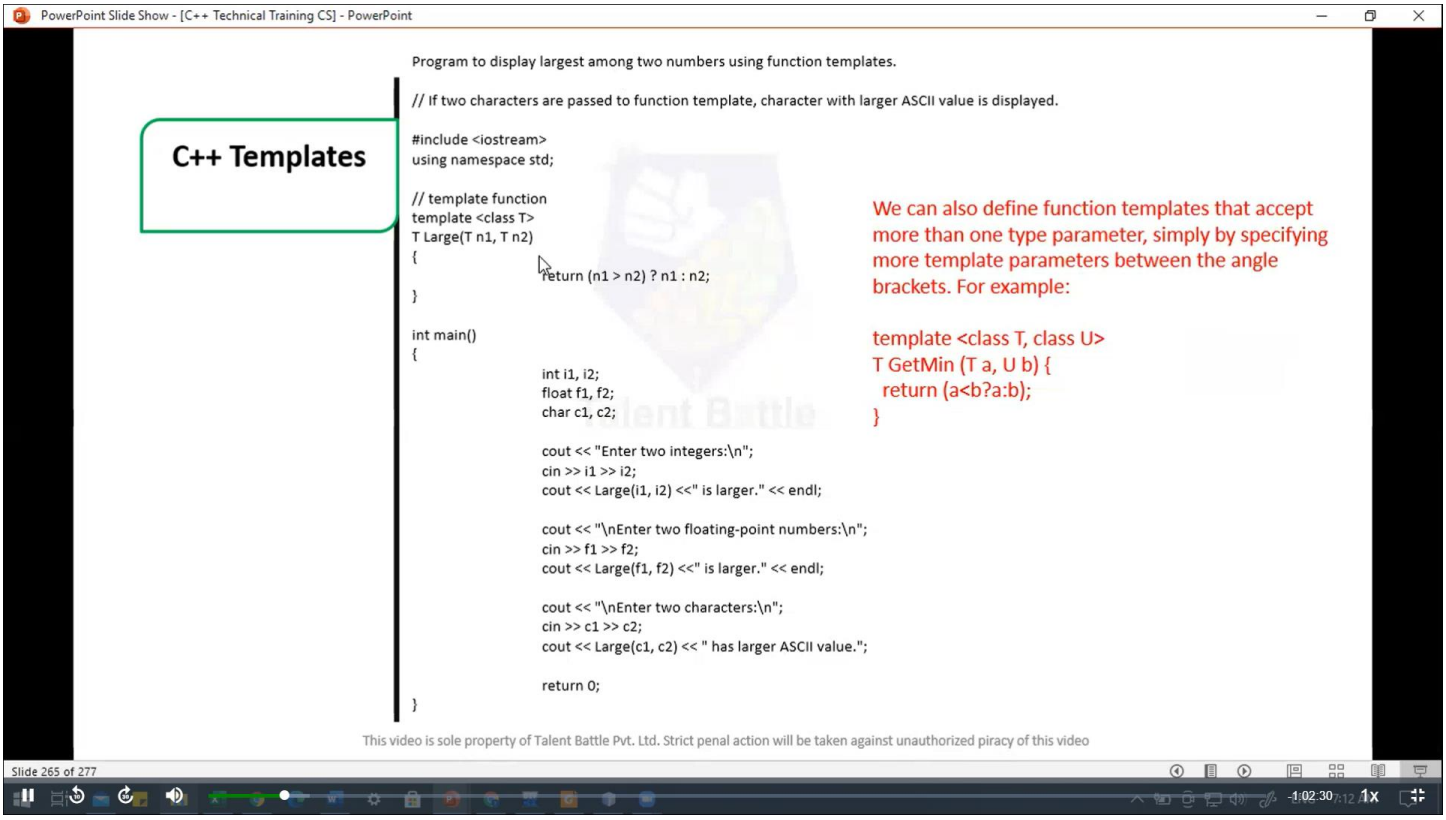
Day 9

C++ STL


PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint
C++ Templates
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.
This video is sole property of Talent Battle Pvt. Ltd. Strict penal action will be taken against unauthorized piracy of this video
Slide 266 of 277
-1:07:05
1x


PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint
C++ Templates
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.
#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";
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;
}
We can also define function templates that accept more than one type parameter, simply by specifying more template parameters between the angle brackets. For example:
template <class T, class U>
T GetMin (T a, U b) {
return (a<b?a:b);
}
This video is sole property of Talent Battle Pvt. Ltd. Strict penal action will be taken against unauthorized piracy of this video
Slide 265 of 277
-1:02:30
1x

C++ Templates

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.

C++ Templates

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;
```

```cpp
// 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){
      num1 = n1;
      num2 = n2;
    }

    void displayResult(){
      cout << "Number 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;
```

```
}


/*
Int results:
Number are: 2 and 1.
Addition is: 3
Subtraction is: 1
Product is: 2
Division is: 2

Float results:
Number are: 2.4 and 1.2.
Addition is: 3.6
Subtraction is: 1.2
Product is: 2.88
Division is: 2

--------------------------------
Process exited after 0.1501 seconds with return value 0
Press any key to continue . . .
*/
```

PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint

C++ STL

Introduction to STL: Standard Template Library

STL is an acronym for standard template library. It is a set of C++ template classes that provide generic classes and function that can be used to implement data structures and algorithms .

STL is mainly composed of :
1.Algorithms
2.Containers
3.Iterators

45:43 7:29 1x

PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint

C++ STL

STL provides numerous containers and algorithms which are very useful in completive programming , for example you can very easily define a linked list in a single statement by using list container of container library in STL , saving your time and effort.

STL is a generic library , i.e a same container or algorithm can be operated on any data types , you don't have to define the same algorithm for different type of elements.

For example , sort algorithm will sort the elements in the given range irrespective of their data type , we don't have to implement different sort algorithm for different datatypes.

44:01 7:31 1x

PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint

C++ STL

C++: Containers in STL

Container library in STL provide containers that are used to create data structures like arrays, linked list, trees etc.

These container are generic, they can hold elements of any data types, for example: vector can be used for creating dynamic arrays of char, integer, float and other types.

39:53 / 1:35:35 1x

PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint

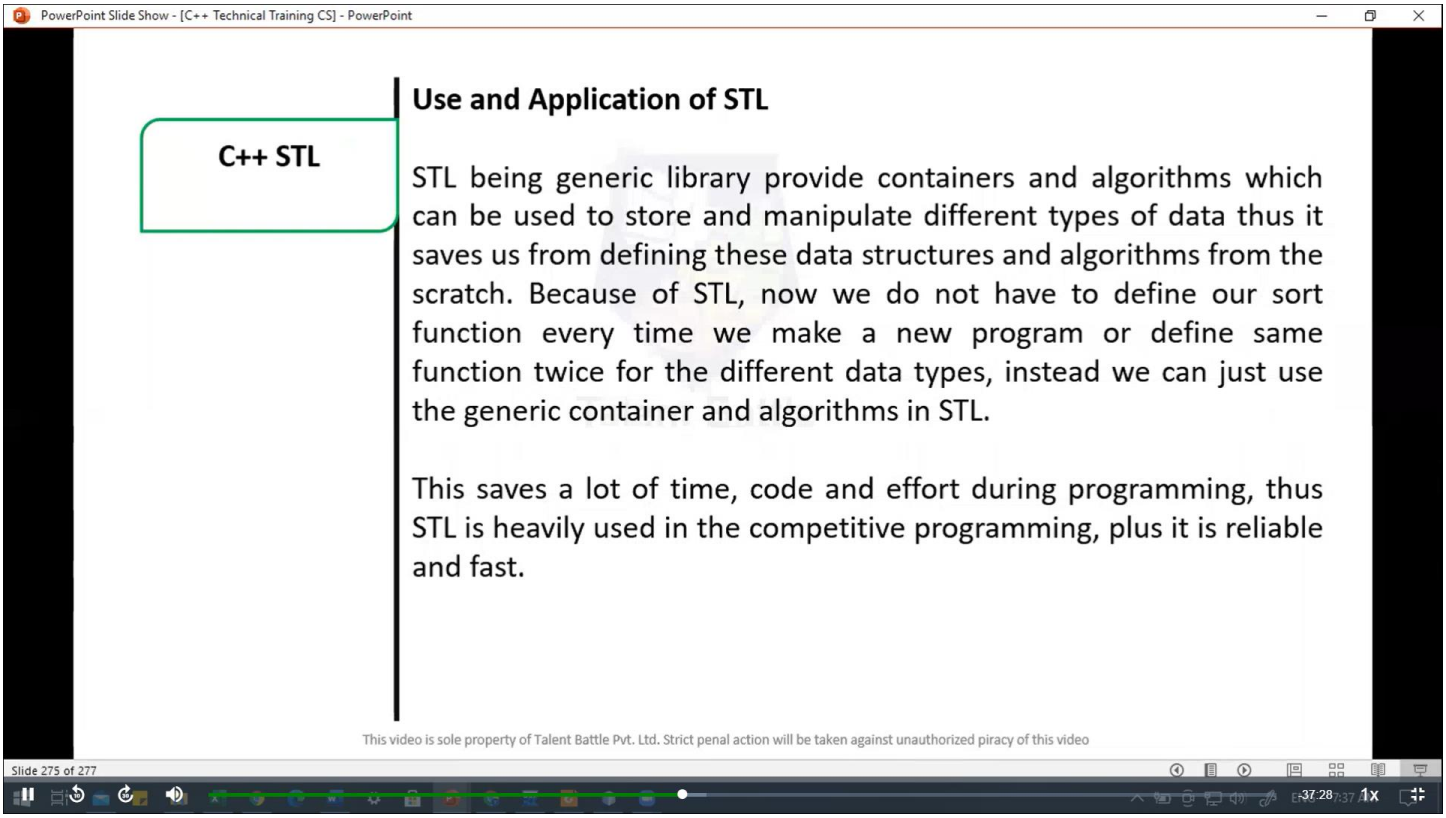
C++ STL

C++: Iterators in STL

Iterators in STL are used to point to the containers. Iterators actually acts as a bridge between containers and algorithms.

For example: sort() algorithm have two parameters, starting iterator and ending iterator, now sort() compare the elements pointed by each of these iterators and arrange them in sorted order, thus it does not matter what is the type of the container and same sort() can be used on different types of containers.

39:13 / 1:35:35 1x




```cpp
//================================================
// C++ STL
// C++ program to implement stack in stl

#include<iostream>
#include<stack>
#include<string>
#include<cstdlib>
using namespace std;

int main(){
  stack<int>st;
  int choice, item;
  while(1){
    cout  << "\n-----------------------------"<<endl;
    cout << "Stack implementation in Stl" << endl;
    cout << "\n-------------------------------"<< endl;
    cout << "1. Insert Element into the Stack"<< endl;
    cout << "2. Delete Element from the Stack" << endl;
    cout << "3. Size of the stack" << endl;
    cout << "4. Top Element of the Stack" << endl;
    cout << "5. Exit" << endl;
    cout << "Enter your Choice: ";
    cin >> choice;

    switch(choice){
```

```
      case 1:
        cout << "Enter value to be inserted: ";
        cin >> item;
        st.push(item);
        break;
      case 2:
        item = st.top();
        st.pop();
        cout << "Element " << item << " Deleted" << endl;
        break;
      case 3:
        cout << "Size of the Queue: ";
        cout << st.size() << endl;
        break;
      case 4:
        cout << "Top Element of the stack: ";
        cout << st.top() << endl;
        break;
      case 5:
        exit(1);
        break;
      default:
        cout << "Wrong choice" << endl;
    }
  }

  return 0;
}


/*

-----------------------------
Stack implementation in Stl

------------------------------
1. Insert Element into the Stack
2. Delete Element from the Stack
3. Size of the stack
4. Top Element of the Stack
5. Exit
Enter your Choice: 1
Enter value to be inserted: 55

-----------------------------
Stack implementation in Stl

------------------------------
1. Insert Element into the Stack
2. Delete Element from the Stack
```

```
3. Size of the stack
4. Top Element of the Stack
5. Exit
Enter your Choice: 1
Enter value to be inserted: 33

------------------------------
Stack implementation in Stl

------------------------------
1. Insert Element into the Stack
2. Delete Element from the Stack
3. Size of the stack
4. Top Element of the Stack
5. Exit
Enter your Choice: 1
Enter value to be inserted: 33

------------------------------
Stack implementation in Stl

------------------------------
1. Insert Element into the Stack
2. Delete Element from the Stack
3. Size of the stack
4. Top Element of the Stack
5. Exit
Enter your Choice: 1
Enter value to be inserted: 22

------------------------------
Stack implementation in Stl

------------------------------
1. Insert Element into the Stack
2. Delete Element from the Stack
3. Size of the stack
4. Top Element of the Stack
5. Exit
Enter your Choice: 3
Size of the Queue: 4

------------------------------
Stack implementation in Stl

------------------------------
1. Insert Element into the Stack
2. Delete Element from the Stack
3. Size of the stack
4. Top Element of the Stack
```

```
5. Exit
Enter your Choice: 4
Top Element of the stack: 22

------------------------------
Stack implementation in Stl

--------------------------------
1. Insert Element into the Stack
2. Delete Element from the Stack
3. Size of the stack
4. Top Element of the Stack
5. Exit
Enter your Choice: 1
Enter value to be inserted: 99

------------------------------
Stack implementation in Stl

--------------------------------
1. Insert Element into the Stack
2. Delete Element from the Stack
3. Size of the stack
4. Top Element of the Stack
5. Exit
Enter your Choice: 4
Top Element of the stack: 99

------------------------------
Stack implementation in Stl

--------------------------------
1. Insert Element into the Stack
2. Delete Element from the Stack
3. Size of the stack
4. Top Element of the Stack
5. Exit
Enter your Choice: 2
Element 99 Deleted

------------------------------
Stack implementation in Stl

--------------------------------
1. Insert Element into the Stack
2. Delete Element from the Stack
3. Size of the stack
4. Top Element of the Stack
5. Exit
Enter your Choice: 5
```

```
--------------------------------
Process exited after 90.62 seconds with return value 1
Press any key to continue . . .
*/
```