**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **26/06/2020** | **Name:** | **Lavanya B** |
| **Course:** | **C++ programming** | **USN:** | **4al17ec043** |
| **Topic:** | **Template**  **Files**  **Challange** | **Semester & Section:** | **6th A** |
| **Github Repository:** | **Lavanya-B** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  **Function Template**  **Functions and classes help to make programs easier to write, safer, and more maintainable.**  **However, while functions and classes do have all of those advantages, in certain cases they can also be somewhat limited by C++'s requirement that you specify types for all of your parameters.**  **With function templates, the basic idea is to avoid the necessity of specifying an exact type for each variable. Instead, C++ provides us with the capability of defining functions using placeholder types, called template type parameters.**  **Eg:**  **#include <iostream>**  **using namespace std;**  **template <class T>**  **T sum(T a, T b) {**  **return a+b;**  **}**  **int main () {**  **double x=7.15, y=15.54;**  **cout << sum(x, y) << endl;**  **}**  **Class template**  **class templates, allowing classes to have members that use template parameters as types.**  **A specific syntax is required in case you define your member functions outside of your class - for example in a separate source file.**  **You need to specify the generic type in angle brackets after the class name.**  **Eg:**  **include <iostream>**  **using namespace std;**  **template <class T>**  **class Pair {**  **private:**  **T first, second;**  **public:**  **Pair (T a, T b):**  **first(a), second(b) { }**  **T bigger();**  **};**  **template <class T>**  **T Pair<T>::bigger() {**  **return (first>second ? first : second);**  **}**  **int main()**  **{**  **Pair <double> obj(23.43, 5.68);**  **cout << obj.bigger();**  **return 0;**  **}**  **Template specialisation**  **In case of regular class templates, the way the class handles different data types is the same; the same code runs for all data types.**  **Template specialization allows for the definition of a different implementation of a template when a specific type is passed as a template argument. Including an empty parameter list. This is because all types are known and no template arguments are required for this specialization, but still, it is the specialization of a class template, and thus it requires to be noted as such.**  **But more important than this prefix, is the <char> specialization parameter after the class template name. This specialization parameter itself identifies the type for which the template class is being specialized (char).**  **Eg:**  **#include <iostream>**  **using namespace std;**  **template <class T>**  **class MyClass {**  **public:**  **MyClass (T x) {**  **cout <<x<<" - not a char"<<endl;**  **}**  **};**  **template < >**  **class MyClass<char> {**  **public:**  **MyClass (char x) {**  **cout <<x<<" is a char!"<<endl;**  **}**  **};**  **int main () {**  **MyClass<int> ob1(42);**  **MyClass<double> ob2(5.47);**  **MyClass<char> ob3('s');**  **}**  **Exceptions**  **Problems that occur during program execution are called exceptions.**  **In C++ exceptions are responses to anomalies that arise while the program is running, such as an attempt to divide by zero.**  **C++ exception handling is built upon three keywords: try, catch, and throw.**  **throw is used to throw an exception when a problem shows up.**  **Eg:**  **#include <iostream>**  **using namespace std;**  **int main()**  **{**  **try {**  **int motherAge = 29;**  **int sonAge = 36;**  **if (sonAge > motherAge) {**  **throw 99;**  **}**  **}**  **catch (int x) {**  **cout<<"Wrong age values - Error "<<x;**  **}**  **return 0;**  **}**  **Files**  **Files requires the standard C++ library called fstream.**  **Three new data types are defined in fstream:**  **ofstream: Output file stream that creates and writes information to files.**  **ifstream: Input file stream that reads information from files.**  **fstream: General file stream, with both ofstream and ifstream capabilities that allow it to create, read, and write information to files.**  **To perform file processing in C++, header files <iostream> and <fstream> must be included in the C++ source file.**  **Eg:**  **#include <iostream>**  **#include <fstream>**  **using namespace std;**  **int main () {**  **ofstream MyFile1("test.txt");**    **MyFile1 << "This is awesome! \n";**  **MyFile1.close();**  **string line;**  **ifstream MyFile("test.txt");**  **while ( getline (MyFile, line) ) {**  **cout << line << '\n';**  **}**  **MyFile.close();**  **}**  **Course completion certificate** |