**National University of Computer & Emerging Sciences, Karachi**

**Computer Science Department**

**Fall 2022, Lab Manual - 12**

|  |  |
| --- | --- |
| **Course Code: CS-1004** | **Course: Object Oriented Programming Lab** |
| **Instructor(s) :** |  |

**Lab # 12**

# Outline:

1. Templates
2. Exception Handling
3. Filling in C++
4. Lab Tasks

## **Templates**

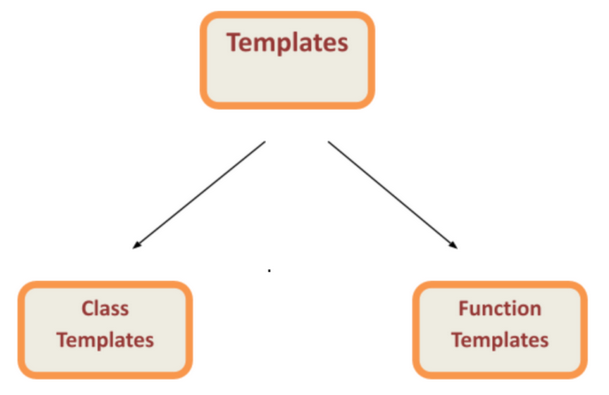
Templates are powerful features of C++, which allows us to write generic programs. Templates are defined as a blueprint or formula for creating a generic class or a function. You can create a single function or single class to work with different data types using templates.

C++ template is also known as generic functions or classes which is a very powerful feature in C++. A keyword “template” in C++ is used for the template’s syntax and angled bracket in a parameter (t), which defines the data type variable.

**Types of Templates in C++:**

There are two types of templates in C++

* Function template
* Class templates



**Function Template:**

Function templates are special functions that can operate with generic types. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type.

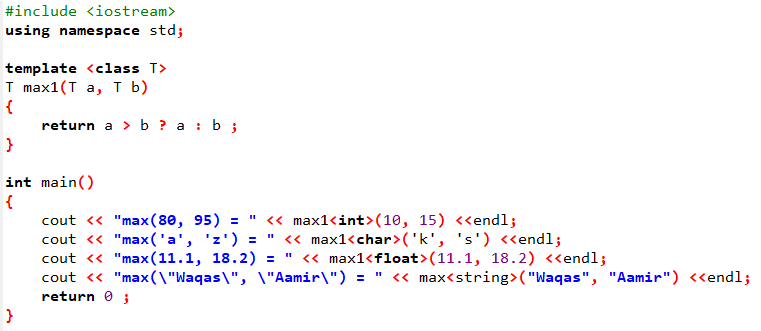
In C++ this can be achieved using template parameters. A template parameter is a special kind of parameter that can be used to pass a type as argument: just like regular function parameters can be used to pass values to a function, template parameters allow to pass also types to a function. These function templates can use these parameters as if they were any other regular type.

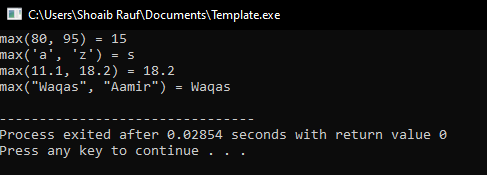
The format for declaring function templates with type parameters is:

* template <class identifier> function\_declaration;
* template<typename identifier> function\_declaration;

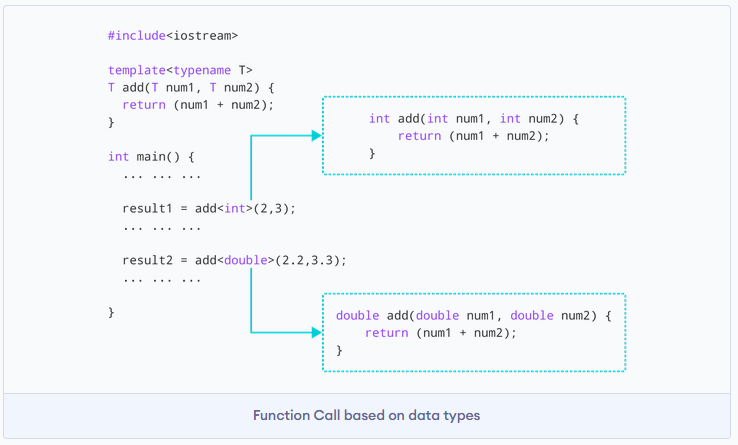
The only difference between both prototypes is the use of either the keyword class or the keyword typename. Its use is indistinct, since both expressions have exactly the same meaning and behave exactly the same way.

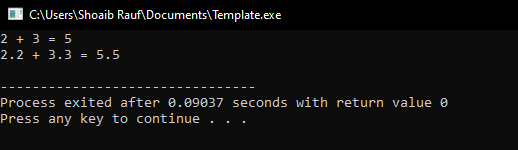
**Sample Code: 01**





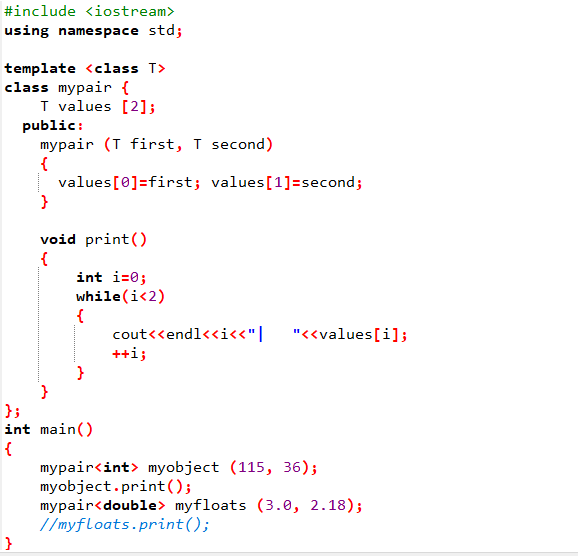
**Sample Code: 02**

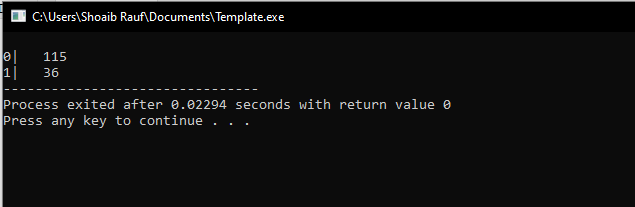


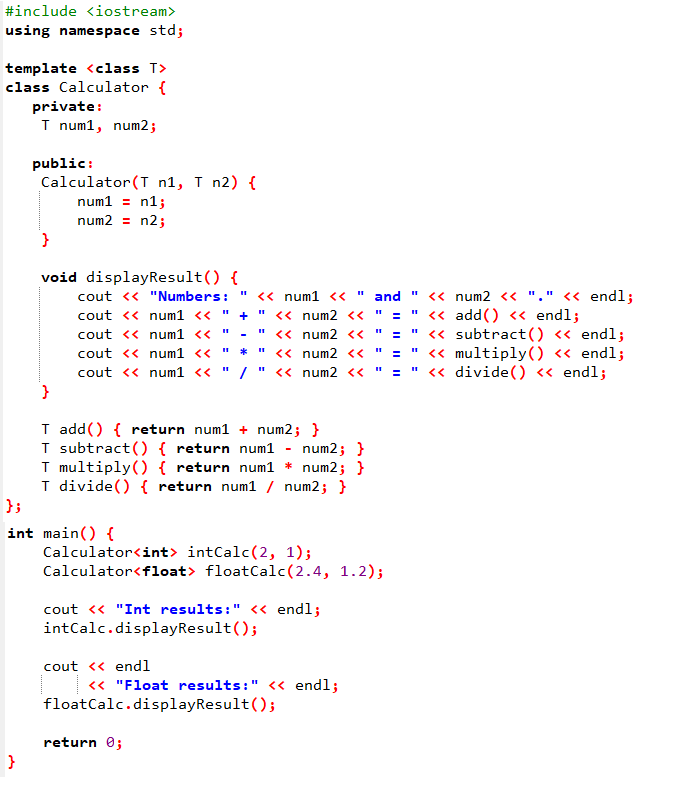


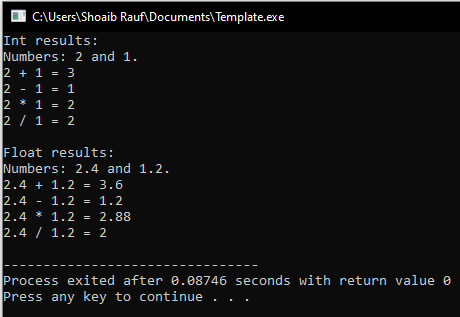
**Class Template:**

If we want to define a different implementation for a template when a specific type is passed as template parameter, we can declare a specialization of that template.









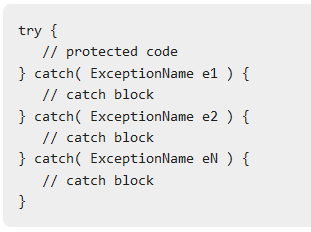
**Exception Handling:**

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw.

* **throw** − A program throws an exception when a problem shows up. This is done using a throw keyword.
* **catch** − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.
* **try** − A try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

Assuming a block will raise an exception, a method catches an exception using a combination of the try and catch keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch as follows

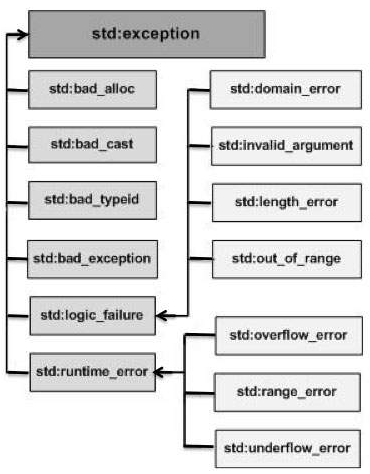


You can list down multiple catch statements to catch different type of exceptions in case your try block raises more than one exception in different situations.

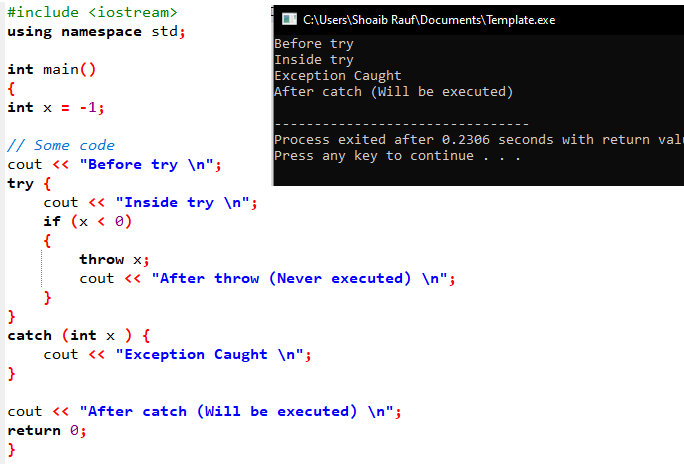
**C++ Standard Exceptions**

C++ provides a list of standard exceptions defined in <exception> which we can use in our programs. These are arranged in a parent-child class hierarchy shown below −

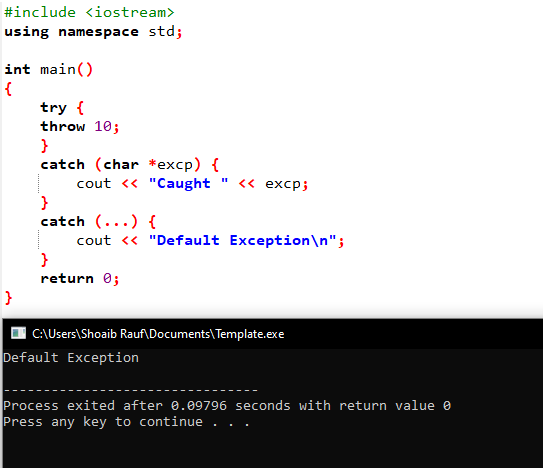
C++ Exceptions Hierarchy



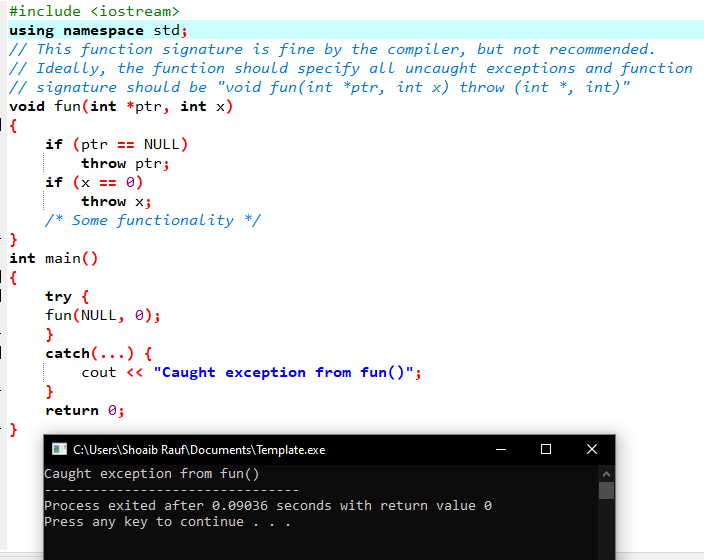
**Sample Code:**



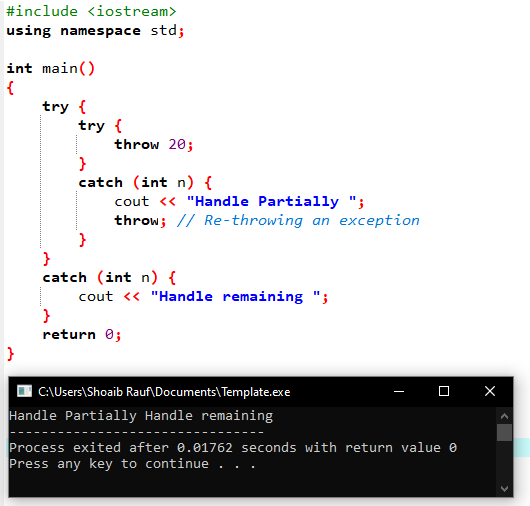
**Sample Code:**



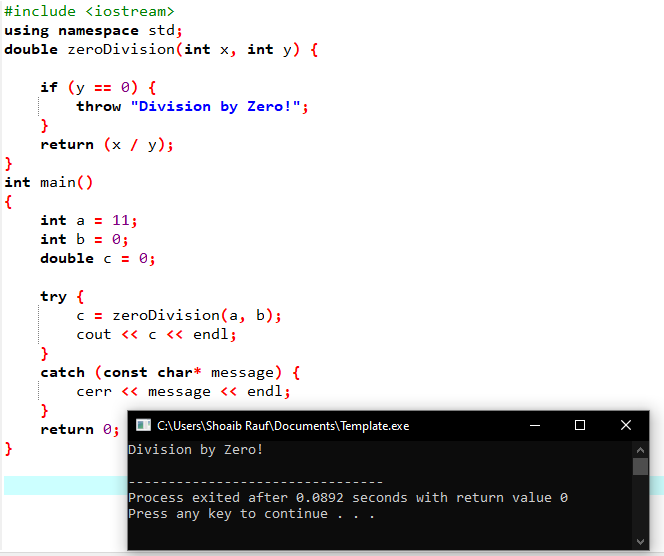
**Sample Code:**



**Sample Code:**



**Sample Code:**

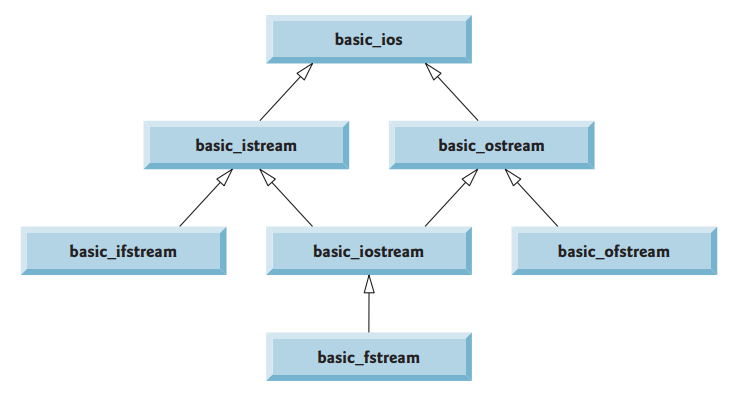


**Filling Handling in C++**

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.

* **ofstream:** Stream class to write on files
* **ifstream:** Stream class to read from files
* **fstream:** Stream class to both read and write from/to files.

**Library Hirerchary:**

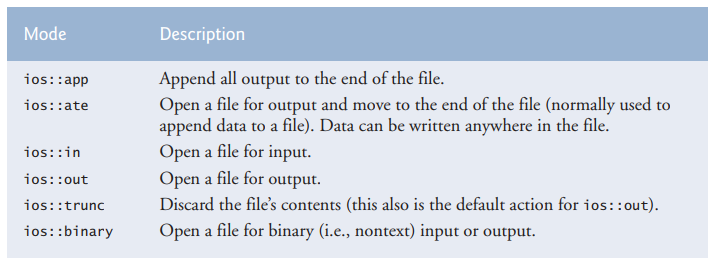


You can read and write data from the files using

1. **Stream Operators** (Reading from the file: >> | Writing to the File: << )
2. **Read(\_)** and **Write(\_)** methods

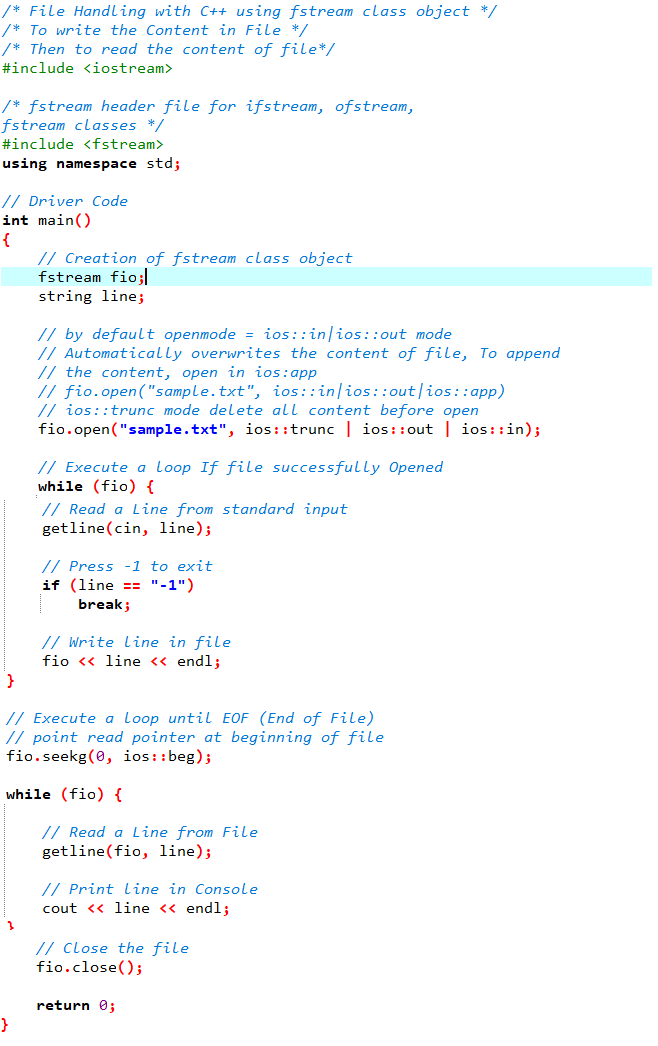
**File Modes**

You can open the file to Read and Write data in the following modes.

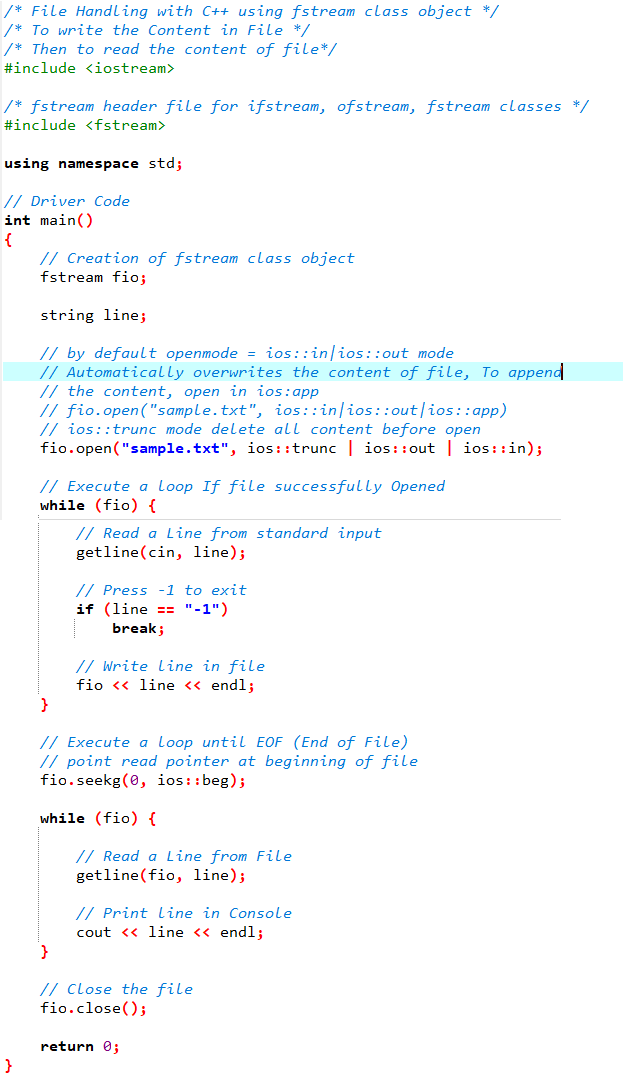


**Stream Operators** (Reading from the file: >> | Writing to the File: << )

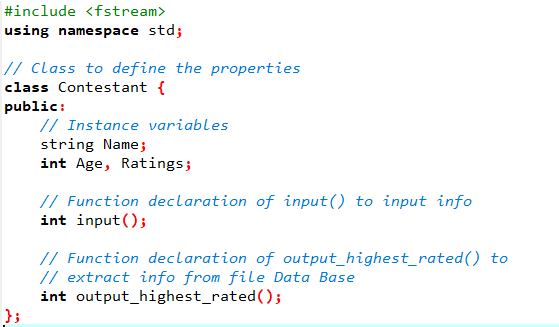
**Writing on the File using Instream operator (<<)**

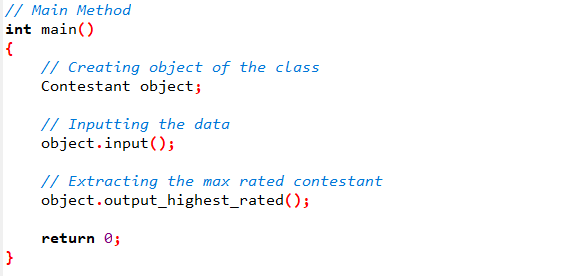


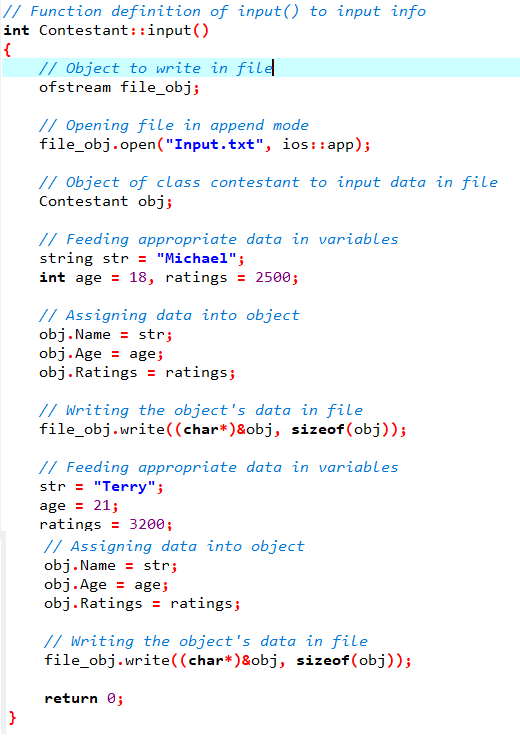
**Reading from File using Outstream operator (>>)**

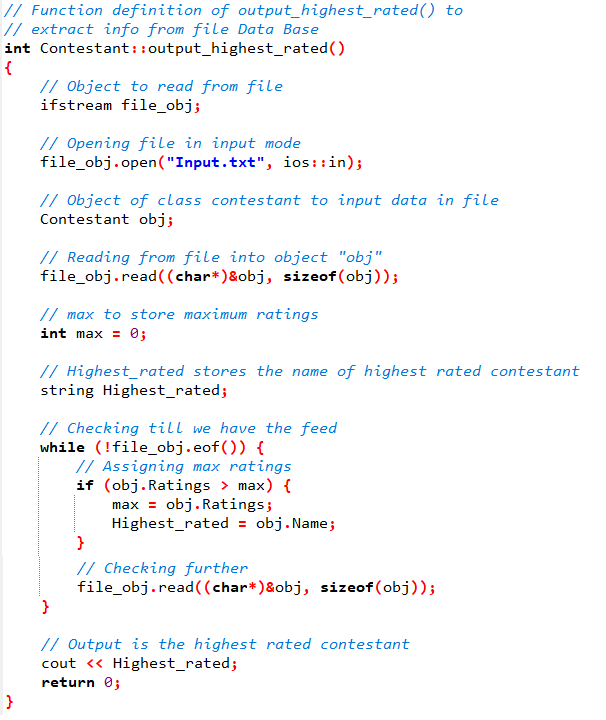


**Using Read(\_) and Write(\_) methods**









**References:**

* https://www.geeksforgeeks.org/templates-cpp/
* https://www.programiz.com/cpp-programming/class-templates
* https://www.tutorialspoint.com/cplusplus/cpp\_templates.htm
* https://www.cplusplus.com/doc/oldtutorial/templates/
* https://www.mygreatlearning.com/blog/templates-in-cpp/
* https://www.tutorialspoint.com/cplusplus/cpp\_exceptions\_handling.htm
* https://www.geeksforgeeks.org/exception-handling-c/?ref=lbp
* https://www.guru99.com/cpp-exceptions-handling.html
* https://www.cplusplus.com/doc/tutorial/exceptions/
* https://www.geeksforgeeks.org/readwrite-class-objects-fromto-file-c/?ref=lbp

**Tasks:**

**Task\_01:**

Create a template class that takes 2 number from the user. These numbers would be integers or float. You need to implement the Add(), Subs() and multiply() for these 2 numbers.

**Task\_02:**

Create a template class that takes N no of elements from the user. These elements of that array would be either integers or characters. You need to implement the sorting algorithm of your own choice to sort the elements in the array.

**Task\_03:**

Write a program to copy the contents of one file to another.

**Task\_04:**

Take a class Person having two attributes name and age. Include a parametrized constructor to give values to all data members.

In main function

i. Create an instance of the person class and name it person1.

ii. Create a binary file person.bin and write person1 object into it.

iii. Read the person1 object from the file.

iv. If file not found then Print “Sorry File not Found”;

**Task\_05:**

Take a class Participant having three attributes (ID, name and score)

* Input () function takes data of the object and stores it in a file name participant.dat
* Output () function takes id from user and show respective data of that id.
* Max () gives the highest score of the Participant in the file