OBJECT ORIENTED PROGRAMMING

Lab 7

OBJECT ORIENTED PROGRAMMING WITH C++

- Chapter Objectives
  - Generic functions
  - Generic classes
  - Exception handling

OBJECT ORIENTED PROGRAMMING WITH C++

- Generic Funtions
  - A generic function defines a general set of operations that will be applied to various types of data.
  - A generic function has the type of data that it will operate upon passed to it as a parameter.
  - In essence, when you create a generic function you are creating a function that can automatically overload itself.
  - A generic function is created using the keyword ***template***.

OBJECT ORIENTED PROGRAMMING WITH C++

- Generic Funtions

```
template <class Ttype >ret_type func_name(parameter list)
{
// body of function
}
```

  - Here *Ttype* is a placeholder name for a data type used by the function.
  - This name can be used within the function definition. However, it is only a placeholder; the compiler will automatically replace this placeholder with an actual data type when it creates a specific version of the function.
  - Although the use of the keyword **class** to specify a generic type in a template declaration is traditional, you can also use the keyword **typename**.

- Generic Classes
- A generic class is a class that defines all algorithms used by that class, but the actual type of the data being manipulated will be specified as a parameter when objects of that class are created.
- Generic classes are useful when a class contains generalizable logic. By using a generic class, you can create a class that will maintain a queue, a linked list, and so on for any type of data.
- The compiler will automatically generate the correct type of object based upon the type you specify when the object is created.

OBJECT ORIENTED PROGRAMMING WITH C++

- Generic Classes
- The general form of a generic class declaration is shown here:

  template <class Ttype > class class_name{ … };

- Here Ttype is the placeholder type name that will be specified when a class is instantiated.
- If necessary, you can define more than one generic data type by using a comma-separated list.
- Once you have created a generic class, you create a specific instance of that class by using the following general form:

  class_name <type> ob;

- Here *type* is the type name of the data that the class will be operating upon.
- Member functions of a generic class are, themselves, automatically generic.
- They need not be explicitly specified as such using **template**.

OBJECT ORIENTED PROGRAMMING WITH C++

- Exception Handling
- C++ provides a built-in error handling mechanism that is called *exception handling*. Using exception handling, you can more easily manage and respond to run-time errors.
- C++ exception handling is built upon three keywords: **try**, **catch**, and **throw**.
- In the most general terms, program statements that you want to monitor for exceptions are contained in a try block.
- If an exception (i.e., an error) occurs within the try block, it is thrown using throw.
- The exception is caught, using catch, and processed.
- As stated, any statement that throws an exception must have been executed from within a try block. A function called from within a try block can also throw an exception.
- Any exception must be caught by a catch statement that immediately follows the try statement that throws the exception.

OBJECT ORIENTED PROGRAMMING WITH C++

- Exception Handling
- The general form of try and catch are shown here:

```
try{// try block}
catch(type1 arg){// catch type1 block}
catch(type2 arg){// catch type2 block}
catch(type3 arg){// catch type3 block}
catch(typeN arg){// catch typeN block}
```

- The try block must contain the portion of your program that you want to monitor for errors.
- When an exception is thrown, it is caught by its corresponding catch statement, which processes the exception.
- There can be more than one catch statement associated with a try.