Empowerment Through Quality Technical Education
AJEENKYA DY PATIL SCHOOL OF ENGINEERING
Dr. D. Y. Patil Knowledge City, Charholi (Bk), Lohegaon, Pune – 412 105
Website: https://dypsoe.in/

LAB MANUAL

OOP and Computer Graphics Laboratory

(217573)

BE (AI&DS) 2019 COURSE

Course Co-ordinator

Prof. Payal R. Deshmukh

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

Department of Artificial Intelligence & Data Science

Vision:

Imparting quality education in the field of Artificial Intelligence and Data Science

Mission:

- To include the culture of R and D to meet the future challenges in AI and DS.
- To develop technical skills among students for building intelligent systems to solve problems.
- To develop entrepreneurship skills in various areas among the students.
- To include moral, social and ethical values to make students best citizens of country.

Program Educational Outcomes:

1. To prepare globally competent graduates having strong fundamentals, domain knowledge, updated with modern technology to provide the effective solutions for engineering problems.

2. To prepare the graduates to work as a committed professional with strong professional ethics and values, sense of responsibilities, understanding of legal, safety, health, societal, cultural and environmental issues.

3. To prepare committed and motivated graduates with research attitude, lifelong learning, investigative approach, and multidisciplinary thinking.

4. To prepare the graduates with strong managerial and communication skills to work effectively as individuals as well as in teams.

Program Specific Outcomes:

1. Professional Skills- The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, networking, artificial intelligence and data science for efficient design of computer-based systems of varying complexities.

2. Problem-Solving Skills- The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.

3. Successful Career and Entrepreneurship- The ability to employ modern computer languages, environments and platforms in creating innovative career paths to be an entrepreneur and to have a zest for higher studies.

Table of Contents

Contents

1. Guidelines to manual usage

This manual assumes that the facilitators are aware of collaborative learning methodologies.

This manual will provide a tool to facilitate the session on Digital Communication modules in collaborative learning environment.

The facilitator is expected to refer this manual before the session.

Icon of Graduate Attributes

K Applying Knowledge	**A** Problem Analysis	**D** Design & Development	**I** Investigation of problems
M Modern Tool Usage	**E** Engineer & Society	**E** Environment Sustainability	**T** Ethics
T Individual & Team work	**O** Communication	**M** Project Management & Finance	**I** Life-Long Learning

Disk Approach- Digital Blooms Taxonomy

1: Remembering / Knowledge
2: Comprehension / Understanding
3: Applying
4: Analyzing
5: Evaluating
6: Creating / Design

Program Outcomes:

1. **Engineering knowledge:** An ability to apply knowledge of mathematics, including discrete mathematics, statistics, science, computer science and engineering fundamentals to model the software application.
2. **Problem analysis:** An ability to design and conduct an experiment as well as interpret data, analyze complex algorithms, to produce meaningful conclusions and recommendations.
3. **Design/development of solutions**: An ability to design and development of software system, component, or process to meet desired needs, within realistic constraints such as economic, environmental, social, political, health & safety, manufacturability, and sustainability.
4. **Conduct investigations of complex problems**:An ability to use research based knowledge including analysis, design and development of algorithms for the solution of complex problems interpretation of data and synthesis of information to provide valid conclusion.
5. **Modern tool usage:** An ability to adapt current technologies and use modern IT tools, to design, formulate, implement and evaluate computer based system, process, by considering the computing needs, limits and constraints.
6. **The engineer and society:** An ability of reasoning about the contextual knowledge of the societal, health, safety, legal and cultural issues, consequent responsibilities relevant to IT practices.
7. **Environment and sustainability:** An ability to understand the impact of engineering solutions in a societal context and demonstrate knowledge of and the need for sustainable development.
8. **Ethics:** An ability to understand and commit to professional ethics and responsibilities and norms of IT practice.
9. **Individual and team work :**An ability to apply managerial skills by working effectively as an individual, as a member of a team, or as a leader of a team in multidisciplinary projects.
10. **Communication:** An ability to communicate effectively technical information in speech, presentation, and in written form
11. **Project management and finance:** An ability to apply the knowledge of Information Technology and management principles and techniques to estimate time and resources needed to complete engineering project.
12. **Life-long learning:** An ability to recognize the need for, and have the ability to engage in independent and life-long learning.

Course Name: Object Oriented Programming

Course Code: (217523)

Course Outcomes

1.CO1:U**nderstand** and **apply** the concepts like inheritance, polymorphism, exceptionhandling and generic structures for implementing reusable programming codes.
2.CO2:Analyze the concept of file and **apply** it while storing and retrieving the data fromsecondary storages.
3.CO3:Analyze and **apply** computer graphics algorithms for line-circle drawing, scanconversion and filling with the help of object oriented programming concepts.
4.CO4:Understand the concept of windowing and clipping and **apply** various algorithms tofill and clip polygons.
5.**CO5:Apply** logic to implement, curves, fractals, animation and gaming programs.

CO to PO Mapping:

PO/CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	1	1	1	-	-	-	-	-	-	-	-
CO2	1	2	1	1	-	-	-	-	-	-	-	1
CO3	2	1	2	2	-	-	-	-	-	-	-	-
CO4	2	1	2	1	-	-	-	-	-	-	-	1
CO5	-	1	-	1	-	-	-	-	-	-	-	-
CO6	-	-	1	-	-	-	-	-	-	-	-	1

CO to PSO Mapping:

	PSO1	PSO2	PSO3
CO1	2	1	-
CO2	2	2	-
CO3	2	1	-
CO4	1	2	-

CO5	2	2	1

2. Laboratory Objective

- Justify the philosophy of object-oriented design and the concepts of encapsulation, abstraction, inheritance, and polymorphism.
- Design, implement, test, and debug simple programs in an object-oriented programming language.
- Describe how the class mechanism supports Inheritance, Polymorphism.

3. Laboratory Equipment/Software

Software Requirements:

For C++ Programming:
Linux Operating System ,
VI Editor or any other Text Editor
GCC Compiler

Hardware Requirements:

Computer nodes with proper configuration.

4. Laboratory Experiment list

Sr. No	Title
	List of Assignments
1	Implement a class Complex which represents the Complex Number data type. Implement thefollowing 1. Constructor (including a default constructor which creates the complex number 0+0i). 2. Overload operator+ to add two complex numbers. 3. Overload operator* to multiply two complex numbers. 4. Overload operators << and >> to print and read Complex Numbers.
2	Develop a program in C++ to create a database of student's information system containing the following information: Name, Roll number, Class, Division, Date of Birth, Blood group, Contact address, Telephone number, Driving license no. and other. Construct the database with suitable member functions. Make use of constructor, default constructor, copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete as well as exception handling.
3	Imagine a publishing company which does marketing for book and audio cassette versions. Create a class publication that stores the title (a string) and price (type float) of publications. From this class derive two classes: book which adds a page count (type int) and tape which adds a playing time in minutes (type float). Write a program that instantiates the book and tape class, allows user to enter data and displays the data members. If an exception is caught, replace all the data member values with zero values.
4	Write a C++ program that creates an output file, writes information to it, closes the file, open itagain as an input file and read the information from the file.
5	Write a function template for selection sort that inputs, sorts and outputs an integer array and afloat array.
6	Write C++ program using STL for sorting and searching user defined records such as personalrecords (Name, DOB, Telephone number etc) using vector container. **OR** Write C++ program using STL for sorting and searching user defined records such as Itemrecords (Item code, name, cost, quantity etc) using vector container.
7	Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index and returns the population of the state.

	Content Beyond Syllabus
1	Explain Storage classes in c++ & Write a program to demonstrate Mutable mutable storage class specifier in C++

4.1. Experiment No. 1

Aim: Implement a class Complex which represents the Complex Number data type. Implement thefollowing

1. Constructor (including a default constructor which creates the complex number 0+0i).
2. Overload operator+ to add two complex numbers.
3. Overload operator* to multiply two complex numbers.

Overload operators << and >> to print and read Complex Numbers.

Objective: To understand the concept of operator overloading.

To understand the concept of constructors, member function, friend function, inline function,dynamic memory allocation and exception handling

Theory:

Operator Overloading : It is a specific case of polymorphism where different operators have different implementations depending on their arguments. In C++ the overloading principle applies not only to functions, but to operators too. That is, of operators can be extended to work not just with built-in types but also classes. A programmer can provide his or her own operator to a class by overloading the built-in operator to perform some specific computation when the operator is used on objects of that class.

An Example of Operator Overloading

Complex a(1.2,1.3); *//this class is used to represent complex numbers*

Complex b(2.1,3); *//notice the construction taking 2 parameters for the real and imaginary part*

Complex c = a+b; *//for this to work the addition operator must be overloaded*

Arithmetic Operators

Arithmetic Operators are used to do basic arithmetic operations like addition, subtraction, multiplication, division, and modulus. The following table list the arithmetic operators used in C++

Operator	Action
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

With C++ feature to overload operators, we can design classes able to perform.

operations using standard operators. Here is a list of all the operators that can be overloaded:

Over loadable operators
+ - * / = <> += -= *= /= <<>>
<<= >>= == != <= >= ++ -- % & ^ ! |
~ &= ^= |= && || %= []

- To overload an operator in order to use it with classes we declare *operator functions*, which are regular functions whose names are the operator keyword followed by the operator signthat we want to overload. The format is:
- type operator operator-symbol (parameters) {/*...*/ }
- The **operator** keyword declares a function specifying what *operator-symbol* means when applied to instances of a class. This gives the operator more than one meaning, or "overloads" it. The compiler distinguishes between the different meanings of an operator by examiningthe types of its operands.

Syntax:

return_typeclass_name :: operator op(arg_list)

{

//function body

}

where,

- Return type is the value returned by the specified operation
- op is the operator to be overload.
- op is proceeding by the keyword operator.
- operator op is

the function name

The Process of Overloading has 3 Steps

1. Create a class that define a data types that is used in the overloading operation

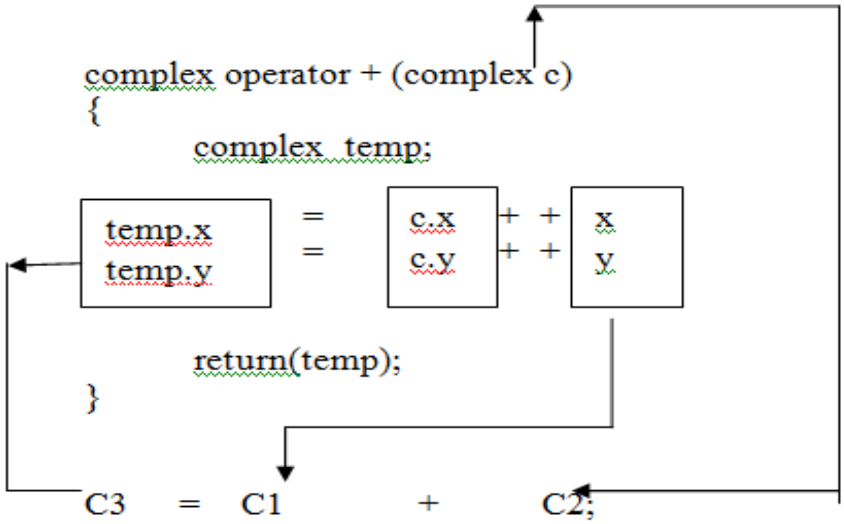
2. Declare the operator function operator op() in the public part of the class.It may be either a member function or a friend function.
3. Define the operator function to implement the required operation

e.g. Overloading Binary operators: A

statement like C = sum (A, B); //

functional notation

This functional notation can be replaced by a natural looking expression



Algorithm:

Step 1: Start the program

Step 2: Create a class complex

Step 3: Define the default constructor.

Step 4: Declare the operator function which are going to be overloaded and display function Step 5: Define the overloaded functions such as +, -,/,* and the display function

For Addition:

(a+bi) + (x + yi) = ((a+x)+(b+y)i)

For Multiplication:

(a+bi) * (x + yi) = (((a*x)-(b*y)) + ((a*y) + (x*b))i)

Step 6: Create objects for complex class in main() function

Step 7:Create a menu for addition, multiplication of complex numbers and display the result Step 8: Depending upon the choice from the user the arithmetic operators will invoke the overloaded operator automatically and returns the result

Step 9: Display the result using display function

Applications:

1. Constructor is used to initialize an object of the class
2. Operator overloading allows C++ operators to have user-defined meanings on user-defined types or classes.

Input:

Default constructor value=

0+0i

Enter the 1st number

Enter the real part7

Enter the imaginary part1

Enter the 2nd number

Enter the real part7

Enter the imaginary part1

Output:

The first number is 7+1i

The second number is 7+1i

The addition is 14+2i

The multiplication is 48+14i

Conclusion:

Hence, we have studied concept of operator overloading.

Outcome:

Upon completion of this experiment, students will be able to:

Experiment level outcome (ELO1): Perform operator overloading and create constructor to create complex number

Questions:

1. What is operator overloading?
2. What are the rules for overloading the operators?
3. State clearly which operators are overloaded and which operator are not overloaded?
4. State the need for overloading the operators.
5. Explain how the operators are overloaded using the friend function.
6. What is the difference between “overloading” and “overriding”?
7. What is operator function? Describe the syntax?
8. When is Friend function compulsory? Give an example?

4.2. Experiment No. 2

Aim:

Develop a program in C++ to create a database of student's information system containing the following information: Name, Roll number, Class, Division, Date of Birth, Blood group, Contact address, Telephone number, Driving license no. and other. Construct the database with suitable member functions. Make use of constructor, default constructor, copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete as well as exception handling.

Objective:

To understand the concept of constructors, member function, friend function, inline function,dynamic memory allocation and exception handling

Theory:

Constructor:

A special method of the class that will be automatically invoked when an instance of the class is created is called as constructor. Following are the most useful features of constructor.

1) Constructor is used for Initializing the values to the data members of the Class.
2) Constructor is that whose name is same as name of class.
3) Constructor gets Automatically called when an object of class is created.
4) Constructors never have a Return Type even void.
5) Constructor is of Default, Parameterized and Copy Constructors. The various types of Constructor are as follows: -

Constructors can be classified into 3 types

1. Default Constructor
2. Parameterized Constructor
3. Copy Constructor

1. **Default Constructor: -** Default Constructor is also called as Empty Constructor which has no arguments and It is Automatically called when we create the object of class but Remember name of Constructor is same as name of class and Constructor never declared with the help of Return Type. Means we can't declare a Constructor with the help of void Return Type., if we never Pass or declare any Arguments then this called as the Copy Constructors.

2. Pa**rameterized Constructor: -** This is another type constructor which has some Arguments and same name as class name but it uses some Arguments So For this, we have to create object of Class by passing some Arguments at the time of creating object with the name of class. When we pass some Arguments to the Constructor then this will automatically pass the Arguments to the Constructor and the values will retrieve by the Respective Data Members of the Class.

3. **Copy Constructor: -** This is also another type of Constructor. In this Constructor we pass the object of class into the Another Object of Same Class. As name Suggests you Copy, means Copy the values of one Object into the another Object of Class .This is used for Copying the values of class object into an another object of class So we call them as Copy Constructor and For Copying the values We have to pass the name of object whose values we wants to Copying and When we are using or passing an Object to a Constructor then we must have to use the & Ampersand or Address Operator.

Destructor: As we know that Constructor is that which is used for Assigning Some Values to data Members and For Assigning Some Values this May also used Some Memory so that to free up the Memory which is Allocated by Constructor, destructor is used which gets Automatically Called at the End of Program and we doesn't have to Explicitly Call a Destructor and Destructor Can't be Parameterized or a Copy This can be only one Means Default Destructor which Have no Arguments. For Declaring a Destructor, we have to use ~tiled Symbol in front of Destructor.

Static members

A class can contain static members, either data or functions. A static member variable has following properties:

- It is initialized to zero when the first object of its class is created. No other initialization is permitted.
- Only one copy of that member is created for the entire class and is shared by all the objects of that class.
- It is the visible only within the class but its lifetime is the entire program

A static member function has following properties

- A static function can have access to only other static members (fun or var) declared in the same class
- A static function can be called using the class name instead of its object name Class_name :: fun_name;

Static member functions are considered to have class scope. In contrast to non static member functions, these functions have no implicit this argument; therefore, they can use only static data members, enumerators, or nested types directly. Static member functions can be accessed without using an object of the corresponding class type.

The following restrictions apply to such static functions:

1. They cannot access non static class member data using the member-selection operators (. or –>).
2. They cannot be declared as virtual.
3. They cannot have the same name as a non-static function that has the same argument types.

E.g. // static members in classes

```
class StaticTest {
private: static int x;
public: static int
count()

{
  return x;
    }
};
```

Friend functions:

In principle, private and protected members of a class cannot be accessed from outside the same class in which they are declared. However, this rule does not affect friends. Friends are functions or classes declared as such. If we want to declare an external function as friend of a class, thus allowing this function to have access to the private and protected members of this class, we do it by declaring a prototype of this external function within the class, and preceding it with the keyword friend.

Properties of friend function:

- It is not in the scope of the class to which it has been declared as friend.
- Since it is not in the scope of the class , it cannot be called using the object of that class
- It can be invoked like a normal function w/o the help of any object.
- It can be declared in private or in the public part of the class.
- Unlike member functions, it cannot access the member names directly and has to use an object name and dot operator with each member name.

Pointers:

A pointer is a derived data type that refers to another data variable by storing the variables memory address rather than data.

Declaration of pointer variable is in the following form :

```
Data_type * ptr_var;
```

Eg int *ptr;

Here ptr is a pointer variable and points to an integer data type.

We can initialize pointer

variable as follows int a,

*ptr; // declaration

ptr = &a //initialization

Pointers to objects:

Consider the following eg item X;// where item is class and X is object Similarly we can define a pointer it_ptr of type item as followsItem *it_ptr ;
Object pointers are useful in creating objects at runtime. We can also access

public members of theclass using pointers.

Eg item X;

item *ptr = &X;

the pointer 'ptr 'is initialized with address of X.

we can access the member functions and data using

pointers as follows

ptr->getdata();

ptr->show();

this pointer:

C++ uses a unique keyword called this to represent an object that invokes a member function. this is a pointer that points to the object for which this function was called. This unique pointer

is automatically passed to a member function when it is

called. Important notes on this pointer.

Applications:

1. To create own student student database
2. To create different function like insert,update & delete the student information
3. to create a more comprehensive student information database system,

Input:

Personnel information such as Name, Date of Birth, Blood group, Height, Weight, Insurance Policy, number, contact address, telephone number, driving license no.

Output:

Display personnel information from database. The result in following format:

Name DOB ……. Driving License No

1

2

.

.

N

Conclusion:

Hence, we have successfully studied concept of constructor, default constructor, copy constructor,destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

Outcome:

Upon completion of this experiment, students will be able to:

Experiment level outcome (ELO1): we have created a student class with constructors, a destructor, and exception handling. It demonstrates the use of constructors for creating student objects, a copy constructor for copying data, and the destructor for cleaning up resources. Additionally, exception handling is used to validate student data.

Questions:

1.What is static Function?

2.What is friend function? State the advantage of using the friend function.

3.What is friend class? Explain with examples.

4.What is this pointer? Explain with examples.
5.State the advantages of this pointer.
6.What are inline functions?
7.How do we declare member of a class static?
8.What are demerits of friend function?
9.What is concept of constructor, destructor? What are types of constructors?

4.3.Experiment No. 3

Aim:

Imagine a publishing company which does marketing for book and audio cassette versions. Create a class publication that stores the title (a string) and price (type float) of publications. From this class derive two classes: book which adds a page count (type int) and tape which adds a playing time in minutes (type float).Write a program that instantiates the book and tape class, allows user to enter data and displays the data members. If an exception is caught, replace all the data member values with zero values.

Objective:

To learn the concept of inheritance

Theory:

Inheritance:

Inheritance in Object Oriented Programming can be described as a process of creating new classes from existing classes. New classes inherit some of the properties and behavior of the existing classes. An existing class that is "parent" of a new class is called a base class. New class that inherits properties of the base class is called a derived class. Inheritance is a technique of code reuse. It also provides possibility to extend existing classes by creating derived classes.

The basic syntax of inheritance is:

Class DerivedClass : accessSpecifier BaseClass

There are 3 access specifiers: Namely **public, private and protected**.

public: This inheritance mode is used mostly. In this the protected member of Base class becomes protected members of Derived class and public becomes public.
protected: In protected mode, the public and protected members of Base class becomes protected members of Derived class.

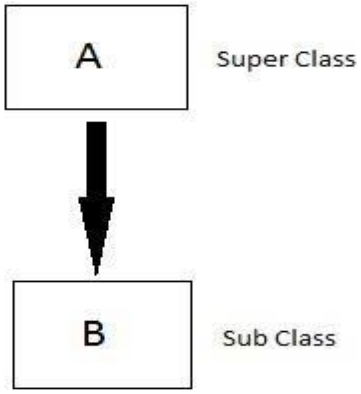
Types of Inheritance

In C++, we have 5 different types of Inheritance. Namely,

1. Single Inheritance
2. Multiple Inheritance
3. Hierarchical Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance

Single Inheritance:

In this type of inheritance one derived class inherits from only one base class. It is the most simplest form of Inheritance.



Syntax:

class subclass_name : access_modebase_class

{

//body of subclass

};

Multiple Inheritance:

In this type of inheritance a single derived class may inherit from two or more than two base classes.

Syntax:

classsubclass_name : access_mode base_class1, ccess_mode base_class2, ...

{

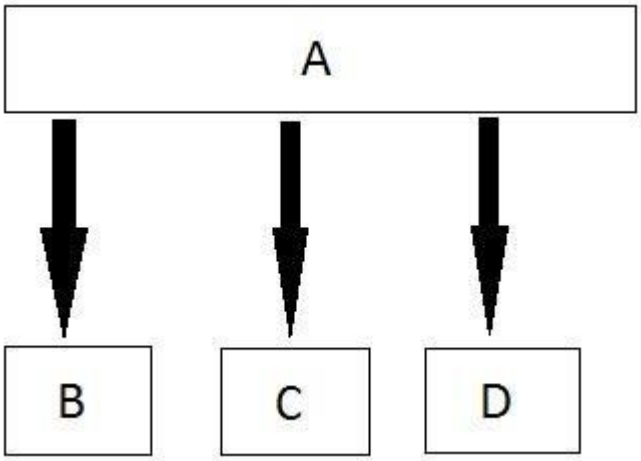
//body of subclass

};

Multilevel Inheritance: In this type of inheritance the derived class inherits from a class, which in turn inherits from some other class. The Super class for one, is sub class for the other.

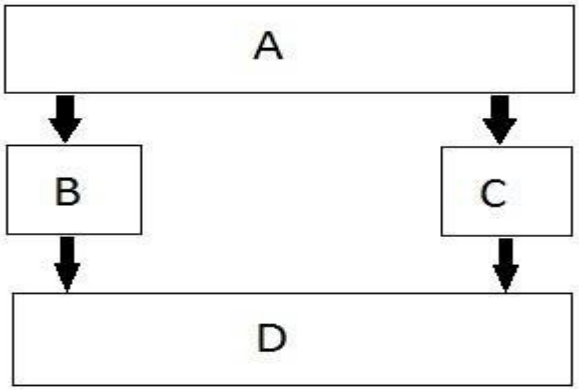
Hierarchical Inheritance:

In this type of inheritance, multiple derived classes inherits from a single base class.



Hybrid Inheritance:

Hybrid Inheritance is combination of any 2 or more types of inheritances



Applications:

1. To create a hierarchy of classes for handling different types of publications (books and audio cassettes) and includes error handling using exceptions.
2. To allow applications to focus on the important words
3. To drop common words

Input:

A class publication that stores the title (a string) and price (type float) of publications. Derives two classes Book and Tape.

Output:

Display title and price from publication class. The result in following format:
Enter Title: OOP
Enter Price: 300
Enter the Pages:1000
Enter the Minutes:60
Title OOP
Price 300
Pages 1000
Minures 60

Conclusion:

Hence, we have successfully studied concept of inheritance and exception handling.

Outcome:

Upon completion of this experiment, students will be able to:

Experiment level outcome (ELO1): Perform how to use inheritance to create a class hierarchy, how to handle user input with exception handling, and how to display information about different types of publications

Questions:

1. What is Inheritance?
2. What are types of Inheritance?
3. What is Single Inheritance?
4. What is Multiple Inheritance?
5. What is Hierarchical Inheritance?
6. What is Multilevel Inheritance?
7. What is Hybrid Inheritance?
8. What is Exception handling?
9. What are try catch block of exception handling?

4.4. Experiment No. 4

Aim:

Write a C++ program that creates an output file, writes information to it, closes the file, open it again as an input file and read the information from the file.

Objective:

To learn the concept of file handling.

Theory:

Stream:

A stream is a sequence of bytes. It acts as source from which the input data can be obtained or as a destination to which the output data can be sent.

1. InputStream

Input Streams are used to hold input from a data producer, such as a keyboard, a file, or a network. The source stream that provides data to the program is called the input stream. A program extracts the bytes from the input stream. In most cases the standard input device is the keyboard. With the cin and "extraction" operator (>>) it is possible to read input from the keyboard.

2. OutputStream

Output Streams are used to hold output for a particular data consumer, such as a monitor, a file, or a printer. The destination stream that receives data from the program is called the output stream. A program inserts the bytes into an output stream. By default, the standard output of a program points at the screen. So with the cout operator and the "insertion" operator (<<) you can print a message onto the screen.iostream standard library provides cin and cout methods for reading from standard input and writing to standard output respectively.file handling provides three new datatypes:

Data Type	Description
ofstream	This data type represents the output file stream and is used to create files and to write information to files.
ifstream	This data type represents the input file stream and is used to read information from files.

Closing a File:

When a C++ program terminates it automatically closes flushes all the streams,release all the allocated memory and close all the opened files. It is always a good practice that a programmer should close all the opened files before program termination. following is the standard syntax for close( ) function, which is a member of fstream ,ifstream, and ofstream objects.

void close( );

Writing to a File

- While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen.
- The only difference is that you use an ofstream or fstream object instead of the cout object.

Reading from a File

- You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard.
- The only difference is that you use an ifstream or fstream object instead of the cin object.

file .read ((char *)&V , sizeof (V)); file . Write ((char *)&V , sizeof (V));

Applications:

1. Data Storage and Retrieval: Storing structured data in a file for later retrieval
2. Data Transformation: Reading data from one file, processing it, and writing the results to another file.

Input:

how many record you want 3
1 abc
2 pqr
3 xyz

Output:

Name=abc

Roll no=12

Conclusion:

Hence, we have successfully studied how to create the output file, writing the data and closing the file successfully.

Outcome:

Upon completion of this experiment, students will be able to:

Experiment level outcome (ELO1):To creates an output file named "output.txt" and writes two lines of text into it.It then closes the output file.

Questions:

1. What is file handling?
2. What are the different benefits of file handling?
3. What is fstream class?
4. How to create object of fsream class?
5. Explain the syntax of read() ?
6. Explain the syntax of write()?

4.5. Experiment No. 5

Aim:

Write a function template for selection sort that inputs, sorts and outputs an integer array and a float array.

Objective:

To learn the concept of file Template.

Theory:

Templates

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.
A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept. There is a single definition of each container, such as vector, but we can define many different kinds of vectors for example, vector <int> or vector <string>.
You can use templates to define functions as well as classes, let us see how do they work: Function Template:
The general form of a template function definition is shown here:

```
template <class type> ret-type func-name(parameter list)
{

// body of function
}
```

Class Template:

Just as we can define function b templates, we can also define class templates. The general form of a generic class declaration is shown here:

```
template <class type> class class-name
{

.

.

.

}
```

Selection Sort:

Selection sort is a sorting algorithm, specifically an in-place comparison sort. It has O(n2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited **How selection sort works?**

Example

For the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value.

So we replace 14 with 10. After one iteration 10, which happens to be the minimum value in the list, appears in the first position of sorted list.

For the second position, where 33 is residing, we start scanning the rest of the list in linear manner.

We find that 14 is the second lowest value in the list and it should appear at the second place. We swap these values

After two iterations, two least values are positioned at the beginning in the sorted manner. The same process is applied on the rest of the items in the array. Pictorial depiction of entire sorting process is as follows –

Input:

Selection sort Integer Element
Enter how many elements you want 5

Enter the Integer element 7

5

8

9

3

Float Element
Enter how many elements
you want 5 Enter the float
element 3.8

9.4
5.5
2.2
6.7

Output:

Sorted list=

3 5 7 8 9

Sorted list=
2.2 3.8 5.6 6.7 9.4

Conclusion:

Hence, we have studied concept of Function Template.

Outcome:

Upon completion of this experiment, students will be able to:

Experiment level outcome (ELO1): Sorts and displays the original and sorted arrays for both integer and float data types using the same selection sort algorithm.

Questions:

1. What is template?
2. What is Function template?
3. What is Class template?
4. Explain selection sort algorithm.
5. Explain template with non-type argument.

4.6. Experiment No. 6

Aim:

Write C++ program using STL for sorting and searching user defined records such as personalrecords (Name, DOB, Telephone number etc) using vector container.

OR

Write C++ program using STL for sorting and searching user defined records such as Item records (Item code, name, cost, quantity etc) using vector container.

Objective:

To learn the concept STL, searching, sorting and vector container.

Theory:

STL:

The Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators. It is a generalized library and so, its components are parameterized. A working knowledge of template classes is a prerequisite for working with STL.
STL has four components

- Algorithms
- Containers
- Functions
- Iterators

Algorithms

- The algorithm defines a collection of functions especially designed to be used on ranges of elements. They act on containers and provide means for various operations for the contents of the containers.

  - Algorithm
  - Sorting
  - Searching
  - Useful Array algorithms
  - Partition Operations

- Numeric

Containers

•Containers or container classes store objects and data. There are in total seven standard "first- class" container classes and three container adaptor classes and only seven header files that provide access to these containers or container adaptors.

- Sequence Containers: implement data structures which can be accessed in a sequential manner.
- vector
- list
- Deque arrays

Functions

• The STL includes classes that overload the function call operator. Instances of such classes are called function objects . Function allow the working of the associated function to be customized with the help of parameters to be passed.

Iterators

•As the name suggests, iterators are used for working upon a sequence of values. They are the major feature that allow generality in STL.

Sorting:

It is one of the most basic functions applied to data. It means arranging the data in a particular fashion, which can be increasing or decreasing. There is a built in function in C++ STL by the name of sort( ). This function internally uses Intro Sort. In more details it is implemented using hybrid of Quick Sort,
Heap Sort and Insertion Sort. By default, it uses Quick Sort but if Quick Sort is doing unfair partitioning and taking more than N*log N time, it switches to Heap Sort and when the array size becomes really small, it switches to Insertion Sort. The prototype for sort is :

sort(startaddress, endaddress)

startaddress: the address of the first element of the array endaddress: the address of the next contiguous location of the last element of the array**.**

Applications:

1. To sort and search user-defined records.
2. To store and manage a collection of records and perform operations like sorting and searching efficiently
3. Manage and manipulate personal records efficiently.

Output:

The array is : 1 5 8 9 0 6 7 3 4 2 0

Let's say we want to search for 2
in the array So, we first sort the
array
The array after sorting is : 0 1 2 3 4
5 6 7 8 9 Now, we do the binary
search
Element found in the array
Now, say we want to search for 10Element not found in the array

Algorithm:

1. Start.
2. Give a header file to use 'vector'.
3. Create a vector naming 'personal_records'.
4. Initialize variables to store name, birth date and telephone number
5. Using iterator store as many records you want to store using predefined functions as push_back().
6. Create another vector 'item_record'
7. Initialize variables to store item code, item name, quantity and cost.
8. Using iterator and predefined functions store the data.
9. Using predefined function sort(), sort the data stored according to user requirements.
10. Using predefined function search, search the element from the vector the user wants to check.
11. Display and call the functions using a menu**.**
12. End.

Input:

Personnel information such as name, DOB, telephone number.

Output:

Display personnel information from database. The result in following format:
***** Menu *****

1. Insert
2. Display
3. Search
4. Delete

5. Sort

6. Exit

Enter your choice:1

Enter Item Name: bat

Enter Item Quantity:2

Enter Item Cost:50

Enter Item Code:1

Conclusion:

Hence, we have successfully studied the concept of STL(Standard Template Library) and how it makes many data structures easy. It briefs about the predefined functions of STL and their uses such a search() and sort()

Outcome:

Upon completion of this experiment, students will be able to:

Experiment level outcome (ELO1): Perform sorting of the records based on a specific field (name in this case) using sorting algorithm.

Questions:

1. What is STL?
2. What are four components of STL?
3. What is Sorting?
4. What is Searching?
5. What vector container?

3.7. Experiment No. 7

Aim:

Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index and returns the population of the state.

Objective:

To learn the concept of map associative container.

Theory:

Map associative container:

Map associative container are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values. **map::operator[]**
This operator is used to reference the element present at position given inside the operator. It is similar to the at() function, the only difference is that the at() function throws an out-of-range exception when the position is not in the bounds of the size of map, while this operator causes undefined behaviour.

***mapname*[*key*]Parameters :**

Key value mapped to the element to be fetched.

Returns :

Direct reference to the element at the given key value.

Examples

```
Input : map mymap;
   mymap['a'] = 1;

mymap['a'];
 Output : 1

  Input : map mymap;
  mymap["abcd"] = 7;

                                  mymap["abcd"];
```

Algorithm:

1. Start.
2. Give a header file to map associative container.
3. Insert states name so that we get values as population of that state.
4. Use population Map. insert().
5. Display the population of states**.**
6. End.

Applications:

1. Data Lookup:This program efficiently looks up state populations using state names as keys
2. Data Retrieval:This concept is applicable in various real-world scenarios such as Database system,Information retrieval system

Input:

Information such as state name to map associative container.

Output:

Size of population Map:5

Brasil: 193 illion

China: 1339 million

India: 1187 million

Indonesia: 234 million

Pakistan: 170 million

Indonesia's populations is 234 million

Conclusion:

Hence, we have successfully studied the concept of map associative container.

Outcome:

Upon completion of this experiment, students will be able to:

Experiment level outcome (ELO1):To creates a std:map to store state populations, with state names as keys and their corresponding populations as values.Easily searches for the entered state name in the map and returns the population of the specified state if found.

Questions:

1. What is an associative container in C++?
2. What is map in C++?
3. How to do declare a map?
4. Explain Associative mapping with example?

5. Appendix

GNU Compiler Collection (GCC) GCC stands for ―GNU Compiler Collection‖. GCC is an integrated distribution of compilers for several major programming languages. These languages currently include C, C++, Objective-C, Objective-C++, Java, Fortran , and Ada.

The abbreviation GCC has multiple meanings in common use. The current official meaning is ―GNU Compiler Collection‖, which refers generically to the complete suite of tools. The name historically stood for ―GNU C Compiler‖, and this usage is still common when the emphasis is on compiling C programs. Finally, the name is also used when speaking of the language independent component of GCC: code shared among the compilers for all supported languages.

The language-independent component of GCC includes the majority of the optimizers, as well as the ―back ends‖ that generate machine code for various processors.

The part of a compiler that is specific to a particular language is called the ―front end‖. In addition to the front ends that are integrated components of GCC, there are several other front ends that are maintained separately. These support languages such as Pascal, Mercury, and COBOL. To use these, they must be built together with GCC proper.

Most of the compilers for languages other than C have their own names. The C++ compiler is G++, the Ada compiler is GNAT, and so on. When we talk about compiling one of those languages, we might refer to that compiler by its own name, or as GCC. Either is correct.

Historically, compilers for many languages, including C++ and Fortran, have been implemented as ―preprocessors which emit another high level language such as C. None of the compilers included in GCC are implemented this way; they all generate machine code directly. This sort of preprocessor should not be confused with the C preprocessor, which is an integral feature of the C, C++, Objective-C and Objective-C++ languages

GCC Command Options

When you invoke GCC, it normally does preprocessing, compilation, assembly and linking. The ―overall options‖ allow you to stop this process at an intermediate stage. For example, the –c option says not to run the linker. Then the output consists of object files output by the assembler. Other options are passed on to one stage of processing. Some options control the preprocessor and others the compiler itself. Yet other options control the assembler and linker; most of these are not documented here, since you rarely need to use any of them. The gcc program accepts options and file names as operands. Many options have multi-letter names; therefore multiple single-letter options may not be grouped: -dr is very different from `-d -r'. You can mix options and other arguments. For the most part, the order you use doesn't matter. Order does matter when you use several options of the same kind; for example, if you specify –L more than once, the directories are searched in the order specified. Many options have long names starting with `-f' or with `-W'—for example, -fmove-loopinvariants, -Wformat and so on. Most of these have both positive and negative forms; the negative form of –ffoo would be -fno-foo. This manual documents only one of these two forms, whichever one is not the default. Compiling C++ Programs C++ source files conventionally use one of the suffixes `.C', `.cc', `.cpp', `.CPP', `.c++', `.cp', or `.cxx'; C++ header files often use `.hh' or `.H'; and pre-processed C++ files use the suffix `.ii'. GCC recognizes files with these names and compiles them as C++ programs even if you call the compiler the same way as for compiling C programs (usually with the name gcc). When you compile C++ programs, you may specify many of the same command-line options that you use for compiling programs in any language; or command-line options meaningful for C and related languages; or options that are meaningful only for C++ programs.

The Vi Editor

All Linux configuration files are written in plain English, easy to read and to adapt. You use a texteditor to write or make changes to such files. The two most popular, powerful, and unfortunately "difficult" text editors, both of which are found in every Linux are Vi and Emacs.

Most GUI-based editors, such as Kedit, are easier to manage. But don't make the mistake of thinking that a GUI-based editor is all you need. There are situations that crop up with Linux that require a text-mode editor -- in other words, when you don't have the luxury of accessing a GUI desktop at all. Vi and Emacs are the only tools that come with every Linux that work in text mode, so learning one or the other is mandatory.

Getting Started

To start Vi, open a terminal or console and simply type "vi" (without the quotation marks) followed by the name of any existing file or a new file you want to create.

Vi works in two main modes, one for editing text and the other for giving commands. To switch between the two modes you use the I (Insert) and Esc keys. The program opens in the Command mode, which is used for cursor movements, delete, cut, copy, paste, and saving changes.

The Insert mode is what you'll work in most of the time. You use it to make changes in an open file. Enter the Insert mode by pressing the I key. Newer Vi versions will display the word "INSERT' on the bottom line while you're in Insert mode.

Press the Esc key to switch Vi back to Command mode. As soon as you hit the Esc key the text "INSERT" on the bottom line disappears. You save your changes to the open file from the Command mode. Press Shift-ZZ to save.

If you make a mistake when saving a file, such as pressing Ctrl-ZZ or closing Vi before saving the file, you'll end up with a swap file (akin to a DOS/Windows temp file) in addition to the original file. Usually the swap file will have the .swp extension. The original file will not contain the recent changes you made; attempting to reopen it will result in an error message.

The swap file is not readable but can be recovered by typing a command like this at the $ prompt and pressing Enter: vi -r {your file name}

In some extreme cases, recovery is not possible. But in most cases, such as closing Vi before saving, a system crash, or a power failure, recovery works very well. After you recover, you must manually delete the swap file using a command like this at the $ prompt: rm .{your file name}.swp

Common Vi Commands

Press Key(s):*	Function:
I	Insert text before the cursor
A	Insert text after the cursor
:	Switch to ex mode
$	Go to last place on the line
^	Go to first place on the line
W	Next word B Previous word Shift
B	Previous word
Shift-G	Last line of the file 20
20 Shift-G	Go to line 20
Y	Copy. (Note: Y3W = copy 3 words; Y3J = copy 4 lines.)
P	Paste
D	Cut
X	Delete character under the cursor