PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

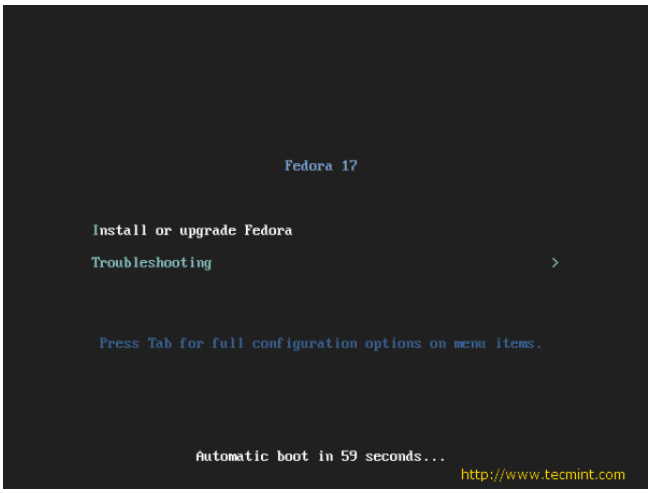
ASSINGMENT NO.	1
TITLE	Install Linux operating system.
PROBLEM STATEMENT /DEFINITION	Install, Configure 64 bit Linux Operating Systems, study basic architecture, memory system, and learn basic administration.
OBJECTIVE	To learn basics regarding installations of Linux.
OUTCOME	Students will learn the installation process.
S/W PACKAGES AND HARDWARE APPARATUS USED	Fedora 64 Bit OS ISO File, Machine supporting 64 bit operating system.
REFERENCES	http://www.tecmint.com/fedora-17-step-by-step-installation-guide-with-screenshots/
STEPS	1. Select **Install** or **Upgrade** Fedora. 2. Select **Basic Storage** device, if your hard drive is locally attached. 3. Set **Hostname** for your Fedora installation. 4. Select nearest city in your **Time Zone.** 5. Set **root** password. 6. Select appropriate **Partition** as per your requirement. 7. Verify **Filesystem** partition here or you can edit filesystem if you want. 8. Select optional **Packages** and click on **Next** install it. 9. Installation started, this may take several minutes as per selection of packages. 10. Packages installation is in progress. 11. Installation completed, Please remove **CD/DVD** and **Reboot** system. 12. Screen of **GRUB Boot Loader**, use arrow keys to select **Fedora Linux** to boot.

	13. Post installation of **Fedora 17**, Welcome Screen.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites: The basic understanding of installation process.

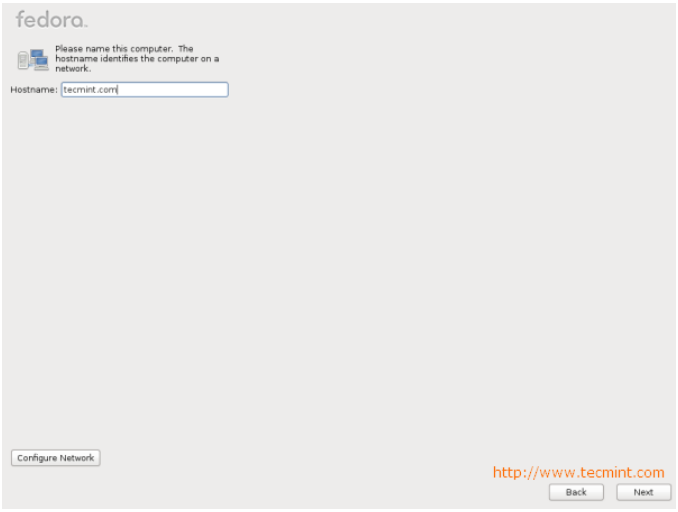
Concepts related Theory :

1. Select **Install** or **Upgrade** Fedora.

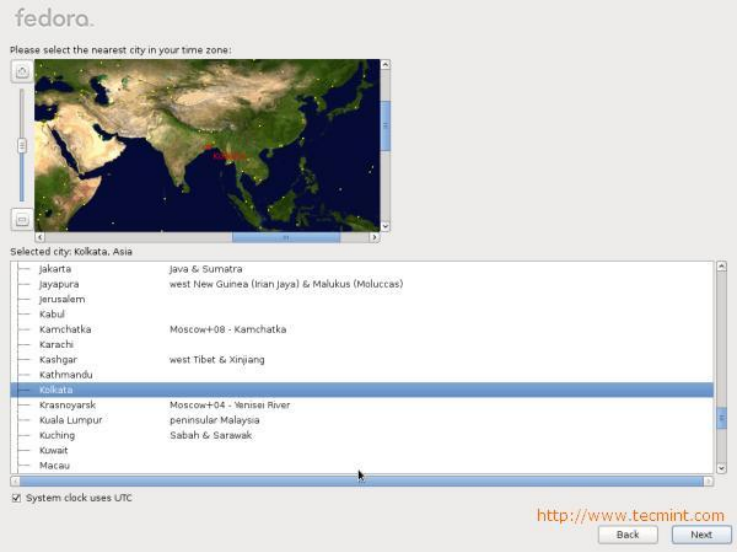


2. Select **Basic Storage** device, if your hard drive is locally attached.

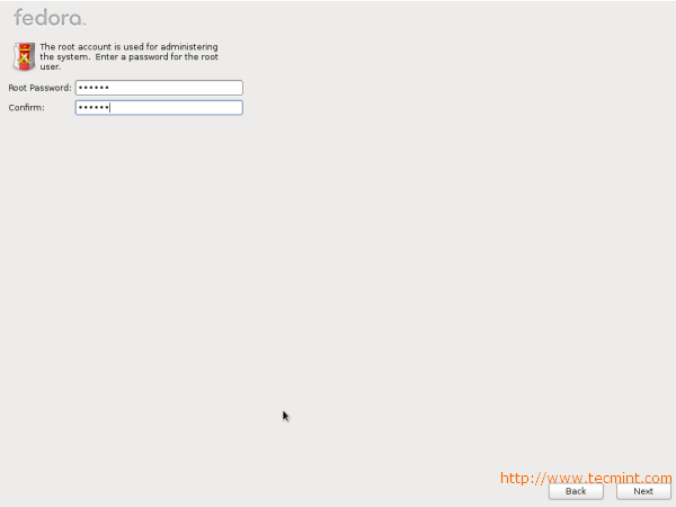
3. Set **Hostname** for your Fedora installation.



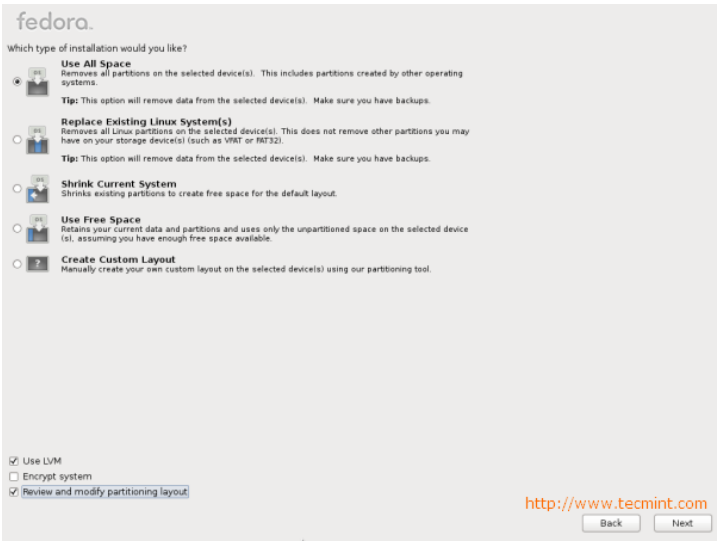
4. Select nearest city in your **Time Zone.**



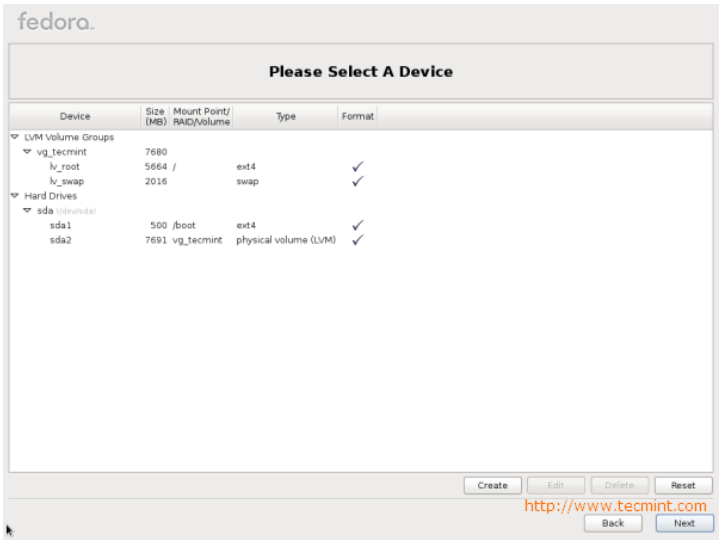
5. Set **root** password.



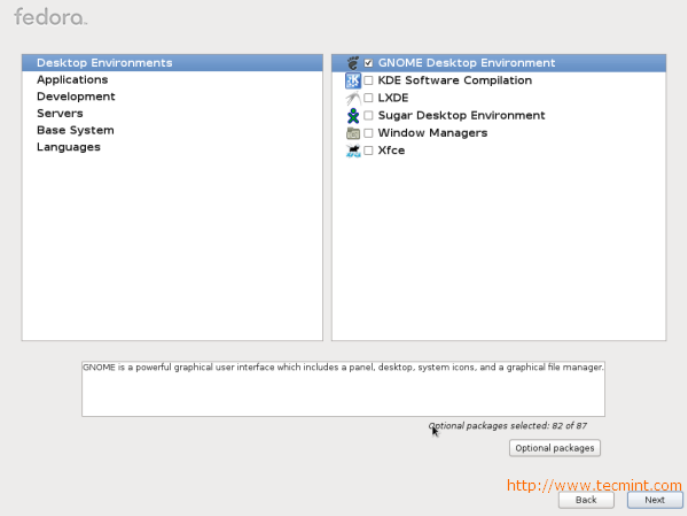
6. Select appropriate **Partition** as per your requirement.



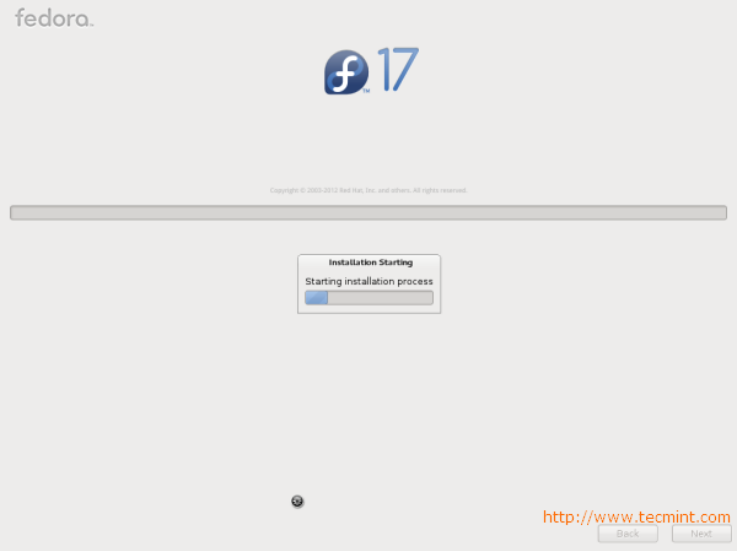
7. Verify **Filesystem** partition here or you can edit filesystem if you want.



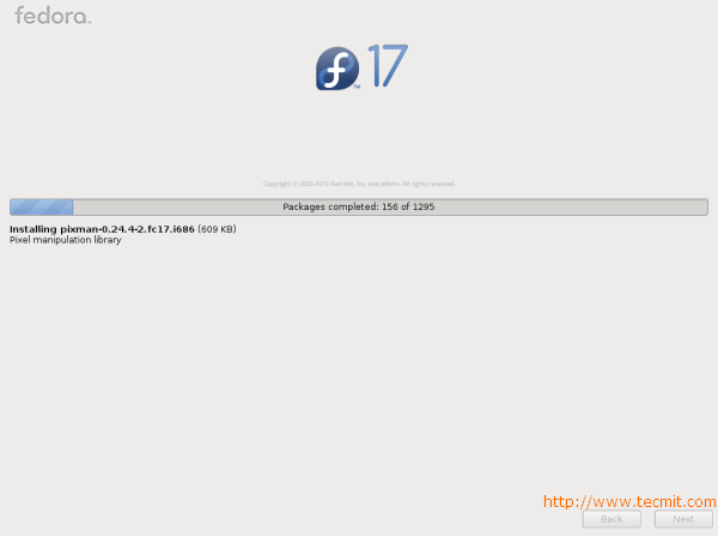
8.Select optional **Packages** and click on **Next** install it.



9. Installation started, this may take several minutes as per selection of packages.

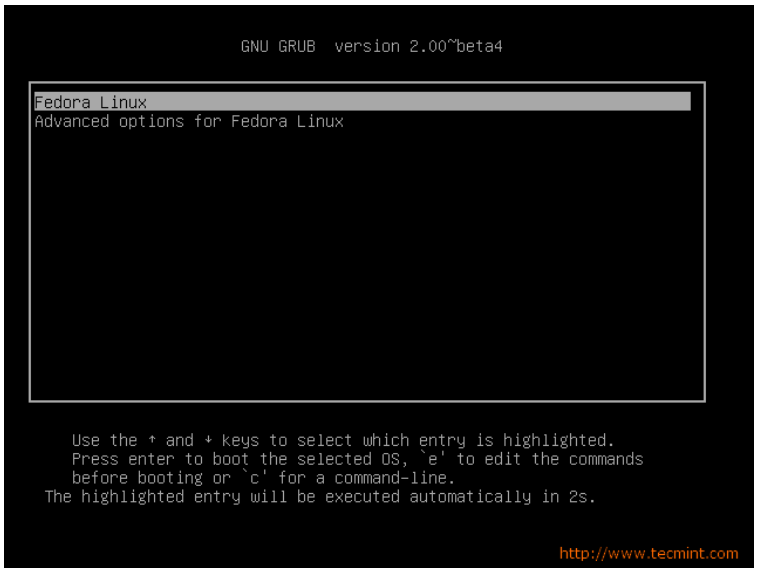


10. Packages installation is in progress.



11. Installation completed, Please remove **CD/DVD** and **Reboot** system.

12. Screen of **GRUB Boot Loader**, use arrow keys to select **Fedora Linux** to boot.



13. Post installation of Fedora 17, Welcome Screen.


Activities
Sun 02:30
narad shrestha
Windows
Applications
Type to search...
All
Accessories
Games
Graphics
Internet
Office
Sound & Video
System Tools
Universal Access
Other
Add/Remove Sof...
AisleRiot Solitaire
Archive Manager
Authentication
Automatic Bug R...
Backup
Brasero
Calculator
Character Map
Cheese
Contacts
Desktop Sharing
http://www.tecmint.com

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	2
TITLE	Implement a class Complex which represents the Complex Number data type.
PROBLEM STATEMENT /DEFINITION	Implement the following operations: 1. Constructor (including a default constructor which creates the complex number 0+0i). 2. Overloaded operator+ to add two complex numbers. 3. Overloaded operator* to multiply two complex numbers. 4. Overloaded <<and >>to print and read Complex Numbers.
OBJECTIVE	To understand arithmetic operation on the Complex number using class Complex.
OUTCOME	Enter number 3 4 3+4i 2 3 2+3i

	what operation you want to perform +,-,*,/ + addition is: 5+7i * multiplication is 6+12i / division is: 1.5+1.33i - substraction is: 1+1i Do you want to check for another two complex numbers(Y/N):N
S/W PACKAGES AND HARDWARE APPARATUS USED	Language C++, Implement in any IDE ex. Eclipse IDE for C++.
REFERENCES	1. Bjarne Stroustrup, ―The C++ Programming language, Third edition, Pearson Education. ISBN 9780201889543.2.Deitel, ―C++ 2. How to Program, 4thEdition, Pearson Education, ISBN:81-297-0276-2 3. Herbert Schildt, ―C++ The complete reference,Eighth Edition, McGraw HillProfessional, 2011, ISBN:978-00-72226805
STEPS	Logical approach to solve above problem: Create class Complex. It contains the following data members :

	• real-denoting real part of a complex number
	• image -denoting imaginary part of a complex number
	It contains the following member functions :
	• complex() -Default constructor.
	• operator+(complex)-Used to add two complex numbers.
	• operator*(complex)-Used to multiply two complex numbers.
	• operator<<(ostream&,complex&)-Overload<<(cout) operator which displays a complex number.
	• operator>>(iostream&,complex&)- Overload >>(cin) operator which takes a complex number as input.
	
	Steps:
	1.Create class Complex.
	2.Declareprivate data members-
	int real//Declaring an integer
	int imag//Declaring an integer
	3.Declare public members-
	complex();
	operator+(complex);
	operator*(complex);
	operator<<(ostream&,complex&);
	operator>>(istream&,complex&);
	};
	
	<u>Default constructor</u>
	
	complex()
	{

	Initializereal and imag as 0;
	}
	
	This function is used to add two complex numbers
	Procedure operator+(complex num1)
	{
	
	real=num1.real+real;
	imag=num1.imag+imag;
	complex num3(real,imag);
	}
	
	This function is used to multiply two complex numbers
	
	Procedure operator*(complex num1)
	{
	
	real=num1.real*real-num1.imag*imag;
	imag=num1.real*imag+num1.imag*real;
	complex num3(real,imag);
	}
	
	Overload <<(cout) operator which displays a complex number

	Procedure operator<<(ostream &dout,complex &c)
	{
	if(c.imag>=0)
	dout<<c.real<<"+"<<c.imag<<"i\n";
	else
	dout<<c.real<<c.imag<<"i\n";
	return (dout);
	}
	
	<u>Overload >>(cin) operator which takes a complex number as input</u>
	
	Procedure operator>>(istream &din,complex &c)
	{
	din>>c.real;
	din>>c.imag;
	return (din);
	}
	
	<u>Main procedure</u>
	
	do
	{
	complex num1,num2;
	read 1st and 2nd complex number in num1 and num2;
	complex num3,num4;
	num3=num1+num2;

	Print "The result of addition is: ";
	Print num3;
	
	num3=num1*num2;
	Print “result of multiplication is: ";
	Print num3;
	
	Read another complex number in num3;
	num1+=num3;
	Print num1;
	
	num2+=num3;
	Print num2;
	Print "Do you want to check for another two complex numbers(Y/N):";
	Read choice in ch;
	}while(ch=='Y'||ch=='y');

INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart

	9. Test cases 10. Conclusion/Analysis

Prerequisites: FPL-I and FPL-II

Concepts related Theory:

Complex numbers are numbers made up of a *real* part and an *imaginary* part. For example:

3.2 + 4i
1 + 3i
1 + 2.3i

In the degenerate case, 0 + 3i is an entirely imaginary number generally written as 3i, and 5 + 0i is an entirely real number generally written as 5. You can represent complex numbers using the complex data type.

Note - The complex arithmetic library (libcomplex) is available only for compatibility mode (-compat[=4]). In standard mode (the default mode), complex number classes with similar functionality are included with the C++ Standard Library libCstd.

The complex arithmetic library implements a complex number data type as a new data type and provides:

- Operators
- Mathematical functions (defined for the built-in numerical types)
- Extensions (for iostreams that allow input and output of complex numbers)
- Error handling mechanisms

Complex numbers can also be represented as an *absolute value* (or *magnitude*) and an *argument* (or *angle*). The library provides functions to convert between the real and imaginary (Cartesian) representation and the magnitude and angle (polar) representation.

The *complex conjugate* of a number has the opposite sign in its imaginary part.

Using the Complex Library

To use the complex library, include the header file complex.h in your program, and compile and link with the -library=complex

The complex arithmetic library defines one class: class complex. An object of class complex can hold a single complex number. The complex number is constructed of two parts:

- The real part
- The imaginary part

```
class complex {
    double re, im;
};
```

The value of an object of class complex is a pair of double values. The first value represents the real part; the second value represents the imaginary part.

Constructors of Class complex

There are two constructors for complex. Their definitions are:

```
complex::complex()                    {re=0.0; im=0.0;}

complex::complex(double r, double i = 0.0) {re=r;
im=i;}
```

If you declare a complex variable without specifying parameters, the first constructor is used and the variable is initialized, so that both parts are 0. The following example creates a complex variable whose real and imaginary parts are both 0:

```
complex aComp;
```

You can give either one or two parameters. In either case, the second constructor is used. When you give only one parameter, that parameter is taken as the value for the real part and the imaginary part is set to 0. For example:

```
complex aComp(4.533);
```

creates a complex variable with the following value:

```
4.533 + 0i
```

If you give two values, the first value is taken as the value of the real part and the second as the value of the imaginary part. For example:

```
complex aComp(8.999, 2.333);
```

creates a complex variable with the following value:

```
8.999 + 2.333i
```

You can also create a complex number using the polar function, which is provided in the complex arithmetic library . The polar function creates a complex value given the polar coordinates magnitude and angle.

There is no destructor for type complex.

Arithmetic Operators

The complex arithmetic library defines all the basic arithmetic operators. Specifically, the following operators work in the usual way and with the usual precedence:

+ - / * =

The subtraction operator (-) has its usual binary and unary meanings.

In addition, you can use the following operators in the usual way:

- Addition assign operator (+=)
- Subtraction assign operator (-=)
- Multiplication assign operator (*=)
- Division assign operator (/=)

However, the preceding four operators do not produce values that you can use in expressions. For example, the following expressions do not work:

```
complex a, b;
...
if ((a+=2)==0) {...}; // illegal
b = a *= b; // illegal
```

You can also use the equality operator (==) and the inequality operator (!=) in their regular meaning.

When you mix real and complex numbers in an arithmetic expression, C++ uses the complex operator function and converts the real values to complex values.

Algorithm: 1.Create class Complex.

2.Declareprivate data members-

int real//Declaring an integer

int imag//Declaring an integer

3.Declare public members-

complex();

4.Accept first complex number.

-accept real.

-accept imaginary.

5.Display first complex number.

6.Accept second complex number.

7.Select the operation to be performed(+,-,/).

8.Display the result.

9.Ask if user want to continue.

Review Questions:

1. What is concept of constructor, destructor**?**

2. What are types of constructors

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	**3**
TITLE	Write a C++ program for the addition of two quadratic polynomials and find the solution of the resultant polynomial.
PROBLEM STATEMENT /DEFINITION	Implement a class Quadratic that represents degree two polynomials i.e., polynomials of type ax^2+bx+c. The class will require three data members corresponding to a, b and c. Implement the following operations: 1. A constructor (including a default constructor which creates the 0 polynomial). 2. Overloaded **operator+** to add two polynomials of degree 2. 3. Overloaded << and >> to print and read polynomials. To do this, you will need to decide what you want your input and output format to look like. 4. A function eval that computes the value of a polynomial for a given value of x. 5. A function that computes the two solutions of the equation ax^2+bx+c=0.
OBJECTIVE	To learn, Operator overloading

OUTCOME	Students will be able to, • Perform the operator overloading • Understand the need of operator overloading
S/W PACKAGES AND HARDWARE USED	• Fedora 20 • C++ IDE (Eclipse) • I3 generation
REFERENCES	1. E Balgurusamy, ―Object Oriented Programming With C++, Fourth edition, McGraw Hill Higher Education.
STEPS	1. Declare data members 2. Declare a class called Quadratic 3. Define the member functions • **To represent two degree polynomial** • **To add two polynomial using 'operator+'** • **To read and print polynomial using << and >> operator** • **To compute the value of polynomial for a given value of x** • **To compute two solution of the equation ax^2+bx+c=0** 4. Print two polynomials 5. Add two polynomials 6. Print two solutions of resultant polynomial
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition

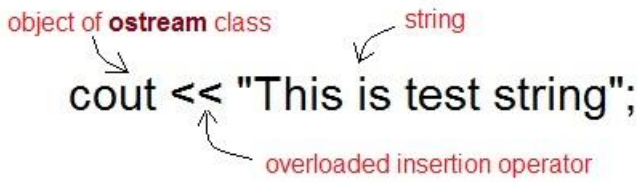
	4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

Basic knowledge of C++ programming with operator overloading concepts.

Concepts related Theory :

Operator overloading is an important concept in C++. It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on user-defined data type. For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String(concatenation) etc.

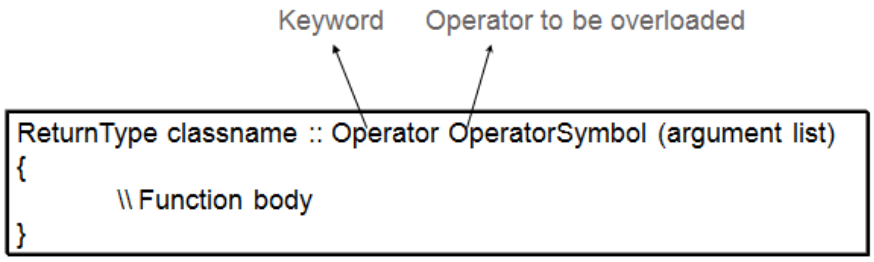


Almost any operator can be overloaded in C++. However there are few operator which can not be overloaded.**Operator that are not overloaded** are follows

- scope operator - ::

- sizeof
- member selector - .
- member pointer selector - *
- ternary operator - ?:

Syntax for operator overloading:



Implementing Operator Overloading:

Operator overloading can be done by implementing a function which can be :

1. Member Function
2. Non-Member Function
3. Friend Function

Operator overloading function can be a member function if the Left operand is an Object of that class, but if the Left operand is different, then Operator overloading function must be a non-member function. Operator overloading function can be made friend function if it needs access to the private and protected members of class.

Restrictions on Operator Overloading:

Following are some restrictions to be kept in mind while implementing operator overloading:

1. Precedence and Associativity of an operator cannot be changed.
2. Arity (numbers of Operands) cannot be changed. Unary operator remains unary, binary remains binary etc.
3. No new operators can be created, only existing operators can be overloaded.
4. Cannot redefine the meaning of a procedure. You cannot change how integers are added.

Algorithm:

1. Declare data members
2. Declare a class called Quadratic
3. Define the member functions
   - **To represent two degree polynomial**
   - **To add two polynomial using 'operator+'**
   - **To read and print polynomial using << and >> operator**
   - **To compute the value of polynomial for a given value of x**
   - **To compute two solution of the equation ax^2+bx+c=0**
4. Create objects for Quadratic class in main
5. Read the two polynomials
6. Call the overloaded operator + to perform the addition
7. Print the addition of the resultant polynomial.

Review Questions:

1. How to declare constructor?
2. What is operator overloading?
3. What is the use of operator overloading?

4. Which operator is overloaded by default by the compiler?

5. Explain overloading unary operator?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	**4**
TITLE	Write a C++ program to implement one dimensional array and perform different functions using operator overloading
PROBLEM STATEMENT /DEFINITION	Implement a class CppArray which is identical to a one-dimensional C++ array (i.e. the index set is a set of consecutive integers starting at 0) except for the following : 1. It performs range checking. 2. It allows one to be assigned to another array through the use of the assignment operator (e.g. cp1= cp2) 3. It supports a function that returns the size of the array. 4. It allows the reading or printing of array through the use of **cout** and **cin**.
OBJECTIVE	To learn, • Operator overloading
OUTCOME	Students will be able to, • Understand the need of operator overloading

	Perform the operator overloading
S/W PACKAGES AND HARDWARE USED	Fedora 20 C++ IDE (Eclipse)
REFERENCES	2. Bjarne Stroustrup, ―The C++ Programming language‖ , Third edition, Pearson Education. ISBN 9780201889543. 3. Deitel, ―C++ How to Program‖ , 4th Edition, Pearson Education, ISBN:81-297-0276-2
STEPS	7. Define a class named CppArray 8. Declare an array as one of the data member 9. Define the member functions to perform **range checking.** **Assignment of one array to another array through the assignment operator (e.g. cp1= cp2)** **Function to returns the size of the array.** **Reading or printing of array through the use of cout and cin.** 10. Create objects for array class in main 11. Read array object1 12. Call overloaded operator = o assign one array to another 13. Print the size of array by calling a function
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm

	8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

Basic concept of c++ with an array and operator overloading.

Concepts related Theory:

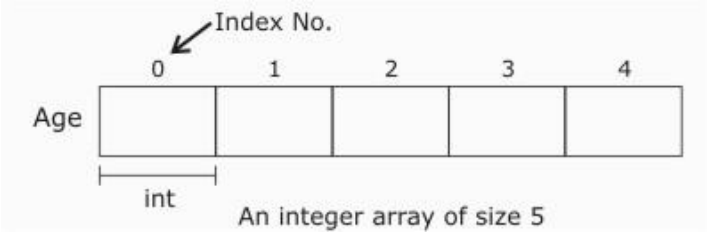
Array

An array is a collection of data elements of same data type. It is described by a single name and each element of an array is referenced by using array name and its subscript no.

Declaration of Array

Type arrayName[numberOfElements];

For example,
int Age[5] ;
float cost[30];



Operator overloading;

The meaning of operators are already defined and fixed for basic types like: int, float, double etc in C++ language. For example: If you want to add two integers then, + operator is used. But, for user-defined types(like: objects), you can define the meaning of operator, i.e, you can redefine the way that operator works. For example: If there are two objects of a class that contain string as its data member, you can use + operator to concatenate two strings. Suppose, instead of strings if

that class contains integer data member, then you can use + operator to add integers. This feature in C++ programming that allows programmer to redefine the meaning of operator when they operate on class objects is known as operator overloading.

Restriction:

- The operators **::** (scope resolution), **.** (member access), **.*** (member access through pointer to member), and **?:**(ternary conditional) cannot be overloaded.
- New operators such as ******, <>, or **&|** cannot be created.
- The overloads of operators **&&** and || lose short-circuit evaluation.
- The overload of operator -> must either return a raw pointer or return an object (by reference or by value), for which operator -> is in turn overloaded.
- It is not possible to change the precedence, grouping, or number of operands of operators.

Algorithm:

1. Define a class named CppArray
2. Declare an array as one of the data member
3. Define the member functions to perform
4. range checking.
5. Assignment of one array to another array through the assignment operator (e.g. cp1= cp2)
6. Function to returns the size of the array.
7. Reading or printing of array through the use of cout and cin.
8. Create objects for array class in main
9. Read array object1
10. Call overloaded operator = o assign one array to another
11. Print the size of array by calling a function

Review Questions:

1. Write the basic operation on one dimensional array.
2. What is difference between one dimensional array and multi dimensional array
3. Write short note on different types of operator.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	**5**
TITLE	Write a C++ program create a calculator for an arithmetic operator (+, -, *, /).
PROBLEM STATEMENT /DEFINITION	Write a C++ program create a calculator for an arithmetic operator (+, -, *, /). The program should take two operands from user and performs the operation on those two operands depending upon the operator entered by user. Use a switch statement to select the operation. Finally, display the result. Some sample interaction with the program might look like this: • Enter first number, operator, second number: 10 / 3 • Answer = 3.333333 • Do another (y/n)? y • Enter first number, operator, second number: 12 + 100 • Answer = 112 • Do another (y/n)? n
OBJECTIVE	To learn, • Arithmetic operation • Switch statement
OUTCOME	Students will be able to, • Perform arithmetic operation

	Understand the need of switch statement
S/W PACKAGES AND HARDWARE USED	Fedora 20 C++ IDE (Eclipse) I3 generation
REFERENCES	4. E Balgurusamy, ―Object Oriented Programming With C++, Fourth edition, McGraw Hill Higher Education.
STEPS	14. Declare data operands 15. Get two operands from user 16. Use switch statement for deciding operator 17. Print the result 18. Ask the user do you want to continue If yes, goes to step 2 and repeat the procedure If not, pause
INSTRUCTIONS FOR WRITING JOURNAL	21. Date 22. Assignment no. 23. Problem definition 24. Learning objective 25. Learning Outcome 26. Concepts related Theory 27. Algorithm 28. Flow chart 29. Test cases 30. Conclusion/Analysis

Prerequisites:

Basic knowledge of C++ programming with switch statement.

Concepts related Theory :

Operators:

Once introduced to variables and constants, we can begin to operate with them by using *operators*. What follows is a complete list of operators. At this point, it is likely not necessary to know all of them, but they are all listed here to also serve as reference.

Assignment operator (=) ;

The assignment operator assigns a value to a variable.

```
x = 5;
```

This statement assigns the integer value 5 to the variable x. The assignment operation always takes place from right to left, and never the other way around:

```
x = y;
```

This statement assigns to variable x the value contained in variable y. The value of x at the moment this statement is executed is lost and replaced by the value of y.

Unary arithmetic operators

There are two unary arithmetic operators, plus (+), and minus (-). If you remember, unary operators are operators that only take one operand.

Operator	Symbol	Form	Operation
Unary plus	+	+x	Value of x
Unary minus	-	-x	Negation of x

The unary plus operator returns the value of the operand. In other words, +5 = 5, and +x = x. Generally you won't need to use this operator since it's redundant. It was added largely to provide symmetry with the unary minus operator.

The unary minus operator returns the operand multiplied by -1. In other words, if x = 5, -x = -5.

For best effect, both of these operators should be placed immediately preceding the operand (eg. -x, not - x).

Do not confuse the unary minus operator with the binary subtraction operator, which uses the same symbol. For example, in the expression x = 5 - -3;, the first minus is the subtraction operator, and the second is the unary minus operator.

Binary arithmetic operators

There are 5 binary arithmetic operators. Binary operators are operators that take a left and right operand.

Operator	Symbol	Form	Operation
Addition	+	x + y	x plus y
Subtraction	-	x - y	x minus y
Multiplication	*	x * y	x multiplied by y
Division	/	x / y	x divided by y
Modulus (Remainder)	%	x % y	The remainder of x divided by y

The addition, subtraction, and multiplication operators work just like they do in real life, with no caveats.

Switch statement:

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

Syntax:

The syntax for a **switch** statement in C++ is as follows:

```
switch(expression){
   case constant-expression  :
      statement(s);
      break; //optional
   case constant-expression  :
      statement(s);
      break; //optional

   // you can have any number of case statements.
   default : //Optional
      statement(s);
}
```

Algorithm:

1. Declare data operands
2. Get two operands from user
3. Use switch statement for deciding operator
4. Print the result
5. Ask the user do you want to continue
    - If yes, goes to step 2 and repeat the procedure
    - If not, pause

Review Questions:

1. What are the uses of arithmetic operator?

2. Explain the rules of writing switch statement.

3. Draw the flow diagram of switch statement.

4. Explain modular operator.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	6
TITLE	Student Database with different OO concept
PROBLEM STATEMENT /DEFINITION	Create student database with appropriate data members that should use the following features of object oriented programming in C++. Class, Object, array of objects, new , delete, default constructor to initialize student class fields, parameterized constructor to set the values into the objects, access specifiers, this pointer.
OBJECTIVE	1. To understand the Object oriented programming Paradigms 2. To understand the concept of class & object. 3. To understand How to create and delete a object 4. To understand the concept of constructor, destructor, & access specifiers, this pointer
OUTCOME	Database is created for student record
S/W PACKAGES AND HARDWARE	64-bit Open source Linux or its derivative Open Source C++ Programming tool like G++/GCC

APPARATUS USED	
REFERENCES	Bjarne Stroustrup, ―The C++ Programming language‖ , Third edition, Pearson Education. ISBN 9780201889543
STEPS	1. Declare the class student with student, roll number and subject as data member. 2. accept() and display() as member functions 3. Declare the class exam with subject code , internal assessment and university examination marks as data member. 4. accept() and display() as member functions 5. Print all information from database.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

Data Structures and problem Solving

Concepts related Theory:

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 creates 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. **P**a**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 Cant 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.

Static data members of a class are also known as "class variables", because there is only one unique value for all the objects of that same class. Their content is not different from one object static members have the same properties as global variables but they enjoy class scope. For that reason, and to avoid them to be declared several times, we can only include the prototype (its declaration) in the class declaration but not its definition (its initialization). In order to initialize a static data-member we must include a formal definition outside the class, in the global scope of this class to another. Because it is a unique variable value for all the objects of the same class, it can be referred to as a member of any object of that class or even directly by the class name.

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.

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 follows

Item *it_ptr ;

Object pointers are useful in creating objects at runtime. We can also access public members of the class 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:

- **this** pointer stores the address of the class instance, to enable pointer access of the members to the member functions of the class.
- **this** pointer is not counted for calculating the size of the object.
- **this** pointers are not accessible for static member functions.
- **this** pointers are not modifiable.

Algorithm:

1. Declare the class student with
    a. student,roll number and subject as data member
    b. accept() and display() as member functions
2. Declare the class exam with
    a. subject code ,internal assessment and university examination marks as data member
    b. accept() and display() as member functions
3. Print all information from database.
4. Stop

Review Questions:

1. Explain with examples pointers to object.
2. What is this pointer? Explain with examples
3. What are inline functions?
4. What is concept of constructor, destructor?
5. What are types of constructors?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	7
TITLE	Student Database with different OO concept
PROBLEM STATEMENT /DEFINITION	Implement C++ program to write a class template to represent a generic vector. Include following member functions: • To create the vector. • To modify the value of a given element • To multiply by a scalar value • To display the vector in the form (10,20,30,…)
OBJECTIVE	1. To understand the concept of Template? 2. To understand the concept of class Template & function Template?
OUTCOME	Class Template multiplies a scalar values.
S/W PACKAGES AND HARDWARE APPARATUS USED	64-bit Open source Linux or its derivative Open Source C++ Programming tool like G++/GCC
REFERENCES	Bjarne Stroustrup, ―The C++ Programming language‖ , Third

	edition, Pearson Education. ISBN 9780201889543
STEPS	Create a Template Create a vector to modify,multiply & display Create main fuction to call display
INSTRUCTIONS FOR WRITING JOURNAL	11. Date 12. Assignment no. 13. Problem definition 14. Learning objective 15. Learning Outcome 16. Concepts related Theory 17. Algorithm 18. Flow chart 19. Test cases 20. Conclusion/Analysis

Prerequisites:

FPL-I and FPL-II

Data Structures and problem Solving

Concepts related Theory :

Template supports generic programming i.e. it is a mechanism that makes it possible to use one function or class to handle many different data types. By using templates we can design a single class/function that operates on data of many types, instead of having to create a separate class/function for each type.

Template <class T>

<return type><name of function>(<argument>)

{

//body of function

}

Algorithm:

template<class T> class vector

{

T *v; int size;

vector(int m); void create(T *a); modify(int); multiply(int); display();

};

template<class T>

vector< T > :: vector(int m)

{

v=new T[size=m]; for(int i=0;i<size;i++) v[i]=0;

}

template<class T>

void vector<T>::create(T*a)

{

for(int i=0;i<size;i++)

{

cin>>a[i];

```
v[i]=a[i];
}
}

template<class T>

void vector<T>::modify(int k)
{
v[k]=v[k]+10;
}

template<class T>
void vector<T>::multiply(int k)

{

 for(int i=0;i<size;i++) v[i]=v[i]*k;
}

template<class T>

void vector<T>::display()
{
cout.setf(ios::showpoint);
cout<<"\n(";
for(int i=0;i<size;i++) cout<<v[i]<<", "; cout<<")\n";
}
```

```
int main()
{
vector <float> v1(5); vector <int> v2(5); float *x;
int *y; int i; int s;
v1.modify(i);
v1.display();
v1.multiply(s);
}
```

Review Questions:

1. What is generic programming? How it is implemented in c++?
2. What is template?
3. Distinguish between function overloading and function template?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	8
TITLE	Rational number
PROBLEM STATEMENT /DEFINITION	Create a class Rational Number (fractions) with the following capabilities: a) Create a constructor that prevents a 0 denominator in a fraction, reduces or simplifies fractions that are not in reduced form and avoids negative denominators. b) Overload the addition, subtraction, multiplication and division operators for this class. c) Overload the relational and equality operators for this class.
OBJECTIVE	• To understand concept of OOP. • To understand class structures for Rational number. • To perform primitive operations on Rational numbers.
OUTCOME	• To write class for Rational number. • To implement primitive operations on Rational numbers. • To implement above stated capabilities..
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems (64-Bit)64-BIT Fedora 17 or latest 64-BIT Up date of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bit) , TC++
REFERENCES	• C++ by B.Stroustrup • OOP Using C++ by Balagurusami

STEPS	Refer to algorithm.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites: Basic of object oriented features (object and class) and basics of rational number. Basics operations on Rational numbers .

Concepts related Theory :

Rational number

A rational number is a number that can be written as a ratio. That means it can be written as a fraction, in which both the numerator (the number on top) and the denominator (the number on the bottom) are whole numbers.

- The number 8 is a rational number because it can be written as the fraction 8/1.
- Likewise, 3/4 is a rational number because it can be written as a fraction.
- Even a big, clunky fraction like 7,324,908/56,003,492 is rational, simply because it can be written as a fraction.

Every whole number is a rational number, because any whole number can be written as a fraction. For example, 4 can be written as 4/1, 65 can be written as 65/1, and 3,867 can be written as 3,867/1.

Algorithm: 1. Create a class called RationalNumber.

2. Accept numerator from user.

3. Accept denominator from user.

4. Validate if denominator is equal to 0, again accept denominator value.

5. Ask what operation needs to be performed.

6. Display the result.

7. Ask if user wants to continue.

8. Exit.

Review Questions: 1. Give the class structure of the above program?

2. Give time complexity of the above problem.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	9
TITLE	Multiple Inheritance
PROBLEM STATEMENT /DEFINITION	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 a publication. 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 classes, 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 write a program that instantiates the book and tape classes, allows user to enter data and displays the data members. -To learn the basic object oriented concepts of inheritance, classes, object creation and exception handling.
OUTCOME	Students will be able to- - write a program that instantiates the book and tape classes, allows user to enter data and displays the data members. - apply the object oriented concepts of inheritance, classes, object creation and exception handling.
	Software:

S/W PACKAGES AND **HARDWARE APPARATUS USED**	-Linux based Operating System(Fedora 20, 64 bit) -GCC compiler -An integrated Development Environment(Eclipse) *Hardware*: A machine with at least the following specifications -Intel® Core™ i3-3240 CPU @ 3.40GHz × 4 processor -4 GBRAM
REFERENCES	1.Robert Lafore, ―Object-Oriented Programming in C++‖ , fourth edition, Sams Publications 2.Herbert Schildt, ―C++ The complete reference‖ , Eighth Edition, McGraw Hill Professional, 2011

STEPS	1.Create a class publication.
	class publication
	{
	private:
	string title;
	float price;
	
	public:
	publication(); //constructor
	void getData()
	{
	system ("cls");
	cout<<"\nEnter the title:";
	getline(cin,title);
	
	cout<<"Enter the price:";
	cin>>price;
	}
	
	void putData()const
	{
	system ("cls");
	cout<<"The Title:"<<title;
	cout<<"The Price:"<<price;
	}
	};

	2.Create a class book and tape inherited from publication class.
	class book: public publication
	{ private:
	int pCount;
	public:
	book();
	void getData()
	{ cout<<"Enter number of pages:";
	cin>> pCount; }
	void putData()
	{ cout<<"The total number of pages:"<<pCount;}};
	class tape: public publication
	private:
	int playTime;
	public:
	tape();
	void getData()
	{ cout << "Play time in minutes: ";
	cin >> playTime;}
	void putData() const
	{ cout << "\nPlaying time: " << playTime;}};
	
	3.Accept book detail and tape details.
	4.Display the result.
	}
	};
	}
	};

INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites: Previous knowledge of basic object oriented concepts like classes, object creation, inheritance and exception handling.

Concepts related 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 behaviour of the existing classes. An existing class that is "parent" of a new class is called a base class.

Types of Inheritance:

1. Single level inheritance: In single level inheritance, there is only one base class and has only one derived class. Example of single level inheritance:

```
class Base
{
};
class Derv: public Base
{

};
```

2. Multi level inheritance:In multi level inheritance, there will be a chain of inheritance with a class derived from only one parent and will have only one child class. Example of multi level inheritance:

```
class A
{
};
class B: public A
{

};

class C: public B
{

};
```

3. Multiple inheritance: One class being derived from multiple parent classes. Example of multiple inheritance:

```
class A
{
};


class B
{

};

class C: public B, public A
{

};
```

4. Hierarchical inheritance: Many classes deriving from one class. Example of hierarchical inheritance:

```
class A
{
};
class B: public A
{

};
```

```
class C: public A
{

};
```

5. Hybrid inheritance: It is a mixture of 2 or more of above types of inheritance. There is no pattern of deriving from classes. Example of hybrid inheritance:

```
class A
{
};
class B
{

};

class C: public A, public B
{

};

class D: public A
{

};
class X: public D, public C
{

};
```

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

Algorithm:

1. **create publication class.**
2. **Create class book and tape inherited from publication class.**
3. Accept book detail and tape details.
4. Display the result.

Review Questions:

1. Give the syntax of hybrid inheritance.

2. Give the syntax of exception handling using try and catch blocks.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	10
TITLE	Display no of lines not starting with alphabet "A"
PROBLEM STATEMENT /DEFINITION	Write a function in C++ to count and display the number of lines not starting with alphabet "A" present in text file "Story.txt". Example: If the "Story.txt" contains following lines: The roses are red. A girl is playing there. There is a playground. An airplane is in sky. Numbers are not allowed in the password. The function should display the output as – 3.
OBJECTIVE	5. To open and read a file. 6. To write a function to perform said problem statement.
OUTCOME	1. To display lines count that is not starting with alphabet "A".
S/W PACKAGES AND HARDWARE APPARATUS USED	64-bit Open source Linux or its derivative Open Source C++ Programming tool like G++
REFERENCES	Data Structures, Algorithms and Applications in C++, Second Edition, Sartaj Sahni.

STEPS	1. Prepare an input file which have contents 2. Open and read input file. 3. Read first character of each line. 4. If read first character is not alphabet "A", increment the output counter by 1. 5. Repeat steps 3 and 4 till the end of file.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

Data Structures and problem Solving.

Concepts Related Theory:

File Handling concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory (Hard disk).

Why use file handling:

- For permanent storage.
- The transfer of input - data or output - data from one computer to another can be easily done by using files.

For read and write from a file you need another standard C++ library called fstream, which defines three new data types:

Data type	Description
Ofstream	This is used to create a file and write data on files
Ifstream	This is used to read data from files
Fstream	This is used to both read and write data from/to files

How to achieve file handling:

For achieving file handling in C++ we need follow following steps

- Naming a file
- Opening a file
- Reading data from file
- Writing data into file
- Closing a file

Functions used in file handling:

Functions	Operation
open()	To create a file
close()	To close an existing file
get()	Read a single character from a file
put()	write a single character in file.
read()	Read data from file
write()	Write data into file.

Defining and opening a file:

The function open() can be used to open multiple files that use the same stream object.

Syntax:

```
file-stream-class   stream-object;

stream-object.open("filename");
```

Example:

```
ofstream  outfile;      // create stream

outfile . open ("data1");  // connect stream to data1
```

Closing a file:

A file must be close after completion of all operation related to file. For closing file we need**close()** function.

Syntax:

```
outfile.close();
```

File opening mode:

Mode	**Meaning**	**Purpose**
ios :: out	Write	Open the file for write only.
ios :: in	read	Open the file for read only.
ios :: app	Appending	Open the file for appending data to end-of-file.
ios :: ate	Appending	take us to the end of the file when it is opened.

Both ios :: app and ios :: ate take us to the end of the file when it is opened. The difference between the two parameters is that the ios :: app allows us to add data to the end of file only, while ios :: ate mode permits us to add data or to modify the existing data anywhere in the file.

The mode can combine two or more parameters using the bitwise OR operator (symbol |)

Example

```
fstream file;

file.Open("data . txt", ios :: out | ios :: in);
```

read() and write() functions:

These functions take two arguments. The first is the address of the variable V , and the second is the length of that variable in bytes. The address of variable must be cast to type char * (i.e pointer to character type).

Example

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

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

Steps:

1. Prepare an input file which have contents
2. Open and read input file.
3. Read first character of each line.
4. If read first character is not alphabet “A”, increment the output counter by 1.
5. Repeat steps 3 and 4 till the end of file.

Questions:

6. Write a C++ program to write number 1 to 100 in a data file NOTES.TXT.
7. Write a function to count the number of blank present in a text file named "OUT.TXT".
8. Which stream class is to only write on files?
9. Which stream class is to only read from files?
10. ios::trunc is used for?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	**11**
TITLE	Write a C++ program using virtual base class and print initial and converted value.
PROBLEM STATEMENT /DEFINITION	Write C++ Program with base class convert declares two variables, val1 and val2, which hold the initial and converted values, respectively. It also defines the functions **getinit( )** and **getconv( )**, which return the initial value and the converted value. These elements of convert are fixed and applicable to all derived classes that will inherit convert. However, the function that will actually perform the conversion, **compute( )**, is a pure virtual function that must be defined by the classes derived from convert. The specific nature of compute( ) will be determined by what type of conversion is taking place.
OBJECTIVE	To learn, • Inheritance • Virtual base class
OUTCOME	Students will be able to, • Perform inheritance

	Understand the need of virtual function
S/W PACKAGES AND HARDWARE USED	Fedora 20 C++ IDE (Eclipse) I3 generation
REFERENCES	5. E Balgurusamy, ―Object Oriented Programming With C++, Fourth edition, McGraw Hill Higher Education.
STEPS	1. Declare the base class convert 2. Declare two variables 3. Declare function **getinit( )** and **getconv( )** 4. Create a derived class from base class 5. Define function **compute ( )** 6. Create derived class object 7.Display the values
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases

	10. Conclusion/Analysis

Prerequisites:

Basic knowledge of C++ programming with inheritance concepts.

Concepts related Theory:

Inheritance:

One of the most important concepts in object-oriented programming is that of inheritance. Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and fast implementation time.

When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the **base** class, and the new class is referred to as the **derived** class.

Types of inheritance :

When deriving a class from a base class, the base class may be inherited through **public, protected** or **private** inheritance. The type of inheritance is specified by the access-specifier as explained above.

We hardly use **protected** or **private** inheritance, but **public** inheritance is commonly used. While using different type of inheritance, following rules are applied:

- **Public Inheritance:** When deriving a class from a **public** base class, **public** members of the base class become **public** members of the derived class and **protected** members of the base class become **protected** members of the derived class. A base class's **private** members are never accessible directly from a derived class, but can be accessed through calls to the **public** and **protected** members of the base class.

- **Protected Inheritance:** When deriving from a **protected** base class, **public** and **protected** members of the base class become **protected** members of the derived class.

- **Private Inheritance:** When deriving from a **private** base class, **public** and **protected** members of the base class become **private** members of the derived class.

Virtual base class:

- An ambiguity can arise when several paths exist to a class from the same base class. This means that a child class could have duplicate sets of members inherited from a single base class.
- C++ solves this issue by introducing a virtual base class. When a class is made virtual, necessary care is taken so that the duplication is avoided regardless of the number of paths that exist to the child class.

Syntax:

```
1. class class_name //This denotes the base class of C++ virtual function
2. {
3. public
4. virtual void virtualfunctioname() = 0 //This denotes the pure virtual function in C++
5. }
```

Example:

```
class A
{
  public:
     int i;
};
class B : virtual public A
```

```
{
  public:
     int j;
};
class C: virtual public A
{
  public:
     int k;
};
class D: public B, public C
{
  public:
     int sum;
};

int main()
{
  D ob;
  ob.i = 10; //unambiguous since only one copy of i is inherited.
  ob.j = 20;
  ob.k = 30;
  ob.sum = ob.i + ob.j + ob.k;
  cout << “Value of i is : ”<< ob.i<<”\n”;
  cout << “Value of j is : ”<< ob.j<<”\n”; cout << “Value of k is :”<< ob.k<<”\n”;
  cout << “Sum is : ”<< ob.sum <<”\n”;

  return 0;
}
```

Algorithm:

1. Declare the base class convert
2. Declare two variables
3. Declare function **getinit( )** and **getconv( )**
4. Create a derived class from base class
5. Define function **compute ( )**
6. Create derived class object
7. Display the values

Review Questions:

6. What is the use of inheritance?
7. What is the use of virtual base class?
8. What is multiple inheritance? Discuss the syntax and rules of multiple inheritance in C++.
9. What is virtual function?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	12
TITLE	Book shop inventory.
PROBLEM STATEMENT /DEFINITION	A book shop maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, price, publisher and stock position. Whenever a customer wants a book, the sales person inputs the title and author and the system searches the list and displays whether it is available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book details and requests for the number of copies required. If the requested copies book details and requests for the number of copies required. If the requested copies are available, the total cost of the requested copies is displayed; otherwise the message ―Required copies not in stock‖ is displayed. Design a system using a class called books with suitable member functions and Constructors. Use new operator in constructors to allocate memory space required. Implement C++ program for the system.
OBJECTIVE	To implement book shop inventory.
OUTCOME	Implementation of book shop inventory.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems (64-Bit)64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bit),TC++
REFERENCES	• C++ by B. Stroustrup • Fundamental of Data Structure in C++.
STEPS	Refer to algorithm

INSTRUCTIONS FOR **WRITING JOURNAL**	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

- Basic of Database.
- Basics of problem solving.

Concepts related Theory

Book shop database is a big database, maintaining records of each book is a tedious task.

Each book is given a unique id called book id. Search for a particular book is done on book id.

Searching

linear search

Linear search is a very simple search algorithm. In this type of search, a sequential search is made over all items one by one. Every items is checked and if a match founds then that particular item is returned otherwise search continues till the end of the data collection.

Binary search

Binary search is a fast search algorithm with run-time complexity of O(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly the data collection should be in sorted form.

Binary search search a particular item by comparing the middle most item of the collection. If match occurs then index of item is returned. If middle item is greater than item then item is searched in sub-array to the right of the middle item other wise item is search in sub-array to the

left of the middle item. This process continues on sub-array as well until the size of subarray reduces to zero.

Algorithm:

```
Step 1: Start.
Step 2: Accept books details from user.
Step 3: Accept book id that is to be searched.
Step 4: Display the book detail, if available.
Step 5: Exit.
```

Review Questions:

1. Give the class structure of the above program?
2. What are the member function and the constructor used in the above program?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	13
TITLE	Create employee bio-data using classes
PROBLEM STATEMENT /DEFINITION	Classes are: - Personal record - Professional record - Academic record Assume appropriate data members and member function to accept required data &print bio-data. Create bio-data using multiple inheritances using C++.
OBJECTIVE	Use of classes and multiple inheritance in c++.
OUTCOME	Enter name:raj enter address:dhankawadi pune Enter date of birth(DD/MM/YYYY):27/08/1990 Enter phone no.:9876543291 SSC percentage: 87 HSC percentage:89 Enter qualification:Btech Enter company name: Morgan Stanley Enter current designation in :software developer biodata: name:raj

	address:Dhankawadi pune DOB:27/08/1990 Phone no:9876543291 SSC percentage: 87 HSC percentage:89 qualification:Btech working company:Morgan Stanley Designation :software developer
S/W PACKAGES AND HARDWARE APPARATUS USED	Language C++, Implement in any IDE ex. Eclipse IDE for C++.
REFERENCES	1.Bjarne Stroustrup, ―The C++ Programming language, Third edition, Pearson Education. ISBN 9780201889543.2.Deitel, ―C++ 2.How to Program, 4thEdition, Pearson Education, ISBN:81-297-0276-2 3.Herbert Schildt, ―C++ The complete reference,Eighth Edition, McGraw HillProfessional, 2011, ISBN:978-00-72226805

STEPS	class personal {
	protected:
	char name[50],address[50];
	long long phno; char dob[12];
	public:
	virtual void displayp()
	{
	//display data Name ,Address,Phone no.,Date of birth
	}

```
};
class academics {
protected: char quali[20];
float ssc,hsc;
public: void displaya()
{
 //display data Qualification,SSC percentage,HSC percentage}
};
class professional {
 Protected: int exp;
char company[30],work[20];

public:
void displaypr()
{
//display data like
//Professional experience,Presently working in company,Current
//designation etc.
}
 };

class biodata:public personal,public academics,public professional
 {
 public: void accept();

 void display()
{
```

	personal::displayp();
	academics::displaya();
	professional::displaypr();
	}
	};
	
	void biodata::accept()
	{
	//accept data for database.
	}
	int main()
	{
	biodata b;
	b.accept();
	b.display();
	return 0;
	}

INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites: Basics of C++ and database. Multiple inheritance**.**

Concepts related Theory:

Algorithm: step1: create class called biodata.

Step2: accept users details-

-personal record

-academics record

-professional record

Step3: Display the record

Step4: exit.

Review Questions:

1. What are the types of inheritance?

2. Give the physical structure of hybrid inheritance.

3. Explain multiple inheritances in C++.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	14
TITLE	Crete User defined exception to check the following conditions and throw the exception if the criterion does not meet.

PROBLEM STATEMENT /DEFINITION	a. User has age between 18 and 55 b. User stays has income between Rs. 50,000 – Rs. 1,00,000 per month c. User stays in Pune/ Mumbai/ Bangalore / Chennai d. User has 4-wheeler Accept age, Income, City, Vehicle from the user and check for the conditions mentioned above. If any of the condition not met then throw the exception.
OBJECTIVE	To learn the concept of exception handling in c++.
OUTCOME	Enter user data 19 52000 Pune 4 User data: Age:19 Income:52000 City:Pune Vehicle_wheels:4 Enter user data 21 52000 Mumbai 2 User data

	Age:21 Income:52000 City:Mumbai Vehicle should be 4 wheeler.//exception thrown.
S/W PACKAGES AND HARDWARE APPARATUS USED	Preferable OS with 64 bit, any IDE for example eclipse IDE for C++.
REFERENCES	1.Yedidyah Langsam, Moshe J Augenstein, Aron M Tenenbaum, ―Data Structures using C and C++, 2.Pearson Education, ISBN 81317-0328-2 A Michael Berman, ―Data Structures via C++: Objects by Evolution, Oxford University Press, ISBN:0-19-510843-4 3.Horowitz and Sahani, ―Fundamentals of Data Structures in C++, University Press, ISBN10:0716782928 ISBN13: 9780716782926.
STEPS	logic to write code : Step 1: Start the program. Step 2: take input from user or u can take it in catch block too. Step 3: Within the try block check whether the value is greater than zero or not. a. if the value of age is not between 18 and 55 and catch the corresponding exception. b. if the value of income between Rs. 50,000 – 1,00,000 per month and catch the corresponding exception. c. if the value of city is not Pune, Mumbai, Banglore, Chennai

	and catch the corresponding exception.
	d. if the value of vehicle is not 4-wheeler and catch the corresponding exception.
	
	Step 4: Read the integer and character values
	Step 5: Stop the program.
	
	<u>Code idea:</u>
	1.Include the library function exception first.
	2.write main function
	int main()
	{
	int age, vehicle_wheels;
	float income;
	string city;
	//take inputs from user here then no need to take it in try block
	try
	{
	//take input from user age
	
	If( )//condition for age like (age<18 && age>55)we need to throw //error
	{
	throw 100;
	}

	//take input from user income
	if( )//condition for income same like age
	{
	throw 1.1f;
	}
	
	//take input from user city
	if( )//condition for city use the "||" operator for city condition.
	{
	throw "a";
	}
	
	if( )//condition for wheels in vehicle like(vehicle_wheels!=4)
	{
	throw 10;
	}

	}//end of try block
	catch(int a)
	{
	//display message
	}

	catch(float b)
	{
	//display message
	}
	
	catch(char c)
	{
	//display message
	}
	catch(…)
	{
	//display message
	}
	return 0;
	
	}

INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases

	10. Conclusion/Analysis

Prerequisites: Basic concept of C++.

Concepts related Theory:

Algorithm:

Step 1: Start the program.

Step 2: take input from user or u can take it in catch block too.

Step 3: Within the try block check whether the value is greater than zero or not.

a. if the value of age is not between 18 and 55 and catch the corresponding exception.

b. if the value of income between Rs. 50,000 – 1,00,000 per month and catch the corresponding exception.

c. if the value of city is not Pune, Mumbai, Banglore, Chennai and catch the corresponding exception.

d. if the value of vehicle is not 4-wheeler and catch the corresponding exception.

Step 4: Read the integer and character values

Step 5: Stop the program.

Review Questions:

1. How exceptions are handled in c?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	15
TITLE	File Handling
PROBLEM STATEMENT /DEFINITION	Write a menu driven program that will create a data file containing the list of telephone numbers in the following form John 23456 Ahmed 9876 .... Use a class object to store each set of data, access the file created and implement the following tasks I.Determine the telephone number of specified person II.Determine the name if telephone number is known III.Update the telephone number, whenever there is a change.
OBJECTIVE	To implement operations of File Handling .
OUTCOME	By the end of the experiment Students will be able to implement all the applications of file handling : 1)Storing data 2)Accessing data 3)Creating , modify and delete .
	Software :

S/W PACKAGES AND **HARDWARE APPARATUS USED**	Linux based operating system,eclipse,JDK. Hardware : Intel core 2 duo,256 MB of RAM,512 MB of disk space .
REFERENCES	Object Oriented Programming with C++ by E. Balaguruswamy
STEPS	1. Accept the data from user. **2.** Accessing data **3. Creating , modify and delete .**
INSTRUCTIONS FOR **WRITING JOURNAL**	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites: Concepts on

1)file handling

2)classes

Concepts related Theory :

File Handling :

As with any OS, **file handling** is a core concept in Linux. Any system programmer would learn it as one of his/her initial programming assignments. This aspect of programming involves system **files**. Through **file handling**, one can perform operations like create, modify, delete etc. on system **files .**

Syntax for opening a file :

void open(const char *filename, ios::openmode mode);

Syntax for ofstream outfile :

typedef basic_ofstream<char> ofstream;

Syntax for ifstream infile :

typedef basic_ifstream<char> ifstream;

Class :

A **class** in **C++** is a user **defined** type or data structure declared with keyword **class** that has data and functions (also called methods) as its members whose access is governed by the three access specifiers private, protected or public (by default access to members of a **class** is private).

Syntax for class in C++ :

class class_name

{

//data members

//functions

};

Algorithm:

1. Accept the data from user.

2. Accessing data

3. Creating, modify and delete .

Review Questions: 1. What is a class in c++?

2.Explain the concept of file handling .

3. Give syntax for opening a file.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	16
TITLE	File Operations
PROBLEM STATEMENT /DEFINITION	Write a C++ program that creates an output file, writes information to it, closes the file and open it again as an input file and read the information from the file.
OBJECTIVE	Students should be able to 1. Create file 2. write in file 3. read information from file 4. Open a file 5. Close a file
OUTCOME	When executed this assignment, student would be able to handle file operations.
S/W PACKAGES AND HARDWARE APPARATUS USED	64-bit Open source Linux or its derivative Open Source C++ Programming tool like G++

REFERENCES	Data Structures, Algorithms and Applications in C++, Second Edition, Sartaj Sahni.
STEPS	1: open a file in write mode. 2**:** write inputted data into the file. 3: close the opened file. 4: open a file in read mode. 5: write the data at the screen. 6: read the data from the file and display it. 7: close the opened file.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

C ++ class concepts , types of files, basic knowledge of use of file.

Concepts related Theory :

So far, we have been using the **iostream** standard library, which provides **cin**and **cout** methods for reading from standard input and writing to standard output respectively.

This tutorial will teach you how to read and write from a file. This requires another standard C++ library called **fstream**, which defines three new data types:

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.
fstream	This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

To perform file processing in C++, header files <iostream> and <fstream> must be included in your C++ source file.

Opening a file:

Syntax: void open(const char *filename, ios::openmode mode);

the first argument specifies the name and location of the file to be opened and the second argument of the **open()** member function defines the mode in which the file should be opened.

Mode Flag	Description
ios::app	Append mode. All output to that file to be appended to the end.
ios::ate	Open a file for output and move the read/write control to the end of the file.
ios::in	Open a file for reading.
ios::out	Open a file for writing.
ios::trunc	If the file already exists, its contents will be truncated before opening the file.

if you want to open a file in write mode and want to truncate it in case it already exists, following will be the syntax:

```
ofstream outfile;
outfile.open("file.dat", ios::out | ios::trunc );
```

Similar way, you can open a file for reading and writing purpose as follows:

```
fstream  afile;
afile.open("file.dat", ios::out | ios::in );
```

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. But 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 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.

Algorithm:

Step 1: open a file in write mode.

Step 2**:** write inputted data into the file.

Step 3: close the opened file.

Step 4: open a file in read mode.

Step 5: write the data at the screen.

Step 6: read the data from the file and display it.

Step 7: close the opened file.

Review Questions:

1. **Name the header files required for file handling programming**
2. **In which modes file can be opened?**

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	17
TITLE	Find and replace word in file using command line arguments
PROBLEM STATEMENT /DEFINITION	Write a C++ program using command line arguments to search for a word in a file and replace it with specified word. The usage of program is: $change <old word> <new word> <file name>
OBJECTIVE	7. To understand basic concepts of command line arguments. 8. To perform replacement of old word with new word in a file.
OUTCOME	2. Word replacement in a file.
S/W PACKAGES AND HARDWARE APPARATUS USED	64-bit Open source Linux or its derivative Open Source C++ Programming tool like G++
REFERENCES	Data Structures, Algorithms and Applications in C++, Second Edition, Sartaj Sahni.
STEPS	1. Accept command line arguments. 2. Search old word in a file. 3. If old word found, replace it with new word in a file. 4. Display the contents of modified file.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm

	8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

Data Structures and problem Solving.

Concepts Related Theory:

In C++ it is possible to accept command line arguments. Command-line arguments are given after the name of a program in command-line operating systems like DOS or Linux, and are passed in to the program from the operating system. To use command line arguments in your program, you must first understand the full declaration of the main function, which previously has accepted no arguments. In fact, main can actually accept two arguments: one argument is number of command line arguments, and the other argument is a full list of all of the command line arguments.

The full declaration of main looks like this:

```
int main ( int argc, char *argv[] )
```

The integer, argc is the ARGument Count (hence argc). It is the number of arguments passed into the program from the command line, including the name of the program.

The array of character pointers is the listing of all the arguments. argv[0] is the name of the program, or an empty string if the name is not available. After that, every element number less than argc is a command line argument. You can use each argv element just like a string, or use argv as a two dimensional array. argv[argc] is a null pointer.

How could this be used? Almost any program that wants its parameters to be set when it is executed would use this. One common use is to write a function that takes the name of a file and outputs the entire text of it onto the screen.

```
#include <fstream>
#include <iostream>

using namespace std;

int main ( int argc, char *argv[] )
{
  if ( argc != 2 ) // argc should be 2 for correct execution
    // We print argv[0] assuming it is the program name
    cout<<"usage: "<< argv[0] <<" <filename>\n";
  else {
    // We assume argv[1] is a filename to open
    ifstream the_file ( argv[1] );
    // Always check to see if file opening succeeded
    if ( !the_file.is_open() )
      cout<<"Could not open file\n";
    else {
      char x;
      // the_file.get ( x ) returns false if the end of the file
      //  is reached or an error occurs
      while ( the_file.get ( x ) )
        cout<< x;
    }
    // the_file is closed implicitly here
  }
}
```

This program is fairly simple. It incorporates the full version of main. Then it first checks to ensure the user added the second argument, theoretically a file name. The program then checks to see if the file is valid by trying to open it. This is a standard operation that is effective and easy. If the file is valid, it gets opened in the process.

Steps:

1. Accept command line arguments.
2. Search old word in a file.
3. If old word found, replace it with new word in a file.
4. Display the contents of modified file.

Questions:

11. What do you mean by command line arguments?
12. How to count command line arguments?
13. What type of arguments can be passed from command line?
14. Does command line arguments concept applies to any other languages than C++?
15. Which header files are required to work on files?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	18
TITLE	Selection sort on integer and float array
PROBLEM STATEMENT /DEFINITION	Write a function template selection Sort. Write a program that inputs, sorts and outputs an integer array and a float array.
OBJECTIVE	To understand basic concept of selection sort. To perform sorting on integer and float array using selection sort.
OUTCOME	Sorted number array in ascending order.
S/W PACKAGES AND **HARDWARE APPARATUS USED**	64-bit Open source Linux or its derivative Open Source C++ Programming tool like G++
REFERENCES	Data Structures, Algorithms and Applications in C++, Second Edition, Sartaj Sahni.
STEPS	1. Set MIN to location 0 2. Search the minimum element in the list 3. Swap with value at location MIN 4. Increment MIN to point to next element 5. Repeat until list is sorted
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm

	8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

Data Structures and problem Solving.

Concepts Related Theory:

Selection sort is a simple sorting algorithm. This sorting algorithm is a in-place comparison based algorithm in which the list is divided into two parts, sorted part at left end and unsorted part at right end. Initially sorted part is empty and unsorted part is entire list.

Smallest element is selected from the unsorted array and swapped with the leftmost element and that element becomes part of sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$ where n are no. of items.

How selection sort works?

We take the below depicted array for our 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. We shall see an pictorial depiction of entire sorting process –

Now, we should learn some programming aspects of selection sort.

Algorithm

Step 1 − Set MIN to location 0.

Step 2 − Search the minimum element in the list.

Step 3 − Swap with value at location MIN.

Step 4 − Increment MIN to point to next element.

Step 5 − Repeat until list is sorted.

Pseudocode

```
procedure selection sort
  list  : array of items
  n     : size of list
  for i = 1 to n - 1
  /* set current element as minimum*/
    min = i
     /* check the element to be minimum */
    for j = i+1 to n
      if list[j] < list[min] then
        min = j;
      end if
    end for
    /* swap the minimum element with the current element*/
    if indexMin != i  then
      swap list[min] and list[i]
    end if
  end for
      end procedure
```

Review Questions:

1. How does selection sort works?
2. What kinds of input selection sort can take?
3. Why selection sort is referred to be simplest sorting algorithm in compare with other sorting algorithms?
4. When to use selection sort?

5. Compare selection sort vs bubble sort.
6. In a selection sort of n elements, how many times is the swap function called?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-1

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	19
TITLE	Template Functions
PROBLEM STATEMENT /DEFINITION	Write a function template selection sort. Write a program that inputs , sorts and outputs an int array and float array.
OBJECTIVE	To understand the concepts of Template, its use and apply it in Programming. To revise sorting methods.
OUTCOME	By the end of the experiment Students will be able to understand and implement the concepts of template in sorting techniques.
S/W PACKAGES AND HARDWARE APPARATUS USED	S/W: Open source 64 bit programming tool like GCC/G++ or an IDE like Eclipse, 64 bit Open source Operating system like Linux. H/W: i3 processor of 3.4GHz, 4GB RAM, 500GB Hard disk
REFERENCES	Bjarne Stroustrup – The C++ Programming Language, Third Edition Herbert Schildt – C++ - The Complete Reference, Eighth Edition

STEPS	1. Accept the data to be sorted 2. Sort the accepted data. 3. Display the sorted data.
INSTRUCTIONS FOR WRITING JOURNAL	Date Assignment no. Problem definition Learning objective Learning Outcome Concepts related Theory Algorithm Flow chart Test cases Conclusion/Analysis

Prerequisites:

Knowledge about implementation of objects and classes, file handling mechanisms such as creation of file, insertion of records in file and deletion of records from file.

Concepts related Theory :

Just as the "iostream" standard library is used to provide the "cin" and "cout" methods for reading from standard input and writing to standard output respectively, the "fstream" standard library is used to provide the methods required for opening a file, reading from a file, writing into a file, closing a file, etc.

There are 3 important classes in the "fstream" standard library :

1) ofstream – Used to create files and write information into files.

2) ifstream – Used to read information from files.

3) fstream – Has the capabilities of both, ofstream and ifstream and can create files, write information to files, and read information from files.

The important functions in this library are :

1) **Opening a file** :

A file must be opened before you can read from it or write to it. Either the **ofstream** or **fstream** object may be used to open a file for writing and ifstream object is used to open a file for reading purpose only.

Following is the standard syntax for open() function, which is a member of fstream, ifstream, and ofstream objects.

```
void open(const char *filename, ios::openmode mode);
```

2) **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. But 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();
```

3) **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.

4) **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.

Algorithm:

1. Accept the data to be sorted
2. Sort the accepted data.
3. Display the sorted data.

Review Questions:

1. Write the syntax for opening a file.
2. Write syntax for writing to a file.
3. Write syntax for closing a file.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	20
TITLE	Linked list
PROBLEM STATEMENT /DEFINITION	Write C++ program using STL for implementation of singly, doubly and circular linked list.
OBJECTIVE	To understand concept of OOP. To understand class structures for SLL in C++. To understand primitive operations on SLL. To understand the concept of merge.
OUTCOME	To write class for SLL in C++ To implement primitive operations on SLL in C++. To implement the merging of two lists..
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems (64-Bit)64-BIT Fedora 17 or latest 64-BIT Up date of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bit) Latest Open source update of Eclipse Programming frame work, , TC++
REFERENCES	C++ by B.Stroustrup OOP Using C++ by Balagurusami

STEPS	Refer to algorithm.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites: Basic of object oriented features (object and class) and basics of singly, doubly and circular linked list. Basics of problem solving.

Concepts related Theory :

Singly linked list

Singly linked lists contain nodes which have a data field as well as a 'next' field, which points to the next node in line of nodes. Operations that can be performed on singly linked lists include insertion, deletion and traversal.

Doubly linked list

In a 'doubly linked list', each node contains, besides the next-node link, a second link field pointing to the 'previous' node in the sequence. The two links may be called 'forward('s') and 'backwards', or 'next' and 'prev'('previous').

Circular linked list

In the last node of a list, the link field often contains a null reference, a special value used to indicate the lack of further nodes. A less common convention is to make it point to the first node of the list; in that case the list is said to be 'circular' or 'circularly linked'; otherwise it is said to be 'open' or 'linear'.

Algorithm: 1.Start

2. Create nodes.

3. Display nodes.

4. Search node.

5. Insert head.

6. Insert after given node.

7. insert last node.

8. Exit.

Review Questions:

1. what is linked list?
2. How singly, doubly and circular linked lists are different from each other.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	21
TITLE	Stack and queue using SLL
PROBLEM STATEMENT /DEFINITION	Write C++ program using STL for implementation of stack and queue using SLL.
OBJECTIVE	To learn stack and queue implementation.
OUTCOME	Implementation of stack and queue using SLL.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems (64-Bit)64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bit),TC++
REFERENCES	C++ by B. Stroustrup Fundamental of Data Structure in C++.
STEPS	Refer to algorithm
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective

	5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

- Basic of stack and queue.
- Basics of problem solving.
- Basics of SLL.

Concepts related Theory :

Stack

A stack is an abstract data type (ADT), commonly used in most programming languages. It is named stack as it behaves like a real-world stack, for example − deck of cards or pile of plates etc.

A real-world stack allows operations at one end only. For example, we can place or remove a card or plate from top of the stack only. Likewise, Stack ADT allows all data operations at one end only. At any given time, We can only access the top element of a stack.

This feature makes it LIFO data structure. LIFO stands for Last-in-first-out. Here, the element which is placed (inserted or added) last, is accessed first. In stack terminology, insertion operation is called **PUSH** operation and removal operation is called **POP** operation.

Queue

Queue is an abstract data structure, somewhat similar to Stack. In contrast to Queue, queue is opened at both end. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

A real world example of queue can be a single-lane one-way road, where the vehicle enters first, exits first. More real-world example can be seen as queues at ticket windows & bus-stops.

Concepts Related Theory:

A. STACK

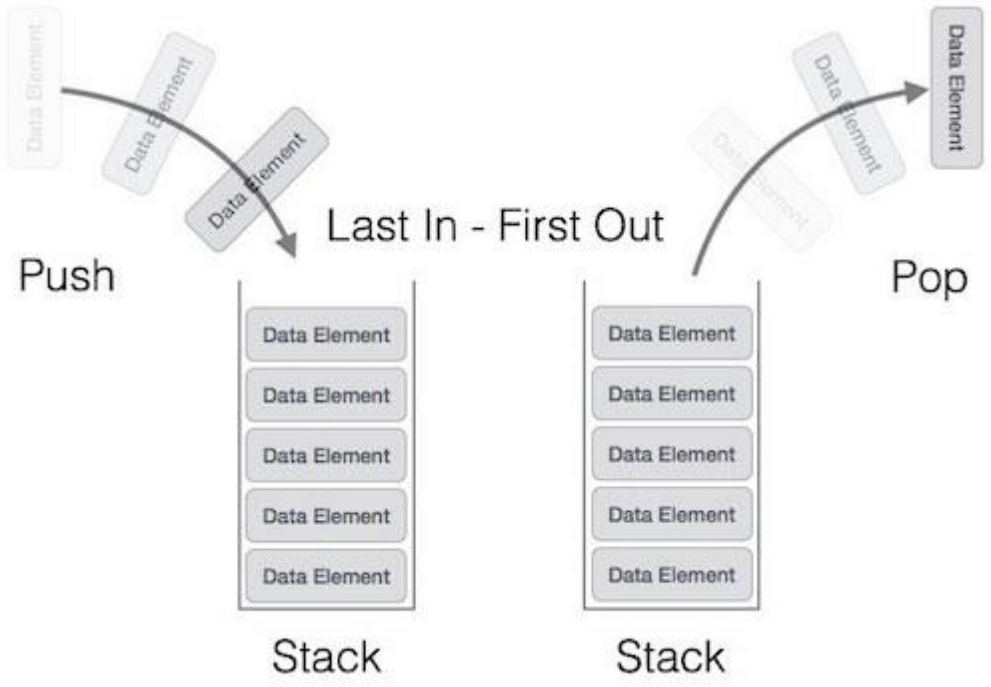
A stack is an abstract data type (ADT), commonly used in most programming languages. It is named stack as it behaves like a real-world stack, for example − deck of cards or pile of plates etc.

A real-world stack allows operations at one end only. For example, we can place or remove a card or plate from top of the stack only. Likewise, Stack ADT allows all data operations at one end only. At any given time, We can only access the top element of a stack.

This feature makes it LIFO data structure. LIFO stands for Last-in-first-out. Here, the element which is placed (inserted or added) last, is accessed first. In stack terminology, insertion operation is called **PUSH** operation and removal operation is called **POP** operation.

Stack Representation

Below given diagram tries to depict a stack and its operations –



A stack can be implemented by means of Array, Structure, Pointer and Linked-List. Stack can either be a fixed size one or it may have a sense of dynamic resizing. Here, we are going to implement stack using arrays which makes it a fixed size stack implementation.

end procedure **Basic Operation**

Stack operations may involve initializing the stack, using it and then de-initializing it. Apart from these basic stuffs, a stack is used for the following two primary operations −

- **push()** − pushing (storing) an element on the stack.
- **pop()** − removing (accessing) an element from the stack.

When data is PUSHed onto stack.

To use a stack efficiently we need to check status of stack as well. For the same purpose, the following functionality is added to stacks −

- **peek()** − get the top data element of the stack, without removing it.

- **isFull()** − check if stack is full.
- **isEmpty()** − check if stack is empty.

At all times, we maintain a pointer to the last PUSHed data on the stack. As this pointer always represents the top of the stack, hence named **top**. The **top** pointer provides top value of the stack without actually removing it.

First we should learn about procedures to support stack functions –

peek() –

```
begin procedure peek
   return stack[top]
```

isfull() –

```
begin procedure isfull
   if top equals to MAXSIZE
      return true
   else
      return false
   endif
   end procedure
```

isempty() –

```
begin procedure isempty
   if top less than 1
      return true
   else
      return false
   endif
```

```
   end procedure
```

push() –

```
begin procedure push: stack, data
   if stack is full
      return null
   endif
      top ← top + 1
      stack[top] ← data
end procedure
```

pop() –

```
begin procedure pop: stack
   if stack is empty
      return null
   endif
      data ← stack[top]
      top ← top - 1
      return data
end procedure
```

B. QUEUE

Queue is an abstract data structure, somewhat similar to Queue. In contrast to Queue, queue is opened at both end. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

A real world example of queue can be a single-lane one-way road, where the vehicle enters first, exits first. More real-world example can be seen as queues at ticket windows & bus-stops.

Queue Representation

As we now understand that in queue, we access both ends for different reasons, a diagram given below tries to explain queue representation as data structure –

Same as Queue, queue can also be implemented using Array, Linked-list, Pointer and Structures. For the sake of simplicity we shall implement queue using one-dimensional array.

Basic Operations

Queue operations may involve initializing or defining the queue, utilizing it and then completing erasing it from memory. Here we shall try to understand basic operations associated with queues −

- **enqueue()** − add (store) an item to the queue.
- **dequeue()** − remove (access) an item from the queue.

Few more functions are required to make above mentioned queue operation efficient. These are –

- **peek()** − get the element at front of the queue without removing it.
- **isfull()** − checks if queue is full.
- **isempty()** − checks if queue is empty.

In queue, we always **dequeue** (or access) data, pointed by**front** pointer and while enqueing (or storing) data in queue we take help of **rear** pointer.

Let's first learn about supportive functions of a queue −

peek() –

```
begin procedure peek

   return queue[front]

end procedure
```

isfull() –

```
begin procedure isfull
   if rear equals to MAXSIZE
      return true
   else
      return false
   endif
   end procedure
```

isempty() –

```
begin procedure isempty
   if front is less than MIN  OR front is greater than rear
      return true
   else
      return false
   endif
   end procedure
```

enqueue() –

```
procedure enqueue(data)
   if queue is full
      return overflow
   endif
      rear ← rear + 1
      queue[rear] ← data
      return true
   end procedure
```

dequeue() –

```
procedure dequeue
   if queue is empty
      return underflow
   end if
   data = queue[front]
   front ← front + 1
    return true
end procedure
```

C. SLL (SINGLY LINKED LIST)

A linked-list is a sequence of data structures which are connected together via links.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list the second most used data structure after array. Following are important terms to understand the concepts of Linked List.

- **Link** − Each Link of a linked list can store a data called an element.
- **Next** − Each Link of a linked list contain a link to next link called Next.
- **LinkedList** − A LinkedList contains the connection link to the first Link called First.

Linked List Representation

Linked list can be visualized as a chain of nodes, where every node points to the next node.

As per above shown illustration, following are the important points to be considered.

- LinkedList contains an link element called first.
- Each Link carries a data field(s) and a Link Field called next.
- Each Link is linked with its next link using its next link.
- Last Link carries a Link as null to mark the end of the list.

Types of Linked List

Following are the various flavours of linked list.

- **Simple Linked List** − Item Navigation is forward only.
- **Doubly Linked List** − Items can be navigated forward and backward way.
- **Circular Linked List** − Last item contains link of the first element as next and and first element has link to last element as prev.

Basic Operation

Following are the basic operations supported by a list.

- **Insertion** − add an element at the beginning of the list.
- **Deletion** − delete an element at the beginning of the list.
- **Display** − displaying complete list.
- **Search** − search an element using given key.
- **Delete** − delete an element using given key.

Algorithm:

Step 1: Start.

Step 2: Insert data(push operation).

Step 3: perform pop operation.

Step 4: Check if stack is full.

Step 5: check if stack is empty.

Step 6: Exit.

Review Questions:

1. What is the difference between a stack and a queue?

2. What do you mean by stack in data structures?
3. What do you mean by queue in data structures?
4. What is stack in computer programming?
5. How many pointers a singly list can have?

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	22
TITLE	Binary number addition.
PROBLEM STATEMENT /DEFINITION	Write C++ program using STL to add binary numbers (assume one bit as one number); use STL stack.
OBJECTIVE	To understand concept of adding two binary number.
OUTCOME	To implement C++ program for adding two binary numbers using STL.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems (64-Bit)64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bit) , TC++
REFERENCES	C++ by B. Stroustrup Fundamental of Data Structure in C++.
STEPS	Refer to algorithm.
INSTRUCTIONS FOR	1. Date 2. Assignment no.

WRITING JOURNAL	3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

- Basic of object oriented features.
- Basics of problem solving.
- Binary addition.
- Basics of stack.

Concepts related Theory :

The simplest arithmetic operation in binary is addition. Adding two single-digit binary numbers is relatively simple, using a form of carrying:

$0 + 0 \rightarrow 0$

$0 + 1 \rightarrow 1$

$1 + 0 \rightarrow 1$

$1 + 1 \rightarrow 0$, carry 1 (since $1 + 1 = 2 = 0 + (1 \times 2^1)$)

Adding two "1" digits produces a digit "0", while 1 will have to be added to the next column. This is similar to what happens in decimal when certain single-digit numbers are added together; if the result equals or exceeds the value of the radix (10), the digit to the left is incremented:

$5 + 5 \rightarrow 0$, carry 1 (since $5 + 5 = 10 = 0 + (1 \times 10^1)$)

$7 + 9 \rightarrow 6$, carry 1 (since $7 + 9 = 16 = 6 + (1 \times 10^1)$)

This is known as *carrying*. When the result of an addition exceeds the value of a digit, the procedure is to "carry" the excess amount divided by the radix (that is, 10/10) to the left, adding it to the next positional value. This is correct since the next position has a weight that is higher by a factor equal to the radix. Carrying works the same way in binary

Algorithm:

Step 1: Start.

Start 2: Accept numbers.

Step 3: Perform addition

sum=(bit1+bit2+carry)%2;

carry=(bit1+bit2+carry)/2;

Step 4: Display the result.

Step 5: Exit.

Review Questions:

1. **Explain stack data structure.**
2. **Explain addition binary two numbers.**

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	23
TITLE	Dqueue
PROBLEM STATEMENT /DEFINITION	Write C++ program using STL for Dqueue (Double ended queue).
OBJECTIVE	To understand concept of OOP. To understand class structures for DLL in C++. To understand primitive operations on DLL circularly.
OUTCOME	To write class for DLL in C++ To implement primitive operations on DCLL with head node in C++.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems (64-Bit)64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bit) , TC++
REFERENCES	C++ by B. Stroustrup Fundamental of Data Structure in C++.

STEPS	Refer to algorithm.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

- Basic of object oriented features.
- Basics of problem solving.
- Basics of Dqueue.

Concepts related Theory :

The C++ STL (Standard Template Library) is a powerful set of C++ template classes to provides general-purpose templatized classes and functions that implement many popular and commonly used algorithms and data structures like vectors, lists, queues, and stacks.

At the core of the C++ Standard Template Library are following three well-structured components:

Component	Description
Containers	Containers are used to manage collections of objects of a certain

	kind. There are several different types of containers like deque, list, vector, map etc.
Algorithms	Algorithms act on containers. They provide the means by which you will perform initialization, sorting, searching, and transforming of the contents of containers.
Iterators	Iterators are used to step through the elements of collections of objects. These collections may be containers or subsets of containers.

All the three components have a rich set of pre-defined functions which help us in doing complicated tasks in very easy fashion. The STL provides a ready-made set of common classes for C++, such as containers and associative arrays, that can be used with any built-in type and with any user-defined type that supports some elementary operations (such as copying and assignment). STL algorithms are independent of containers, which significantly reduces the complexity of the library.

The STL achieves its results through the use of templates. This approach provides compile-time polymorphism that is often more efficient than traditional run-time polymorphism. Modern C++ compilers are tuned to minimize abstraction penalty arising from heavy use of the STL.

The STL was created as the first library of generic algorithms and data structures for C++, with four ideas in mind: generic programming, abstractness without loss of efficiency, the Von Neumann computation model, and value semantics.

Deque

Deques are one of the many standard template library (STL) containers available in C++. Their name suggests a good deal about their intended usage and what is possible with them. They are containers, meaning that they are designed to hold multiple data elements. They are templates, meaning that the constructors and methods have all been templatized, allowing for the use of any of the standard C++ data types, as well as user defined classes with the proper constructors defined. And the name of the template itself stands for double ended queue, implying behavior similar to a queue, except with fast access from both ends of the container. This will attempt to address all of the methods available for the deque container. Since many of the methods are also available for other STL containers, learning the syntax and usage of deques should provide a good start for figuring out some of the other elements of the STL.

Algorithm:

Step 1: Start

Step 2: Enter element at front

Step 3: Enter element at Rear

Step 4: Delete element at front

Step 5: Delete element at Rear.

Step 6: Display front element.

Step 7: Display back element.

Step 8: Display the element of dequeue.

Step 9: Exit

Conclusion: Implementation of Dqueue operation using stack.

Review Questions:

1. Explain add and delete operation on dqueue.

2. How dqueue is different from queue.

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2016-17

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: S.E. **SEMESTER: I**

SUBJECT: Object Oriented Programming Lab

ASSINGMENT NO.	24
TITLE	Sorting and searching with user defined records.
PROBLEM STATEMENT /DEFINITION	Write C++ program using STL for sorting and searching with user defined records such as person record (Name, birth, date, telephone no), item record (item code, item name, quantity and cost).
OBJECTIVE	To learn sorting and searching with user defined records.
OUTCOME	Implementation of C++ program for sorting and searching with user defined record
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems (64-Bit)64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version and update of Microsoft Windows 7 Operating System onwards Programming Tools (64-Bi), TC++
REFERENCES	C++ by B. Stroustrup Fundamental of Data Structure in C++.
STEPS	Refer to algorithm.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition

	4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Flow chart 9. Test cases 10. Conclusion/Analysis

Prerequisites:

- Basic of object oriented features.
- Basics of problem solving.
- Basics of searching and sorting.

Concepts related Theory :

Sorting

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are numerical or lexicographical order.

Importance of sorting lies in the fact that data searching can be optimized to a very high level if data is stored in a sorted manner. Sorting is also used to represent data in more readable formats. Some of the examples of sorting in real life scenarios are following.

- **Telephone Directory** − Telephone directory keeps telephone no. of people sorted on their names. So that names can be searched.
- **Dictionary** − Dictionary keeps words in alphabetical order so that searching of any work becomes easy.

In-place sorting and Not-in-place

Sorting algorithms may require some extra space for comparison and temporary storage of few data elements. These algorithms do not require any extra space and sorting is said to be happened in-place, or for example, within the array itself. This is called **in-place sorting**. Bubble sort is an example of in-place sorting.

But in some sorting algorithms, the program requires space which more than or equal to the elements being sorted. Sorting which uses equal or more space are called **not-in-place sorting**. Merge-sort is an example of not-in-place sorting.

Stable and Not Stable Sorting

If a sorting algorithm, after sorting the contents, does not change the sequence of similar content in which they appear, it is called **stable sorting**.

If a sorting algorithm, after sorting the contents, changes the sequence of similar content in which they appear, it is called **unstable sorting**.

stability of an algorithm matters when we wish to maintain the sequence of original elements, like in a tuple for example.

Adaptive and Non-Adaptive Sorting Algorithm

A sorting algorithm is said to be adaptive, if it takes advantage of already 'sorted' elements in the list that is to be sorted. That is, while sorting if the source list has some element already sorted, adaptive algorithms will take this in to account and will try not to re-order them.

A non-adaptive algorithm is one which does not take into account the elements which are already sorted. They try to force every single element to be re-order to confirm their sortedness.

Important terms

Some terms are generally coined while discussing sorting techniques, here is a brief introduction to them −

Increasing Order

A sequence of values is said to be in **increasing order**, if the successive element is greater than the previous one. For example, 1, 3, 4, 6, 8, 9 are in increasing order, as every next element is greater than the previous.

Decreasing Order

A sequence of values is said to be in **decreasing order**, if the successive element is less than the current one. For example, 9, 8, 6, 4, 3, 1 are in decreasing order, as every next element is less than the previous.

Non-Increasing Order

A sequence of values is said to be in **non-increasing order**, if the successive element is less than or equal to its previous element in the sequence. This order occurs when the sequence contains

duplicate values. For example, 9, 8, 6, 3, 3, 1 are in non-increasing order, as every next element is less than or equal to (in case of 3) but not greater than the any previous element.

Non-Decreasing Order

A sequence of values is said to be in **non-decreasing order**, if the successive element is greater than or equal to its previous element in the sequence. This order occurs when the sequence contains duplicate values. For example, 1, 3, 3, 6, 8, 9 are in non-decreasing order, as every next element is greater than or equal to (in case of 3) but not less than the previous one.

Searching

linear search

Linear search is a very simple search algorithm. In this type of search, a sequential search is made over all items one by one. Every items is checked and if a match founds then that particular item is returned otherwise search continues till the end of the data collection.

Binary search

Binary search is a fast search algorithm with run-time complexity of O(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly the data collection should be in sorted form.

Binary search search a particular item by comparing the middle most item of the collection. If match occurs then index of item is returned. If middle item is greater than item then item is searched in sub-array to the right of the middle item other wise item is search in sub-array to the left of the middle item. This process continues on sub-array as well until the size of subarray reduces to zero.

Algorithm:

Step 1: Start.

Step 2: Insert the record.

Step 3: Display the record.

Step 4: Search the record by their code.

Step 5: Sort the record.

Step 6: Delete the record selected by user.

Step 7: Exit.

Conclusion: Sorting and searching with user defined records is implemented in C++ using STL.

Review Questions: 1. Give time complexity for searching.

2. Give time complexity for sorting.