Assignment No: 1

Title: Arithmetic operations on complex numbers using operator overloading.

Problem Statement:

Implement a class Complex which represents the Complex Number data type.

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.

Objectives:

To learn the concept of constructor, default constructor, operator overloading using member function and friend function.

Theory:

Operator Overloading

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

An Example of Operator Overloading

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

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

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

Arithmetic Operators

Arithmetic Operators are used to do basic arithmetic operations like addition, subtraction, multiplication, division, and modulus.

With C++ feature to overload operators, we can design classes able to perform operations using standard operators. Here is a list of all the operators that can be overloaded:

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

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

Syntax:

return_typeclass_name :: operator op(arg_list)

{

//function body

}

where,

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

Process of the overloading has 3 steps

1. Create a class that define a data types that is used in the overloading operation
2. Declare the operator function operator op() in the public part of the class.

It may be either a member function or a friend function.

3. Define the operator function to implement the required operation

e.g.

Overloading Binary operators:

A statement like

```
C = sum (A, B);                    // functional notation
```

This functional notation can be replaced by a natural looking expression

```
C = A+B;                              // arithmetic notation
```

By overloading the + operator using an operator+ () function.

Algorithm:

Step 1: Start the program

Step 2: Create a class complex

Step 3: Define the default constructor.

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

For Addition:

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

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

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

Step 7:Create a menu for addition, multiplication of complex numbersand display the result

Step 8: Depending upon the choice from the user the arithmetic operators will invoke the overloaded operator automatically and returns the result.

Step 9: Display the result using display function

Output:

```
Default constructor value=0+0i

Enter the 1st number
Enter the real part2
Enter the imaginary part4

Enter the 2nd number
Enter the real part4
Enter the imaginary part8

The first number is 2+4i
The second number is 4+8i

The addition is 6+12i


The multiplication is -24+32i
```

Conclusion:

Hence, we have studied concept of operator overloading.

Questions:

1. What is operator overloading?
2. What are the rules for overloading the operators?
3. State clearly which operators are overloaded and which operator are not overloaded?
4. State the need for overloading the operators.
5. Explain how the operators are overloaded using the friend function.
6. What is the difference between “overloading” and “overriding”?

7. What is operator function? Describe the syntax?
8. When is Friend function compulsory? Give an example?

Assignment No:2

Title:- Student Database Management system.

Problem statement:-

Develop an object oriented program in C++ to create a database of student information system containing the following information: Name, Roll number, Class, division, Date of Birth, Blood group, Contact address, telephone number, driving licence no. etc Construct the database with suitable member functions for initializing and destroying the data viz constructor, default constructor, Copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

Objectives:

To learn the concept of constructor, default constructor, copy, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

Theory:

Constructor:

A special method of the class that will be automatically invoked when an instance of the class is created is called as constructor. Following are the most useful features of constructor. 1) Constructor is used for Initializing the values to the data members of the Class.

2) Constructor is that whose name is same as name of class.

3) Constructor gets Automatically called when an object of class is created.

4) Constructors never have a Return Type even void.

5) Constructor is of Default, Parameterized and Copy Constructors.

The various types of Constructor are as follows: -

Constructors can be classified into 3 types

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

1. **Default Constructor: -**

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

2. Pa**rameterized Constructor: -**

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

3. **Copy Constructor: -**

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

Destructor:

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

Static members:-

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

• It is initialized to zero when the first object of its class is created. No other initialization is permitted.

- Only one copy of that member is created for the entire class and is shared by all the objects of that class.
- It is the visible only within the class but its lifetime is the entire program.

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 (of course this is only valid for static members.

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.

this pointer:

C++ uses a unique keyword called this to represent an object that invokes a member function. thisis 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.

• 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. Start
2. Read personnel information such as Name, Date of Birth, Blood group, Height, Weight, Insurance Policy, number, Contact address, telephone number, driving license no..
3. Print all information from database.
4. Stop

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

Output:-

```
Default values:
    Sachin    0   I  A 11/11/1111A   A       city 9000000000A0101010           A0101010

Sachin(Object) is destroyed!

Enter:name,roll,Class,Div,Dob,bg,contact,phone,license
 abc 1 II Linux 12/12/2000 B+ Pune 98634555 B123345

abc         1         II            Linux         12/12/2000        B+          Pune          98634555        B123345

abc(Object) is destroyed!

Use of copy constructor :

 abc         1         II            Linux         12/12/2000        B+          Pune          98634555        B12334
```

```
abc(Object) is destroyed!
How many objects u want to create?:1

Enter:name,roll,Class,Div,Dob,bg,contact,phone,license

abc 1 II Linux 12/12/2000 B+ Pune 98634555 B123345

abc      1      II       Linux      12/12/2000      B+      Pune      98634555      B123345
```

Conclusion:

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

Questions:

1. What is concept of constructor, destructor?
2. What are types of constructors?
3. What is static Function?
4. What is friend function? State the advantage of using the friend function
5. What is this pointer? Explain with examples.

Assignment No:3

Title: Creating a class which uses the concept of inheritance, displays data and data members and uses the concept of exception handling.

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

Objectives: To learn the concept of inheritance and exception handling.

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. New class that inherits properties of the base class is called a derived class. Inheritance is a technique of code reuse. It also provides possibility to extend existing classes by creating derived classes.
The basic syntax of inheritance is:

Class DerivedClass : accessSpecifier BaseClass

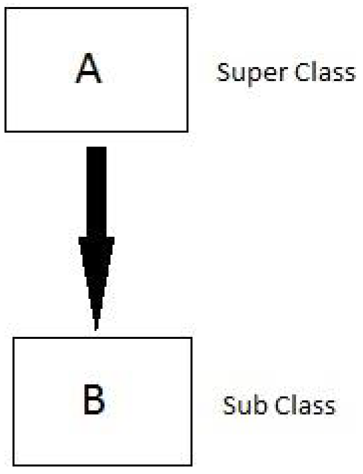
There are 3 access specifiers:
Namely public, private and protected.
public:
This inheritance mode is used mostly. In this the protected member of Base class becomes protected members of Derived class and public becomes public.
protected:

In protected mode, the public and protected members of Base class becomes protected members of Derived class.
private:
In private mode the public and protected members of Base class become private members of Derived class.

Types of Inheritance

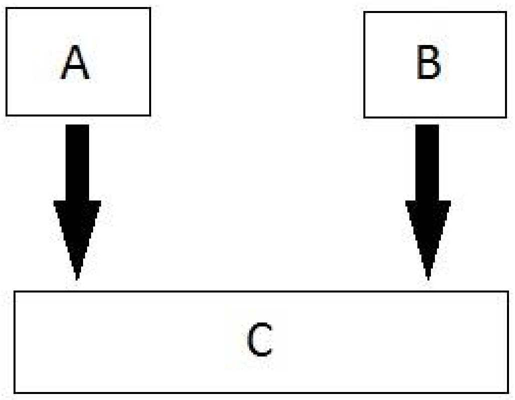
In C++, we have 5 different types of Inheritance. Namely,
1. Single Inheritance
2. Multiple Inheritance
3. Hierarchical Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance

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



Syntax:
class subclass_name : access_modebase_class
{
//body of subclass
};

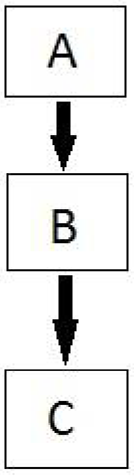
Multiple Inheritance:
In this type of inheritance a single derived class may inherit from two or more than two base classes .



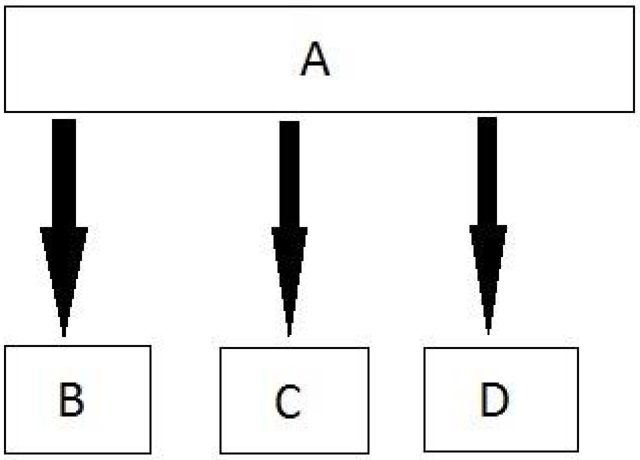
Syntax:

```
classsubclass_name : access_mode base_class1, access_mode base_class2, ....
{
//body of subclass
};
```

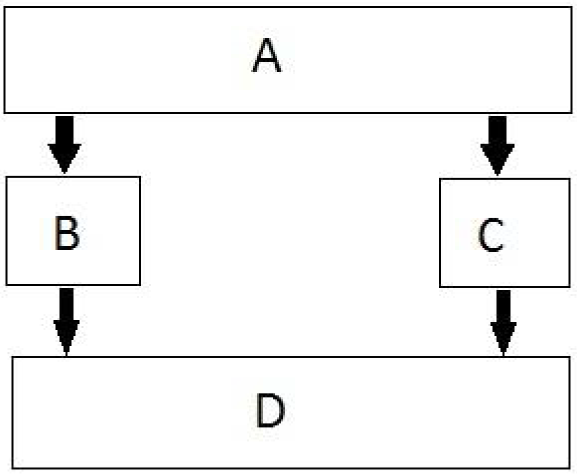
Multilevel Inheritance:
In this type of inheritance the derived class inherits from a class, which in turn inherits from some other class. The Super class for one, is sub class for the other.



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



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



Exception Handling:
Exception handling is part of C++ and object oriented programming. they are added in C++ to handle the unwanted situations during program execution. If we do not type the program correctly then ot might result in errors. Main purpose of exception handling is to identify and report the runtime error in the program.
Famous examples are divide by zero, array index out of bound error, file not found, device not found, etc.
C++ exception handling is possible with three keywords iz. try, catch and throw. Exception handling performs the following tasks:-

- •Find the problem in the given code. It is also called as hit exception.

- We receive the roe info. It is called as catching the exception.
- •It takes the corrective action.It is called as exception handling.

TRY:- It is block code in which there are chances of runtime error.This block is followed by one or more catch block.Most error prone code is added in try block.
CATCH:- This is used to catch th exception thrown by the try blok. In catch block we take corrective action on throwing exception. If files are openend , we can take corrective action like closing file handles,closing database connetions,saving unsaved work ,etc.
THROW:- Program throws exception when problem occurs.It is possible with throw keyword.

Syntax:-

```
//normal program code

ry{
throw exception
}
catch(argument)
{
...
...
}
//rest of the code
/
```

Algorithm:

```
1. Start.
2. Create classes Publication, book and tape.
3 .Publication class having data members title, price.
4.Class Book having data members pages and member functions getdata() and pudata().
5. Class Tape having data members minutes and member functions getdata() and pudata().
6. Create an object bof class book and object t of class tape.
7. Stop.
```

Input:- Title, price, no.of pages for book and playing time for tape.

Output:-

run your own program and paste output here.

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

1. What is Inheritance?
2. What are types of Inheritance?
3. What is Exception handling?
4. What are try catch block of exception handling?

Assignment No: 4

Title: File handing

Problem Statement : 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.

Objectives: To learn concept of file handling.

Theory:

Stream:

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

1. InputStream

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

2. OutputStream

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

file handling provides three new datatypes:

Data Type	**Description**
ofstream	This data type represent output file stream and used to create files and to write information to files.
ifstream	This data type represent input file stream and used to read information from files.
fstream	This data type represent file stream, and have both the functinality of ofstream and ifstream . So it can create file,write into file,read from file.

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

Opening a File

1. A file must be opened before you can read from it or write to it.

2. 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.

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

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

4. Here, 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.

in	File open for reading: the internal stream buffer supports input operations.
out	File open for writing: the internal stream buffer supports output operations.
binary	Operations are performed in binary mode rather than text.
ate	The output position starts at the end of the file.
app	All output operations happen at the end of the file, appending to its existing contents.
trunc	Any contents that existed in the file before it is open are discarded.

• You can combine two or more of these values by ORing them together.
• For example, 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 f.iles .

It is always a good practice that a programmer should close all the opened files before programtermination.
• 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.

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

Reading from a File

You read information from a file into your program using the stream extraction operator • The only difference is that you use an ifstream or fstream object instead of the cinobject.

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

• These function 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) .

Algorithm:

1. Start

2. Create aclass

3. Define data members roll number andname.

4. Define accept() to take name and roll number fromuser.

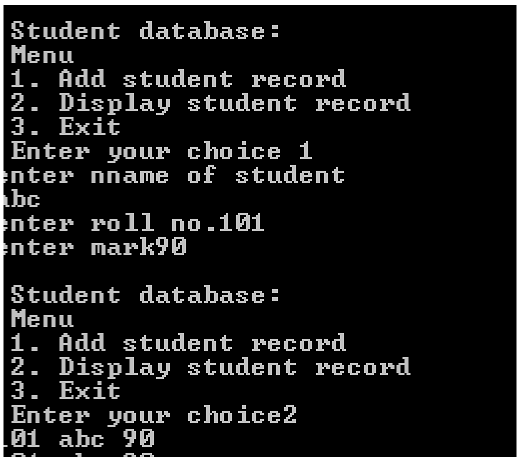
5. Define display() to display therecord.

6. In main() create the object of class and fstream class.

7.Take a limit from user in nvariable.

8. Open the file in out mode , call accept() to take record from user,then call write() to write that record into the file and at the end close that file.
9. Open the file in in mode, read the record from the file ,call display() function to display the record and at the end close that file.
10.Stop.

Input & Output:



```
Student database:
Menu
1. Add student record
2. Display student record
3. Exit
Enter your choice 1
nter nname of student
bc
nter roll no.101
nter mark90

Student database:
Menu
1. Add student record
2. Display student record
3. Exit
Enter your choice2
01 abc 90
```

Conclusion:
Hence, we have studied concept of File handing

Questions:

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

Assignment No: 5

Title:- Function Template

Problem Statement:- Implement a function template selection Sort. Write a program that inputs, sorts and outputs an integer array and a float array.

Objectives :- To learn the concept of template

Theory:

Templates :-

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept. There is a single definition of each container, such as vector, but we can define many different kinds of vectors for example, vector <int> or vector <string>.

You can use templates to define functions as well as classes, let us see how do they work

Function Template:

The general form of a template function definition is shown here:

```
template <class type> ret-type func-name(parameter list)
{
// body of function
}
```

Here, type is a placeholder name for a data type used by the function. This name can be used within the function definition.

Class Template:

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

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

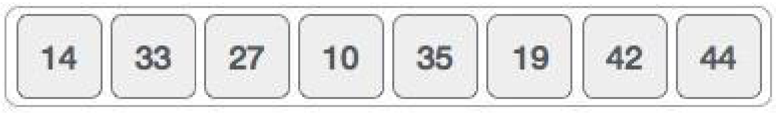
Here, type is the placeholder type name, which will be specified when a class is instantiated. You can define more than one generic data type by using a comma-separated list.

Selection Sort:

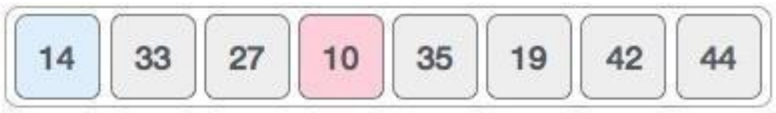
Selection sort is a sorting algorithm, specifically an in-place comparison sort. It has O(n2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated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

How selection sort works?

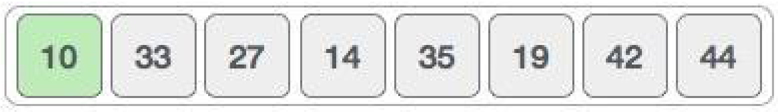
Example



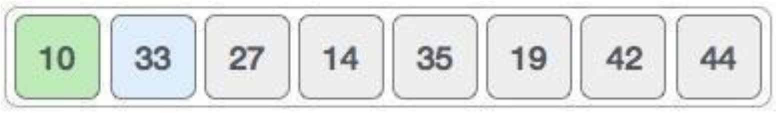
For the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value .



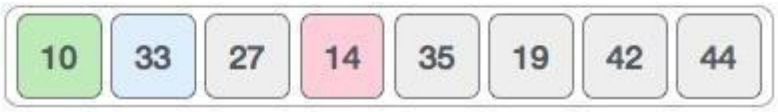
So we replace 14 with 10. After one iteration 10, which happens to be the minimum value in the list, appears in the first position of sorted list.



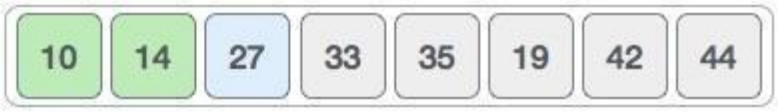
For the second position, where 33 is residing, we start scanning the rest of the list in linear manner.

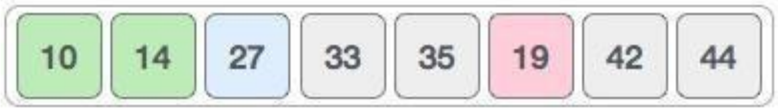


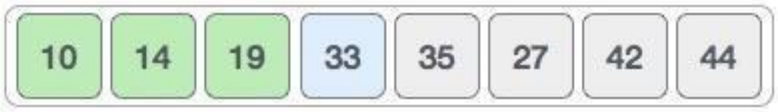
We find that 14 is the second lowest value in the list and it should appear at the second place. We swap these values.

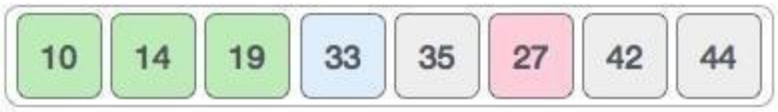


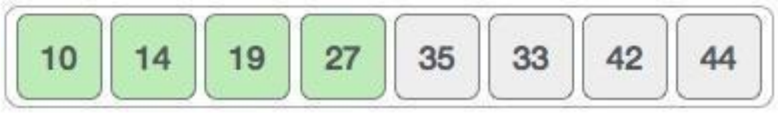
After two iterations, two least values are positioned at the beginning in the sorted manner.
The same process is applied on the rest of the items in the array.
Pictorial depiction of entire sorting process is as follows −

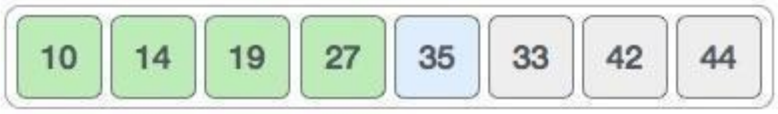


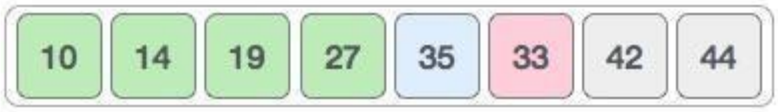


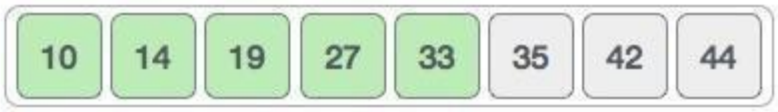


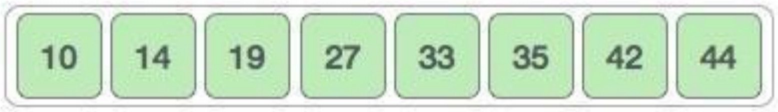












Algorithm:

1. Start

2. Declare the template parameter T.

3. Define template function for selection sort.

4. In main() Define two arrays, one for integer and another for float. and take a input for both the arrays and call sorting function template to sort the number.

5. Stop

Input:-

```
Enter Choice  1

Enter no. of elements in an array 5

Enter elements in an array
34
78
90
56
12
```

Output:-

```
Selection Sort
----------- Menus--------------------
1. Sorting of integer array
2. Sorting of float array
3. Exit
Array before sorting
34
78
```

```
90

56

12

Array after sorting

12

34

56
78
90
```

Conclusion:

Hence, we have studied concept of Function Template.

Questions:

1. What is template?
2. What is Function template?
3. What is Class template?
4. Explain template with function overloading.
5. Explain template with non-type argument

Assignment No:6

Title:- Personnel information system using sorting and searching for STL and vector container.

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

Objectives:

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

Theory:

STL:

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

STL has four components

- Algorithms
- Containers
- Functions
- Iterators

Algorithms

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

• Useful Array algorithms

• Partition Operations

• Numeric

Containers

• Containers or container classes store objects and data. There are in total seven standard “first-class” container classes and three container adaptor classes and only seven header files that

provide access to these containers or container adaptors.

• Sequence Containers: implement data structures which can be accessed in a sequential manner.

• vector

• list

• deque

• arrays

• forward_list(Introduced in C++11)

• Container Adaptors : provide a different interface for sequential containers.

• queue

• priority_queue

• stack

• **Associative Containers :** implement sorted data structures that can be quickly searched (O(log n) complexity)

. • set

• multiset

• map

• multimap

Unordered Associative Containers : implement unordered data structures that can be quickly searched • unordered_set

• unordered_multiset

• unordered_map

• unordered_multimap

Functions

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

Iterators

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

Sorting:

It is one of the most basic functions applied to data. It means arranging the data in a particular fashion, which can be increasing or decreasing. There is a builtin function in C++ STL by the name of sort(). This function internally uses IntroSort. In more details it is implemented using hybrid of QuickSort .

HeapSort and InsertionSort.By default, it uses QuickSort but if QuickSort is doing unfair partitioning and taking more than N*logN time, it switches to HeapSort and when the array size becomes really small,

The prototype for sort is :

```
sort(startaddress, endaddress)
startaddress: the address of the first element of the array
endaddress: the address of the next contiguous location of the last element of the
array.
So actually sort() sorts in the range of [startaddress,endaddress)
```

Searching:

It is a widely used searching algorithm that requires the array to be sorted before search is applied. The main idea behind this algorithm is to keep dividing the array in half (divide and conquer) until the element is found, or all the elements are exhausted.

It works by comparing the middle item of the array with our target, if it matches, it returns true otherwise if the middle term is greater than the target, the search is performed in the left sub-array. If the middle term is less than target, the search is performed in the right sub-array.

Algorithm:

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

Input:

Personnel information such as name, DOB, telephone number.

Output:

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

```
***** Menu *****
1.Insert
2.Display
3.Search
4.Sort
5.Delete
6.Exit
Enter your choice:1
Enter Item Name: bat
Enter Item Quantity:2
Enter Item Cost:50
Enter Item Code:1
```

Conclusion:

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

Questions:

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

Assignment No:7

Title:- Use of map associative container

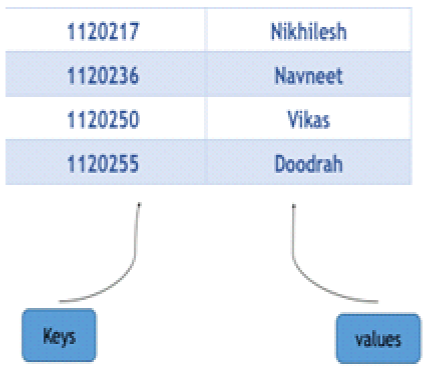
Problem Statement:- Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an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

Objectives:
To learn the concept of map associative container

Theory:
Map associative container:
Map associative container are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values.

Maps contain sorted key-value pair, in which each key is unique and cannot be changed, and it can be inserted or deleted but cannot be altered.

For Example:-



A map of students where:

roll number is the key and name is the value.

Keys are always arranged in sorted order

In case the keys are of string type, they are sorted lexicographically.

Creating Map:-

- **Syntax**

map<key_type , value_type> map_name;

Map<int,string>m1;

  - This will create a map with key of type Key_type and value of type value_type.
  - key of a map and corresponding values are always inserted as a pair, you cannot insert only key or just a value in a map.

- create a map m with keys 1,2,3 and their corresponding values 2,3,4

```
map<int,int> m { {1,2} , {2,3} , {3,4} };
```

- create a map with keys of type character and values of type integer

```
map<string,int> map1;
map1["abc"]=100;    // inserts key = "abc" with value = 100
map1["b"]=200;      // inserts key = "b" with value = 200
map1["c"]=300;      // inserts key = "c" with value = 300
map1["def"]=400;    // inserts key = "def" with value = 400
```

Member Functions of Map in C++ STL

1. at and [ ]

Both at and [ ] are used for accessing the elements in the map.

The only difference between them is that at throws an exception if the accessed key is not present in the map, on the other hand operator [ ] inserts the key in the map if the key is not present already in the map.

2. empty, size and max_size

a) empty() returns boolean true if the map is empty, else it returns Boolean false.

b) size() returns number of entries in the map, an entry consist of a key and a value.

c) max_size() returns the upper bound of the entries that a map can contain (maximum possible entries) based on the memory allocated to the map.

3)Insert

insert() is used to insert entries in the map. Since keys are unique in a map, it first checks that whether the given key is already present in the map or not,

if it is present the entry is not inserted in the map and the iterator to the existing key is returned

otherwise new entry is inserted in the map.

- There are two variations of insert():

: **a)insert(pair)**

In this variation, a pair of key and value is inserted in the map. The inserted pair is always inserted at the appropriate position as keys are arranged in sorted order.

b) insert(start_itr , end_itr):

This variation inserts the entries in range defined by **start_itr** and **end_itr** of another map.

4. Find()

find() returns the iterator to the entry having key equal to given key (passed as parameter).

iterator=map_name.find(key)

or

constant iterator=map_name.find(key

Algorithm:
1. Start.
2. Give a header file to map associative container.
3. Insert states name &population of that state.
4. Use populationMap.insert().
5. Display the population of states**.**
6. Enter name of state for search
7. Display population of that state
8. End.

Input:
Information such as state name & population to map associative container.

Output:-

```
How many entries in a map 5

Enter state name maharashtra

Enter population 6543332

Enter state name bihar

Enter population 3441123

Enter state name kerala
```

```
 Enter population 3494985898

 Enter state name karnataka

 Enter population 345681

Enter state name punjab

 Enter population 45958986987

The map state is :

            State    Population

            bihar    3441123

            karnataka        3456810

            kerala  349498766

            maharashtra    6543332

            punjab 459589869

 Enter state kerala

Key-value pair present :

kerala->349498766
```

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

Questions:

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