# **ZEAL EDUCATION SOCIETY’s**

**ZEAL COLLEGE OF ENGINEEIRNG AND RESEARCH, NARHE, PUNE**

**DEPARTMENT OF COMPUTER ENGINEERING**

## **SEMESTER-I**

**[A.Y. : 2022 - 2023]**



**OBJECT ORIENTED PROGRAMMING**

**(210247)**

**LABORATORY MANUAL**

**Institute and Department Vision and Mission**

| **INSTITUTE**  **VISION** | To impart value added technological education through pursuit of academic excellence, research and entrepreneurial attitude. |
| --- | --- |
| **INSTITUTE**  **MISSION** | **M1:** To achieve academic excellence through innovative teaching and learning process.  **M2:** To imbibe the research culture for addressing industry and societal needs**.**  **M3:** To provide conducive environment for building the entrepreneurial skills.  **M4:** To produce competent and socially responsible professionals with core human values. |

| **DEPARTMENT VISION** | “To emerge as a department of repute in Computer Engineering which produces competent professionals and entrepreneurs to lead technical and betterment of mankind.” |
| --- | --- |
| **DEPARTMENT**  **MISSION** | **M1:** To strengthen the theoretical and practical aspects of the learning process by teaching applications and hands on practices using modern tools and FOSS technologies.    **M2:** To endeavor innovative interdisciplinary research and entrepreneurship skills to serve the needs of Industry and Society.  **M3:** To enhance industry academia dialog enabling students to inculcate professional skills.  **M4:** To incorporate social and ethical awareness among the students to make them conscientious professionals. |



**Department**

**Program Educational Objectives(PEOs)**

| **PEO1:** To Impart fundamentals in science, mathematics and engineering to cater the needs of society and Industries. |
| --- |
| **PEO2:** Encourage graduates to involve in research, higher studies, and/or to become entrepreneurs. |
| **PEO3:** To Work effectively as individuals and as team members in a multidisciplinary environment with high ethical values for the benefit of society. |

| **Savitribai Phule Pune University**  **Second Year of Computer Engineering (2019 Course)**  **210247: Object Oriented Programming Laboratory** | | |
| --- | --- | --- |
| **Teaching Scheme:** | **Credit** | **Examination Scheme:** |
| PR: 04 Hours/Week | 02 | TW: 25 Marks |
|  |  | PR: 50 Marks |

**Course Objectives:**

* To explore & understand the principles of Object Oriented Programming (OOP) .
* To use the object-oriented paradigm in program design.
* To provide object-oriented programming insight using C++.
* To lay a foundation for advanced programming.

**Course Outcomes:**

On completion of the course, student will be able to-

| CO1: | **Analyze** the strengths of object oriented programming. |
| --- | --- |
| CO2: | **Design** and apply OOP principles for effective programming. |
| CO3: | **Develop** the application using object oriented programming language(C++) |
| CO4: | **Apply** object-oriented concepts for advanced programming. |
|  |  |

## 

## **List of Assignments**

| **Sr.**  **No.** | **GROUP A** | **Page No.** | **Date of Conduction** | **Date of Submission** | **Sign** |
| --- | --- | --- | --- | --- | --- |
| 01 | Implement a class Complex which represents the Complex Number data type. Implement the following 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. |  |  |  |  |
| 02 | 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 license no. and other. 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 as well as exception handling. |  |  |  |  |
| 03 | 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. |  |  |  |  |
| **Group B** | |  |  |  |  |
| 04 | 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. |  |  |  |  |
| 05 | Write a function template selection Sort. Write a program that inputs, sorts and outputs an integer array and a float array. |  |  |  |  |
| **Group C** | |  |  |  |  |
| 06 | Write C++ program using STL for sorting and searching with user defined records such as person record(Name, DOB, Telephone number etc) using vector container. |  |  |  |  |
| 07 | 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. |  |  |  |  |



**Group A : Assignment No. 01**

**Problem Statement:**

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

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 operator overloading, constructor and data hiding.

**Outcome:**

To understand operator overloading concept.

**Software Requirement:**

1. Computer System with Linux/Open Source Operating System.

2. Open source programming tool like G++/GCC Compiler.

**Input:** Values for operator + and \*.

**Output**: Complex class implemented.

**Theory Concepts :**

**Complex Number:**

A **complex number** is a **number** that can be expressed in the form a + bi, where a and b are real **numbers** and i is the imaginary unit, that satisfies the equation i2 = −1. In this expression, a is the real part and b is the imaginary part of the **complex number**.

### **Operations on Complex number:**

**Addition of Complex Numbers:**

Theaddition of two complex numbers a + b i and c + d i is defined as follows

(a+bi) + (c+di) = (a+c) + (b+d) i

This is similar to grouping like terms: real parts are added to real parts and imaginary parts are added to imaginary parts.

**Subtraction of Complex Numbers:**

The subtraction of two complex numbers a + b i and c + d i is defined as follows.

(a + b i) - (c + d i) = (a - b) + (b - d) i

### **Multiplication Complex Numbers:**

The multiplication of two complex numbers a + b i and c + d i is defined as follows.

(a + b i)(c + d i) = (a c - b d) + (a d + bc) i

However you do not need to memorize the above definition as the multiplication can be carried out using properties similar to those of the real numbers and the added property i 2 = -1.

### **OOP Features & Concepts used in this practical :**

**Operator Overloading:**

Operator overloading is a compile-time polymorphism in which the operator is overloaded to provide the special meaning to the user-defined data type. Operator overloading is used to overload or redefines most of the operators available in C++. It is used to perform the operation on the user-defined data type. For example, C++ provides the ability to add the variables of the user-defined data type that is applied to the built-in data types.The advantage of Operators overloading is to perform different operations on the same operand.

**Operator that cannot be overloaded are as follows:**

* Scope operator (::)
* Sizeof
* member selector(.)
* member pointer selector(\*)
* ternary operator(?:)

**Following is the list of operators which can be overloaded:**

| + | - | \* | / | % | ^ |
| --- | --- | --- | --- | --- | --- |
| & | | | ~ | ! | , | = |
| < | > | <= | >= | ++ | -- |
| << | >> | == | != | && | || |
| += | -= | /= | %= | ^= | &= |
| |= | \*= | <<= | >>= | [ ] | ( ) |
| -> | ->\* | new | new [ ] | delete | delete [ ] |

## **Syntax of Operator Overloading**

return\_type class\_name  : : operator op(argument\_list)

{

     // body of the function.

}

Where, **return type** is the type of value returned by the function.

**class\_name** is the name of the class.

**operator op** is an operator function where op is the operator being overloaded, and the operator is the keyword.

## **Rules for Operator Overloading:**

* Existing operators can only be overloaded, but the new operators cannot be overloaded.
* The overloaded operator contains atleast one operand of the user-defined data type.
* We cannot use friend function to overload certain operators. However, the member function can be used to overload those operators.
* When unary operators are overloaded through a member function take no explicit arguments, but, if they are overloaded by a friend function, takes one argument.
* When binary operators are overloaded through a member function takes one explicit argument, and if they are overloaded through a friend function takes two explicit arguments.

**Conclusion:** Thus, we have implemented a class Complex which represents the Complex Number data type.

### 

**Question Bank:**

1. What is class?
2. What is data type? What are the different data types in C++?
3. What is constructor? Explain with example.
4. Give any one examples of operator overloading.



**Group A : Assignment No. 02**

**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 license no. and other. 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 as well as exception handling.

**Objective:**

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.

**Outcome:**

To understand constructor, destructor, static member functions, dynamic memory allocation, friend class, this pointer, inline code and operators-new and delete concept.

**Software Requirement:**

1. Computer System with Linux/Open Source Operating System.

2. Open source programming tool like G++/GCC Compiler.

**Input:** The details of student:Name, Roll number, Class, division, Date of Birth, Blood group, Contact address, telephone number, driving licence no. etc

**Output**: Display database of student information.

**Theory Concepts :**

**Constructor:**

A special method of the class that will be automatically invoked when an instance of the class is created

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.

### **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.
   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:**

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 :

1.A static function can have access to only other static members (fun or var) declared in the same class

2.A static function can be called using the class name instead of its object name

*Class\_name :: function\_name;*

The following restrictions apply to such static functions:

* They cannot access non static class member data using the member-selection operators (**.** or **>**).
* cannot be declared as **virtual**.
* 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.

### **Friend classes**

Just as we have the possibility to define a friend function, we can also define a class as friend of another one, granting that second class access to the protected and private members of the first one.

### **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 p, \* ptr ; // declaration

ptr = & a ; // initialization

### **Pointers to objects:**

Consider the following example

*item P ; // where item is class & P is object*

*Similarly, we can define a pointer item\_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.*

*Ex. 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.

**Inline function:**

 Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call. This substitution is performed by the C++ compiler at compile time. Inline function may increase efficiency if it is small.  
The syntax for defining the function inline is:

inline return-type function-name(parameters)

{

// function code

}

Remember, inlining is only a request to the compiler, not a command. Compiler can ignore the request for inlining. Compiler may not perform inlining in such circumstances like:  
1) If a function contains a loop. (for, while, do-while)  
2) If a function contains static variables.  
3) If a function is recursive.  
4) If a function return type is other than void, and the return statement doesn’t exist in function body.  
5) If a function contains switch or goto statement.

### **Algorithm:**

1. Start
2. Read personnel information such as Name, Roll number, Class, division, Date of Birth, Blood group, Contact address, telephone number, driving licence no. etc
3. Print all information from database.
4. Stop
5. **Input :** Personnel information such as Name, Roll number, Class, division, Date of Birth, Blood group, Contact address, telephone number, driving licence no. Etc
6. **Output :**Display personnel information from database.

### **Conclusions:**

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.

**Question Bank:**

* + 1. What is Pointer?
    2. What is Dynamic memory allocation operators-new and delete?



**Group A : Assignment No. 03**

**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 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 learn the concept of inheritance and exception handling.

**Outcome:**

To understand inheritance and exception handling concept.

**Software Requirement:**

1. Computer System with Linux/Open Source Operating System.

2. Open source programming tool like G++/GCC Compiler.

**Input:** Give the title (a string) and price (type float) of a publication.

**Output**: Base class derives the properties from base class.

**Theory Concepts :**

**Inheritance:**

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

The basic syntax of inheritance is:

**Class Derived\_Class : accessSpecifier Base\_Class**

**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

**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:-

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

2.It informs error has occurred. It is called as throwing the exception.

3.We receive the roe info. It is called as catching the exception.

4.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 try{

throw exception

}

catch(argument)

{

...

...

}

//rest of the code

// Exception Handling #include <iostream> using namespace std; int main()

int x = -1;

// Some code

cout<< "Before try \n"; try {

cout<< "Inside try \n"; if (x < 0)

{

throw x;

cout<< "After throw (Never executed) \n";

}

}

catch (int x ) {

cout<< "Exception Caught \n";

}

cout<< "After catch (Will be executed) \n"; return 0;

}

Output:

Before try Inside try

Exception Caught

After catch (Will be executed)

**Algorithm:**

1. Start.
2. Create classes Publication, book and tape.

3 .Publication class having data members title, price.

1. Class Book having data members pages and member functions getdata() and pudata().
2. Class Tape having data members minutes and member functions getdata() and pudata().
3. Create an object bof class book and object t of class tape.
4. Stop.

| **Input:** | A class publication that stores the title (a string) and price (type float) of publications. Derives two classes Book and Tape. |
| --- | --- |
| **Output:** |  |

| Display title and price from publication class. The result in following format: Enter Title: OOP  Enter Price: 300  Enter Pages: 250 Enter Title: POP Enter Price: 200  Enter Minutes: 60 Title: OOP  Price: 300  Pages: 250 Title: POP Price: 200  Minutes: 60 |
| --- |
| **Conclusion:** |
| Hence, we have successfully studied concept of inheritance and exception handling. |

### 

**Question Bank:**

1. Explain types of inheritance with examples?
2. What is Exception?



**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.

**Objective**: To learn the concept of Input/Output File.

**Outcome:** To understand the concept of Input/Output File.

**Theory Concepts :**

**Software and Hardware Requirement**

1. Computer System with Linux/Open Source Operating System.
2. Open source programming tool like G++/GCC Compiler.

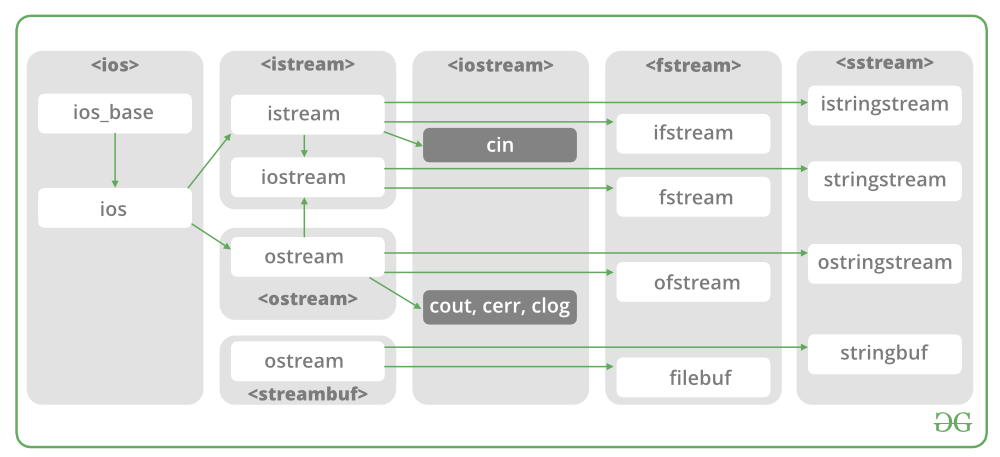
**Input:** Read the information written in the file.

**Output:** Write the information in output file.

**File Handling through C++ Classes**

File handling is used for store a data permanently in computer. Using file handling we can store our data in secondary memory (Hard disk).  
How to achieve the File Handling  
For achieving file handling we need to follow the following steps:-  
 STEP 1-Naming a file  
 STEP 2-Opening a file  
 STEP 3-Writing data into the file  
 STEP 4-Reading data from the file  
 STEP 5-Closing a file.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.   
**ofstream:** Stream class to write on files   
**ifstream:** Stream class to read from files   
**fstream:** Stream class to both read and write from/to files.



Now, the first step to open the particular file for read or write operation. We can open file by   
1. Passing file name in constructor at the time of object creation   
2. Using the open method

**For e.g.**

**Open File by using constructor**   
ifstream (const char\* filename, ios\_base::openmode mode = ios\_base::in);   
ifstream fin(filename, openmode) by default openmode = ios::in   
ifstream fin(“filename”);

**Open File by using open method**   
Calling of default constructor   
ifstream fin;  
fin.open(filename, openmode)   
fin.open(“filename”);

**Modes :**

| Member Constant | Stands For | Access |
| --- | --- | --- |
| in \* | Input | File open for reading: the internal stream buffer supports input operations. |
| Out | Output | File open for writing: the internal stream buffer supports output operations. |
| Binary | Binary | Operations are performed in binary mode rather than text. |
| Ate | at end | The output position starts at the end of the file. |
| App | Append | All output operations happen at the end of the file, appending to its existing contents. |
| Trunc | Truncate | Any contents that existed in the file before it is open are discarded. |

**Default Open Modes :**

| Ifstream | ios::in |
| --- | --- |
| ofstream | ios::out |
| Fstream | ios::in | ios::out |

**Conclusion:**

Thus we have created a program to create a file. Write information in it and then close the file. Again open file and read the written information.

**Question Bank:**

1. What is File Handling? What are the classes used for File Handling in C++?

2. What are the modes of File Handling?



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

**Objective:** To learn the concept of Selection Sort.

**Outcome:** To understand the concept of Selection Sort.

**Theory Concepts :**

**Software and Hardware Requirement**

1. Computer System with Linux/Open Source Operating System.
2. Open source programming tool like G++/GCC Compiler.

**Input:** Accept input of integer numbers and float numbers and store in integer array and float array..

**Output:** Sort the data using Function Template Selection Sort and display sorted data.

**Function templates**

[Selection sort](https://pages.cs.wisc.edu/~siff/CS367/Notes/search-sort.txt), as written, does not rely on the elements of data being of type int, it only requires that the elements of data be comparable (i.e. have a < operator defined for them), allow them to b initialized (think default constructor) and assigned to. Integers have these properties, but so do a host of other types.

Goal: parameterize sort so that it can operate on arrays of any types which support those basic operations.

How do we do this? A simple idea is to use the typedef mechanism.

**The statement:**

typedef t foo

makes foo a synonym for the type t (whatever type t is). In our case, we might write:

typedef int ElementType

and then rewrite sort using ElementType in place of some or all of occurrences of int.

Now if we want to change this sort function to operate on arrays of strings instead, we just change one line:

typedef String ElementType

This has its limitations: what happens if we want to use an int sort in one part of our program and a String sort in another?

A more flexible solution is to use function templates. A template is, like it sounds, a form where "blanks" get filled in at a later date. For sort, template function looks like:

template <class T>

void SelectionSort(T A[], int N)

{

T min;

int j, k, minIndex;

for (k = 0; k < N; k++) {

min = A[k]; minIndex = k;

for (j = k+1; j < N; j++) {

if (A[j] < min ) { min = A[j]; minIndex = j; }

}

Swap(A[k], A [minIndex]);

}

}

**Question Bank:**

1. Define Function Template.

2. Explain the use of typedef .

**Conclusion:**

Thus we have created function template selection sort, accept data from user, sort it and display.



**Problem Statement:**

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

**Objective:** To learn the concept of STL.

**Outcome:** To understand the concept of STL.

**Theory Concepts :**

**Software and Hardware Requirement**

1. Computer System with Linux/Open Source Operating System.
2. Open source programming tool like G++/GCC Compiler.

**Input:** Accept input from user personal records like Name, DOB, Telephone number and Item records like Item code, Name, Cost, and Quantity.

**Output:** Sort the data using Standard Template Library (STL)

**The C++ Standard Template Library (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. Working knowledge of [template classes](https://www.geeksforgeeks.org/templates-cpp/) is a prerequisite for working with STL.

**STL has 4 components:**

**Algorithms :** Algorithms act on containers. They provide the means by which you will perform initialization, sorting, searching, and transforming of the contents of containers.

**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.

**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 :** Iterators are used to step through the elements of collections of objects. These collections may be containers or subsets of containers.

**Conclusion:**

Thus we have created using STL, sort and search Personal and Item records using vector containers.

**Question Bank:**

1. What is STL.

2. What are the components of STL?

3. What is Vector 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.

**Objective:** To learn the concept of STL.

**Outcome:** To understand the concept of STL.

**Theory Concepts :**

**Software and Hardware Requirement**

1. Computer System with Linux/Open Source Operating System.
2. Open source programming tool like G++/GCC Compiler.

**Input:** Accept the name of state as an input from user.

**Output:** According to the input state from the user, looks in map and display the population of that state using Map Associative Container.

**Map in C++ Standard Template Library (STL)**

In C++, a **MAP** is an associative container storing items in a mapped form. Each item in the map is composed of key-value and a mapped value. Two mapped values cannot share the same key values.

The key values are good for sorting and identifying elements uniquely. The mapped values are for storing content associated with the key. The two may differ in types, but the member type combines them via a pair type that combines both.

## **Why use std::map?**

Here are reasons for using map:

* std:: map stores unique keys only in sorted order based on chosen sorting criteria.
* It’s easy and faster to search for elements using the key.
* Only one element is attached to each key.
* std::map can be used as an associative array.
* std::map is implementable using binary trees (balanced).

**Syntax:**

To declare std::map, use this syntax:

std::map<key\_datatype, value\_datatype>map\_name;

* The key\_datatype denotes the datatype of the map keys.
* The value\_datatype denotes the datatype of the values corresponding to the map keys.
* The map\_name is the name of the map.

**For example:**

map<string, int> my\_map;

We declared a map named my\_map. The map will have a string as **key** datatypes and integer as **values** datatype.

**Member types:**

The member functions can use the following members types as either parameters or return type:

* **key\_type:** Key (The first parameter in the template)
* **mapped\_type:** T (The second parameter in the template)
* **key\_compare:** Compare (The third parameter in the template)
* **allocator\_type:** Alloc (The fourth parameter in the template)
* **value\_type:** pair<const key\_type,mapped\_type>
* **value\_compare:** Nested function class for comparing elements
* **reference:** allocator\_type::reference
* **const\_reference:** allocator\_type::const\_reference
* **pointer:** allocator\_type::pointer
* **const\_pointer:** allocator\_type::const\_pointer
* **iterator:** a bi-directional iterator to the value\_type
* **const\_iterator:** a bi-directional iterator to the const value\_type
* **reverse\_iterator:** a reverse iterator
* **const\_reverse\_iterator:** a constant reverse iterator
* **difference\_type:** ptrdiff\_t
* **size\_type:** size\_t

**Built-in Functions**

std::map comes with inbuilt functions. Some of these include:

* **begin()-** This function returns the iterator to the first item of the map.
* **size()-**This function returns the number of items in a map.
* **empty()-**This function returns a Boolean value denoting whether a map is empty.
* **insert( pair(key, value))-** This function inserts new key-value pair to a map.
* **find(val)-** This function gives the iterator to the val element if it’s found. Otherwise, it will return m.end().
* **Erase (iterator position)-** This function deletes the item at the position pointed by the iterator.
* **erase(const g)**– This function deletes key-value g from a map.
* **Clear ()-**This function deletes all items from a map.

## **Iterating over Map Elements**

You can iterate over the map elements. We simply need to create an iterator and use it for this.

**For example:**

**Example 1:**

#include <iostream>

#include <string>

#include <map>

using namespace std;

int main() {

map<int, string> Students;

Students.insert(std::pair<int, string>(200, "Alice"));

Students.insert(std::pair<int, string>(201, "John"));

cout << "Map size is: " << Students.size() << endl;

cout << endl << "Default map Order is: " << endl;

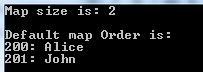
for (map<int, string>::iterator it = Students.begin(); it != Students.end(); ++it) {

cout << (\*it).first << ": " << (\*it).second << endl;

}

}

**Output:**



**Conclusion:**

Thus we have created program using Map Associative Container, user input state and program give the population of given state accordingly.

**Question Bank:**

1. What is Map Associative Container?

2. Explain the member types and built \_in functions in Map.