* Object Oriented Programming (OOP) - Practical Number - 1.

Name :- Kaustubh Shrikant Kabra.
Class :- Second Year Engineering.
Div :- A Roll Number :-
Batch :-
Department :- Computer Department
College :- AISSMS's IOIT.

Title :-

Difference between Object Oriented Programming and Procedural Oriented Programming

Learning Objectives :-

To understand the basic difference OOP and Procedural Oriented Programming Learning.

Outcome :-

Student are able to write C++ programs using basic concepts.

Aim :-

Write a simple C++ program to display person's name and number.

Software and Hardware Required :-

Linux Operating Systems, GCC, Eclipse frame-work.

Theory :-

	Procedure Oriented Programming	Object Oriented Programming
Divided Into	In POP, program is divided into small parts called functions.	In OOP, program is divided into parts called objects.
Approach	POP follows Top Down approach	OOP follows Bottom Up approach.
Access Specifier	POP does not have any access specifier	OOP has access specifier named Public, Private, Protected, etc.
Data Moving	In POP, Data can move freely from function to function in the system.	In OOP, objects can move and communicate with each other through member functions.
Expansion	To add new data and function in POP is not so easy.	OOP provides an easy way to add new data and function.
Data Hiding	POP does not have any proper way for hiding data so it is less secure.	OOP provides Data hiding so provides more security.

Algorithm :-

1. Declare class with variables and member variable.
2. Accept name and number.
3. Display name and number.
4. Stop.

Test Case:-

Input:-

Enter the Name - Kaustubh

Enter the Number - 29.

Expected Output:-

Kaustubh

29.

Conclusion:-

Thus we have understood difference between procedure and object oriented programming language. Also we have studied and implemented C++ program using basic constructor of OOP.

* Object Oriented Programming (OOP) - Practical Number - 2 (Group-A)

Name:- Kaustubh Shrikant Kabra.
Class:- Second Year Engineering.
Div:- A Roll Number:-
Batch:-
Department:- Computer Department.
College:- AISSMS's IOIT.

Title:-

Demonstrate basic concepts of object oriented programming for a class to store details of students.

Objectives:-

1) To understand use of different types of constructor and destructor.
2) To understand use of static members and inline keyword.
3) To understand this pointer, dynamic memory allocation operators like new and delete.

Problem Statement:-

Develop a program in C++ to create a database of student's information system containing the following information: Name, Roll Number, Class, Division, Date of Birth, Blood group, Contact address, Telephone number, Driving License number and other. Construct the database with suitable member functions. Make use of destructor, static member function, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete as well as exception handling.

Outcomes :-

1) Students will be able to demonstrate different types of constructor and destructor.
2) Student will be able to demonstrate use of static member and inline keyword.
3) Student will be able to demonstrate this pointer, dynamic memory allocation operators like new and delete.

Hardware Requirement :-

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1GB Hard Disk or more.

Software Requirement :-

64 bit Linux / Windows Operating System, G++ compiler.

Theory :-

Constructor -

Constructor is a special member function whose task is to initialize the objects of its class.

Characteristics of a constructor -

1) They should be declared in the public section.
2) Constructor name and class name must be same.
3) They cannot be inherited.
4) They cannot be virtual.

Types of constructor - 1) Default constructor
2) Parameterised constructor
3) Copy constructor.

Destructors:-

Destructor is a special class function which destroys the object as soon as the scope of objects ends. The destructor is called automatically by the compiler when the object goes out of scope.

Allocation of memory:-

There are two ways that memory get allocated for data storage:-

1) Compile Time (or static) Allocation:-

Memory for named variables is allocated by the compiler. Exact size and type of storage must be known at compile time. For standard array declarations, this is why the size has to be constant.

2) Dynamic Memory Allocation:-

Memory allocated "on the fly" during run time dynamically allocated space usually placed in a program segment known as the heap or the free store.

Algorithm:-

1) Create a class named student with data members as required.
2) Create the object of classes with default constructors, copy constructo
3) Define two member function viz set data, display data.
4) Define static member function Increment count, show_total_count.
5) Use this pointer to call display.
6) Initialize all student records, and display them.

Test Case:-

How many student you have?

1

Name :- Kaustubh
Roll Number :- 20
Class :- SE
Div :- A
Address :- Pune
Phone :- 9168100204
Date of Birth :- 29052001
Blood Group :-

Thanks you for deleting details.

Conclusion :- Constructors and destructors saves memory so can be used to create database efficiently.

* Object Oriented Programming (OOP) - Practical Number - 3 (Group - A)

Name :- Kaustubh Shrikant Kabra.
Class :- Second Year Engineering.
Div :- A Roll Number :-
Batch :-
Department :- Computer Department.
College :- AISSMS's IOIT.

Title :-

Arithmetic operations on complex number using operator overloading.

Objective :-

1. To understand operator overloading overloading, constructor and data hiding.
2. To demonstrate overloading of binary operator, insertion and extraction operator.

Problem Statement :-

Implement a class complex which represent the complex Number data type. Implement the following operations :-

1. Constructor
2. Overload operator + to add two complex number.
3. Overload operator * to multiply two complex number.
4. Overload << and >> to print and read complex number.

Outcomes :-

1) Student will be able to demonstrate use of constructor.

2) Student will be able to demonstrate binary operator overloading.
3) Student will be able to demonstrate Overloading of Insertion and extraction operator using friend function.

Hardware requirement:-
Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1GB Hard Disk or more.

Software requirement:-
64 bit Linux/ Windows Operating System, G++ compiler.

Prerequisite:-
Object Oriented Concepts.

Theory:-
Operator Overloading-
To define an additional task to an operator, we must specify what it means in relation to class to which the operator is applied. This is done with the help of a special functions, called operator function.

Unary Operator OverLoading-
The unary operators operate on a single operand and following are example of unary operators:-
1) Increament (++)
2) Decreament (--)
3) Unary minus (-)
4) Logical not (!)

Date :
Page :

Binary Operator Overloading -

In overloading binary operator, a friend function will have arguments, while a member function will have one argument

Rules for overloading operators :-

1) Only existing operators can be overloaded. New cant be.

2) The overloaded operator must have at least one operand that is of user defined type.

3) Overloaded operators follow the syntax rules of the original operators. They cannot be overridden.

4) Binary operators overloaded through a member function take one explicit argument and those which are overloaded through a friend function takes two explicit arguments.

5) When using binary operators overload through a member function, the left hand operand must be an object of the relevant class.

Algorithm :-

1) Define class for complex number containing real and imaginary part.
2) Accept 2 complex number from user.
3) Display the menu.
4) Choose the operation on complex numbers.
5) Display the result.

Date :
Page :

Test case:-

Enter real and img. part of 1st complex number :-
4 5
Enter real and img part of 2nd complex number :-
6 7.

1) Addition
2) Subtraction
3) Multiplication
4) Division
5) Exit.
Enter your choice:- 1.

Addition of complex numbers are :- 10+12i.

Enter your choic:- 3.

Multiplication of complex numbers are :- - 11 + 58i.

Enter your choise:- 5.

Thank You.

Conclusion:-

Here we have studied used and demonstrated use of binary operator overloading and insertion extraction operator overloding using friend function.

* Object Oriented Programming (OOP) - Practical Number - 4 (Group - A)

Name :- Kaustubh Shrikant Kabra.
Class :- Second Year Engineering
Div :- A Roll Number :-
Batch :-
Department :- Computer Department
College :- AISSMS's IOIT.

Title :-
Demonstrate reusability of code thru Inheritance and use of exception handling.

Objective :-
1) To learn and understand code reusability and demonstrate it using Inheritance concepts.
2) To learn, understand and demonstrate exception handling in object oriented environment.

Problem Statement :-
Write a program that instantiates the book and type classes, allows user to enter data and display the data members. If an exception is caught, replace all the data member values with zero value.

Outcomes :-
1) Students will be able to learn and understand inheritance and exception handling.
2) Students will be able to demonstrate inheritance and exception handling.

Hardware Requirement :-

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1GB Hard Disk or more.

Software Requirement :-

64 bit Linux / Windows Operating System, G++ compiler

Theory :-

Inheritances -

Inheritance is basically done by creating new classes, reusing the properties of the existing ones. The mechanism of deriving a new class from an old one is called inheritance.

Exception Handling -

An exception occurs when an unexpected error or unpredictable behaviors happened on your program not caused by the operating system itself.

Algorithm :-

1) Start
2) Create a base class named media with data members.
3) Derive 2 classes named book and cassette from base class.
4) Add page count and playing time as additional data members to inherited classes.
5) Use getdata and pushdata to take in data and run user and display entered data respectively.
6) Allows user to select media whether he wants to select books as media type or the other with as switch statement.

Date :
Page :

7) Use try catch mechanism to handle exceptions.

Test case:-

1) Add books
2) Add cassettes
3) Display available books
4) Display available cassettes.

Enter your choice:
1

Enter number of books
1.
Enter Title of book: KK
Enter number of pages in book: 999
Enter Price of the book: 500.

Do you want to continue? (Y/N)
Y
Enter your choice:
2
Enter details of cassette 1.
Enter Title of cassettes: KK
Enter the running time of cassette in minutes: 29.
Enter price of the cassette: 10.

Do you want to continue? (Y/N)
Y

CLASSMATE
Date :
Page :

Enter your choice:
3
Details of book 1:-
:
:
Do you want to continue? (Y/N)
Y
Enter your choice:
4
Details of cassette 1:-
:
:
Do you want to continue? (Y/N)
N
Thanks You

Conclusion:-

Here, we learned to use and demonstrate concept of inheri-tance and exception handling.

Date :
Page :

* Object Oriented Programming (OOP) - Practical Number - 5 (Group - B)

Name :- Kaustubh Shrikant Kabra.
Class :- Second Year Engineering
Div :- A Roll Number :-
Batch :-
Department :- Computer Department
College :- AISSMS's IOIT.

Title :-

Demonstrate various file operations on file using C++.

Objectives :-

1) To learn and understand streams and files in object oriented paradigm.
2) To demonstrate file operations like create, open, read write and close a file.

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.

Outcomes :-

1) Student will be able to learn and understand concepts of streams and files in C++.
2) Student will be able to demonstrate various operations like creating a new file, opening an existing file, reading from file, closing file.

Hardware requirements :-

Any CPU with Pentium processor, 256 MB RAM or more, 1GB Hard Drive or more.

Software requirements :-

64-bit Linux / Windows Operating System, G++ compiler.

Theory :-

The iostream library is an object-oriented library that provides inputs and outputs functionality using streams.

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 inputs 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.

```
# include <iostream>
# include <fstream>
using namespace std;
int() main() {
of stream myfile;
myfile. open ("example.txt");
```

```
myfile << "Writing this to a file.\n";
myfile.close();
return 0;
}
```

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 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);

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:: trunk - If the file already exists, its contents will be truncated before opening the file.

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. Following is the standard syntax for close() function, which is a member of fstream,
ifstream, and of stream objects
void close();

Writing a File :-

While doing C++ programming you write information to a file from your program using 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.
Writing operations on text files are performed in the same way we operated with cout:

```
# include < iostream>
# include < fstream>
using namespace std;
int main ( ) {
ofstream myfile ("example. txt");
if (myfile ! is _ open ( ))
{
myfile << "This is a line. \n";
myfile << "This is another line. \n";
myfile close ( )
}
}
```

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.

Reading from a file can also be performed in the same way that we did with cin:

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;

int main() {
string line;
ifstream myfile ("example.txt");

if (myfile.is_open())
{
while (getline(myfile, line))
{
 cout << line << '\n';
}
myfile.close();
}
else cout << "Unable to open file";
return 0;
}
```

Algorithm :-

1) Start.
2) In main(), first create O/P stream and connect file using open() function.
3) Accept I/P from user
4) Write accepted data into file.
5) Close file using closes function.
6) Create I/P stream of connect file to it.
7) Check the end of the file.
8) Read data from file.
9) Display data on screen. Repeat 1 to 9.
10) If end of file detected Stop reading data.
11) Close the file.
12) Stop.

Test case :-

Enter the Data to be stored in file.

Enter Student Name :- Kaustubh Kabra.
Enter Roll Number :- 20.
Enter Division :- A
Enter Class :- SE Comp-1.
Enter Contact Number :-

Data Stored in file.
Kaustubh Kabra.
20
A
SE Comp-1.

CLASSMATE
Date :
Page :

Conclusion :-

Hence we studied and learned various file handling operations and methods available in and stream header file.

* Object Oriented Programming (OOP) - Practical Number - 6 (Group - B)

Name :- Kaustubh Shrikant Kabra
Class :- Second Year Engineering
Div :- A Roll Number :-
Batch :-
Department :- Computer Department
College :- AISSMS's IOIT.

Title :-
Demonstrate function templates for sorting.

Objective :-
1) To learn and understand templates.
2) To demonstrate function templates for selection sort.

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

Outcomes :-
1) Student will be able to learn and understand working and use of function template.
2) Student will be able to demonstrate function template for selection sort.

Hardware Requirement :-
Any CPU with pentium processor or similar, 256 MB RAM or more, 1GB Hard Disk or more.

Software Requirements:-
64 bit Linux/Windows Operating System, G++ compiler.

Theory:-

Templates-

Templates are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or to work on many different data types without being rewritten for each one templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

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

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

// body of function
}
```

Selection Sort-

Selection sort is a simple 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.

This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$ where n are number of items.

Step1- Set MIN to location 0.

Step2- Search the minimum element in the list.

Step3- Swap with value at location MIN.

Step4- Increment MIN to point to next element.

Step5- Repeat until list is sorted.

Algorithm :-

1) Start.
2) Declare function template for selection sort.
3) Template <class T> T selection (T a[], Tn, {})
4) In main declare variable for choice.
5) Accept choice from user.
6) Call function template for int array x and n=size.
7) Display sorted array.
8) If choice == 2) accept size and element in float
9) Go to step6
10) Stop.

Test case:-

Number of element:- 7

Enter elements:- 27 9 -27 7 5 19 -19

After pass 1:- -27 27 9 7 5 19 -19

After pass 2:- -27 -19 27 9 7 5 19

After pass 3:- -27 -19 5 27 9 7 19

After pass 4:- -27 -19 5 7 27 9 19

After pass 5:- -27 -19 5 7 9 27 19

After pass 6:- -27 -19 5 7 9 19 27.

After sorting:- -27 -19 5 7 9 19 27.

Conclusion:-

Here we demonstrated use of function template for selection sort.

* Object Oriented Programming (OOP) - Practical Number - 7 (Group - C)

Name :- Kaustubh Shrikant Kabra.
Class :- Second Year Engineering.
Div :- A
Roll Number :-
Batch :-
Department :- Computer Department
College :- AISSMS's IOIT.

Title :-

Demonstration of STL for Sorting and Searching with user-defined records such as Item Record.

Objective :-

1) To learn and understand concepts of Standard Template Library.
2) To demonstrate STL for implemention of sorting and searching operations

Problem Statement :-

Write C++ program using STL for sorting and searching user defined records such as Item records using vector container.

Outcomes :-

1) Student will be able to learn and understand concepts of STL.
2) Student will be able to demonstrate various operations for Sorting and searching using STL.

Hardware requirement :-

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirement :-
64 bit linux / Windows Operating System, G++ compiler.

Theory :-
The C++ STL (Standard Template Library) is a powerful set of C++ template classes to provide general-purposed templatized classes and function that implement many popular and commonly

"With this, we are almost ready to translate the example shown above to find the highest value in an array, using the STL tools. I will first present the implementation, and then explain the details:

```
vector<int>:: iterator current = value.begin();
int high =* current++;
while ( current != value.end())
{
if (* current > high)
{
high =* current.
}
current++;
}
```

Algorithm :-

1) A vector container is used as STL.
2) It supports a random access iterator provides an efficient implementation for frequently used operations.
3) Use random access iterator for accessing vector elements.
4) If we want to sort the elements of use algorithm sort.
5) For searching find it give location of specified element.

Conclusion :-

We have successfully implemented vector container using STL for searching and sorting data.

* Object Oriented Programming (OOP) - Practical Number 8 (Group-C)

Name:- Kaustubh Shrikant Kabra.
Class:- Second Year Engineering.
Div:- A Roll Number:-
Batch:-
Department:- Computer Department.
College:- AISSMS's IOIT.

Title:-
Demonstration of implementation of map associative container.

Objective:-
1) To learn and understand concepts of Standard Template Libary.
2) To demonstrate STL for implementation of map associative container.

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

Outcomes:-
1) Student will be able to learn and understand concepts of STL.
2) Student will be able to implementation map associative container concepts.

Theory :-

Associative containers are those that provide direct access to its elements for storage and retrieval purposes. The elements are accessed via keys, also known as search keys. There are four ordered and four unordered associative containers in C++ such as multiset, set, multimap, map and unordered_multiset, unordered_set, unordered_multimap and unordered_map

- Map Associative Containers :-

The map associative container stores elements as key-value pairs. It uses unique keys to perform fast storage and retrieval of its associative values.

Elements can be inserted and removed from anywhere in the map. If we do not want the constraint of ordering the keys, we can use its unordered version, called the unordered_map.

Example -

```
#include <iostream>
#include <map>
#include <iterator>
#include <string>
using namespace std;

int main()
{
    map<int, string, less<int>> weekdays;

    weekdays.insert(make_pair(1, "Sunday"));
    weekdays.insert(make_pair(2, "Monday"));
    weekdays.insert(make_pair(3, "Wednesday"));
    weekdays.insert(make_pair(5, "Thursday"));
```

```
weekdays.insert(make_pair(7,"Saturday"));
weekdays.insert(make_pair(5,"Thursday"));
weekdays.insert(make_pair(3,"Tuesday"));
weekdays.insert(make_pair(6,"Friday"));

for(auto day: weekdays)
    cout<<day.first<<"_"<< day.second<<endl;

cout <<"\n----------"<<endl
weekdays[2] = "Monday";

for(auto day: weekdays)
    cout<<day.first<<"_"<<day.second<< endl.

    return 0;
}
```

- Multimap Associative Container -

Similar to the map, the multimap associative container is also an associative container. The elements of multimap also are stored in key-value pairs. The relationship between key-value pairs, therefore, is of one-to-many. If we do not want the constraint of ordering the keys, we can used its unordered version called the unordered_multimap.

Example -

```
#include<iostream>
#include<map>
#include<iterator>
#include<string>
```

Date :
Page :

```
using namespace std;
int main()
{
  multimap<int, string, less<int>> box;

  box.insert(make_pair(1, "socks"));
  box.insert(make_pair(3, "T-Shirt"));
  box.insert(make_pair(6, "Gloves"));
  box.insert(make_pair(4, "Shirt"));
  box.insert(make_pair(2, "Jacket"));

  for (auto day : box)
     cout << day.first << "_" << day.second << endl;

  cout << "There are" << box.count(1) << "pair of socks" << endl;

  return 0;
}
```

Algorithm :-

Conclusion :-

We have successfully implemented map associative container.

* Object Oriented Programming (OOP) - Practical Number 9

Name :- Kaustubh Shrikant Kabra.
Class :- Second Year Engineering
Div :- A Roll Number :-
Batch :-
Department :- Computer Department
College :- AISSMS's IOIT.

Title :-
Program demonstrates command Line argument.

Objective :-
1) To learn and understand concepts of command line argument.
2) To demonstrates implementation of conramnd line argument.

Problem Statement :-
Write a program in C++ for command Line argument.

Outcomes :-
1) Student will be able to learn and understand command line argument.
2) Student will be able to implement command line argument.

Hardware requirement :-
Any CPU with Pentium processor, 256 MB RAM or more, 1GB Hard Drive or more.
Software requirement :- 64 bit linux / Windows Operating System, G++ compiler.

Theory:-

Command Line Argument:-

Command line arguments are given after the name of the program in command-line shell of Operating System.

To pass command line argument, we typically define main() with two arguments : first argument is the number of command line argument and second is list of command line argument.

```
int main(int argc, char *argv[]){/*...*/}
```

Properties:-

1. They are passed to main() function.
2. They are parameters/arguments supplied to the program when it is invoked.
3. argv[argc] is a Null pointer
4. argv[0] holds the name of the program.
5. argv[1] points to the first command line argument and argv[n] points last argument.

Conclusion:-

We have learned and stued the implementation of Command Line Argument.