Department of Artificial Intelligence & Machine Learning

VISION

To Develop skilled professionals in the field of Artificial Intelligence & Machine Learning Contributing globally to the benefit of industry and society**.**

MISSION

- To impart Knowledge in cutting edge Artificial Intelligence technologies that meets industry standards.
- To collaborate with industry to uplift innovative research and development in Artificial Intelligence & Machine Learning and related domains to meet societal demands.
- To produce successful computer science and Engineering graduates with a specialization in Artificial Intelligence & Machine Learning with personal and professional responsibilities, and a Commitment to lifelong learning.

LAB OBJECTIVE

- Introduces object-oriented programming concepts using the C++ language.
- Introduces the principles of data abstraction, inheritance and polymorphism;
- Introduces handling of filesand exception handling

Lab Exercises

Q. No	Problem Statement	CO, PO,PSO
1.	Write a C++ program to find the sum of all the natural numbers from 1 to n.	Co1,Po1,Po2,P03,Po5, PSo1
2.	Write a C++ program to sort the elements in ascending and descending order.	Co1,Po1,Po2,P03,Po5, PSo1
3.	Write a C++ program to swap 2 values by writing a function that uses call by reference technique.	Co1,Po1,Po2,P03,Po5, PSo1
4.	Write a C++ program to demonstrate function overloading for the following prototypes. add(int a, int b) add(double a, double b)	Co1,Po1,Po2,P03,Po5, PSo1
5.	Create a class named Shape with a function that prints "This is a shape". Create another class named Polygon inheriting the Shape class with the same function that prints "Polygon is a shape". Create two other classes named Rectangle and Triangle having the same function which prints "Rectangle is a polygon" and "Triangle is a polygon" respectively. Again, make another class named Square having the same function which prints "Square is a rectangle".Now, try calling the function by the object of each of these classes.	Co3,Po1,Po2,P03,Po5, PSo1
6.	Suppose we have three classes Vehicle, FourWheeler, and Car. The class Vehicle is the base class, the class FourWheeler is derived from it and the class Car is derived from the class FourWheeler. Class Vehicle has a method 'vehicle' that prints 'I am a vehicle', class FourWheeler has a method 'fourWheeler' that prints 'I have four wheels', and class Car has a method 'car' that prints 'I am a car'. So, as this is a multi-level inheritance; we can have access to all the other classes methods from the object of the class Car. We invoke all the methods from a Car object and print the corresponding outputs of the methods. So, if we invoke the methods in this order, car(), fourWheeler(), and vehicle(), then the output will be 16-2-2023 3 I am a car I have four wheels I am a vehicle Write a C++ program to demonstrate multilevel inheritance using this.	Co3,Po1,Po2,P03,Po5, PSo1
7.	Write a C++ program to create a text file, check file created or not, if created it will write some text into the file and then read the text from the file.	Co4,Po1,Po2,P03,Po5, PSo1
8.	Write a C++ program to write and read time in/from binary file using fstream	Co4,Po1,Po2,P03,Po5, PSo1
9.	Write a function which throws a division by zero exception and catch it in catch block. Write a C++ program to demonstrate usage of try, catch and throw to handle exception.	Co4,Po1,Po2,P03,Po5, PSo1
10.	Write a C++ program function which handles array of bounds exception using C++.	Co4,Po1,Po2,P03,Po5,PSo 1

Introduction

Concepts of Object Oriented programming using C++

1.Classes and Objects:

A class is a user defined data type.Object is an instance of that class. By default the members are private.

Syntax of class declaration

```
Class classname name
{
public:
data members;
method members;
private:
data members;
method members;
};
```

Ex:

```
class Student
{
string name;
public:
void read()
        {
cin>>name;
        }
void display()
        {
cout<<name;
        }
};
```

Syntax for object declaration

Classname variable;

Ex: Student s;

Private members are not accessible outside the class. But public members are accessible outside the class. To access the members of a class first the object of that class is created. Then the object name and dot(.) operator is used.

Ex. sdisplay();

2.Abstraction and Encapsulation

- Hiding implementation details is called abstraction. Encapsulation is Binding data and method as single unit. In the above example data(name)and methods(read( )and display()) are put together.

3.Inheritance

one class inherits all the attributes of some other class.

Types of inheritance:

1.Single level 2. Multilevel 3.Multiple 4. Hierarchical 5. Hybrid inheritance.

4.Polymorphism:

It is the ability of a single object to appear in many forms.

Types of Polymorphism

1.Compile time Polymorphism(Function overloading and Operator overloading)

The resolution of the function happens at compile-time

2.Run timePolymorphism (Function Overriding)

The resolution of the function happens at compile-time

3.Parametric Polymorphism(Templates)

Exception

Exceptions are runtime anomalies or abnormal conditions that a program encounters .

Exception handling in C++ consist of **three keywords: try , throw and catch** .All exceptions are derived from std::exception class.

- try **-** code that may raise an exception
- throw - throws an exception when an error is detected
- catch - code that handles the exception thrown by the throw keyword

The basic syntax for exception handling in C++ is given below:

```
try{
    //code that may raise an exception
throw argument;
}
catch(exception){
//code to handle exception
}
```

Here, we have placed the code that might generate an exception inside the try block.

Every try block is followed by the catchblock.

When an exception occurs, the throw statement throws an exception, which is caught by the catch block.

The catchblock cannot be used without the try block.

Advantage

It maintains the normal flow of the application. In such case, rest of the code is executed even after exception.

File Handling

File handling is a mechanism to store the output of a program in a file and help perform various operations on it. Files help to store these data permanently on a storage device. The term "Data" is commonly referred to as known facts or information. In the present era, data plays a vital role.

Types of file

1.binary file 2.text file

File processing consists of creating, storing, and/or retrieving the contents of a file .

File handling functions:

fprintf(), fscanf(), fgetc(), fputc(), fgets(), fputs() will help you in writing and reading to a file.

The fstream library allows us to work with files.Functions for file handling.

There are three classes included in the fstream library, which are used to create, write or read files:

class	Description
oftream	Creates and writes to files
ifstream	Reads from files
fstream	Acombination of ofstream and ifstream:creates,reads and writes to files

File Opearations:

No.	Function	Description
1	**open()**	This is used to create a file.
2	**read()**	This is used to read the data from the file
3	**write()**	reads data from the file
4	**close()**	This is used to close the file.

Syntax for opening a file

open(FileName , Mode);

Syntax for writing into a file

FileName<<"Insert the text here";

Syntax for reading from a file

FileName>>Variable

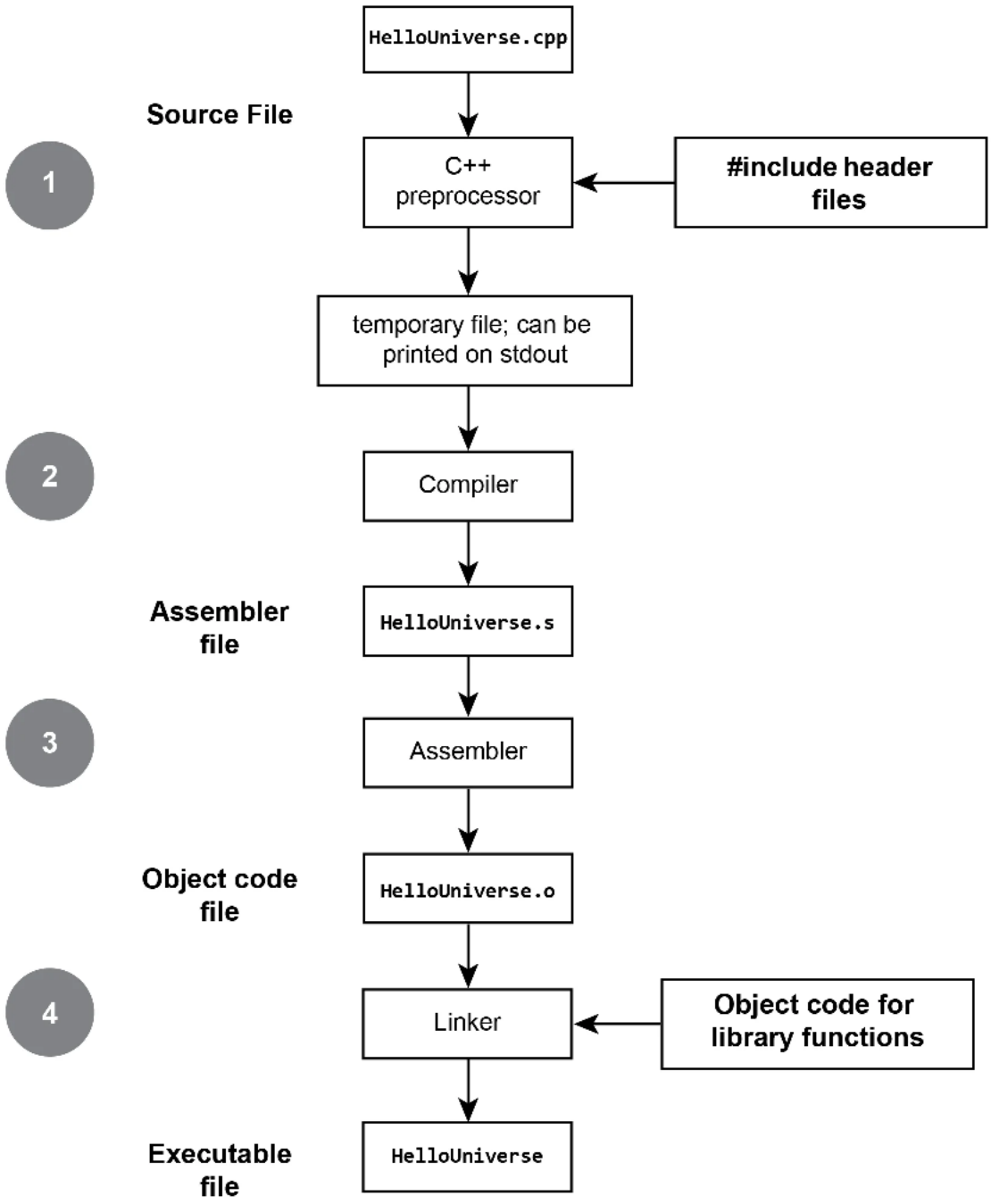
Syntax for closing a file

FileName.close();

Compiling C++ source code into machine-readable code consists of the following four processes:

1. Preprocessing the source code.
2. Compiling the source code.
3. Assembling the compiled file.
4. Linking the object code file to create an executable file.

The following diagram shows C++ compilation process.



IDE Used : Code blocks

Code Blocks:

It is a cross platform that supports compiling and running multiple programming languages.It can work with a variety of compilers.

Steps for using code blocks

1.click codeblocks icon on the desktop

2.select<file>select <new> and then select <project>

3.Double click <console application>

4.click<next> and select <c++>

5.Write the project name in <project title>, write the path or folder to create project.

6. click<next>

7. click<finish>

8. workspace is created with the given project name

9.Double click Sources main.cpp file is created. Double click main.cpp template for main is created. Then start writing program

10.Afterentering the program save it.

11.click on <build> to compile the program

12.click on <run> button to execute the program.

Program 1:

Write a C++ program to find the sum of all the natural numbers from 1 to n.

```
#include <iostream>
using namespace std;
void naturalsum(int n)
 {
int sum=0, i;
for(i=1;i<=n;i++)
sum=sum+i;
cout<<"sum of first "<< n <<" natural numbers : "<<sum<<endl;
  }
int main( )
 {
int n;
cout<<"enter the value of n:  ";
cout<<endl;
cin>>n;
naturalsum(n);
}
```

Sample Intput:

```
enter the value of n:
5
```

Output:

```
sum of  first  5 natural numbers : 15
```

Program 2:

Write a C++ program to sort the elements in ascending and descending order.

```
#include <iostream>
using namespace std;
//method to sort elements in ascending order
void ascending(int a[], int n)
  {
        int i, j, temp;
        for(i=0;i<n-1;i++)
        {
                for(j=0; j<n-1-i; j++)
                 {
                if(a[j]>a[j+1])
                {
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
 }
 }
 }
 }
//method to sort elements in descending order
void descending(int a[ ], int n)
  {
        int i, j, temp;
        for(i=0; i<n-1; i++)
         {
                for(j=0; j<n-1-i; j++)
```

```
                {
                        if(a[j]<a[j+1])
                {
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
                }
        }
    }
}
```

//main method

```
int main( )
{
int a[10], n, i, choice;
char ch;
do
  {
                cout<<"enter the size of the array";
                cin>>n;
                cout<<"enter the elements";
                for(i=0; i<n; i++)
                cin>>a[i];
                cout<<"elements before sorting";
                cout<<endl;
                for(i=0; i<n; i++)
                cout<<a[i]<<" ";
                cout<<endl;
```

```
            cout<<"1. ascending order sort  2. descending order sort";
            cout<<endl<<"enter your choice:  ";
            cin>>choice;
            switch (choice)
        {
                case 1:
                    cout<<"elements after sorting";
                    ascending(a,n);
                    cout<<endl;
                    cout<<endl;
                    for(i=0; i<n; i++)
                        cout<<a[i]<<" ";
                    break;
                case 2:
                    cout<<"elements after sorting";
                    descending(a,n);
                    cout<<endl;
                    cout<<endl;
                    for(i=0; i<n; i++)
                        cout<<a[i]<<" ";
                    break;
                default:
                    cout<<"invalid option";
             }
cout<<endl<<"do u want to continue y/n";
cin>>ch;
}while(ch=='y'|| ch=='Y');
cout<<"bye";
}
```

Sample Input

enter the size of the array

4

enter the elements

23

67

88

11

elements before sorting

23 67 8 11

1. ascending order sort 2. descending order sort

enter your choice: 1

elements after sorting

Output:

8 11 23 67

do u want to continue y/n y

Program 3:

Write a C++ program to swap 2 values by writing a function that uses call by reference technique

```
#include <iostream>
using namespace std;
void swap(int &p,int &q)
{
int temp;
temp=p;
   p=q;
   q=temp;
}
int main()
{
int a,b;
cout<<"enter the two numbers"<<endl;
cin>>a>>b;
cout<<"Elements before swapping"<<endl;
cout<<"a= "<<a<<endl;
cout<<"b= "<<b<<endl;
swap(a,b);
cout<<"Elements after swapping"<<endl;
cout<<"a= "<<a<<endl;
cout<<"b= "<<b<<endl;
return 0;
}
```

Sample input:

Enter two numbers

10 20

Elements before swapping

a=10

b=20

Output:

Elements after swapping

a=20

b=10

OR

Call by address program

```
#include <iostream>
using namespace std;
void swap(int *p, int *q)
{
int temp;
temp=*p;
   *p=*q;
   *q=temp;
}
int main()
{
int a,b;
cout<<"enter the two numbers"<<endl;
cin>>a>>b;
cout<<"elements before swapping"<<endl;
cout<<"a= "<<a<<endl;
cout<<"b= "<<b<<endl;
swap(&a, &b);
cout<<"elements after swapping"<<endl;
cout<<"a= "<<a<<endl;
cout<<"b= "<<b<<endl;
return 0;
}
```

Sample input:

Enter two numbers

10 20

Elements before swapping

a=10

b=20

Output:

Elements after swapping

a=20

b=10

Program 4:

Write a C++ program to demonstrate function overloading for the following prototypes. add(int a, int b) add(double a, double b)

```
#include <iostream>
using namespace std;
void add(int p, int q)
  {
int c;
     c=p+q;
cout<< c;
  }
void add(double p, double q)
  {
double c;
     c=p+q;
cout<< c;
  }
int main( )
  {
int a,b;
double x, y;
cout<<"enter two integer numbers";
cin>>a>>b;
cout<<a<<"+"<<b<<"=";
add(a,b);
cout<<endl<<"enter two double numbers";
cin>>x>>y;
cout<<x<<"+"<<y<<"=";
add(x,y);
return  0;
  }
```

Sample Input:
enter two integer numbers: 12 34
Output
12+34=46
Sample Input:
enter two double numbers: 6.7 8.9
Output:
6.7+8.9=15.6

Program 5:

Create a class named Shape with a function that prints "This is a shape". Create another class named Polygon inheriting the Shape class with the same function that prints "Polygon is a shape". Create two other classes named Rectangle and Triangle having the same function which prints "Rectangle is a polygon" and "Triangle is a polygon" respectively. Again, make another class named Square having the same function which prints "Square is a rectangle".Now, try calling the function by the object of each of these classes.

```
#include <iostream>
using namespace std;
class Shape
{
public:
void display()
  {
cout<<endl<< "This is a shape";
  }
};
class Polygon:public Shape
{
public:
void display()
  {
    Shape::display();
cout<<endl<< "Polygon is a shape";
  }
};
class Rectangle:public Polygon
{
public:
void display()
  {
        Shape::display();
cout<<endl<<"Rectangle is a Polygon ";
  }
};
class Triangle:public Polygon
{
public:
void display()
  {
        Shape::display();
cout<<endl<<"Triangle is a Polygon ";
```

```
  }
};
class Square: public Rectangle
{
public:
void display()
  {
        Shape::display();
cout<<endl<<"Square is Rectangle  ";
  }
};
int main()
{
     Polygon p;
     Rectangle r;
     Triangle t;
     Square sq;
p.display();
r.display();
t.display();
sq.display();
return 0;
 }
```

Output:

```
This is a shape

Polygon is a shape

This is a shape

Rectangle is a Polygon

This is a shape

Triangle is a Polygon

 This is a shape

Square is Rectangle
```

Program 6: Suppose we have three classes Vehicle, FourWheeler, and Car. The class Vehicle is the base class, the class FourWheeler is derived from it and the class Car is derived from the class FourWheeler. Class Vehicle has a method 'vehicle' that prints 'I am a vehicle', class FourWheeler has a method 'fourWheeler' that prints 'I have four wheels', and class Car has a method 'car' that prints 'I am a car'. So, as this is a multi-level inheritance; we can have access to all the other classes methods from the object of the class Car. We invoke all the methods from a Car object and print the corresponding outputs of the methods. So, if we invoke the methods in this order, car(), fourWheeler(), and vehicle(), then the output will be I am a car I have four wheels I am a vehicle Write a C++ program to demonstrate multilevel inheritance using this.

```
#include <iostream>
using namespace std;
class Vehicle
{
public:
void vehicle()
  {
cout<<"i am a vehicle"<<endl;
  }
};
classFourWheeler: public Vehicle
{
public:
voidfourwheeler()
  {
cout<<"i have four wheels"<<endl;
  }
};
classCar:public FourWheeler
{
public:
void car()
  {
cout<<"i am a car"<<endl;
  }
};
int main()
{
  Car c;
c.vehicle();
c.fourwheeler();
c.car();
return 0;
}
```

Output:

i am a vehicle
i have four wheels
i am a car

Program7:

Write a C++ program to create a text file, check file created or not, if created it will write some text into the file and then read the text from the file.

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

int main( )
{
fstream file; //object of fstream class

  //opening file "sample.txt" in out (write) mode
file.open("sample.txt", ios::out);

if(!file)
  {
cout<<"Error in creating file!!!"<<endl;
return 0;
  }

cout<<"File created successfully."<<endl;
  //write text into file
file<<"ABCD.";
  //closing the file
file.close();

  //again open file in read mode
file.open("sample.txt",ios::in);
if(!file)
  {
cout<<"Error in opening file!!!"<<endl;
return 0;
  }
  //read untill end of file is not found.
charch; //to read single character
cout<<"File content: ";

while(!file.eof())
  {
file>>ch; //read single character from file
cout<<ch;
  }
file.close(); //close file
return 0;
}
```

Output:ABCD..

Program 8:

Write a C++ program to write and read time in/from binary file using fstream

```
#include <iostream>
#include <fstream>
#include <iomanip> //for setfill() and setw()

using namespace std;

#define FILE_NAME "time.dat"

//function to write time into the file
void writeTime(int h, int m, int s){

	charstr[10];
	fstream file;
	file.open(FILE_NAME, ios::out|ios::binary);

	if(!file){
		cout<<"Error in creating file!!!"<<endl;
		return;
	}

	//make string to write
	sprintf(str,"%02d:%02d:%02d",h,m,s);

	//write into file
	file.write(str,sizeof(str));
	cout<<"Time "<<str<<" has been written into file."<<endl;

	//close the file
	file.close();
}
//function to read time from the file
void readTime(int *h, int *m, int *s){

	charstr[10];
	intinH,inM,inS;

	fstreamfinC;
	finC.open(FILE_NAME, ios::in|ios::binary);
	if(!finC){
		cout<<"Error in file opening..."<<endl;
		return;
	}
```

```
        if(finC.read((char*)str,sizeof(str))){
                //extract time values from the file
                sscanf(str,"%02d:%02d:%02d",&inH,&inM,&inS);
                //assign time into variables, which are passing in function
                *h=inH;
                *m=inM;
                *s=inS;
        }
        finC.close();
}
int main( )
{
        int m,h,s;
        cout<<"Enter time:\n";
        cout<<"Enter hour: ";     cin>>h;
        cout<<"Enter minute: "; cin>>m;
        cout<<"Enter second: "; cin>>s;
//write time into file
        writeTime(h,m,s);
        //now, reset the variables
        h=m=s=0;
        //read time from the file
        readTime(&h,&m,&s);

        //print the time
        cout<<"The time is
"<<setw(2)<<setfill('0')<<h<<":"<<setw(2)<<setfill('0')<<m<<":"<<setw(2)<<setfill('0')
<<s<<endl;
        return 0;
}
```

Input:

Enter time:

Enter hour:12

Enter minute:32

Enter second:22

Output:

Time 12:32:22 has been written into file

The time is 12:32:22

Program 9:

Write a function which throws a division by zero exception and catch it in catch block. Write a C++ program to demonstrate usage of try, catch and throw to handle exception.

```
#include <iostream>
using namespace std;
void divide(int a, int b)
 {
if(b!=0)
      {
int c=a/b;
cout<<endl<<"result="<<c;
      }
else
throw(b);
   }
int main( )
{
try
   {
divide(4,3);
divide(4,0);
   }
catch(int e)
   {
cout<<endl<<"divide by zero";
   }
return 0;
}
```

Sample Input:

4 3

Output :

result=1

input:

4 0

Output:

divide by zero

Program10:

Write a C++ program function which handles array of bounds exception using C++.

```
#include <iostream>
using namespace std;
void check_array_index(int a[10],intn,inti)
{
        try
         {
                if (i>=0 &&i<n)
                        cout<<"the element at position  "<<i<<"  is: "<<a[i];
                else
                        throwi;
         }
        catch(int e)
         {
                cout<<endl<<"array index out of bound";
          }
  }
int main()
{
int a[10],n,pos;
cout<<"enter the size of  an array :  ";
cin>>n;
cout<<"enter the elements of an  array "<<endl;
for(inti=0;i<n;i++)
cin>>a[i];
cout<<"enter the index of element in an array : ";
cin>>pos;
check_array_index(  a, n, pos);
return 0;
}
```

Output:

Run1:

Sample input:

enter the size of an array : 3
enter the elements of an array
89
56
77
enter the index of element in an array: 2

Output:

the element at position 2 is: 7

Run 2:

Sample input:

enter the size of an array : 4
enter the elements of an array
33
44
67
98
enter the index of element in an array: 5

Output:

array index out of bound