 **R N S Institute of Technology**

**(***AICTE Approved, VTU Affiliated, NAAC Accredited with 'A' Grade)*

**Dr. Vishnuvardhan Road, R.R Nagar Post, Channasandra, Beganluru-560098.**

**ESTD: 2001**

***An Institute with a Difference***

**C++ Programming Lab**

**(18MCA16)**

|  |
| --- |
| College  **VISION**  *" Building RNSIT into a world-class institution "*  **MISSION**  *" To impart high quality education in Engineering and Technology and Management with a difference, enabling students to excel in their career "* |
| Department  **VISION**  *“Synergizing Computer Applications for real world”*  **MISSION**  *Produce technologists of highest caliber to engage in design research and development, so as to enable the nation to be self-reliant*  *Give conceptual orientation in basic computer applications and mathematics, motivate the students for lifelong learning*  *Integrate project environment experiences at every level of the post graduate curriculum to give a firm practical foundation.* |

**Prepared by**

**Rajatha S Asst. Prof ,**

**Department of MCA, RNSIT**

**© MCA- 2019-20**

**SEMESTER – I CREDITS –02**

Subject Code: 18MCA16 CIE Marks: 40

SEE Hours: 3 SEE Marks: 60

Number of Lecture Hours/Week: 03 Hours Laboratory and 01 Hour Tutorial

**Course objectives:** This course (18MCA16) will enable student to:

OB1: Use the knowledge of C++ Editors and demonstrate Modularity concepts using different types of calling function.

OB2: Formulate the function templates and use the concept of inline functions.

OB3: Implement the concept of friend function, friend class and static polymorphism concept using function and operator overloading.

OB4: Demonstrate the concept of constructors, destructors and Inheritance using runtime polymorphism with virtual functions.

OB5: Apply Exception handling; create files with C++ Input/output Streams and use Standard template library.

**Course Outcomes:** At the end of the course, students will be able to:

CO1: Differentiate the effect of different types of calling a function with C++ programs.

CO2: Design the function template and implement the inline function.

CO3: Apply the static polymorphism concept using friend class, friend function and

operator overloading

CO4: Write programs to demonstrate the run-time polymorphism with constructor,

destructor and virtual functions

CO5: Design an application to handle errors with exceptions and store data in the files.

**PART -A**

**=================================================================**

1. **Write a C++ program to find the sum for the given variables using function with default arguments.**

**//Program:** Demonstrate Default Argument

#include <iostream>

using namespace std;

void sum (int a =2, int b=3, int c=4, int d=5)

{

int res;

res = a + b + c + d;

cout << "Sum =" << res;

}

int main ( )

{

int a, b, c, d;

cout << "\n Enter 4 Nos : ";

cin >> a >>b >>c >>d;

cout<<"-- Output with 1 argument-- "<<endl;

sum (a) ; // 3 values are default

cout<<"-- Output with 2 arguments-- "<<endl;

sum (a, b) ; // 2 values are default

cout<<"-- Output with 3 arguments-- "<<endl;

sum (a, b, c) ; // 1 value is default

return 0;

}

1. **Write a C++ program to swap the values of two variables and demonstrates a function using call by value.**

***//Program: Call-by-Value***

#include <iostream>

using namespace std;

**void swap (int , int );** *// function prototype*

**int main ()**

**{**

int a, b;

*cout<<"\*\*\*\*\*\*\*\* SWAPPING USING CALL BY VALUE \*\*\*\*\*\*\n"<<endl;*

cout << "Input a = ";

cin >> a;

cout << "Input b = ";

cin >> b;

*cout << "\*\*\*\* Output Before swap \*\*\*\*"<< endl;*

cout << " a = " << a << endl;

cout << " b = " << b << endl;

**swap (a, b);**  *// calling a function as: Call-by-value*

*cout << "\*\*\*\* Output After swap \*\*\*\*"<< endl;*

cout << " a = " << a << endl;

cout << " b = " << b << endl;

return 0;

**}**

**void swap (int a, int b) // function definition of swap for Call-by-value**

**{**

int temp;

temp = a;

a = b;

b = temp;

**}**

1. **Write a C++ program the swap the values of two variables and demonstrates a function using Call by reference using reference variable (&).**

**// Program: Call by reference using ‘&’ operator.**

#include <iostream>

using namespace std;

void swap (int &x, int &y)

{

int temp;

temp = x;

x = y;

y = temp;

}

int main ()

{

cout<<"Swapping Using Call By Reference Using(&)\n"<<endl;

int a = 100;

int b = 200;

cout << "\*\*\*\* Output Before swap \*\*\*\*"<< endl;

cout << " a = " << a << endl;

cout << " b = " << b << endl;

swap (a, b);

cout << "\*\*\*\* Output After swap \*\*\*\*"<< endl;

cout << " a = " << a << endl;

cout << " b = " << b << endl;

return 0;

}

1. **Write a C++ program the swap the values of two variables and demonstrates a function using Call by reference using pointer.**

**// Program: Call by reference using ‘&’ operator.**

#include <iostream>

using namespace std;

void swap (int \*x, int \*y)

{

int temp;

temp = \*x;

\*x = \*y;

\*y = temp;

}

int main ()

{

cout<<"Swapping Using Call By ReferenceUsingPointer(\*)\n"<<endl;

int a = 100;

int b = 200;

cout << "\*\*\*\* Output Before swap \*\*\*\*"<< endl;

cout << " a = " << a << endl;

cout << " b = " << b << endl;

swap (&a, &b);

cout << "\*\*\*\* Output After swap \*\*\*\*"<< endl;

cout << " a = " << a << endl;

cout << " b = " << b << endl;

return 0;

}

1. **Write a C++ program to swap the values of two dynamically allocated variables and release the memory after swapping. (use new & delete operators)**

#include <iostream>

using namespace std;

void swap (int \*x, int \*y)

{

int temp;

temp = \*x;

\*x = \*y;

\*y = temp;

}

int main ()

{

cout<<"Swapping Using Dynamic Variable\n"<<endl;

int \*a;

int \*b;

a = new int (50);

b = new int (100);

cout << "\*\*\*\* Output Before swap \*\*\*\*"<< endl;

cout << " a = " << \*a << endl;

cout << " b = " << \*b << endl;

77

swap (a, b);

cout << "\*\*\*\* Output After swap \*\*\*\*"<< endl;

cout << " a = " << \*a << endl;

cout << " b = " << \*b << endl;

return 0;

}

**6. Write a program to find the largest, smallest & second largest of three numbers. (Use inline function MAX and MIN to find largest & smallest of 2 numbers)**

#include<iostream>

using namespace std;

inline int MAX (int a, int b)

{

return (a>b) ? a : b;

}

inline int MIN (int a, int b)

{

return (a<b) ? a : b;

}

int main()

{

cout<<" \*\*\*\*\*\*\*\* Demonstrate INLINE Function \*\*\*\*\*\*\*";

int a, b, c, large, small, secLargest;

cout<<"\n Enter values for : a, b and c :";

cin>>a>>b>>c;

large = MAX (a, MAX (b, c) );

small = MIN (a, MIN (b, c) );

cout<<"\n Largest number :"<< large <<endl;

cout<<"\n Smallest number :"<< small << endl;

secLargest = (a+b+c) -large - small;

cout<<"\n Second largest :"<<secLargest;

return 0;

}

|  |
| --- |
| **7.Write a program to calculate the volume of different geometric shapes like cube, cylinder and sphere and hence implement the concept of Function Overloading.** |

#include<iostream>

using namespace std;

float volume ( float, int);

float volume (float);

int volume (int);

int main()

{

float cRadius, sRadius, height;

int side;

cout <<"Enter cylinder Details:" << endl ;

cout <<"Cylinder Radius and height= ";

cin >> cRadius >>height;

cout<<endl<< "Enter Cube Details:";

cout <<"Cube Side = " ;

cin>>side;

cout<<endl<< "Enter Sphere Details:";

cout <<"Sphere Radius = " ;

cin>>sRadius;

cout << "Cube Volume = "<< volume(side);

cout << "Cylinder Volume = "<< volume(cRadius, height);

cout << "Sphere Volume = "<< volume(sRadius);

return 0;

}

float volume (float rad, int height)

{

return ( 3.14 \* rad \* rad \* height );

}

float volume ( float rad )

{

return ( (4/3.0) \* 3.14 \* rad \* rad \* rad);

}

int volume ( int side )

{

return (side \* side \* side);

}

8. **Write a C++ program to create a template function for Bubble Sort and demonstrate sorting of integers and doubles.**

#include<iostream>

using namespace std;

template <class T > void bubble(T a[], int n)

{

int i,j;

T temp;

for(i=1;i<n;i++)

{

for(j=0;j<n-i;j++)

{

if(a[j]>=a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

int main()

{

int intarr[10],i,n,m;

double dbarr[10];

cout<<”Enter size of an integer array”;

cin>>n;

cout<<”enter elements for integer array”;

for(i=0;i<n;i++)

cin>>intarr[i];

cout<<”Enter size of an double array”;

cin>>m;

cout<<”enter elements for double array”;

for(i=0;i<m;i++)

cin>>dbarr[i];

bubble(intarr,n);

bubble(dbarr,m);

cout<<”Sorted integer array elements”<<endl;

for(i=0;i<n;i++)

cout<<intarr[i]<<endl;

cout<<”Sorted double array elements”<<endl;

for(i=0;i<n;i++)

cout<<dbarr[i]<<endl;

return 0;

}

**PART B**

--------------------------------------------------------------

1. Define a STUDENT class with USN, Name, and Marks in 3 tests of a subject. Declare an array of 10 STUDENT objects. Using appropriate functions, find the average of the two better marks for each student. Print the USN, Name and the average marks of all the students.

#include<iostream>

using namespace std;

class Student

{

char USN[11], name[15];

float m1, m2, m3, avg;

public:

void readStudent();

void computeAvg();

void showStudent();

};

void Student:: readStudent()

{

cout<<"Enter the USN:";

cin>>USN;

cout<<"Enter the name:";

cin>>name;

cout<<"Enter marks of test1, test2 and test3 "<<endl;

cin>>m1 >> m2 >> m3;

}

void Student::computeAvg()

{

float small;

small =((m1<=m2)?((m1<=m3)?m1:m3):((m2<=m3)?m2:m3));

avg=(m1+m2+m3-small)/2;

}

void Student:: showStudent()

{

cout << USN<< "\t" ;

cout << name << "\t";

cout << m1 << "\t" << m2<< "\t" << m3 << "\t" ;

cout<< avg << endl;

}

int main()

{

Student st [10];

int n;

cout<<"\*\*\*\*\*\*\*\*\*\*STUDENT INFORMATION\*\*\*\*\*\*\*\*\*\*\*\*"<< endl;

cout<<"Enter the number of students:";

cin>>n;

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

{

cout<<endl<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " <<endl;

cout<<"Enter Details of student:"<< i + 1<<endl<<endl;

st[i].readStudent();

}

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

st[i].computeAvg();

cout<<"STUDENT DETAILS WITH AVG MARKS " <<endl;

cout<<"----------------------------------"<<endl;

cout<<"USN: \t NAME \t Test1 \t Test2 \t Test3 \t AVG" << endl;

cout<<"---------------------------------"<<endl;

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

st[i].showStudent();

cout << "---------------------------------" << endl;

return 0;

}

1. Write a C++ program to create a class called COMPLEX and implement the following overloading functions ADD that return a complex number:
   1. ADD (a, s2) – where “a‟ is an integer (real part) and s2 is a complex number
   2. ADD (s1, s2) – where s1 and s2 are complex numbers

#include<iostream>

using namespace std;

class COMPLEX

{

**private:**

int real, img;

**public:**

COMPLEX Add (int, COMPLEX);

COMPLEX Add (COMPLEX, COMPLEX);

void read();

void show();

};

COMPLEX COMPLEX : : Add(int n, COMPLEX s2)

{

COMPLEX t;

t.real = n + s2.real;

t. img = s2.img;

return t;

}

COMPLEX COMPLEX : : Add(COMPLEX s1, COMPLEX s2)

{

COMPLEX t;

t.real = s1.real + s2.real;

t. img = s1. img + s2. img;

return t;

}

void COMPLEX : : read()

{

cout<< "Real part = ";

cin>>real;

cout<<"Imaginary part = ";

cin>> img;

}

void COMPLEX : : show()

{

cout << real << " + " << img << "i" << endl;

}

int main( )

{

COMPLEX s1, s2, s3;

int a;

cout <<"Enter Complex Number and store in s2:" <<endl;

s2.read( );

cout <<"Read Integer Number To Add To s2 = " ;

cin >> a;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl << endl ;

cout << endl << "Perform s1 = a + s2 using ADD function" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl << endl;

s1 = s1.Add(a, s2); // function returns an object and store in obj s1

cout <<"Complex Num s2 = ";

s2.show();

cout <<"Number to add = " << a << endl;

cout <<"-----------------------------------" << endl;

cout <<"Complex Result s1 = ";

s1.show( );

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl << endl ;

cout << endl << "Perform s3 = s1 + s2 using ADD function" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl << endl ;

s3 = s1.Add(s1, s2);

cout <<" s1 = ";

s1.show( );

cout <<" s2 = ";

s2.show( );

cout <<"-----------------------------------" << endl;

cout <<" s3 = ";

s3.show( );

return 0;

}

Output:

Enter Complex Number and store in s2:

Real part = 6

Imaginary part = 5

Read Integer Number To Add To s2 = 2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Perform s1 = a + s2 using ADD function

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Complex  Num  s2 = 6 + 5i

Number  to  add = 2

-----------------------------------

Complex Res  s1 = 8 + 5i

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Perform s3 = s1 + s2 using ADD function

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

  s1 = 8 + 5i

  s2 = 6 + 5i

-----------------------------------

  s3 = 14 + 10i

**3a. Friend functions and friend classes**

Write a program to define class name HUSBAND and WIFE that holds the income respectively. Calculate and display the total income of a family using Friend function.

#include<iostream>

using namespace std;

**class WIFE;** //prototype for class WIFE

**class HUSBAND**

**{**

**private:**

char name[10];

int age;

float salary;

**public:**

**void getdata()**

**{** cout << "Enter Name : ";

cin >> name;

cout << "Enter Age : ";

cin >> age;

cout << "Enter Salary : ";

cin >> salary;

**}**

**void display()**

**{** cout << "Name : " << name << endl;

cout << "Age : " << age << endl;

cout << "Salary : " << salary << endl;

**}**

**friend void totSal (HUSBAND, WIFE);**

**};**

**class WIFE**

**{**

**private:**

char name[10];

int age;

float salary;

**public :**

**void getdata()**

**{** cout << "Enter Name : " ;

cin >> name;

cout << "Enter Age : ";

cin >> age;

cout << "Enter Salary : ";

cin >> salary;

**}**

**void display()**

**{**

cout << "Name : " << name << endl;

cout << "Age : " << age << endl;

cout << "Salary : " << salary << endl;

**}**

**friend void totSal (HUSBAND, WIFE);**

**};**

**void totSal (HUSBAND H,WIFE W)**

**{**

cout<< "Husband salary = " << H.salary << endl;

cout<< "Wife salary = " << W.salary << endl;

cout<< "Family Salary = " << H.salary + W.salary << endl;

**}**

**int main()**

**{**

cout<<"Program To Find The Total Salary of the Family"<<endl;

**HUSBAND Hus;**

**WIFE Wf;**

cout << endl << "Enter Husband Details : " << endl;

**Hus.getdata();**

cout << endl << "Enter Wife Details : " << endl;

**Wf.getdata();**

cout << endl << "Husband Details : " << endl;

**Hus.display();**

cout << endl <<"Wife Detials : " << endl;

**Wf.display();**

cout << endl <<"-Total Salary of the Family is-"<<endl;

**totSal(Hus, Wf);**

**return 0;**

**}**

|  |
| --- |
| **3b. Friend functions and friend classes:**  Write a program to accept the student detail such as name and 3 different marks by get\_data() method and display the name and average of marks using display() method. Define a friend class for calculating the average of marks using the method mark\_ avg(). |

include<iostream>

using namespace std;

**class Student**

**{**

**private:**

char name[15];

float m1,m2,m3,avg;

**public:**

void getData();

friend class AVGMARKS;

void showData();

**};**

**class AVGMARKS**

**{**

**public:**

int getAvg(Student t)

{

t.avg = (t.m1 + t.m2 + t.m3) / 3.0;

return t.avg;

}

**};**

**void Student:: getData()**

**{**

cout << "Enter Student name : " ;

cin >> name;

cout << "Enter test marks of 3 Subjects :" << endl;

cout << "Test1 = " ;

cin >> m1;

cout << "Test2 = " ;

cin >> m2;

cout << "Test3 = " ;

cin >> m3;

**}**

**void Student:: showData()**

**{**

cout<<"--------------------------------------"<<endl;

cout<<name<<" Details are:"<<endl;

cout<<"--------------------------------------"<<endl;

cout<<"Name:"<<name<<endl;

cout<<"Test1:"<<m1<<endl;

cout<<"Test2:"<<m2<<endl;

cout<<"Test3:"<<m3<<endl;

cout<<"---------------------------------------\n";

**}**

**int main()**

**{**

**Student s1;**

**AVGMARKS ob;**

**s1.getData();**

**s1.showData();**

cout<< "Avgerage Marks = " ;

cout << **ob.getAvg(s1)** ;

return 0;

**}**

4.Create a class called MATRIX using two-dimensional array of integers. Implement the following operations by overloading the operator == which checks the compatibility of two matrices to be added and subtracted. Perform the addition and subtraction by overloading the + and – operators respectively. Display the results by overloading the operator <<. **If (m1== m2)** then **m3 = m1 + m2** and **m4 = m1 - m2** else display error.

#include <iostream>

using namespace std;

class MATRIX

{

private:

int r, c, i, j;

int mat[10][10];

public:

void read\_order();

void read\_matrix();

int operator == (MATRIX);

MATRIX operator + (MATRIX);

MATRIX operator - (MATRIX);

friend ostream& operator <<(ostream&, MATRIX);

};

void MATRIX:: read\_order()

{

cout<<"\n Enter the number of rows and column:";

cin>>r>>c;

}

void MATRIX :: read\_matrix()

{

cout<<"\n enter the matrix elements:";

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

for ( int j = 0; j<c; j++)

cin >> mat[i][j];

}

int MATRIX :: operator == (MATRIX m2)

{

if ( ( r == m2.r ) && ( c == m2.c ) )

return 1;

else

return 0;

}

ostream& operator<<(ostream& out, MATRIX m)

{

for ( int i = 0; i < m.r; i++)

{

for ( int j = 0; j < m.c ; j++){

out<< m.mat[i][j] << " " ;

}

out<<endl;

}

return out;

}

MATRIX MATRIX :: operator + (MATRIX m2)

{

MATRIX temp;

cout << "Performing Addition of two Matrix :"<<endl;

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

for ( int j = 0 ; j < c ; j++)

temp.mat[i][j] = this->mat[i][j] + m2.mat[i][j];

temp.r=r;

temp.c=c;

return temp;

}

MATRIX MATRIX :: operator - (MATRIX m2)

{

MATRIX temp;

cout << "Performing Subtraction of two Matrix :"<<endl;

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

for ( int j = 0; j < c ; j++)

temp.mat[i][j] = this->mat[i][j] - m2.mat[i][j];

temp.r=r;

temp.c=c;

return temp;

}

int main()

{

MATRIX m1,m2,m3,m4;

cout<<"first matrix";

m1.read\_order();

cout<<"second matrix";

m2.read\_order();

if(m1==m2)

{

m1.read\_matrix();

m2.read\_matrix();

cout<<endl<<"First Matrix :"<<endl;

cout<<m1;

cout<<endl<<"Second Matrix :"<<endl;

cout<<m2<<endl;

cout<<endl<<"Matrix Addition result:"<<endl;

m3 = m1 + m2;

cout<<m3;

cout<<endl<<"Matrix Subtraction result:"<<endl;

m4 = m1 - m2;

cout<<m4<<endl;

}

else

cout<<"addition/subtraction is not possible";

return 0;

}

|  |
| --- |
| 5.Write a program to create an HUMAN class with features Head, Legs, Hands.(NOTE:Head, Legs and Hands are of integer/float types)   * 1. Create an object HUMAN1 using default constructor. (Default features to have 1 Head, 2 Legs and 2 Hands)   2. Create an object HUMAN2 with customized inputs using Parameterized Constructor   3. Create an object HUMAN3 using existing object HUMAN1 (Copy Constructor).   4. All Humans die after their lifetime. (Destructor) |

#include<iostream>

#include<string.h>

**using namespace std;**

**class HUMAN**

**{**

**private:**

char name[30];

int head;

int legs;

int hands;

**public:**

**HUMAN()**

**{**

strcpy(name , "\n--man1--\n");

head = 1;

legs = 2;

hands = 2;

**}**

**HUMAN (int head, int legs, int hands)**

**{**

strcpy(name , "\n--man2--\n");

this->head = head;

this->legs = legs;

this->hands = hands;

**}**

**HUMAN ( HUMAN &man)**

**{**

strcpy(name , "\n--man3--\n");

head = man.head;

legs = man.legs;

hands = man.hands;

**}**

**~HUMAN()**

**{**

cout << name << " is killed "<<endl;

getch();

**}**

**void show()**

**{**

cout << "head=" << head << endl << "legs=" << legs <<endl;

cout << "hands=" << hands << endl;

**}**

**};**

**int main()**

**{**

**HUMAN** **man1, man2(3,4,5), man3 = man1;**

cout << "Human1 values:"<< endl;

cout << "--------------"<< endl;

**man1.show();**

cout << "Human2 values:" << endl;

cout << "--------------" << endl;

**man2.show();**

cout << "Human3 values:" <<endl;

cout <<"--------------" <<endl;

**man3.show();**

**return 0;**

**}**

|  |
| --- |
| 1. Demonstrate Simple Inheritance concept by creating a base class FATHER with data members FirstName, SurName, DOB and BankBalance and creating a derived class SON, which inherits SurName and BankBalance feature from base class but provides its own feature FirstName and DOB. Create and initialize F1 and S1 objects with appropriate constructors and display the Father & Son details. *(Hint : While creating S1 object, call Father base class parameterized constructor through derived class by sending values)* |

#include<iostream>

#include<string.h>

using namespace std;

**class FATHER**

**{**

**private:**

char fname[20];

char DOB [20];

**protected:**

float bal;

char surName[20];

**public:**

**FATHER () { }**

**FATHER (char \*fname, char \*surName, char \*fdob, float bal )**

**{**

strcpy(this->fname, fname);

strcpy(this->surName, surName);

strcpy(DOB, fdob);

this->bal = bal;

**}**

**void showFather()**

**{**

cout<<endl<<"\*\*\*\*\*\*\* FATHERS DETAILS \*\*\*\*\*\*\*\*"<<endl;

cout<<"Father name = "<< fname <<endl;

cout<<"Surname = "<< surName <<endl;

cout<<"DOB = "<< DOB <<endl;

**}**

**};**

**class SON : public FATHER**

**{**

**private:**

char name[20];

char DOB [10];

**public:**

**SON(char \*fname, char \*surName, char \*fdob, float bal) : FATHER(fname, surName, fdob, bal)**

**{**

cout<<endl<<"-----Enter Son Details-------"<<endl;

cout<<"Son NAme=";

cin>>name;

cout<<"DOB = ";

cin>>DOB;

**}**

**void** **showSon()**

**{**

cout<<"Son Name ="<<name<<endl;

cout<<"Son Dob ="<<DOB<<endl;

cout<<"Surname ="<<surName<<endl;

cout<<"Family BAnk Balance ="<<bal<<endl;

**}**

**};**

**int main()**

**{**

char fName[10], surName[10], fDOB[10];

float bal;

cout<<" \*\*\*\*\*\* Input FATHER Details\*\*\*\*\*\*"<<endl;

cout<< " Father name:"<<endl;

cin>>fName;

cout<<"Surname:"<<endl;

cin>>surName;

cout<<"Father DOB :"<<endl;

cin>>fDOB;

cout<<"Father Bank balance :"<<endl;

cin>>bal;

**SON s1 (fName, surName, fDOB, bal);**

**s1.showFather();**

**s1.showSon();**

return 0;

**}**

1. Create an abstract base class EMPLOYEE with data members: Name, EmpID and BasicSal and a pure virtual function Cal\_Sal().Create two derived classes MANAGER (with data members: DA and HRA) and SALESMAN (with data members: DA, HRA and TA). Write appropriate constructors and member functions to initialize the data, read and write the data and to calculate the net salary. The main() function should create array of base class pointers/references to invoke overridden functions and hence to implement run-time polymorphism

#include<iostream>

using namespace std;

class EMPLOYEE

{

protected:

char Name[50], EmpID[10];

double BasicSal, netSal;

public:

virtual void cal\_sal() = 0;

};

class MANAGER: public EMPLOYEE

{

private:

double DA, HRA;

public:

MANAGER()

{

cout << "\nEnter Manager Details"<<endl;

cout<<"Name: ";

cin >> Name;

cout << " Employee ID: ";

cin >> EmpID;

cout << "Basic salary: ";

cin >> BasicSal;

}

void cal\_sal()

{

DA = BasicSal \* 0.10;

HRA = BasicSal \* 0.20;

netSal = BasicSal + DA + HRA;

cout<<"Manager \t"<<EmpID<<"\t"<<Name<<"\t";

cout<<BasicSal<<"\t"<<DA<< "\t"<<HRA<<"\t\t"<<netSal <<endl;

}

};

class SALESMAN : public EMPLOYEE

{

private:

double DA, HRA, TA;

public:

SALESMAN()

{

cout << "\nEnter Salesman Details"<<endl;

cout<<"Name: ";

cin >> Name;

cout << "Employee ID: ";

cin >> EmpID;

cout << "Basic salary: ";

cin >> BasicSal;

}

void cal\_sal()

{

DA = BasicSal \* 0.10;

HRA = BasicSal \* 0.20;

TA = BasicSal \* 0.30;

netSal = BasicSal + DA + HRA + TA;

cout<<"Salesman \t"<<EmpID<<"\t"<<Name<<"\t";

cout<<BasicSal<<"\t"<<DA<< "\t"<<HRA<<"\t"<<TA<< "\t"<<netSal <<endl;

}

};

int main()

{

EMPLOYEE \*emp[2];

emp[0] = new MANAGER();

emp[1] = new SALESMAN();

cout<<"\n\*\*\*\*\*\* DETAILS OF Employee \*\*\*\*\*\*\*\n" ;

cout<<" Post \t\t EmpID \t Name \t Basic \t DA \t HRA \t TA \t NetSalary"<<endl;

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

emp[0]->cal\_sal();

emp[1]->cal\_sal();

return 0;

}

1. Write a program to implement FILE I/O operations on characters. I/O operations includes inputting a string, Calculating length of the string, Storing the string in a file, fetching the stored characters from it, etc

#include <fstream>

#include <iostream>

void writeToFile (ofstream &outfile)

{

char str[50];

cout << "Read string from Keyboard : " << endl;

cin.getline(str, 50);

// Find length of string

for(int len=0; str[len]!= '\0' ; len++);

cout << "Length of string = " << len << endl;

// write input string into the file.

outfile << str << endl;

cout<< “String success fully stored in file”<<endl;

}

void readFromFile (ifstream &infile)

{

char str[50];

cout << "Reading from the file to string str.." << endl;

while(!infile.eof() )

{

infile >> str;

cout << str << " "; // write the string data on the screen.

}

}

void main ()

{

char str[50];

ofstream outfile; //create output file (Write mode)

outfile.open("data.txt");

writeToFile(outfile); // write string data to file

outfile.close(); // close the opened file.

ifstream infile; //create output file (Write mode)

infile.open("data.txt"); // Read string data from file

readFromFile(infile);

infile.close();

}

|  |
| --- |
| 1. Write a program to implement Exception Handling with minimum 5 exceptions Classes including two built-in exceptions. |

#include <iostream.h>

#include<conio.h>

**class BagException**

**{**

**public:**

char\* what()

{

return " Bag weight is exceeding 40 kg.. Not allowed ";

}

**};**

**class AgeException**

**{**

**public:**

char\* what()

{

return " Age is Less than 10 years..Not allowed to travel";

}

**};**

**class LuggageException**

**{**

**public:**

char\* what()

{

return " No. of Luggage are exceeding 10..Not allowed";

}

**};**

**void main()**

**{**

int bagWeight, age, luggageCount, seatNo;

float cost;

cout<<"Enter Bag weight :"<<endl;

cin>>bagWeight;

cout<<"Enter Passsenger Age :"<<endl;

cin>>age;

cout<<"Enter number of Luggage Passenger has :"<<endl;

cin>>luggageCount;

cout<<"Enter Seat No :"<<endl;

cin>>seatNo;

cout<<"Enter Flight Cost Paid:"<<endl;

cin>>cost;

**try**

**{**

if(bagWeight>40)

throw BagException(); //throw exception of type **BagException**

if(age<10)

throw AgeException(); //throw exception of type **AgeException**

if(luggageCount>10)

throw LuggageException(); //throw exception of type **LuggageException**

if(seatNo < 5)

throw seatNo; //throw exception of type **int**

if(cost < 0)

throw cost; //throw exception of type **float**

cout<<"Passenger can board the Plane ...";

**}**

**catch(BagException e)**

**{**

cout<<"\nCaught an exception : "<<endl;

cout<<e.what()<<endl;

**}**

**catch(AgeException e)**

**{**

cout<<"\nCaught an exception : "<<endl;

cout<<e.what()<<endl;

**}**

**catch(LuggageException e)**

**{**

cout<<"\nCaught an exception : "<<endl;

cout<<e.what()<<endl;

**}**

**catch(int seatNo)**

**{**

cout<<"\nCaught an exception :"<<endl;

cout<<"Seat no below -"<< seatNo <<" are a reserved....”<<endl

**}**

**catch(float price)**

**{**

cout<<"\nCaught an exception :"<<endl;

cout<<"Cost :” << price << “ is less than zero and Invalid....”<<endl

**}**

cout<<"\n End";

**}**

**OUTPUT**

|  |
| --- |
| **10) a)** Write a C++ program to concatenate 2 strings using STL string class function. |

#include <iostream>

using namespace std;

int main()

{

string s1, s2, result;

cout << "Enter string s1: ";

getline (cin, s1);

cout << "Enter string s2: ";

getline (cin, s2);

result = s1 + s2;

cout << "Resultant String = "<< result;

return 0;

}

**Output:**

|  |
| --- |
| Write a simple C++ program to store and display integer elements using STL Vector class |

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

int n, element;

cout << "Size of Vector=" << v.size() << endl;

//Putting values into the vector

cout << "Enter No of elemnts to add :" << endl;

cin>>n;

cout << "Enter "<< n <<" Elemnts :" << endl;

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

{

cin >> element;

v.push\_back(element);

}

//Size after adding values

cout << "Size of Vector=" << v.size() << endl;

//Display the contents of vector

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

{

cout << v[i] << " ";

}

}