SATHYABAMA UNIVERSITY

(Established under Section 3, UGC Act 1956)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCSX4003 - PROGRAMMING IN C++ LAB MANUAL

List of Experiments

1. Write a program to calculate final velocity using the formula: v=u + a*t, with initial velocity, acceleration and time as input.

2. Write a program to swap two characters of different data types using function overloading concept.

3. Write a program to change the sign of an operands using unary operator overloading concept.

4.Write a program to add two complex numbers using binary operator overloading concept.

5.Write a program to find mean value of two integers using friend function concept.

6. Write a program to multiply and divide two different data type using inline function concept.

7.Write a program to Implement Matrix class with dynamic memory allocation and necessary methods. Give proper constructor, destructor, copy constructor, and overloading of assignment operator.

8.Write a program to enter the sale value and print the agent's commission using single inheritance.

9.Write a program to enter salary and output income tax and net salary using multiple inheritance concept.

10. Write a program to enter the unit reading and output the customer's telephone bill using hierarchical inheritance.

11.Write a program to find the grade of the students based on academic marks and sports using multilevel inheritance.

12.Write a program having student as an abstract class and create many derived classes such as Engineering, Medical etc from student's class. Create their objects and process them.

13. Write a program to count the words and characters in given text using virtual function.

14. Write a program to calculate net pay of employee using virtual base class concept.

15. Write a program to calculate division of two number with a try block to detect and throw an exception if the condition" divide –by-zero" occurs.

1. CALCULATING FINAL VELOCITY

AIM:

To Write C++ program to calculate final velocity using the formula: v=u + a*t, with initial velocity, acceleration and time as input.

ALGORITHM:

Step 1: Declare the class, velocity with the variables, u, a and t as integer type.
Step 2: Define a function, getdata() to get the input from the user.
Step 3: Define a function, calculate() to find final velocity v=u+a*t and display the result.
Step 4: Call the getdata() and calculate() functions from the main function by creating an instance of the class velocity.

2. FUNCTION OVERLOADING

AIM:

To swap two data items of different data types using function overloading concept .

ALGORITHM:

Step 1: Read two integer values
Step 2: Read two character values
Step 3: Display the values before swap
Step 4: Call the function swap() for swapping integers
Step 5: Call the function swap() for swapping characters
Step 6: Display the values after swap

Algorithm for swap(int &x,int &y)
Step 1: int t=x
Step 2: x=y
Step 3: y=t

Algorithm for swap(char &x,char &y)
Step 1: char t=x
Step 2: x=y
Step 3: y=t

3. UNARY OPERATOR OVERLOADING

AIM:

To change the sign of an operand using unary operator overloading concept.

ALGORITHM:

Step 1: Create a class and declare the required data members.
Step 2: Read two integers x and y
Step 3: In the main function invoke the code for overloading the unary operator (unary minus)
Step 4: Display the result.

Algorithm for operator-()
Step 1: x=-x
Step 2: y=-y

4. COMPLEX NUMBER ADDITION USING BINARY OPERATOR OVERLOADING

AIM:

To add two complex numbers using binary operator overloading concept.

ALGORITHM:

Step 1: Create a class and declare the required data members.
Step 2: Read the real value ‘x’ and imaginary value ‘y’ .
Step 3: In the main function invoke a code for overloading the binary operator (+ OR -)
Step 4: Display the result.

Algorithm for operator+(complex z)
Step 1: Declare the complex variabe ‘t’
Step 2: t.x=x+z.x
Step 3: t.y=y+z.y
Step 4: return t

5. FRIEND FUNCTION

AIM:

To find the mean value of two integers using friend function concept.

ALGORITHM:

Step 1: Create a class ‘abc’ and declare the required data members.
Step 2: Read the values ‘ x’ and ‘ y’.
Step 3: Call the friend function mean() /*(Pass the object of the class as argument)*/ from the main function

Algorithm for mean(abc ob)
Step 1: Average=(ob.x+ob.y)/2
Step 2: Display the result.

6. INLINE FUNCTION

AIM:

To write a C++ program to multiply and divide two different data types using inline function concept.

ALGORITHM:

Step 1: Declare the class and declare two data members ‘a’ as integer type and ‘b’ as float type.
Step 2: Define the functions, multiply() and divide() using the keyword inline.
Step 3: Create an instance and invoke the inline functions.

Algorithm for inline float mul(int x,float y)
Step 1: return x*y
Algorithm for inline float div(int x,float y)
Step 1: return y/x

7. DYNAMIC MEMORY ALLOCATION WITH CONSTRUCTORS, DESTRUCTORS AND OPERATOR OVERLOADING

AIM:

To implement matrix class with dynamic memory allocation by giving proper constructor, destructor, copy constructor, and overloading of assignment operator.

ALGORITHM:

Step 1: Create a class named matrix
Step 2: Declare the constructor, copyconstructor, assignment operator and destructor as public data members of the class.
Step 3: In the main function create the object of the class thereby the constructor gets automatically invoked and then create the second object to perform operator overloading and third object for copy constructor.
Step 3.1: Invoke the assignment operator by assigning object2=object1
Step 3.2: Invoke the copy constructor by copying object3=object2
Step 4: Display the result
Step 5: Finally the destructor is invoked automatically

Algorithm for the constructor matrix()

Step 1: Read the number of rows 'dx' and columns 'dy'
Step 2: Call allocarr() for dynamic memory allocation
Step 3: Read the matrix values
Step 3.1: For i = 0 to dx
Step 3.1.1: For j = 0 to dy
Step 3.1.1.1: Read *(*(a+i)+j)

Algorithm for allocarr()

Step 1: a=new int*[dx];
Step 2: For i = 0 to dx
Step 2.1: a[i]=new int[dy];

Algorithm for the copy constructor matrix(const matrix &m**)**

Step 1: dx=m.dx;
Step 2: dy=m.dy;
Step 3: For i = 0 to dx
Step 3.1: For j = 0 to dy
Step 3.1.1: a[i][j]=m.a[i][j]

Algorithm for overloading assignment operator
matrix & matrix::operator=(const matrix &m)

Step 1: dx=m.dx;
Step 2: dy=m.dy;
Step 3: For i = 0 to dx

Step 3.1: For j = 0 to dy
Step 3.1.1: a[i][j]=m.a[i][j]
Step 4: return *this

Algorithm for disp()
Step 1: For i = 0 to dx
Step 1.1: For j = 0 to dy
Step 1.1.1: Write *(*(a+i)+j)

Algorithm for the destructor ~ matrix ()
Step 1: For i = 0 to dy
Step 1.1: delete a[i]
Step 2: delete a

8. SINGLE INHERITANCE

AIM:

To enter the sale value and print the agent's commission using single inheritance.

ALGORITHM:

Step 1: Declare the class 'agency' and define the member function getdata() and get the salesprice as input.
Step 2: Declare the class 'agent' which is derived from 'agency' class and define the member function calc () to calculate the commission as 10% of salesprice and display the result.
Step 3: In the main (), create an instance of the derived class and invoke the getdata() and calc() member functions of the base class and derived class respectively.

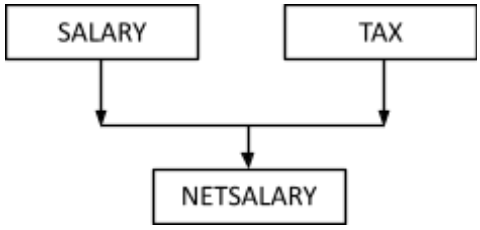
9. MULTIPLE INHERITANCE

AIM:

To read salary and calculate income tax and net salary using multiple inheritance concept.

ALGORITHM:

Step 1: Declare the class 'salary' with variables Basic, HRA and DA.
Step 2: Define the member function getsal() in the class 'salary' to find the gross=basic+hra+da.
Step 3: Declare the class ' tax' and find the tax amount as to be deducted in the member function deduct()
Step 4: Declare the class 'netsalary' which is publicly derived from classes salary and tax.
Step 5: Define the member function netpay() int the 'netsalary' class to deduce the tax from total.
Step 6: Create an instance for the derived class netsalary and invoke the member functions of both base classes and the derived class.



10. HIERARCHICAL INHERITANCE

AIM:

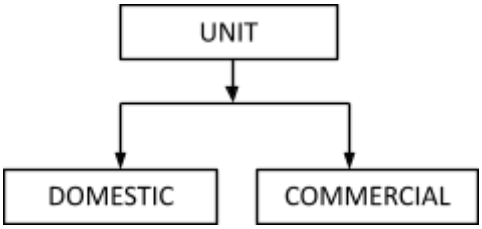
To write a c++ program to read the unit reading and calculate the customer's telephone bill using hierarchical inheritance.

ALGORITHM:

Step 1: Declare the class 'unitread'
Step 2: Define the member function getname() to get the name of the consumer and getunit() to get the number of units of electricity consumed, in the base class 'unitread'.
Step 3: Declare the class 'domestic' and calculate the bill amount as ₹3 per unit in the member function calc_domestic()

Step 4: Declare the class 'commercial' and calculate the bill amount as ₹6 per unit in the member function calc_commercial()

Step 5: Create instances for the derived classes and display the results.



11. MULTILEVEL INHERITANCE

AIM:

To write a c++ program to find the grade of students based on academic marks and sports using multilevel inheritance.

ALGORITHM:

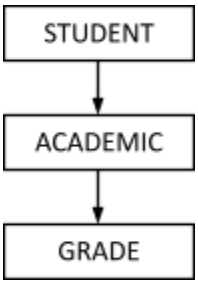
Step 1: Declare the class ' student' with variables rollno and name.

Step 2: Define the member function getdata() in the student class to get the rollno and name.

Step 3: Declare the class ' academic' which is derived publicly from student class and read the marks in member function getmarks() of the academic class and calculate the average.

Step 4: Declare the class 'grade' which is publicly derived from academic class and declare the member function calc() to calculate the grade of each student

Step 5: Create array of objects for the derived class 'grade' and calculate the grade for a set of students.



12. ABSTRACT CLASS

AIM:

To write a c++ program having student as an abstract class and create derived classes such as engineering, medical, arts and science from student class.

ALGORITHM:

Step 1: Declare the abstract class ' student' with variables name and percentage .

Step 2: Declare the classes 'engg ' , 'medical', 'arts' which are derived publicly from the abstract class 'student'

Step 3: In the derived classes set some eligibility criterial and display whether the student is eligible or not.

Step 4: Create array of objects for the derived classes separately and invoke the corresponding member functions

13. VIRTUAL FUNCTION

AIM:

To write a c++ program to count the words and characters in given text using virtual function.

ALGORITHM:

Step 1: Declare the base class ' mystring'

Step 2: Create the constructor for 'mystring' class and initialise a string and calculate string length.

Step 3: Create the virtual member function disp(), for the base class 'mystring' and display the string and its length

Step 4 : Declare the class 'char_ct' which is publicly derived from the class 'mystring'

Step 5: Create the constructor for the 'char_ct' class and calculate the number of characters in the given string

Step 6: Create the member function disp(), for the derived class 'char_ct' and display the number of characters

Step 7: Declare the class 'word_ct' which is publicly derived from the class 'mystring'

Step 8: Create the constructor for the 'word_ct' class and calculate the number of characters in the given string

Step 9: Create the member function disp(), for the derived class

'word_ct' and display the number of words.

Step 10: In main(), create object for base and derived classes. Create a base class pointer and invoke all the functions using base pointer.

14. VIRTUAL BASE CLASS

AIM:

To write a c++ program to calculate net pay of employee using virtual base class concept.

ALGORITHM:

Step 1: Declare the class ' employee' and define the member function getname() to get the employee name.

Step 2: Declare the class ' salary ' which is publicly derived from 'employee' with the keyword virtual. Define the member function getdetails() to get basic , HRA and DA and find the total.

Step 3: Declare the class ' taxcal' which is publicly derived from 'employee' with the keyword virtual. Find the tax amount to be deducted in the function deduct().

Step 4: Declare the class ' netsalary 'which is publicly derived from the classes salary and tax. Define the member function netpay() to deduce the tax from total.

Step 5: Create an instance for the derived class netsalary and invoke all functions.

15. EXCEPTION HANDLING

AIM:

To write c++ a program to calculate division of two number with a try block to detect and throw an exception if the condition" divide –by-zero" occurs.

ALGORITHM:

Step 1: Declare variables a,b,c.

Step 2: Read values for a,b,c,.

Step 3: Inside the try block check the condition.

Step 3.1: if(a-b!=0) then calculate the value of d and display.

Step 3.2: else throw the exception.

Step 4: Catch the exception and display the appropriate message.

******