**Q1. Write a C++ Program to Derive a class called employee2 from the employee class. This new class should add a type double data item called compensation and, also an enum type called period to indicate whether the employee is paid hourly, weekly, or monthly. For simplicity you can change the manager, scientist and labourer classes so they are derived from employee2 instead of employee. However, note that in many circumstances it might be more in the spirit of OOP to create a separate base class called compensation and three new class manager2, scientist2 and labour2 and use multiple inheritances to derive these classes from the original manager, scientist and labourer classes and from compensation. This way none of the original classes needs to be modified.**

#include <iostream>

#include <conio.h>

using namespace std;

const int LEN = 80; //maximum length of names

////////////////////////////////////////////////////////////////

class employee //employee class

{

private:

char name[LEN]; //employee name

unsigned long number; //employee number

public:

void getdata()

{

cout << "\n Enter last name : "; cin >> name;

cout << " Enter number :"; cin >> number;

}

void putdata() const

{

cout << "\n Name : " << name;

cout << "\n Number : " << number;

}

};

////////////////////////////////////////////////////////////////

class employee2 :public employee

{

private:

double compen;

enum paytype{ hourly, weakly, monthly };

paytype period;

public:

void getdata(void)

{

char a;

employee::getdata();

cout << "Enter compensation: ";

cin >> compen;

cout << "Enter payment period (h,w,m): ";

cin >> a;

switch (a)

{

case 'h':

period = hourly;

break;

case 'w':

period = weakly;

break;

case 'm':

period = monthly;

break;

}

}

void putdata(void) const

{

employee::putdata();

cout << "Employee compensation: " << compen << endl;

switch (period)

{

case hourly:

cout << "hourly" << endl;

break;

case weakly:

cout << "weakly" << endl;

break;

case monthly:

cout << "monthly" << endl;

break;

}

}

};

////////////////////////////////////////////////////////////////

class manager : public employee2 //management class

{

private:

char title[LEN]; //"vice-president" etc.

double dues; //golf club dues

public:

void getdata()

{

employee2::getdata();

cout << " Enter title : "; cin >> title;

cout << " Enter golf club dues : "; cin >> dues;

}

void putdata() const

{

employee2::putdata();

cout << "\n Title : " << title;

cout << "\n Golf club dues : " << dues;

}

};

////////////////////////////////////////////////////////////////

class scientist : public employee2 //scientist class

{

private:

int pubs; //number of publications

public:

void getdata()

{

employee2::getdata();

cout << " Enter number of pubs : "; cin >> pubs;

}

void putdata() const

{

employee2::putdata();

cout << "\n Number of publications : " << pubs;

}

};

////////////////////////////////////////////////////////////////

class laborer : public employee2 //laborer class

{};

////////////////////////////////////////////////////////////////

int main()

{

manager m1, m2;

scientist s1;

laborer l1;

cout << endl; //get data for several employees

cout << "\nEnter data for manager 1";

m1.getdata();

cout << "\nEnter data for manager 2";

m2.getdata();

cout << "\nEnter data for scientist 1";

s1.getdata();

cout << "\nEnter data for laborer 1";

l1.getdata();

//display data for several employees

cout << "\nData on manager 1";

m1.putdata();

cout << "\nData on manager 2";

m2.putdata();

cout << "\nData on scientist 1";

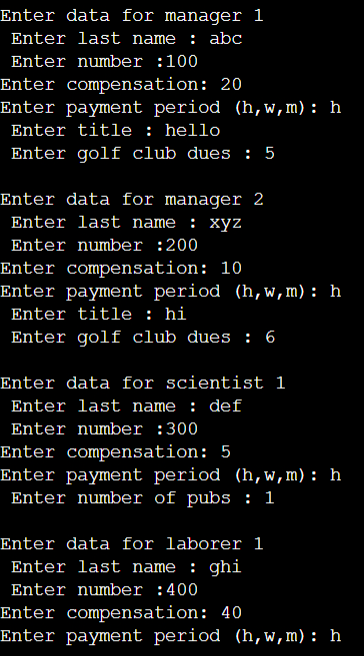
s1.putdata();

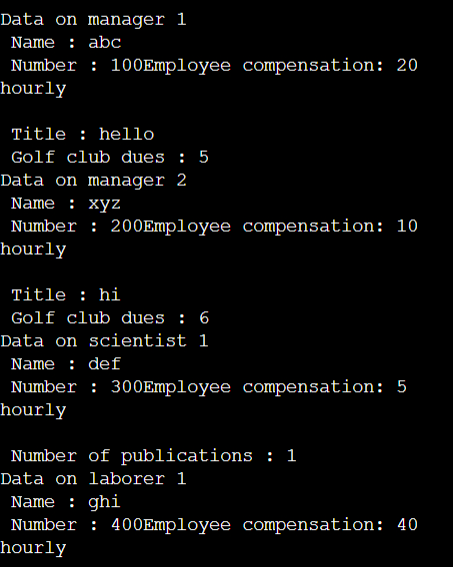
cout << "\nData on laborer 1";

l1.putdata();

cout << endl;

}





**Q2. Write a C++ program which start with the publication, book and tape classes. Add a base class sale that holds an array of three floats so that it can record the dollar sales of a particular publication for the last three months. Include a getdata( ) function to get three sales amounts from the user and a putdata( ) function to display the sales figures. After the book and tape classes so they are derived from both publication and sales. An object of class book or tape should input and output sales data along with its other data. Write a main ( ) function to create a book object and a tape object and exercise their input/output capabilities.**

#include <iostream>

#include <string>

#include <conio.h>

using namespace std;

class publication

{

private:

string title;

float price;

public:

void getdata(void)

{

string t;

float p;

cout << "Enter title of publication: ";

cin >> t;

cout << "Enter price of publication: ";

cin >> p;

title = t;

price = p;

}

void putdata(void)

{

cout << "Publication title: " << title << endl;

cout << "Publication price: " << price << endl;

}

};

class sales

{

private:

float s1, s2, s3;

public:

void getdata(void)

{

cout << "Enter month 1 sale: $";

cin >> s1;

cout << "Enter month 2 sale: $";

cin >> s2;

cout << "Enter month 3 sale: $";

cin >> s3;

}

void putdata(void)

{

cout << "Month 1 sale: $" << s1 << endl;

cout << "Month 2 sale: $" << s2 << endl;

cout << "Month 3 sale: $" << s3 << endl;

}

};

class book :public publication,public sales

{

private:

int pagecount;

public:

void getdata(void)

{

publication::getdata();

sales::getdata();

cout << "Enter Book Page Count: ";

cin >> pagecount;

}

void putdata(void)

{

publication::putdata();

sales::putdata();

cout << "Book page count: " << pagecount << endl;

}

};

class tape :public publication,public sales

{

private:

float ptime;

public:

void getdata(void)

{

publication::getdata();

sales::getdata();

cout << "Enter tap's playing time: ";

cin >> ptime;

}

void putdata(void)

{

publication::putdata();

sales::putdata();

cout << "Tap's playing time: " << ptime << endl;

}

};

int main()

{

book b;

tape t;

b.getdata();

t.getdata();

b.putdata();

t.putdata();

}



**Q4. Write a C++ Program to implement the binary tree by writing the definition a class tree. This class contains various attribute of a tree and various operation on the tree like insertion, deletion, traversing. In addition to this, the class drive three new classes complete binary tree, binary search tree, and full binary tree.**

#include <iostream>

using namespace std;

class BST

{

int data;

BST \*left, \*right;

public:

// Default constructor.

BST();

// Parameterized constructor.

BST(int);

// Insert function.

BST\* Insert(BST\*, int);

// Inorder traversal.

void Inorder(BST\*);

};

// Default Constructor definition.

BST ::BST()

: data(0)

, left(NULL)

, right(NULL)

{

}

// Parameterized Constructor definition.

BST ::BST(int value)

{

data = value;

left = right = NULL;

}

// Insert function definition.

BST\* BST ::Insert(BST\* root, int value)

{

if (!root)

{

// Insert the first node, if root is NULL.

return new BST(value);

}

// Insert data.

if (value > root->data)

{

// Insert right node data, if the 'value'

// to be inserted is greater than 'root' node data.

// Process right nodes.

root->right = Insert(root->right, value);

}

else

{

// Insert left node data, if the 'value'

// to be inserted is greater than 'root' node data.

// Process left nodes.

root->left = Insert(root->left, value);

}

// Return 'root' node, after insertion.

return root;

}

// Inorder traversal function.

// This gives data in sorted order.

void BST ::Inorder(BST\* root)

{

if (!root) {

return;

}

Inorder(root->left);

cout << root->data << endl;

Inorder(root->right);

}

// Driver code

int main()

{

BST b, \*root = NULL;

root = b.Insert(root, 50);

b.Insert(root, 30);

b.Insert(root, 20);

b.Insert(root, 40);

b.Insert(root, 70);

b.Insert(root, 60);

b.Insert(root, 80);

b.Inorder(root);

return 0;

}

