Assignment - 31 A Job Ready Bootcamp in C++, DSA and IOT MySirG

new and delete operator, Inheritance

1. Define a class Person with instance members name and age. Also define member

functions setName(), setAge(), getName(), getAge(). Now define class Employee by

inheriting Person class. In the Employee class define empid and salary as instance

members. Also define setEmpid, setSalary, getEmpid, getSalary.

#include <iostream>

#include <string.h>

using namespace std;

class Person

{

private:

char \*name;

int age;

public:

void setName(char name[]) { strcpy(this->name, name); }

void setAge(int age) { this->age = age; }

char \*getName() { return name; }

int getAge() { return age; }

};

class Employee : public Person

{

private:

int empId;

float salary;

public:

void setEmpId(int empId) { this->empId = empId; }

void setSalary(float salary) { this->salary = salary; }

int getEmpId() { return empId; }

float getSalary() { return salary; }

void displayData()

{

cout << "\nName - " << getName() << "\nAge - " << getAge() << "\nEmp Id - " << empId << "\nEmp Salary - " << salary << endl;

}

};

int main()

{

Employee e1;

e1.setName("Sachin Payasi");

e1.setAge(20);

e1.setEmpId(001);

e1.setSalary(54.34f);

e1.displayData();

return 0;

}

2. Write a C++ program to add two numbers using single inheritance. Accept these two

numbers from the user in base class and display the sum of these two numbers in

derived class.

#include <iostream>

using namespace std;

class Base

{

private:

float x, y;

public:

void acceptData()

{

cout << "Enter First Number: ";

cin >> x;

cout << "Enter Second Number: ";

cin >> y;

}

protected:

float getX() { return x; }

float getY() { return y; }

};

class Derived : public Base // single inheritance

{

private:

float sum;

public:

void doSum()

{

sum = getX() + getY();

}

float getSum() { return sum; }

};

int main()

{

Derived obj;

obj.acceptData();

obj.doSum();

cout << obj.getSum() << endl;

}

3. Write a C++ program to calculate the percentage of a student using multi-level

inheritance. Accept the marks of three subjects in base class. A class will be derived

from the above mentioned class which includes a function to find the total marks

obtained and another class derived from this class which calculates and displays the

percentage of students.

#include <iostream>

using namespace std;

class Student

{

private:

int sub1, sub2, sub3;

public:

void setSubData(int sub1, int sub2, int sub3)

{

this->sub1 = sub1;

this->sub2 = sub2;

this->sub3 = sub3;

}

int getSub1() { return sub1; }

int getSub2() { return sub2; }

int getSub3() { return sub3; }

};

class TotalMarks : public Student // single inheritance

{

private:

int total;

public:

void doTotal() { total = getSub1() + getSub2() + getSub3(); }

int getTotal() { return total; }

};

class Percentage : public TotalMarks // multilevel inheritance

{

private:

float percentage;

public:

void findPercentage() { percentage = getTotal() / 3.0f; } // (total \* 300)/ 100.0

float getPercentage() { return percentage; }

};

int main()

{

Percentage student1;

student1.setSubData(94, 100, 35); // 3 subjects marks

student1.doTotal();

student1.findPercentage();

cout<<student1.getPercentage();

return 0;

}

4. Write a C++ program to design a base class Person (name, address,

phone\_no). Derive a class Employee (eno, ename) from Person. Derive a

class Manager (designation, department name, basic-salary) from

Employee. Write a menu driven program to:

a. Accept all details of 'n' managers.

b. Display manager having highest salary

|  |  |
| --- | --- |
|  | #include <bits/stdc++.h> |
|  | using namespace std; |
|  | class Person |
|  | { |
|  | protected: // no need to made getter setter |
|  | char pName[20], pAddress[20]; |
|  | int phoneNo; |
|  | }; |
|  | class Employee : public Person // Single Inheritance |
|  | { |
|  | protected: |
|  | int eNo; |
|  | char eName[20]; |
|  | }; |
|  | class Manager : public Employee // Multilevel Inheritance |
|  | { |
|  | protected: |
|  | char desigNation[30], departName[30]; |
|  | float basicSalary; |
|  |  |
|  | public: |
|  | void acceptDetails() |
|  | { |
|  | cout << "\nEnter details of Manager"; |
|  | cout << "\n-------------------------"; |
|  | cout << "\nEnter Employee Number: "; |
|  | cin >> eNo; |
|  | cout << "\nEnter Name: "; |
|  | cin >> eName; |
|  | cout << "\nEnter Address: "; |
|  | cin >> pAddress; |
|  | cout << "\nEnter PhoneNo. "; |
|  | cin >> phoneNo; |
|  | cout << "\nEnter Designation: "; |
|  | cin >> desigNation; |
|  | cout << "\nEnter Department Name: "; |
|  | cin >> departName; |
|  | cout << "\nEnter Basic Salary: "; |
|  | cin >> basicSalary; |
|  | } |
|  | float getSalary() { return basicSalary; } |
|  | char \*getName() { return eName; } |
|  | }; |
|  | int main() |
|  | { |
|  | int i, totalManager, temp = INT\_MIN; |
|  | Manager man[100]; |
|  | cout << "\nHow many Manager you want enter? "; |
|  | cin >> totalManager; // Accepts details for 'n' manager |
|  |  |
|  | for (i = 0; i < totalManager; i++) |
|  | man[i].acceptDetails(); |
|  |  |
|  | int ind = -1; |
|  | for (i = 0; i < totalManager; i++) |
|  | { |
|  | if (temp < man[i].getSalary()) |
|  | { |
|  | temp = man[i].getSalary(); |
|  | ind = i; |
|  | } |
|  | } |
|  |  |
|  | cout<<"\nManager with highest salary is: "<<temp<<endl; |
|  | cout<<"\nAnd Manager Name is: "<<man[i].getName()<<endl; |
|  | return 0; |
|  | } |

5. Write a C++ program to define a base class Item (item-no, name, price).

Derive a class Discounted-Item (discount-percent). A customer purchases

'n' items. Display the item-wise bill and total amount using appropriate

format.

#include <bits/stdc++.h>

using namespace std;

class ItemDetails

{

protected:

char itemName[20];

int itemNo;

double itemPrice;

};

class discountItem : public ItemDetails

{

public:

float discountPercent;

float discountPrice;

void acceptDetails()

{

cout << "\nEnter item No. - ";

cin >> itemNo;

cout << "\nEnter Item Name: ";

cin >> itemName;

cout << "\nEnter Item Price: ";

cin >> itemPrice;

cout << "\nEnter discount Percent: ";

cin >> discountPercent;

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

discountPrice = itemPrice - ((itemPrice \* discountPercent) / 100.0f);

}

float getItemPrice() { return itemPrice; }

void displayBill()

{

cout << "\nItem Name: " << itemName;

cout << "\nItem No. " << itemNo;

cout << "\nItem Price: " << itemPrice;

cout << "\nDiscount Percent: " << discountPercent;

cout << "\nDiscount Price: " << discountPrice;

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

}

};

int main()

{

int i, noOfItem;

float totalDiscount = 0, totalPrice = 0;

discountItem item[100];

cout << "\nHow many item you want to purchase?";

cin >> noOfItem;

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

item[i].acceptDetails();

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

{

item[i].displayBill();

totalDiscount += item[i].getItemPrice() - item[i].discountPrice;

totalPrice += item[i].getItemPrice();

}

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

cout << "\nTotal Price: " << totalPrice;

cout << "\nTotal Discount: " << totalDiscount;

cout << "\nFinal Price: " << totalPrice - totalDiscount;

return 0;

}

6. Write a C++ program to demonstrate how a common friend function can

be used to exchange the private values of two classes. (Use call by

reference method).

#include <bits/stdc++.h>

using namespace std;

class B;

class A

{

protected:

int x;

public:

void setX(int x) { this->x = x; }

void showX() { cout << "X = " << x << endl; }

friend void swap(A &, B &);

};

class B

{

protected:

int y;

public:

void setY(int y) { this->y = y; }

void showY() { cout << "Y = " << y << endl; }

friend void swap(A &, B &);

};

void swap(A &a, B &b)

{

int temp = a.x;

a.x = b.y;

b.y = temp;

}

int main()

{

A a;

B b;

a.setX(5);

b.setY(4);

cout << "\nBefore swapping: ";

a.showX();

b.showY();

swap(a, b);

cout << "\nAfter swapping: ";

a.showX();

b.showY();

return 0;

}

7. Write class declarations and member function definitions for a C++ base

class to represent an Employee (emp-code, name).

Derive two classes as Fulltime (daily rate, number of days, salary) and

Parttime (number of working hours, hourly rate, salary).

Write a menu driven program to:

1. Accept the details of ‘n’ employees.

2. Display the details of ‘n’ employees.

3. Search a given Employee by emp-code.

#include <iostream>

using namespace std;

class Employee

{

protected:

int empCode;

char name[20];

public:

void acceptData()

{

cout << "\nEnter employee no. : ";

cin >> empCode;

cout << "\nEnter employee name : ";

cin >> name;

}

int getEmpid() { return empCode; }

};

class FullTime : public Employee

{

protected:

int noOfDays;

float salary, dailyRate;

float calcFullTimeEmpSalary()

{

salary = noOfDays \* dailyRate;

return salary;

}

public:

void acceptDataFullTimeEmp()

{

acceptData();

cout << "\nEnter no. of days: ";

cin >> noOfDays;

cout << "\nEnter daily rate: ";

cin >> dailyRate;

}

void showFullTimeEmpDetails()

{

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

cout << "\nEmployee Code: " << empCode;

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

cout << "\nSalary: " << calcFullTimeEmpSalary();

cout << "\nStatus:\tFullTime";

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

}

};

class PartTime : public Employee

{

private:

int workingHrs;

float hourlyRate, salary;

float calcPartTimeEmpSalary()

{

salary = workingHrs \* hourlyRate;

return salary;

}

public:

void acceptDataPartTimeEmp()

{

acceptData();

cout << "\nEnter working hours: ";

cin >> workingHrs;

cout << "\nEnter hourly rate: ";

cin >> hourlyRate;

}

void showPartTimeEmpDetails()

{

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

cout << "\nEmployee Code: " << empCode;

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

cout << "\nSalary: " << calcPartTimeEmpSalary();

cout << "\nStatus:\tPartTime";

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

}

};

int main()

{

int fullTimeEmp, partTimeEmp;

cout << "\nHow many full time emp you want enter? ";

cin >> fullTimeEmp;

FullTime f[fullTimeEmp];

cout << "\nHow many Part time emp you want enter? ";

cin >> partTimeEmp;

PartTime p[partTimeEmp];

int emp1 = 0, emp2 = 0;

do

{

int choice1, choice2;

cout << "\n\n1. Enter Record: ";

cout << "\n2. Display Record: ";

cout << "\n3. Search Record: ";

cout << "\n4. Quit: ";

cout << "\n\tEnter your choice: ";

cin >> choice1;

switch (choice1)

{

case 1:

cout << "\n\n1. FullTime Employee: ";

cout << "\n2. PartTime Employee: ";

cin >> choice2;

switch (choice2)

{

case 1:

f[emp1].acceptDataFullTimeEmp();

emp1++;

break;

case 2:

p[emp2].acceptDataPartTimeEmp();

emp2++;

break;

// defualt case

}

break;

case 2:

// show all emp details

// First - for full time employee

for (int i = 0; i < emp1; i++)

f[i].showFullTimeEmpDetails();

// Second - for Part time employee

for (int i = 0; i < emp2; i++)

p[i].showPartTimeEmpDetails();

break;

case 3:

int searchId;

cout << "\nEnter emp id: ";

cin >> searchId;

for (int i = 0; i < emp1; i++)

{

if (f[i].getEmpid() == searchId)

{

f[i].showFullTimeEmpDetails();

break;

}

}

// Second - for Part time employee

for (int i = 0; i < emp2; i++)

if (p[i].getEmpid() == searchId)

{

p[i].showPartTimeEmpDetails();

break;

}

break;

case 4:

exit(0);

// default case

}

} while (emp1 <= fullTimeEmp || emp2 <= partTimeEmp);

return 0;

}

8 - In a bank, different customers have savings account. Some customers may

have taken a loan from the bank. So bank always maintain information about

bank depositors and borrowers.

Design a Base class Customer (name, phone-number). Derive a class

Depositor(accno, balance) from Customer.

Again, derive a class Borrower (loan-no, loan-amt) from Depositor.

Write necessary member functions to read and display the details of ‘n’

customers.

#include <iostream>

using namespace std;

class Customer

{

protected:

char custName[20];

long mobileNo;

void acceptData()

{

cout << "\n\nEnter Customer Name: ";

cin >> custName;

cout << "\nEnter mobile no. : ";

cin >> mobileNo;

}

public:

void showCustomerDetails()

{

cout << "\n\nDetails of Customer: ";

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

cout << "\nCustomer Name: " << custName;

cout << "\nCustomer Mobile No. : " << mobileNo;

}

};

class Dipositors : public Customer

{

protected:

long accountNo, balance;

public:

void acceptDipositorsData()

{

acceptData();

cout << "\nEnter Customer Account No. : ";

cin >> accountNo;

cout << "\nEnter Customer balance: ";

cin >> balance;

}

void showDipositorsData()

{

showCustomerDetails();

cout << "\nA/c No. : " << accountNo;

cout << "\nBalance: " << balance;

}

};

class Borrowers : public Customer

{

protected:

long loanNo, loanAmount;

public:

void acceptBorrowersData()

{

acceptData();

cout << "\nEnter Loan No. : ";

cin >> loanNo;

cout << "\nEnter Loan Amount: ";

cin >> loanAmount;

}

void showBorrowersDetails()

{

showCustomerDetails();

cout << "\nLoan No. : " << loanNo;

cout << "\nLoan Amount: " << loanAmount;

}

};

int main()

{

Dipositors \*d;

Borrowers \*b;

int bCust, dCust; // you can use menu drivern program also for simplicity

cout << "\nHow many depositers customer details you want to enter? ";

cin >> bCust;

cout << "\nHow many borrowers customer details you want to enter? ";

cin >> dCust;

d = new Dipositors[dCust];

b = new Borrowers[bCust];

// dipositors cust details

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

d->acceptDipositorsData();

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

d->showDipositorsData();

// borrowers cust details

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

b->acceptBorrowersData();

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

b->showBorrowersDetails();

return 0;

}

9. Write a C++ program to implement the following class hierarchy:

Student: id, name

StudentExam (derived from Student): Marks of 6 subjects

StudentResult (derived from StudentExam) : percentage

Define appropriate functions to accept and display details.

Create 'n' objects of the StudentResult class and display the marklist.

#include <iostream>

using namespace std;

class Student

{

protected:

int stId;

char name[20];

void acceptStudentInfo()

{

cout << "\n\nEnter student id: ";

cin >> stId;

cout << "\nEnter student name: ";

cin >> name;

}

void showStudentInfo()

{

cout << "\n\n student id: ";

cout << stId;

cout << "\n student name: ";

cout << name;

}

};

class StudentExam : public Student

{

protected:

float sub1, sub2, sub3, sub4, sub5, sub6;

public:

void acceptSubData()

{

acceptStudentInfo();

cout << "\nEnter sub1 No. : ";

cin >> sub1;

cout << "\nEnter sub2 No. : ";

cin >> sub2;

cout << "\nEnter sub3 No. : ";

cin >> sub3;

cout << "\nEnter sub4 No. : ";

cin >> sub4;

cout << "\nEnter sub5 No. : ";

cin >> sub5;

cout << "\nEnter sub6 No. : ";

cin >> sub6;

}

float totalMarks()

{

return (sub1 + sub2 + sub3 + sub4 + sub5 + sub6);

}

void showStudentInfoWithSub()

{

showStudentInfo();

cout << "\nMarks: ";

cout << "\nSub1: " << sub1;

cout << "\nSub2: " << sub2;

cout << "\nSub3: " << sub3;

cout << "\nSub4: " << sub4;

cout << "\nSub5: " << sub5;

cout << "\nSub6: " << sub6;

}

};

class StudentResult : public StudentExam

{

protected:

float percentage;

public:

void calPercentage()

{

percentage = totalMarks() / 6.0;

}

void showStudentDetails()

{

showStudentInfoWithSub();

showPercentage();

}

void showPercentage()

{

cout << "\nPercentage: " << percentage;

}

};

int main()

{

StudentResult s[5]; // you can enter by user

int noOfStudent = 1;

// u can use loop also

s[0].acceptSubData();

s[0].showStudentDetails();

return 0;

}

10. Consider two base classes

worker(int code, char name, float salary),

officer(float DA, HRA)

class manger(float TA(is 10% of salary), gross salary) is derived from both base classes.

Write necessary member functions.

#include <iostream>

#include <string.h>

using namespace std;

class worker

{

protected:

int code;

char name[20];

float salary;

public:

worker() {}

worker(int code, char \*n, float salary)

{

this->code = code;

strcpy(name, n);

this->salary = salary;

}

void showDetailsOfWorker()

{

cout << "\n\nCode: " << code;

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

cout << "\nSalary: " << salary;

}

};

class Officer

{

protected:

float DA, HRA;

public:

Officer() {}

Officer(float d, float h)

{

DA = d;

HRA = h;

}

void showDetailsOfOfficer()

{

cout << "\nDA: " << DA;

cout << "\nHRA: " << HRA;

}

};

class Manager : public worker, public Officer

{

private:

float TA, gSalary;

public:

Manager() {}

Manager(int c, char \*n, float s, float d, float h) : worker(c, n, s), Officer(d, h)

{

}

void showDetailsManager()

{

showDetailsOfWorker();

showDetailsOfOfficer();

TA = 0.10 \* salary;

cout << "\nTA: " << TA;

gSalary = DA + HRA + TA + salary;

cout << "\nGross Salary is: " << gSalary;

}

};

int main()

{

int cnt, i;

cout << "\nEnter Manager count: ";

cin >> cnt;

Manager \*m = new Manager[cnt];

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

{

cout << "\n\nEnter worker information for " << i + 1 << endl;

int c;

cout << "\nEnter code: ";

cin >> c;

char name[20];

cout << "\nEnter name: ";

cin >> name;

float salary;

cout << "\nEnter salary: ";

cin >> salary;

float d;

cout << "\nEnter DA: ";

cin >> d;

float h;

cout << "\nEnter HRA: ";

cin >> h;

m[i] = Manager(c, name, salary, d, h);

}

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

{

cout<<"\nmanager information: "<<endl;

m[i].showDetailsManager();

}

return 0;

}