Program 1:

#include <iostream>

#include <string>

using namespace std;

class Person {

protected:

string name;

int age;

public:

void getData() {

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

}

void displayData() {

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

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

}

};

class Admin : public Person {

protected:

float adminSalary;

public:

void getData() {

Person::getData();

cout << "Enter admin salary: ";

cin >> adminSalary;

}

void displayData() {

Person::displayData();

cout << "Admin Salary: " << adminSalary << endl;

}

float calculateBonus() {

return 0.05 \* adminSalary;

}

};

class Account : public Person {

protected:

float accountSalary;

public:

void getData() {

Person::getData();

cout << "Enter account salary: ";

cin >> accountSalary;

}

void displayData() {

Person::displayData();

cout << "Account Salary: " << accountSalary << endl;

}

float calculateBonus() {

return 0.1 \* accountSalary;

}

};

class Master : public Admin, public Account {

public:

void getData() {

Admin::getData();

Account::getData();

}

void displayData() {

cout << "Master's Details:" << endl;

Admin::displayData();

Account::displayData();

}

float calculateBonus() {

return Admin::calculateBonus() + Account::calculateBonus(); // Add bonuses from Admin and Account

}

};

int main() {

Master master;

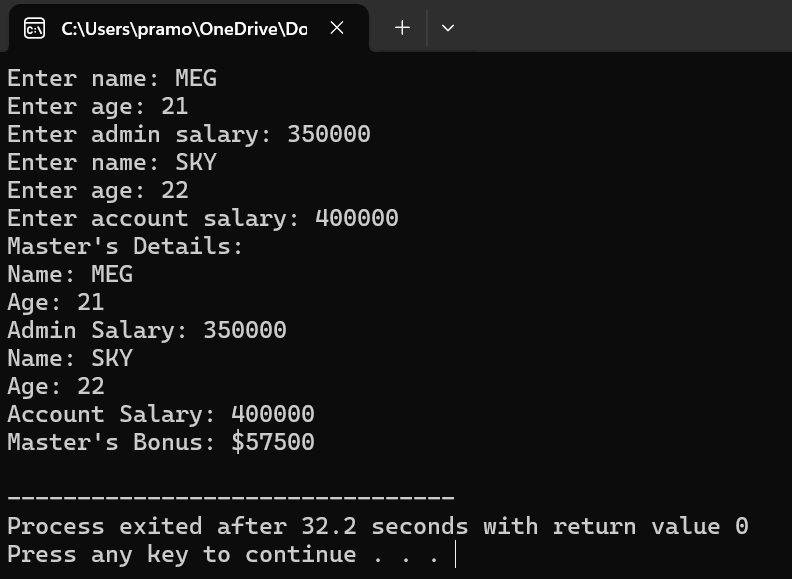
master.getData();

master.displayData();

cout << "Master's Bonus: $" << master.calculateBonus() << endl;

return 0;

}



Program 2:

#include <iostream>

class Shape {

public:

virtual float area() = 0;

virtual float volume() = 0;

};

class Rectangle {

protected:

float length;

float width;

public:

Rectangle(float l, float w) : length(l), width(w) {}

float area() {

return length \* width;

}

};

class Cuboid : public Rectangle, public Shape {

private:

float height;

public:

Cuboid(float l, float w, float h) : Rectangle(l, w), height(h) {}

float area() override {

return 2 \* (length \* width + length \* height + width \* height);

}

float volume() override {

return length \* width \* height;

}

};

int main() {

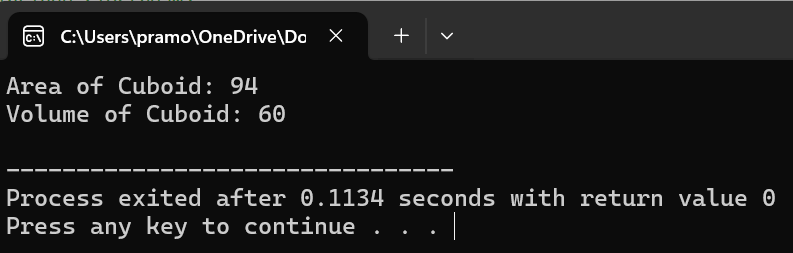
Cuboid cuboid(3, 4, 5);

std::cout << "Area of Cuboid: " << cuboid.area() << std::endl;

std::cout << "Volume of Cuboid: " << cuboid.volume() << std::endl;

return 0;

}



Program 3:

#include <iostream>

class Number {

protected:

int num;

public:

Number(int n) : num(n) {}

virtual void print() {

std::cout << num << " ";

}

};

class Skipper : public Number {

private:

int skip;

public:

Skipper(int n, int s) : Number(n), skip(s) {}

void printRange(int start, int end) {

for (int i = start; i <= end; i += skip) {

Number::print();

num += skip;

}

}

};

int main() {

int M, N, K;

std::cout << "Enter the value of M: ";

std::cin >> M;

std::cout << "Enter the value of N: ";

std::cin >> N;

std::cout << "Enter the value of K: ";

std::cin >> K;

Skipper skipper(M, K);

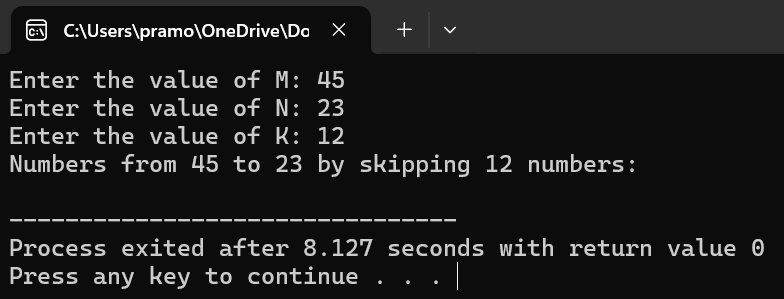
std::cout << "Numbers from " << M << " to " << N << " by skipping " << K << " numbers: ";

skipper.printRange(M, N);

std::cout << std::endl;

return 0;

}



Program 4;

#include <iostream>

class Grandfather {

protected:

double propertyValue;

public:

Grandfather(double value) : propertyValue(value) {}

double getProperty() {

return propertyValue;

}

};

class Grandson : public Grandfather {

public:

Grandson(double value) : Grandfather(value) {}

double getInheritedProperty() {

return propertyValue;

}

};

int main() {

double propertyValue = 500.0;

Grandfather grandpa(propertyValue);

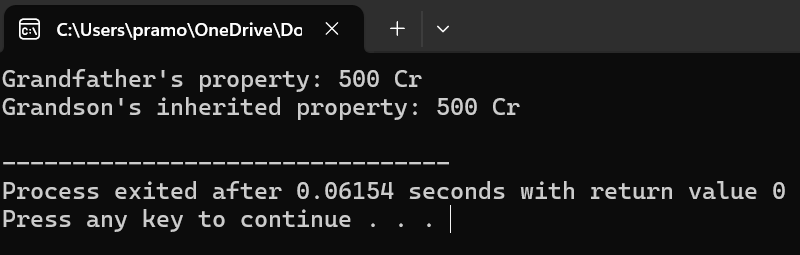
Grandson grandson(propertyValue);

std::cout << "Grandfather's property: " << grandpa.getProperty() << " Cr" << std::endl;

std::cout << "Grandson's inherited property: " << grandson.getInheritedProperty() << " Cr" << std::endl;

return 0;

}



Program 5:

#include <iostream>

#include <vector>

#include <string>

class Author {

private:

std::string name;

bool didWork;

public:

Author(const std::string& n, bool work) : name(n), didWork(work) {}

bool didContribute() const {

return didWork;

}

std::string getName() const {

return name;

}

};

class Paper {

private:

std::vector<Author> authors;

public:

Paper(const std::vector<Author>& auths) : authors(auths) {}

void addAuthor(const Author& auth) {

authors.push\_back(auth);

}

void identifyNotWorkedAuthor() const {

for (const auto& author : authors) {

if (!author.didContribute()) {

std::cout << author.getName() << " didn't contribute to the paper." << std::endl;

}

}

}

};

int main() {

std::vector<Author> authors {

Author("Author 1", true),

Author("Author 2", true),

Author("Author 3", true),

Author("Author 4", false)

};

Paper paper(authors);

paper.identifyNotWorkedAuthor();

paper.addAuthor(Author("Author 5", true));

return 0;

}

