**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСТИТЕТ**

**им. И.Раззакова**

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра «**Программное обеспечение компьютерных систем**»

Направление: 710400 «**Программная инженерия**»

Дисциплина: ««**Объектно-ориентированное программирование**»»

ОТЧЕТ

Лабораторная работа №6

Тема: Наследование

Выполнил: Кудайбердиев Эрлан

Группа: ПИ-2-21

Проверил: Мусабаев Э. Б.

Бишкек – 2024

**Задание №1**

Проблему экономии времени, доходов, а также увеличения надежности создаваемой объектно-ориентированной программы можно решить с помощью одного из главных принципов ООП - наследования, используя для создания класса-потомка готовый протестированный и отлаженный код класса-родителя. Для решения этой проблемы используйте преимущества **простого наследования**. Для этого создайте родительский класс работник (**Employee)**, протестируйте и отладьте его. В описании этого класса есть поля**:** имя работника - занимаемая должность - position[64], заработная плата за месяц- salary;

методы: конструктор по умолчанию, конструктор с аргументами, деструктор, расчет заработной платы за год, размещение информации о классе-родителе на консоли.

На основе отлаженного класса-родителя **Employee** создайте с помощью **механизма наследования** **public** класс-потомок менеджер (**Manager)**.

Класс - потомок Manager состоит из**:**

полей**:** ежегодный бонус **(annual\_bonus)**, машина компании (**company\_car[64]**)**,** опционы на акции (**stock\_options**) типа (**int**);

методов: конструктор по умолчанию, конструктор с аргументами, деструктор, размещение информации о классе-потомке на консоли.

*Определитесь с идентификаторами доступа к членам класса, не нарушая основного принципа ООП – инкапсуляции в наследовании.*

**Код программы:**

#include <iostream>

using namespace std;

class Employee {

protected:

string name;

string position;

double salary;

public:

Employee() {

this->name = "";

this->position = "";

this->salary = 0;

}

Employee(const string& name, const string& position, double salary) {

this->name = name;

this->position = position;

this->salary = salary;

}

double calculateAnnualSalary() {

return salary \* 12;

}

void displayEmployee() {

cout << "Имя: " << name << endl;

cout << "Позиция: " << position << endl;

cout << "Зарплата: " << salary << endl;

}

};

class Manager : public Employee {

private:

double annual\_bonus;

string company\_car;

int stock\_options;

public:

Manager() : Employee(){

this->annual\_bonus = 0;

this->company\_car = "";

this->stock\_options = 0;

}

Manager(const string& name, const string& position, double salary,

double annual\_bonus, const string& company\_car, int stock\_options)

: Employee(name, position, salary) {

this->annual\_bonus = annual\_bonus;

this->company\_car = company\_car;

this->stock\_options = stock\_options;

}

void displayManager() {

displayEmployee();

cout << "Ежегодный бонус: " << annual\_bonus << endl;

cout << "Машина компании: " << company\_car << endl;

cout << "Опционные акции: " << stock\_options << endl;

}

};

int main() {

setlocale(LC\_ALL, "rus");

Employee emp1("Кудайбердиев Эрлан", "java разработчик", 6000);

cout << "Информация о сотруднике:" << endl;

emp1.displayEmployee();

cout << "Ежегодный бонус: " << emp1.calculateAnnualSalary() << endl;

cout << endl;

Manager mgr1("Камиев Абулхаир", "Проект менеджер", 17000, 2000, "BMV x5", 100);

cout << "Информация о менеджеров:" << endl;

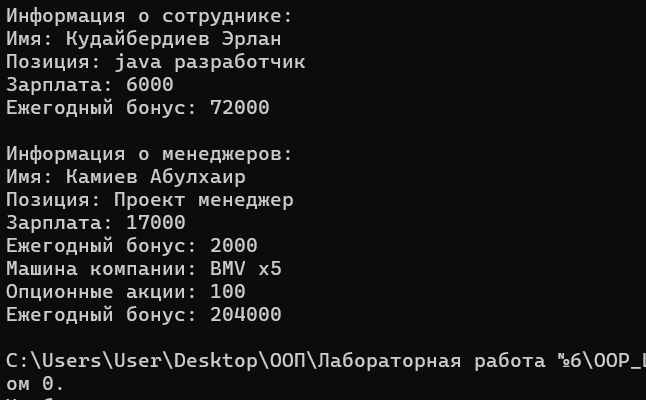
mgr1.displayManager();

cout << "Ежегодный бонус: " << mgr1.calculateAnnualSalary() << endl;

return 0;

}

**Результат:**



**Задание №2**

Используйте преимущества **множественного наследования,** а именно: разработайте, протестируйте и отладьте родительские классы Дисплей **(*Display*)** и Материнская плата (M***otherBoard),*** и на их основе создайте класс-потомокКомпьютер **(*Сomputer*)**:

В описании *класса-родителя* ***Display*** есть

поля: тип монитора char type[32], количество цветов long colors, разрешение по оси х int x\_resolution, разрешение по оси у int y\_resolution.

методы: конструктор с аргументами; размещение информации о классе-родителе на консоли.

В описании *класса-родителя* ***MotherBoard*** *имеет*

поля: тип процессора int processor, скорость процессора int speed, объем оперативной памяти int RAM;

методы: конструктор с аргументами; размещение информации о классе-родителе на консоли.

- *Класс-потомок* ***Сomputer*** создать на основе родительских классовс помощью механизма наследования **public** и отобразить его на экране**,** добавив:

поля: марка компьютера (name [64]), объем жесткого диска (hard\_disk**)**.

методы: конструктор с аргументами, метод Show().

**Код программы:**

#include <iostream>

#include <string>

using namespace std;

class Display {

protected:

string type;

long colors;

int x\_resolution;

int y\_resolution;

public:

Display(const string& t, long c, int x\_res, int y\_res) : type(t), colors(c), x\_resolution(x\_res), y\_resolution(y\_res) {}

void ShowInfo() {

cout << "Тип дисплея: " << type << endl;

cout << "Количество цветов: " << colors << endl;

cout << "Разрешение: " << x\_resolution << "x" << y\_resolution << endl;

}

};

class MotherBoard {

protected:

int processor;

int speed;

int RAM;

public:

MotherBoard(int proc, int spd, int ram) : processor(proc), speed(spd), RAM(ram) {}

void ShowInfo() {

cout << "Тип процессора: " << processor << endl;

cout << "Скорость процессора: " << speed << " МГц" << endl;

cout << "Объем оперативной памяти: " << RAM << " ГБ" << endl;

}

};

class Computer : public Display, public MotherBoard {

protected:

string name;

int hard\_disk;

public:

Computer(const string& n, int hd, const string& display\_type, long display\_colors, int display\_x\_res, int display\_y\_res, int mb\_processor, int mb\_speed, int mb\_ram)

: Display(display\_type, display\_colors, display\_x\_res, display\_y\_res), MotherBoard(mb\_processor, mb\_speed, mb\_ram), name(n), hard\_disk(hd) {}

void Show() {

cout << "Название компьютера: " << name << endl;

MotherBoard::ShowInfo();

Display::ShowInfo();

cout << "Жесткий диск: " << hard\_disk << " ГБ" << endl;

}

};

int main() {

setlocale(LC\_ALL,"rus");

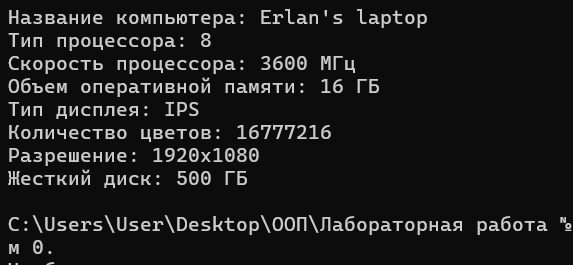
Computer myComputer("Erlan's laptop", 500, "IPS", 16777216, 1920, 1080, 8, 3600, 16);

myComputer.Show();

return 0;

}

**Результат:**



**Задание №3**

В задании № 1 изменить механизм наследования так, чтобы закрыть доступ к членам базового класса из производного класса. А затем восстановить этот доступ всеми известным Вам способами.

**Код программы:**

#include <iostream>

using namespace std;

class Employee {

public:

string name;

private:

string position;

protected:

double salary;

public:

Employee() {

this->name = "";

this->position = "";

this->salary = 0;

}

Employee(const string& name, const string& position, double salary) {

this->name = name;

this->position = position;

this->salary = salary;

}

double calculateAnnualSalary() {

return salary \* 12;

}

void displayEmployee() {

cout << "Имя: " << name << endl;

cout << "Позиция: " << position << endl;

cout << "Зарплата: " << salary << endl;

}

};

class Manager : private Employee {

private:

double annual\_bonus;

string company\_car;

int stock\_options;

public:

Employee::displayEmployee;

Employee::calculateAnnualSalary;

public:

Manager() : Employee(){

this->annual\_bonus = 0;

this->company\_car = "";

this->stock\_options = 0;

}

Manager(const string& name, const string& position, double salary,

double annual\_bonus, const string& company\_car, int stock\_options)

: Employee(name, position, salary) {

this->annual\_bonus = annual\_bonus;

this->company\_car = company\_car;

this->stock\_options = stock\_options;

}

void displayManager() {

displayEmployee();

cout << "Ежегодный бонус: " << annual\_bonus << endl;

cout << "Машина компании: " << company\_car << endl;

cout << "Опционные акции: " << stock\_options << endl;

}

};

int main() {

setlocale(LC\_ALL, "rus");

Employee emp1("Кудайбердиев Эрлан", "java разработчик", 6000);

cout << "Информация о сотруднике:" << endl;

emp1.displayEmployee();

cout << "Ежегодный бонус: " << emp1.calculateAnnualSalary() << endl;

cout << endl;

Manager mgr1("Камиев Абулхаир", "Проект менеджер", 17000, 2000, "BMV x5", 100);

cout << "Информация о менеджеров:" << endl;

mgr1.displayManager();

cout << "Ежегодный бонус: " << mgr1.calculateAnnualSalary() << endl;

return 0;

}

**Результат:**

