Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

ОТЧЕТ

к лабораторной работе №4

**«ПРИНЦИП ПОЛИМОРФИЗМА. ВИРТУАЛЬНЫЕ ФУНКЦИИ. АБСТРАКТНЫЕ КЛАССЫ»**

|  |  |
| --- | --- |
|  | |
|  |
|  |
|  |
|  |  | | |
|  | | |  | |  |
|  | | |  | |  |
| Студент гр. 324402 | | |  | | Цевелюк А. И. |
| Проверил | | |  | | Булыга Д. И. |

Минск 2024

**1 Цель работы**

Цель работы:изучить реализацию принципа динамического и статического полиморфизма, особенности использования перегрузки операторов, методов, переопределения функций.

**2 Теоретические сведения по лабораторной работе**

Полиморфизм — это свойство объектно-ориентированного программирования, которое позволяет использовать один и тот же интерфейс для разных типов объектов. Полиморфизм означает, что одно и то же имя функции или оператора может применяться к разным объектам, и в зависимости от типа объекта будет вызываться соответствующая реализация.

Полиморфизм позволяет писать более универсальный и гибкий код, который не зависит от конкретных классов, а работает с общими абстракциями. Полиморфизм также способствует повторному использованию кода, так как одна и та же функция или оператор может работать с разными классами, наследующими общий интерфейс.

Виртуальные функции — это функции, которые объявляются с ключевым словом virtual в базовом классе и могут быть переопределены в производных классах. Виртуальные функции позволяют реализовать динамический полиморфизм, то есть выбор нужной реализации функции во время выполнения программы в зависимости от типа объекта.

Виртуальные функции образуют цепочку виртуальных функций, которая связывает функции с одинаковым именем и сигнатурой в иерархии классов. Чтобы вызвать виртуальную функцию для объекта, используется указатель или ссылка на базовый класс. Компилятор определяет, какая из функций в цепочке должна быть вызвана, на основе типа объекта, на который указывает указатель или ссылка.

Виртуальные функции могут быть чистыми или нечистыми. Чистые виртуальные функции — это функции, которые не имеют реализации в базовом классе и должны быть переопределены во всех производных классах. Чистые виртуальные функции обозначаются = 0 в конце объявления. Класс, содержащий хотя бы одну чистую виртуальную функцию, называется абстрактным классом и не может быть использован для создания объектов. Нечистые виртуальные функции — это функции, которые имеют реализацию в базовом классе, но могут быть переопределены в производных классах.

Абстрактные классы — это классы, которые не предназначены для создания объектов, а только для наследования от них. Абстрактные классы содержат или наследуют без переопределения хотя бы одну чистую виртуальную функцию. Абстрактные классы определяют интерфейс для переопределения производными классами.

Абстрактные классы используются для описания общих свойств и поведения группы схожих объектов, которые могут иметь различные детали реализации. Абстрактные классы позволяют абстрагироваться от конкретных классов и работать с общими понятиями.

Например, можно определить абстрактный класс Shape, который представляет геометрическую фигуру. Этот класс может содержать чистые виртуальные функции для вычисления площади и периметра фигуры. От этого класса можно наследовать конкретные классы, такие как Rectangle, Circle, Triangle и т.д., которые будут реализовывать эти функции по-своему. Тогда можно создать указатель или ссылку на Shape и вызывать виртуальные функции для разных объектов, не зная их точного типа.

**3 Формулировка индивидуального задания**

Создать набор классов по теме «Расчет заработной платы», одним из которых будет класс Employee (сотрудник) с полями ФИО, стаж, часовая заработная плата, количество отработанных часов, оклад, премия. Создать массив сотрудников компании. Реализовать ввод данных всех сотрудников с клавиатуры. Рассчитать с помощью методов класса заработную плату за отработанное время, и премию, размер которой определяется в зависимости от стажа работника (при стаже до 1 года 1%, до 3 лет 5%, до 5 лет 8%, свыше5 лет 15%). С помощью метода печати, реализовать вывод информации о работнике на экран. Предусмотреть метод для записи в файл данных о работнике. Создать методы доступа к полям класса.

**4 Код решения индивидуального задания**

#include <fstream>

#include <iostream>

#include <string>

#include <vector>

class employee

{

protected:

std::string full\_name\_;

unsigned experience\_, number\_of\_hours\_;

double hourly\_wage\_, salary\_, bonus\_;

virtual void set\_bonus() = 0;

public:

virtual ~employee() = default;

employee() = default;

employee(const std::string& full\_name, unsigned experience, unsigned number\_of\_hours, double hourly\_wage);

employee(const employee& obj);

std::string get\_full\_name() { return full\_name\_; }

unsigned get\_experience() const { return experience\_; }

unsigned get\_number\_of\_hours() const { return number\_of\_hours\_; }

double get\_hourly\_wage() const { return hourly\_wage\_; }

double get\_salary() const { return salary\_; }

double get\_bonus() const { return bonus\_; }

void print() const;

void print\_to\_file() const;

};

employee::employee(const std::string& full\_name, const unsigned experience, const unsigned number\_of\_hours, const double hourly\_wage)

{

full\_name\_ = full\_name;

experience\_ = experience;

number\_of\_hours\_ = number\_of\_hours;

hourly\_wage\_ = hourly\_wage;

salary\_ = number\_of\_hours \* hourly\_wage;

}

void employee::print() const

{

std::cout << "Full name: " << full\_name\_ << ", experience: " << experience\_ << ", number of hours: " << number\_of\_hours\_ << ", hourly wage: " << hourly\_wage\_ << ", salary: " << salary\_ << ", bonus: " << bonus\_ << '\n';

}

void employee::print\_to\_file() const

{

std::fstream file;

file.open("lab4\_logs.txt", std::ofstream::app);

file << "Full name: " << full\_name\_ << ", experience: " << experience\_ << ", number of hours: " << number\_of\_hours\_ << ", hourly wage: " << hourly\_wage\_ << ", salary: " << salary\_ << ", bonus: " << bonus\_ << '\n';

file.close();

}

class employee\_1\_y final : public employee

{

protected:

void set\_bonus() override

{

bonus\_ = salary\_ \* 0.01;

}

public:

employee\_1\_y(const std::string& full\_name, const unsigned experience, const unsigned number\_of\_hours, const double hourly\_wage) : employee(full\_name, experience, number\_of\_hours, hourly\_wage) { employee\_1\_y::set\_bonus(); }

};

class employee\_3\_y final : public employee

{

protected:

void set\_bonus() override

{

bonus\_ = salary\_ \* 0.05;

}

public:

employee\_3\_y(const std::string& full\_name, const unsigned experience, const unsigned number\_of\_hours, const double hourly\_wage) : employee(full\_name, experience, number\_of\_hours, hourly\_wage) { set\_bonus(); }

};

class employee\_5\_y final : public employee

{

protected:

void set\_bonus() override

{

bonus\_ = salary\_ \* 0.08;

}

public:

employee\_5\_y(const std::string& full\_name, const unsigned experience, const unsigned number\_of\_hours, const double hourly\_wage) : employee(full\_name, experience, number\_of\_hours, hourly\_wage) { set\_bonus(); }

};

class employee\_more\_5\_y final : public employee

{

protected:

void set\_bonus() override

{

bonus\_ = salary\_ \* 0.15;

}

public:

employee\_more\_5\_y(const std::string& full\_name, const unsigned experience, const unsigned number\_of\_hours, const double hourly\_wage) : employee(full\_name, experience, number\_of\_hours, hourly\_wage) { set\_bonus(); }

};

void clear\_file()

{

std::fstream file;

file.open("lab4\_logs.txt", std::ofstream::out | std::ofstream::trunc);

file.close();

}

void clear()

{

system("pause");

system("cls");

}

int main()

{

clear\_file();

std::vector<employee\*> employees;

std::string buf;

std::cout << "Enter amount of employees: ";

std::getline(std::cin, buf);

const int amount = stoi(buf);

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

{

std::string full\_name;

std::cout << "Enter full name of employee No " << i + 1 << ": ";

std::getline(std::cin, full\_name);

std::cout << "Enter experience in years of employee No " << i + 1 << ": ";

std::getline(std::cin, buf);

const unsigned experience = stoi(buf);

std::cout << "Enter number of hours of employee No " << i + 1 << ": ";

std::getline(std::cin, buf);

const unsigned number\_of\_hours = stoi(buf);

std::cout << "Enter hourly wage of employee No " << i + 1 << ": ";

std::getline(std::cin, buf);

const double hourly\_wage = stoi(buf);

if (experience < 1)

{

employee\_1\_y\* push\_back\_employee = new employee\_1\_y(full\_name, experience, number\_of\_hours, hourly\_wage);

employees.push\_back(push\_back\_employee);

}

else if (experience >= 1 && experience < 3)

{

employee\_3\_y\* push\_back\_employee = new employee\_3\_y(full\_name, experience, number\_of\_hours, hourly\_wage);

employees.push\_back(push\_back\_employee);

}

else if (experience >= 3 && experience < 5)

{

employee\_5\_y\* push\_back\_employee = new employee\_5\_y(full\_name, experience, number\_of\_hours, hourly\_wage);

employees.push\_back(push\_back\_employee);

}

else

{

employee\_more\_5\_y\* push\_back\_employee = new employee\_more\_5\_y(full\_name, experience, number\_of\_hours, hourly\_wage);

employees.push\_back(push\_back\_employee);

}

}

clear();

std::cout << "Your employee list:\n";

for (const employee\* employeeptr : employees)

{

employeeptr->print();

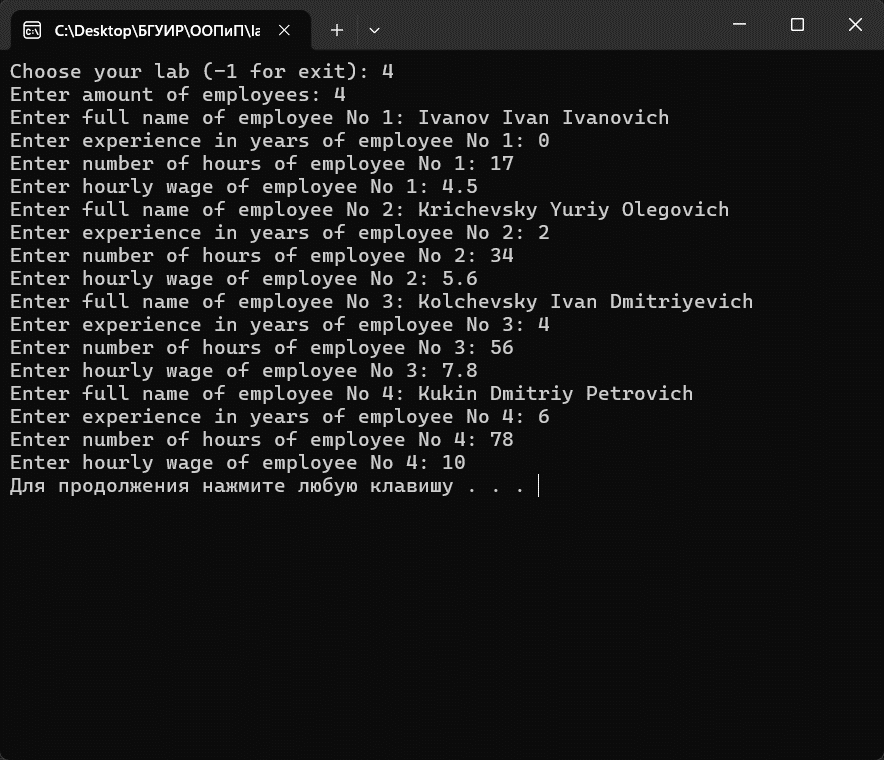
employeeptr->print\_to\_file();

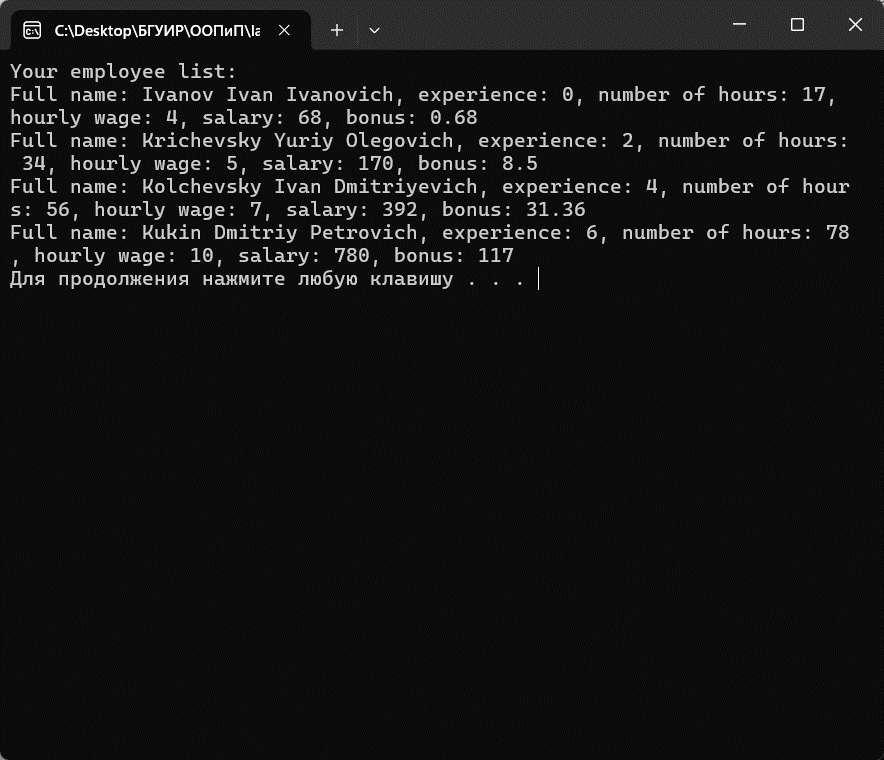
}

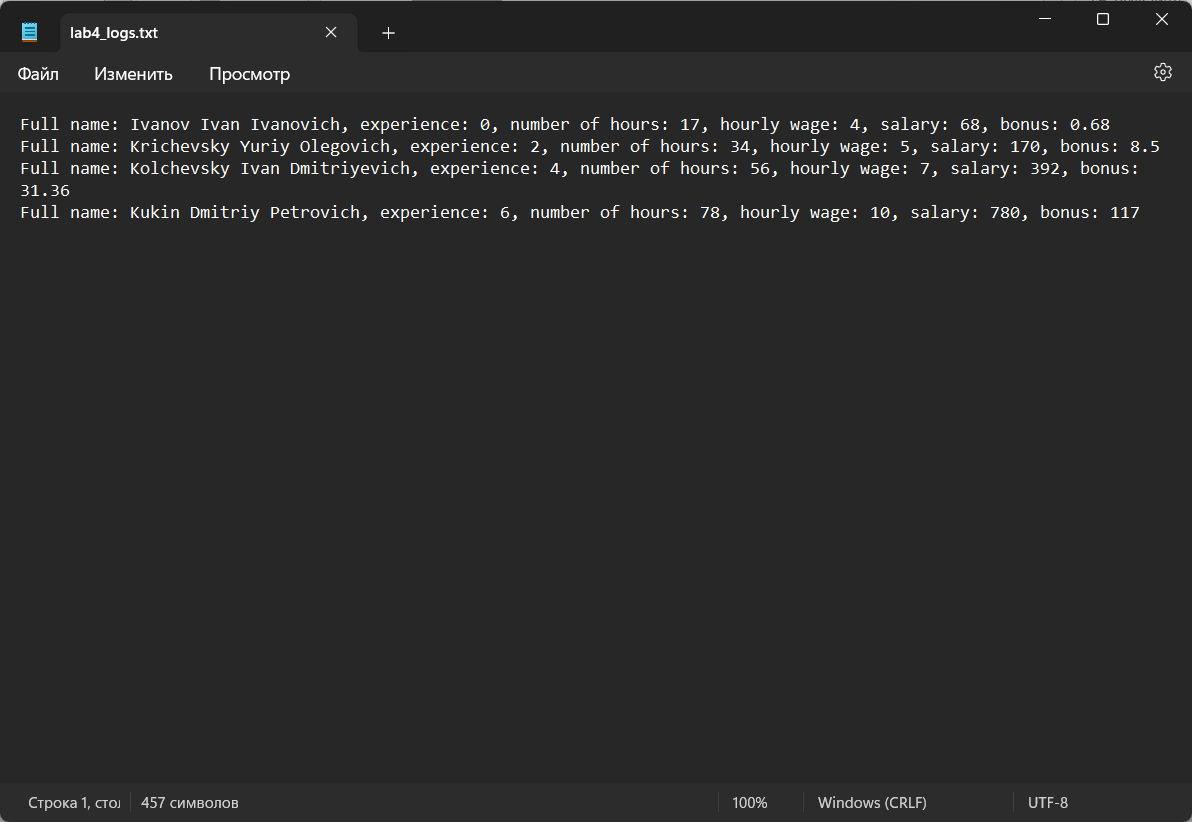
return 0;

}

**5 Скриншоты выполнения индивидуального задания**







**6 Ответы на контрольные вопросы**

1. Какая функция называется виртуальной?

Виртуальной называется функция, которая объявлена с ключевым словом virtual в базовом классе и может быть переопределена в производных классах. Виртуальные функции позволяют реализовать динамический полиморфизм, то есть выбор нужной реализации функции во время выполнения программы в зависимости от типа объекта.

1. Какая функция называется чисто виртуальной?

Чисто виртуальной называется функция, которая не имеет реализации в базовом классе и должна быть переопределена во всех производных классах. Чистые виртуальные функции обозначаются = 0 в конце объявления. Класс, содержащий хотя бы одну чистую виртуальную функцию, называется абстрактным классом и не может быть использован для создания объектов.

1. Чем виртуальные функции отличаются от перегруженных?

Виртуальные функции отличаются от перегруженных тем, что виртуальные функции имеют одинаковое имя и сигнатуру в разных классах, связанных наследованием, а перегруженные функции имеют одинаковое имя, но разную сигнатуру в одном классе или в разных классах, не связанных наследованием. Виртуальные функции позволяют реализовать динамический полиморфизм, а перегруженные функции позволяют реализовать статический полиморфизм.

1. Какой класс является абстрактным?

Абстрактным является класс, который не предназначен для создания объектов, а только для наследования от него. Абстрактный класс содержит или наследует без переопределения хотя бы одну чистую виртуальную функцию. Абстрактный класс определяет интерфейс для переопределения производными классами.

1. Для чего необходимо класс делать абстрактным?

Класс необходимо делать абстрактным, когда он представляет общее понятие, которое не имеет смысла реализовывать самостоятельно, а только через его конкретные подвиды. Абстрактный класс позволяет абстрагироваться от деталей реализации и работать с общим интерфейсом. Абстрактный класс также защищает от создания объектов неполного или некорректного типа.

1. В чем состоит различие раннего и позднего связывания?

Раннее и позднее связывание — это способы определения, какая функция должна быть вызвана для объекта. Раннее связывание означает, что выбор функции происходит на этапе компиляции на основе типа указателя или ссылки, через который вызывается функция. Позднее связывание означает, что выбор функции происходит на этапе выполнения на основе типа объекта, на который указывает указатель или ссылка. Раннее связывание используется для обычных функций, а позднее связывание используется для виртуальных функций.

1. Может ли конструктор быть виртуальным?

Конструктор не может быть виртуальным, потому что виртуальные функции работают на основе таблицы виртуальных функций, которая создается для каждого объекта при его конструировании. Поэтому виртуальные функции не могут быть вызваны до того, как объект будет создан, а конструктор как раз создает объект. Кроме того, конструктор не может быть переопределен в производных классах, а только вызван через список инициализации.

1. Опишите назначение виртуального деструктора.

Виртуальный деструктор — это деструктор, который объявлен с ключевым словом virtual в базовом классе и может быть переопределен в производных классах. Виртуальный деструктор позволяет корректно удалять объекты производных классов через указатель или ссылку на базовый класс. Если деструктор не виртуальный, то при удалении такого объекта будет вызван только деструктор базового класса, что может привести к утечке памяти или некорректному освобождению ресурсов.

1. Как защитить объект класса от копирования?

Чтобы защитить объект класса от копирования, можно сделать конструктор копирования и оператор присваивания приватными или удаленными. Тогда нельзя будет создать копию объекта через эти методы. Также можно объявить класс как final, то есть запретить наследование от него. Тогда нельзя будет создать копию объекта через приведение типов.

1. Для чего предназначены ключевые слова virtual и override?

Ключевые слова virtual и override предназначены для работы с виртуальными функциями. Ключевое слово virtual используется для объявления виртуальной функции в базовом классе, которая может быть переопределена в производных классах. Ключевое слово override используется для указания, что функция в производном классе переопределяет виртуальную функцию из базового класса. Это помогает избежать ошибок, связанных с разной сигнатурой или видимостью функций.

**7 Выводы по лабораторной работе**

Вывод: я изучил реализацию принципа динамического и статического полиморфизма, особенности использования перегрузки операторов, методов, переопределения функций.