КАФЕДРА №

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

Наследование классов, базовый класс, производный класс

по курсу: Объектно-ориентированное программирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

Условие

10 вариант

Цель работы:
Изучить механизм создания нового класса на основе уже существующего, варианты доступа к элементам базового класса из производного.

10. Создать класс Работник фирмы(фио, образование, Год поступления на работу, оклад). В классе должен быть метод вывода данных о работнике. На основе класса работника фирмы создать производные классы Стажер(должность, продолжительность испытательного строка, надбавка за прилежность), Руководящий работник(наименование отдела, количество подчиненных, надбавка за руководство), Директор(количество отделов, надбавка). В производных классах предусмотреть методы для расчета зарплаты и вычисления стажа работы, для Директора – подсчет количества подчиненных.

Листинг программы

main.cpp

```
#include <iostream>
using namespace std;

#include <cmath>

#include "Employee.h"
#include "Intern.h"
#include "Leading_worker.h"
#include "Director.h"

namespace global {
  int count_count_subordinates = 0;
}

// рисует линию в терминале
void draw_line(int size = 20) {
  for (int i = 0; i < size; i++)
    cout << '-';
  cout << endl;
}

int main() {
    // смена кодировки
  system("chcp 65001");

  // Стажер
  Intern intern((char*)"test1", 2013, 13000, (char*)"тест", 100,
1000);
  intern.info();

  draw_line();

  // Руководящий работник
  Leading_worker leading_worker((char*)"test2", 2020, 20000,
(char*)"Завод", 10);
  leading_worker.info();

  draw_line();
```

```
  // Директор
  Director director((char*)"test3", 2002, 30000, 2, 100);
  director.info();

    return 0;
```

Intern.h

```
#include <iostream>
using namespace std;

// Стажер
class Intern: public Employee {
public:
  Intern(char* full_name, int Year_of_admission, double Salary,
char* Job_title, int Duration, double Diligence_bonus);
  void info();
  double calculate_money();

protected:
  char* job_title; // Должность
  int duration; // Продолжительность испытательного строка
  double diligence_bonus; // Надбавка за прилежность
};

Intern::Intern(char* full_name, int Year_of_admission, double Sal-
ary, char* Job_title, int Duration, double Diligence_bonus) :Em-
ployee(full_name, Year_of_admission, Salary) {
  job_title = Job_title;
  duration = Duration;
  diligence_bonus = Diligence_bonus;
}

void Intern::info() {
  cout << "Стажер" << endl;
  show_info();
  printf("Должность: %s\n", job_title);
  cout << "Продолжительность испытательного строка: " << duration
<< " дней" << endl;
  cout << "Надбавка за прилежность: " << diligence_bonus << " руб"
<< endl;
  printf("Саммарное зп: %.2lf руб\n", calculate_money());
}

double Intern::calculate_money() {
  return calculate_salary() + diligence_bonus;
}
```

Employee.h

```
#include <iostream>
#include <ctime>
using namespace std;

namespace global {
  extern int count_count_subordinates;
}
```

```
int get_now_year() {
  time_t now = time(0);

  tm *ltm = localtime(&now);
  return (1900 + ltm -> tm_year);
}

class Employee {
public:
  Employee(char* full_name, int Year_of_admission, double Salary);

  void show_info();
  double calculate_salary();
  int get_experience();

protected:
  char* full_name;       // ФИО
  int year_of_admission;  // Год поступления на работу
  double salary;          // Оклад

};

// конструктор
Employee::Employee(char* Full_name, int Year_of_admission, double
Salary) {
  full_name = Full_name;
  year_of_admission = Year_of_admission;
  salary = Salary;

  //if (director) count_subordinates++;
}

// вывод информации
void Employee::show_info() {
  //cout << "ФИО: " << full_name << end;
  printf("ФИО: %s\n", full_name);
  cout << "Год поступления на работу: " << year_of_admission <<
endl;
  cout << "Оклад: " << salary << " руб" << endl;
  cout << "Стаж работы: " << get_experience() << " лет" << endl;
}

int Employee::get_experience() {
  return (get_now_year() - year_of_admission);
}

// рассчёт зарплаты
double Employee::calculate_salary() {
  return (get_experience() * 12) * salary
```

Director.h

```
#include <iostream>
using namespace std;

namespace global {
  extern int count_count_subordinates;
}
```

```
// Директор
class Director: public Employee {
public:
  Director(char* full_name, int Year_of_admission, double Salary,
int Amoun_departments, int Allowance);

  void info();
  double calculate_money();
  int get_count_subordinates();

protected:
  int amoun_departments; // количество отделов
  int allowance; // надбавка
};

Director::Director(char* full_name, int Year_of_admission, double
Salary, int Amoun_departments, int Allowance) :Employee(full_name,
Year_of_admission, Salary) {
  amoun_departments = Amoun_departments;
  allowance = Allowance;
}

void Director::info() {
  cout << "Директор" << endl;
  show_info();
  cout << "Количество отделов: " << amoun_departments << endl;
  cout << "Надбавка: " << allowance << " руб"<< endl;
  printf("Саммарное зп: %.2lf руб\n", calculate_money());
  cout << "Количество подчинённых: " << get_count_subordinates()
<< endl;
}

int Director::get_count_subordinates() {
  return global::count_count_subordinates;
}

double Director::calculate_money() {
  return calculate_salary();
}
```

Leading_worker.h

```
#include <iostream>
using namespace std;

namespace global {
  extern int count_count_subordinates;
}

// Руководящий работник
class Leading_worker: public Employee {
public:
  Leading_worker(char* full_name, int Year_of_admission, double
Salary, char* Name_department, int Count_subordinates);

  void info();
  double calculate_money();

protected:
```

```
  char* name_department; // наименование отдела
  int count_subordinates; // количество подчиненных
};

Leading_worker::Leading_worker(char* full_name, int Year_of_admis-
sion, double Salary, char* Name_department, int Count_subordi-
nates) :Employee(full_name, Year_of_admission, Salary) {
  name_department = Name_department;
  count_subordinates = Count_subordinates;

  global::count_count_subordinates += Count_subordinates;
}

void Leading_worker::info() {
  cout << "Руководящий работник" << endl;
  show_info();
  printf("Наименование отдела: %s\n", name_department);
  cout << "Количество подчиненных: " << count_subordinates <<
endl;
  printf("Саммарное зп: %.2lf руб\n", calculate_money());
}

double Leading_worker::calculate_money() {
  return calculate_salary();
```

Скриншоты

Вывод

Мы изучили механизм создания нового класса на основе уже существующего и варианты доступа к элементам базового класса из производного.