Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение

высшего образования

«**Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные

системы»

**ОТЧЁТ**

**по лабораторной работе №18.4**

Дисциплина: «Информатика»

Тема: Простое наследование. Принцип подстановки.

Вариант 13

Выполнил:

Студент группы ИВТ-20-2б

Сафронов Владислав Владиславович

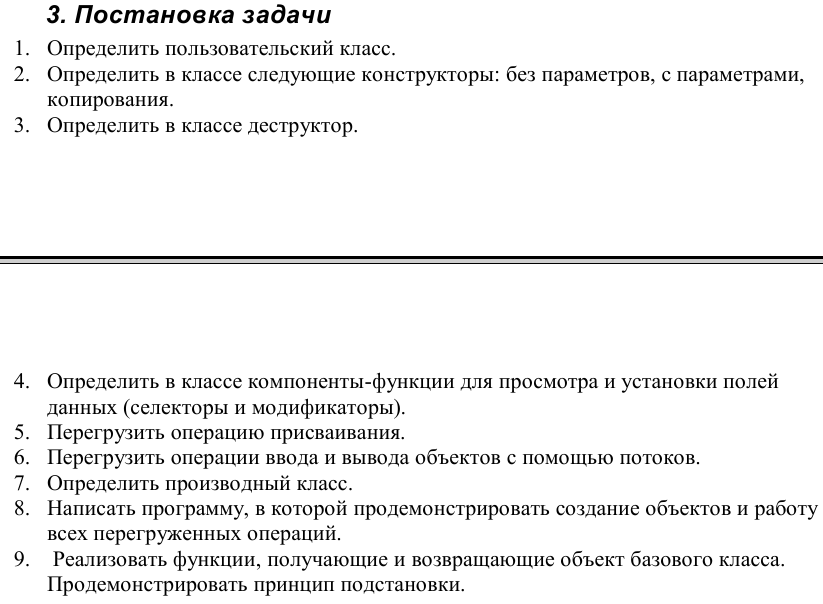
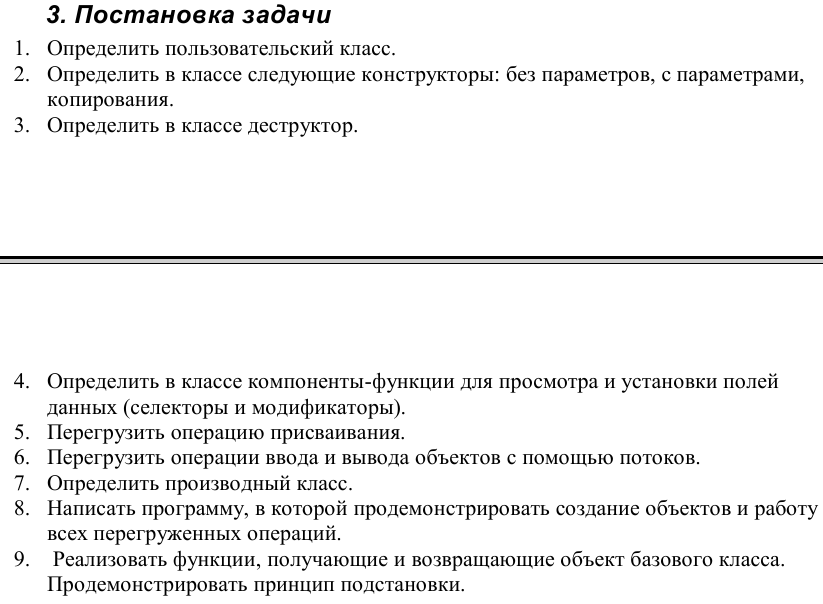
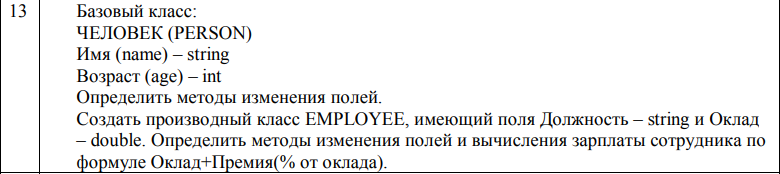
Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**

****

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Разработать класс Person для хранения сведений о человеке.
   2. Разработать класс Employee для хранения сведений о работнике. Наследуется от класса Person.
   3. Разработать необходимые методы для ввода данных в поля классов, конструкторы, деструкторы.
   4. Организовать перегрузку операторов >>, <<, =.
2. В ходе работы были использованы следующие типы данных:
   1. Класс Person для хранения сведений о человеке. Имеет 2 поля: строковое поле name хранит ФИО, целочисленное поле age – возраст.
3. class Person
4. {
5. string name;
6. int age;
   * 1. Конструкторы класса Person.
7. Person()
8. {
9. name = "None";
10. age = 0;
11. }
12. Person(string name, int age)
13. {
14. this->name = name;
15. this->age = age;
16. }
17. Person(const Person& obj)
18. {
19. name = obj.name;
20. age = obj.age;
21. }
    * 1. Деструктор описывать явно не требуется, так как в классе не используется динамическая память.
22. ~Person() {}
    * 1. Сеттеры для присвоения значений полям класса Person.
23. void SetName(string a)
24. {
25. this->name = a;
26. }
27. void SetAge(int n)
28. {
29. this->age = n;
30. }
    * 1. Геттеры для возвращения значений полей класса Person.
31. string GetName() { return name; }
32. int GetAge() { return age; }
    * 1. Метод Show для вывода значений полей в консоль.
33. void Show()
34. {
35. cout << "ФИО: " << name << endl;
36. cout << "Возраст: " << age << endl;
37. }
    * 1. Перегрузка операции присваивания = для класса Person.
38. Person& Person::operator=(Person& obj)
39. {
40. if (this == &obj) return \*this;
41. name = obj.name;
42. age = obj.age;
43. return \*this;
44. }
    * 1. Перегрузка операторов << и >> для класса Person.
45. istream& operator>> (istream& cin, Person& obj)
46. {
47. cout << "Введите ФИО: ";
48. getline(cin, obj.name);
49. cout << "Введите возраст: ";
50. cin >> obj.age;
51. return cin;
52. }
53. ostream& operator<< (ostream& cout, Person& obj)
54. {
55. return (cout << "Имя: " << obj.name << endl << "Возраст: " << obj.age << endl);
56. }
    1. Класс Employee наследуется от класса Person и имеет 2 поля: строковое поле position хранит должность работника, вещественное поле salary хранит оклад.
57. class Employee : public Person
58. {
59. string position;
60. double salary;
    * 1. Конструкторы класса Employee
61. Employee() :Person(), position("None"), salary(0) {}
62. Employee(string pos, double sal) :Person(), position(pos), salary(sal) {}
63. Employee(string pos) :Person(), position(pos), salary(0) {}
64. Employee(string name, int age, string pos, double sal) :Person(name, age), position(pos), salary(sal) {}
65. Employee(double sal) :Person(), salary(sal) {}
66. Employee(Person p, string pos, double sal) :Person(p), position(pos), salary(sal) {}
    * 1. Сеттеры и геттеры для изменения и возвращения значений полей класса.
67. void SetPosition(string pos) { this->position = pos; }
68. void SetSalary(double sal) { this->salary = sal; }
69. string GetPosition() { return position; }
70. double GetSalary() { return salary; }
    * 1. Метод GetPay возвращает вещественное число: оклад + премия работника. Премия составляет 25% от оклада.
71. double GetPay() { return salary \* 1.25; }
    * 1. Метод All\_Info для вывода значений полей класса Employee. Сначала вызывается метод Show класса Person, затем выводятся поля класса Employee, а также оклад + премия.
72. void All\_Info()
73. {
74. this->Show();
75. cout << "Должность: " << position << endl;
76. cout << "Оклад: " << salary << endl;
77. cout << "Оклад + премия: " << GetPay() << endl;
78. }
    * 1. Деструктор описывать явно не требуется, так как в классе не используется динамическая память.
79. ~Employee() {}
    * 1. Перегрузка оператора присваивания = для класса Employee
80. Employee& operator=(Person& p)
81. {
82. this->SetName(p.GetName());
83. this->SetAge(p.GetAge());
84. return \*this;
85. }
86. };
87. Описание функции main()
    1. Создание экземпляров классов Person и Employee используя разные конструктооры.
88. int main()
89. {
90. setlocale(LC\_ALL, "Russian");
91. system("chcp 1251>nul");
93. Person a("Кондратьев Артём Иванович", 23);
94. Person b(a);
95. a.Show(); b.Show();
96. cout << "----------------------------------\n";
98. Employee c(a, "Тестировщик", 55000);
99. c.All\_Info();
100. cout << "----------------------------------\n";
     1. Демонстрация перегрузки операции присваивания.
101. Person Empty;
102. Employee d("Смирнов Сергей Георгиевич", 20, "Системный аналитик", 80000);
103. Empty.Show();
104. Empty = b;
105. Empty.Show();
106. cout << "----------------------------------\n";
     1. Демонстрация перегрузки операций >> и <<
107. cout << "Демонстрация перегрузки '<<' и '>>':\n";
108. cout << "Ввод (>>):\n";
109. Person obj;
110. cin >> obj;
111. cout << "Вывод (<<):\n";
112. cout << obj;
113. obj.Show();
114. }

**Блок-схема программы**

**Код программы**

**Файл Person.h:**

1. #pragma once
2. #include <iostream>
3. using namespace std;
5. class Person
6. {
7. string name;
8. int age;
9. public:
10. Person()
11. {
12. name = "None";
13. age = 0;
14. }
15. Person(string name, int age)
16. {
17. this->name = name;
18. this->age = age;
19. }
20. Person(const Person& obj)
21. {
22. name = obj.name;
23. age = obj.age;
24. }
26. ~Person() {}
28. void SetName(string a)
29. {
30. this->name = a;
31. }
32. void SetAge(int n)
33. {
34. this->age = n;
35. }
37. string GetName() { return name; }
38. int GetAge() { return age; }
40. void Show()
41. {
42. cout << "ФИО: " << name << endl;
43. cout << "Возраст: " << age << endl;
44. }
46. Person& operator=(Person& obj);
47. friend istream& operator >> (istream& cin, Person& obj);
48. friend ostream& operator << (ostream& cin, Person& obj);
49. };
51. Person& Person::operator=(Person& obj)
52. {
53. if (this == &obj) return \*this;
54. name = obj.name;
55. age = obj.age;
56. return \*this;
57. }
58. istream& operator>> (istream& cin, Person& obj)
59. {
60. cout << "Введите ФИО: ";
61. getline(cin, obj.name);
62. cout << "Введите возраст: ";
63. cin >> obj.age;
64. return cin;
65. }
66. ostream& operator<< (ostream& cout, Person& obj)
67. {
68. return (cout << "Имя: " << obj.name << endl << "Возраст: " << obj.age << endl);
69. }

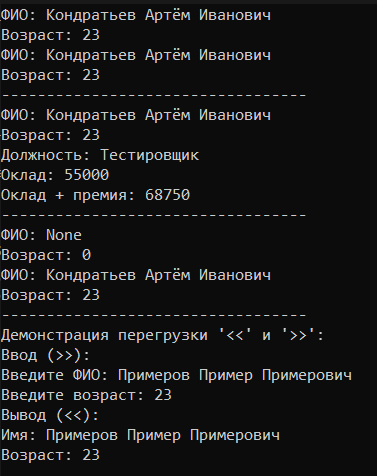
**Файл Employee.h:**

1. #pragma once
2. #include <iostream>
3. #include "Person.h"
4. using namespace std;
6. class Employee : public Person
7. {
8. string position;
9. double salary;
10. public:
11. Employee() :Person(), position("None"), salary(0) {}
12. Employee(string pos, double sal) :Person(), position(pos), salary(sal) {}
13. Employee(string pos) :Person(), position(pos), salary(0) {}
14. Employee(string name, int age, string pos, double sal) :Person(name, age), position(pos), salary(sal) {}
15. Employee(double sal) :Person(), salary(sal) {}
16. Employee(Person p, string pos, double sal) :Person(p), position(pos), salary(sal) {}
18. void SetPosition(string pos) { this->position = pos; }
19. void SetSalary(double sal) { this->salary = sal; }
20. string GetPosition() { return position; }
21. double GetSalary() { return salary; }
23. double GetPay() { return salary \* 1.25; }
25. void All\_Info()
26. {
27. this->Show();
28. cout << "Должность: " << position << endl;
29. cout << "Оклад: " << salary << endl;
30. cout << "Оклад + премия: " << GetPay() << endl;
31. }
33. ~Employee() {}
35. Employee& operator=(Person& p)
36. {
37. this->SetName(p.GetName());
38. this->SetAge(p.GetAge());
39. return \*this;
40. }
41. };

**Файл main.cpp:**

1. #include <iostream>
2. #include <string>
3. #include "Person.h"
4. #include "Employee.h"
5. using namespace std;
7. int main()
8. {
9. setlocale(LC\_ALL, "Russian");
10. system("chcp 1251>nul");
12. Person a("Кондратьев Артём Иванович", 23);
13. Person b(a);
14. a.Show(); b.Show();
15. cout << "----------------------------------\n";
17. Employee c(a, "Тестировщик", 55000);
18. c.All\_Info();
19. cout << "----------------------------------\n";
21. Person Empty;
22. Employee d("Смирнов Сергей Георгиевич", 20, "Системный аналитик", 80000);
23. Empty.Show();
24. Empty = b;
25. Empty.Show();
26. cout << "----------------------------------\n";
28. cout << "Демонстрация перегрузки '<<' и '>>':\n";
29. cout << "Ввод (>>):\n";
30. Person obj;
31. cin >> obj;
32. cout << "Вывод (<<):\n";
33. cout << obj;
34. obj.Show();
35. }

**Скриншоты результатов работы программы**

****

**Ответы на вопросы**



1. Наследование позволяет реализовать типизированное повторное использование объектов, создание иерархии «тип-подтип» с сохранением подтипом всех свойств своего предка. Можно рассматривать наследование для всех трех составляющих объекта.



2. Когда вы открыто наследуете родительский класс, то унаследованные public-члены остаются public.



3. Когда вы открыто наследуете родительский класс, то унаследованные private-члены остаются недоступными для дочернего класса.



4. К protected-членам родительского класса доступ открыт для членов дочернего класса.



5. 



6. Конструкторы не наследуются.



7. Деструкторы не наследуются.



8. 



9. 

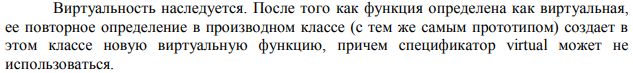


10. 

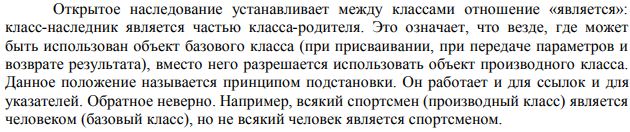


11. 



12. 



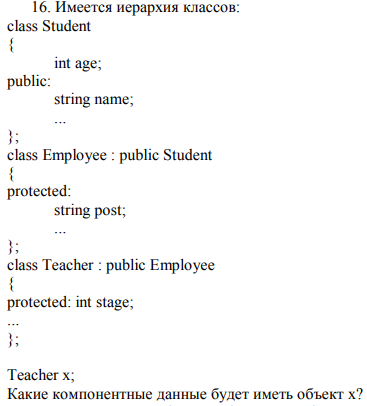
13. 



14. 



15. Пункт 13 и 14 – принцип подстановки.



16. age, name, post, stage.



17. Student(); Employee() : Student(); Teacher() : Employee();