Министерство образование Республики Беларусь

Учреждение образования «Полоцкий государственный университет»

Информационно-технологический факультет

Кафедра технологий программирования

Лабораторная работа №2 по теме «Наследования и виртуальные функции»

По дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Выполнил: Ролевич М.А.

Группа 19-ИТ-1.

Проверил: Васильева Д. М.

Новополоцк

2021 г.

**Вариант 3**

**Цель работы**: Получить практические навыки создания иерархии классов и использования статических компонентов класса.

**Задание 1. Постановка задачи**

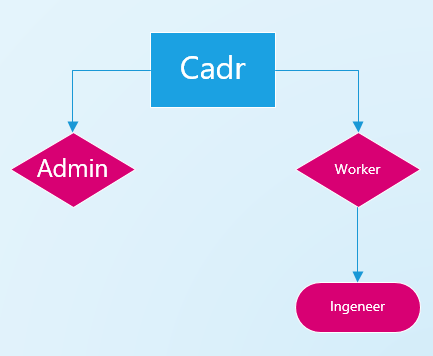
Для получения практических навыков создания иерархии классов и использования статических компонент, предложено реализовать демонстрационную программу, в которой создаются объекты различных классов и помещаются в список, после чего список просматривается. Необходимо выполнить следующие задания:

1. Определить иерархию классов (в соответствии с вариантом).
2. Определить в классе статическую компоненту - указатель на начало связанного списка объектов и статическую функцию для просмотра списка.
3. Реализовать классы.
4. Написать демонстрационную программу, в которой создаются объекты различных классов и помещаются в список, после чего список просматривается.
5. Сделать соответствующие методы не виртуальными и посмотреть, что будет.

Для моего варианта предложены следующие классы:

1. Рабочий
2. Кадры
3. Инженер
4. Администрация

Я составил следующее наследование: сначала идёт Кадры, от которого идут админ и рабочий, от которого идёт инженер.



**Задание 1. Определение пользовательских классов.**

Прототип абстрактного класса Cadr, затем его реализация:

1. сlass Cadr {
2. public:
3. int man;
4. Cadr(int x) {
5. man = x;
6. cout << "the number of card created " << this->man <<
7. endl;
8. };
9. ~Cadr() {
10. cout << "Cadr deleted" << endl;
11. };
12. };

Прототип класса Admin, который наследуется от абстрактного Cadr и его реализация:

1. class Admin :public Cadr{
2. public:
3. int slevel;
4. Admin(int a, int x) : Cadr(x) {
5. slevel = a;
6. cout << "An Admin is created with a level of work " <<
7. this->slevel << endl;
8. }
9. ~Admin() {
10. cout << "Deleted Admin element";
11. }
12. };

Прототип класса Worker, который наследуется от абстрактного Cadr и его реализация:

1. class Worker :public Cadr {
2. public:
3. int staj;
4. Worker(int s, int x) :Cadr(x) {
5. staj = s;
6. cout << "An experienced Worker item is being created "<<
7. this->staj << endl;
8. };
9. ~Worker(){
10. cout << "Deleted Worker element" << endl;
11. };
12. };

Прототип класса Engineer, который наследуется от Worker и его реализация:

1. class Engineer :public Worker {
2. public:
3. int expe;
4. Engineer(int f, int s, int x) :Worker(s, x) {
5. expe = f;
6. cout << "Created element engineer with experience " <<
7. this->expe << endl;
8. }
9. ~Engineer() {
10. cout << "Deleted engineer element" << endl;
11. }
12. };

**Задание 4. Демонстрационная программа**

1. #include <iostream>
2. #include "Admin.h"
3. #include "Worker.h"
4. #include "Cadr.h"
5. #include "Engeneer.h"
6. #include <list>
7. int main() {
8. cout << "\n\n";
9. typedef unique\_ptr<Product> TList;
10. list <TList> list1;
11. list1.push\_back(TList(new Worker));
12. list1.push\_back(TList(new Engeneer));
13. list1.push\_back(TList(new Admin));
14. cout << "\n\n";
15. for (TList const& lst : list1) {
16. lst->show();
17. cout << endl;
18. }
19. cout << endl;
20. }

Демонстрационная программа выполняет подключения всех необходимых классов, а затем создает три объекта от каждого из класса кроме абстрактного Person. Затем выполняется вызов статического метода для выведения информации о каждом из объектов.

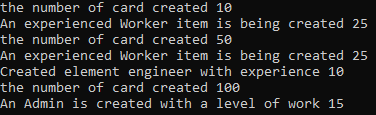


Рис. 1 – Вызов конструкторов



Рис. 2 – вызов метода show ()

На рис. 2 показан результат вызова метода show (), в котором объекты добавлялись в список с параметрами.

**Задание 5. О виртуальных методах.**

**Виртуальная функция** в языке С++ — это особый тип функции, которая, при её вызове, выполняет «наиболее» дочерний метод, который существует между родительским и дочерними классами. Это свойство еще известно, как **полиморфизм**. Дочерний метод вызывается тогда, когда совпадает сигнатура (имя, типы параметров и является ли метод константным) и тип возврата дочернего метода с сигнатурой и типом возврата метода родительского класса. Такие методы называются переопределениями (или «переопределенными методами»).