ИСТ-АДМО-202Б

Трубин В.Д.

Орлов К.О.

3241.406315.000 ПЗ

Геоинформационных систем

Языки программирования

Уфа – 2025

Полиморфизм

Содержание

[Введение 3](#_Toc199390111)

[1. Задание 1 4](#_Toc199390112)

[2. Задание 2 7](#_Toc199390113)

[Заключение 12](#_Toc199390114)

[Список литературы 13](#_Toc199390115)

Введение

Цель практических занятий: закрепление и углубление знаний, полученных в теоретической части дисциплины, развитие навыков объектно-ориентированной технологии проектирования алгоритмов и программ, развитие творческих способностей в области программирования.

# Задание 1

Описать родительский класс для объектов в соответствии с вариантом задания. Для создаваемого класса предусмотреть четкое выполнение принципов Инкапсуляции (все данные объекта недоступны для прямого обращения). Все данные задаются и выводятся посредством вызова интерфейсных методов. В классе должны быть предусмотрены виртуальные методы, которые будут реализованы в классах потомках

#include <iostream>

#include <string>

#include <ctime>

using namespace std;

// task 1

class Person {

protected:

string snils;

string lastName;

string firstName;

string middleName;

string gender;

string birthDate;

public:

// Конструктор по умолчанию

Person() : snils(""), lastName(""), firstName(""), middleName(""), gender(""), birthDate("") {}

// Сеттеры

virtual void setSnils(const string& s) { snils = s; }

virtual void setLastName(const string& ln) { lastName = ln; }

virtual void setFirstName(const string& fn) { firstName = fn; }

virtual void setMiddleName(const string& mn) { middleName = mn; }

virtual void setGender(const string& g) { gender = g; }

virtual void setBirthDate(const string& bd) { birthDate = bd; }

// Геттеры

virtual string getFullName() const {

return lastName + " " + firstName + " " + middleName;

}

virtual string getShortName() const {

if (!middleName.empty())

return lastName + " " + firstName[0] + "." + middleName[0] + ".";

return lastName + " " + firstName[0] + ".";

}

virtual int getCurrentAge() const {

// Упрощенный расчет возраста

if (birthDate.empty()) return 0;

// Получаем текущий год

time\_t now = time(0);

tm\* ltm = localtime(&now);

int currentYear = 1900 + ltm->tm\_year;

// Парсим год рождения (формат DD.MM.YYYY)

int birthYear = stoi(birthDate.substr(6, 4));

return currentYear - birthYear;

}

virtual int getAgeAtDate(const string& date) const {

if (birthDate.empty() || date.empty()) return 0;

int targetYear = stoi(date.substr(6, 4));

int birthYear = stoi(birthDate.substr(6, 4));

return targetYear - birthYear;

}

virtual string getDateWhenAgeReached(int age) const {

if (birthDate.empty()) return "";

int birthYear = stoi(birthDate.substr(6, 4));

int targetYear = birthYear + age;

return birthDate.substr(0, 6) + to\_string(targetYear);

}

// Виртуальные методы для переопределения в потомках

virtual void inputData() = 0;

virtual void printData() const = 0;

# Задание 2

Описать не менее двух производных (дочерний) классов объектов в соответствии с вариантом. В производных классах предусмотреть описание дополнительных атрибутов, а также реализовать в соответствии с принципом полиморфизма методы, которые необходимы для выполнения операций над дочерними объектами, т.е. методы, описанные в классе-родителе должны быть описаны в родительском классе, а реализованы в классах потомках

#include <iostream>

#include <string>

#include <ctime>

using namespace std;

// task 1

class Person {

protected:

string snils;

string lastName;

string firstName;

string middleName;

string gender;

string birthDate;

public:

// Конструктор по умолчанию

Person() : snils(""), lastName(""), firstName(""), middleName(""), gender(""), birthDate("") {}

// Сеттеры

virtual void setSnils(const string& s) { snils = s; }

virtual void setLastName(const string& ln) { lastName = ln; }

virtual void setFirstName(const string& fn) { firstName = fn; }

virtual void setMiddleName(const string& mn) { middleName = mn; }

virtual void setGender(const string& g) { gender = g; }

virtual void setBirthDate(const string& bd) { birthDate = bd; }

// Геттеры

virtual string getFullName() const {

return lastName + " " + firstName + " " + middleName;

}

virtual string getShortName() const {

if (!middleName.empty())

return lastName + " " + firstName[0] + "." + middleName[0] + ".";

return lastName + " " + firstName[0] + ".";

}

virtual int getCurrentAge() const {

// Упрощенный расчет возраста

if (birthDate.empty()) return 0;

// Получаем текущий год с использованием localtime\_s

time\_t now = time(0);

tm ltm;

localtime\_s(&ltm, &now);

int currentYear = 1900 + ltm.tm\_year;

// Парсим год рождения (формат DD.MM.YYYY)

int birthYear = stoi(birthDate.substr(6, 4));

return currentYear - birthYear;

}

virtual int getAgeAtDate(const string& date) const {

if (birthDate.empty() || date.empty()) return 0;

int targetYear = stoi(date.substr(6, 4));

int birthYear = stoi(birthDate.substr(6, 4));

return targetYear - birthYear;

}

virtual string getDateWhenAgeReached(int age) const {

if (birthDate.empty()) return "";

int birthYear = stoi(birthDate.substr(6, 4));

int targetYear = birthYear + age;

return birthDate.substr(0, 6) + to\_string(targetYear);

}

// Виртуальные методы для переопределения в потомках

virtual void inputData() = 0;

virtual void printData() const = 0;

};

// task 2

class MilitaryPerson : public Person {

private:

string rank;

string position;

int serviceMonths;

int rewardsCount;

int penaltiesCount;

public:

MilitaryPerson() : Person(), rank(""), position(""), serviceMonths(0), rewardsCount(0), penaltiesCount(0) {}

// Уникальные методы

void promote() { rewardsCount++; }

void penalize() { penaltiesCount++; }

// Переопределение виртуальных методов

void inputData() override {

cout << "Enter military person's data:\n";

cout << "SNILS: "; cin >> snils;

cout << "Last name: "; cin >> lastName;

cout << "First name: "; cin >> firstName;

cout << "Middle name: "; cin >> middleName;

cout << "Gender: "; cin >> gender;

cout << "Birth date (DD.MM.YYYY): "; cin >> birthDate;

cout << "Rank: "; cin >> rank;

cout << "Position: "; cin >> position;

cout << "Service duration (months): "; cin >> serviceMonths;

}

void printData() const override {

cout << "\nMilitary person data:\n";

cout << "Full name: " << getFullName() << "\n";

cout << "SNILS: " << snils << "\n";

cout << "Birth date: " << birthDate << " (Age: " << getCurrentAge() << " years)\n";

cout << "Rank: " << rank << "\n";

cout << "Position: " << position << "\n";

cout << "Service duration: " << serviceMonths << " months\n";

cout << "Rewards: " << rewardsCount << "\n";

cout << "Penalties: " << penaltiesCount << "\n";

cout << "Seniority (with preferential coefficient): "

<< serviceMonths \* 1.5 << " months\n";

}

// Уникальные геттеры

int getServiceMonths() const { return serviceMonths; }

int getRewardsCount() const { return rewardsCount; }

int getPenaltiesCount() const { return penaltiesCount; }

};

class PlatoonCommander : public Person {

private:

string rank;

string position;

int subordinatesCount;

int monthlyRewards;

int monthlyPenalties;

public:

PlatoonCommander() : Person(), rank(""), position(""), subordinatesCount(0), monthlyRewards(0), monthlyPenalties(0) {}

// Уникальные методы

void rewardSubordinate() { monthlyRewards++; }

void penalizeSubordinate() { monthlyPenalties++; }

// Переопределение виртуальных методов

void inputData() override {

cout << "Enter platoon commander's data:\n";

cout << "SNILS: "; cin >> snils;

cout << "Last name: "; cin >> lastName;

cout << "First name: "; cin >> firstName;

cout << "Middle name: "; cin >> middleName;

cout << "Gender: "; cin >> gender;

cout << "Birth date (DD.MM.YYYY): "; cin >> birthDate;

cout << "Rank: "; cin >> rank;

cout << "Position: "; cin >> position;

cout << "Number of subordinates: "; cin >> subordinatesCount;

}

void printData() const override {

cout << "\nPlatoon commander data:\n";

cout << "Full name: " << getFullName() << "\n";

cout << "SNILS: " << snils << "\n";

cout << "Birth date: " << birthDate << " (Age: " << getCurrentAge() << " years)\n";

cout << "Rank: " << rank << "\n";

cout << "Position: " << position << "\n";

cout << "Number of subordinates: " << subordinatesCount << "\n";

cout << "Monthly rewards: " << monthlyRewards << "\n";

cout << "Monthly penalties: " << monthlyPenalties << "\n";

if (subordinatesCount > 0) {

cout << "Rewards per subordinate: "

<< static\_cast<double>(monthlyRewards) / subordinatesCount << "\n";

cout << "Penalties per subordinate: "

<< static\_cast<double>(monthlyPenalties) / subordinatesCount << "\n";

}

}

// Уникальные геттеры

double getRewardsPerSubordinate() const {

return subordinatesCount > 0 ? static\_cast<double>(monthlyRewards) / subordinatesCount : 0;

}

double getPenaltiesPerSubordinate() const {

return subordinatesCount > 0 ? static\_cast<double>(monthlyPenalties) / subordinatesCount : 0;

}

};

int main() {

// Создаем объекты через указатели на базовый класс

Person\* person1 = new MilitaryPerson();

Person\* person2 = new PlatoonCommander();

// Ввод данных

person1->inputData();

person2->inputData();

// Вывод данных

person1->printData();

person2->printData();

// Приведение типов для вызова специфических методов

MilitaryPerson\* soldier = dynamic\_cast<MilitaryPerson\*>(person1);

if (soldier) {

soldier->promote(); // Поощрение военнослужащего

soldier->printData();

}

PlatoonCommander\* commander = dynamic\_cast<PlatoonCommander\*>(person2);

if (commander) {

commander->rewardSubordinate(); // Поощрение подчиненного

commander->penalizeSubordinate(); // Взыскание подчиненному

commander->printData();

}

delete person1;

delete person2;

return 0;

}

Заключение

Входе данной практической работы были изучены основы полиморфизма в рамках объектно-ориентированной парадигме программирование в языке C++. Была написана программа, которая демонстрирует принципы полиморфизма, программа включает в себя один родительский класс Person, а так же два дочерних класса, в методах которых переопределяются виртуальные функции родительского класса.

Список литературы

1. Кубенский, А. А. Функциональное программирование : учебник и практикум для академического бакалавриата / А. А. Кубенский. — М. : Издательство Юрайт, 2019. — 348 с.
2. Трофимов, В. В. Основы алгоритмизации и программирования : учебник для СПО / В. В. Трофимов, Т. А. Павловская ; под ред. В. В. Трофимова. — М. : Издательство Юрайт, 2019. — 137 с