1 ПОСТАНОВКА ЗАДАЧИ

Создать аналогичный базовый класс «Пассажироперевозчик» и производные классы «Самолёт», «Поезд», «Автомобиль». Определить время и стоимость передвижения.

2 ДИАГРАММА КЛАССОВ

Диаграмма классов представлена на рисунке 2.1.

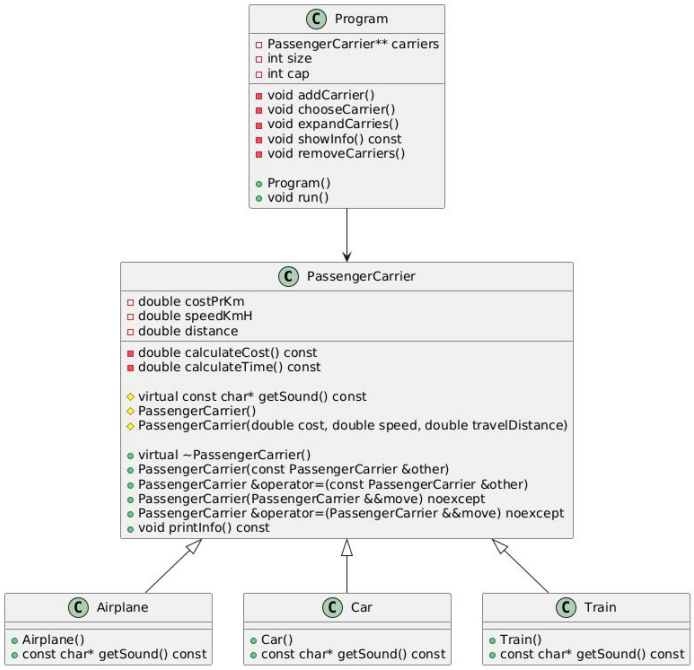

Рисунок 2.1 – Диаграмма классов

3 ЛИСТИНГ КОДА

Файл main.cpp

```
#include "program.h"
#include <iostream>

int main() {
    Program program;

    program.run();
    return 0;
}
```

Файл passenger_carrier.h

```
#pragma once

class PassengerCarrier {
    double costPrKm;
    double speedKmH;
    double distance;

    double calculateCost() const;
    double calculateTime() const;

  protected:
    virtual const char *getSound() const;
    PassengerCarrier();

    explicit PassengerCarrier(double cost, double speed, double
travelDistance);

  public:
    virtual ~PassengerCarrier();

    PassengerCarrier(const PassengerCarrier &other);
    PassengerCarrier &operator=(const PassengerCarrier &other);
    PassengerCarrier(PassengerCarrier &&move) noexcept;
    PassengerCarrier &operator=(PassengerCarrier &&move) noexcept;
    void printInfo() const;
};
```

Файл passenger_carrier.cpp

```
#include "passenger_carrier.h"
#include <iostream>

PassengerCarrier::PassengerCarrier() : costPrKm(0.0), speedKmH(0.0),
distance(0.0) {
}

PassengerCarrier::PassengerCarrier(double cost, double speed, double
travelDistance)
    : costPrKm(cost), speedKmH(speed), distance(travelDistance) {
}

PassengerCarrier::~PassengerCarrier() {
    costPrKm = 0;
    speedKmH = 0;
```

```
    distance = 0;
}

PassengerCarrier::PassengerCarrier(const PassengerCarrier &other) = default;

PassengerCarrier::PassengerCarrier(PassengerCarrier    &&move)    noexcept    =
default;

PassengerCarrier &PassengerCarrier::operator=(const PassengerCarrier &other) =
default;

PassengerCarrier     &PassengerCarrier::operator=(PassengerCarrier     &&move)
noexcept = default;

const char *PassengerCarrier::getSound() const {
    return "NONE";
}

double PassengerCarrier::calculateCost() const {
    return distance / costPrKm;
}

double PassengerCarrier::calculateTime() const {
    return distance / speedKmH;
}

void PassengerCarrier::printInfo() const {
    double cost = calculateCost();
    double time = calculateTime();

    std::cout << getSound() << " The cost(in BYN) per distance travelled is "
<< cost << std::endl;
    std::cout << getSound() << " The time(in Hours) per distance travelled is
" << time << std::endl;
}
```

Файл car.h

```
#pragma once

#include "passenger_carrier.h"

class Car : public PassengerCarrier {
  public:
    const char *getSound() const override;

    Car();
};
```

Файл car.cpp

```
#include "car.h"
#include "consts.h"
#include "utils.h"
#include <iostream>

const char *Car::getSound() const {
    return kCarSound;
}

Car::Car()
```

```
    : PassengerCarrier(getNumber("\nPlease enter car cost (per km): ",
kCarMinCostPrKm, kCarMaxCostPrKm),
                       getNumber("\nPlease enter car speed (km/h): ",
kCarMinSpeedKmH, kCarMaxSpeedKmH),
                       getNumber("\nPlease enter car travel distance (km): ",
kCarMinDistanceKm, kCarMaxDistanceKm)) {
}
```

Файл train.h

```
#pragma once

#include "passenger_carrier.h"

class Train : public PassengerCarrier {
  public:
    const char *getSound() const override;

    Train();
};
```

Файл train.cpp

```
#include "train.h"
#include "consts.h"
#include "utils.h"
#include <iostream>

const char *Train::getSound() const {
    return kTrainSound;
}

Train::Train()
    : PassengerCarrier(
          getNumber("\nPlease enter train cost (per km): ", kTrainMinCostPrKm,
kTrainMaxCostPrKm),
          getNumber("\nPlease enter train speed (km/h): ", kTrainMinSpeedKmH,
kTrainMaxSpeedKmH),
          getNumber("\nPlease enter train travel distance(km): ",
kTrainMinDistanceKm, kTrainMaxDistanceKm)) {
}
```

Файл airplane.h

```
#pragma once

#include "passenger_carrier.h"

class Airplane : public PassengerCarrier {
  public:
    const char *getSound() const override;

    Airplane();
};
```

Файл airplane.cpp

```
#include "airplane.h"
#include "consts.h"
#include "utils.h"
```

```
#include <iostream>

const char *Airplane::getSound() const {
    return kAirplaneSound;
}

Airplane::Airplane()
    : PassengerCarrier(
          getNumber("\nPlease enter airplane cost (per km): ",
kAirplaneMinCostPrKm, kAirplaneMaxCostPrKm),
          getNumber("\nPlease enter airplane speed (km/h): ",
kAirplaneMinSpeedKmH, kAirplaneMaxSpeedKmH),
          getNumber("\nPlease enter airplane travel distance(km): ",
kAirplaneMinDistanceKm, kAirplaneMaxDistanceKm)) {
}
```

Файл menus.h

```
#pragma once

void showTaskMenu();
void showCarriersMenu();
```

Файл menus.cpp

```
#include "menus.h"
#include <iostream>

void showTaskMenu() {
    std::cout << "\n\t\t\t\tTASK" << std::endl;
    std::cout << "Create a base class 'PassengerCarrier' and derived classes
'Airplane', 'Train', 'Car'." << std::endl;
    std::cout << "Define methods to calculate time and cost of travel." <<
std::endl;

    std::cout << "\n\t\t\t\tMENU" << std::endl;
    std::cout << "1. Add a carrier." << std::endl;
    std::cout << "2. Show information (cost and time for a given distance)." <<
std::endl;
    std::cout << "3. Exit the program." << std::endl;
}

void showCarriersMenu() {
    std::cout << "\n\t\t\t\tSELECT CARRIER TYPE" << std::endl;
    std::cout << "1. Car" << std::endl;
    std::cout << "2. Train" << std::endl;
    std::cout << "3. Airplane" << std::endl;
}
```

Файл consts.h

```
#pragma once

inline constexpr const char *kCarSound = "BIP BIP!";
inline constexpr const char *kTrainSound = "CHOOH CHOOH!";
inline constexpr const char *kAirplaneSound = "WHOOOSH!";

inline constexpr const double kCarMinCostPrKm = 0.1;
inline constexpr const double kCarMaxCostPrKm = 5.0;
inline constexpr const double kCarMinSpeedKmH = 10.0;
```

```
inline constexpr const double kCarMaxSpeedKmH = 250.0;
inline constexpr const double kCarMinDistanceKm = 1.0;
inline constexpr const double kCarMaxDistanceKm = 1000.0;

inline constexpr const double kTrainMinCostPrKm = 0.05;
inline constexpr const double kTrainMaxCostPrKm = 1.0;
inline constexpr const double kTrainMinSpeedKmH = 30.0;
inline constexpr const double kTrainMaxSpeedKmH = 350.0;
inline constexpr const double kTrainMinDistanceKm = 10.0;
inline constexpr const double kTrainMaxDistanceKm = 2000.0;

inline constexpr const double kAirplaneMinCostPrKm = 0.5;
inline constexpr const double kAirplaneMaxCostPrKm = 10.0;
inline constexpr const double kAirplaneMinSpeedKmH = 300.0;
inline constexpr const double kAirplaneMaxSpeedKmH = 1200.0;
inline constexpr const double kAirplaneMinDistanceKm = 100.0;
inline constexpr const double kAirplaneMaxDistanceKm = 10000.0;

inline constexpr const char *kWhiteColor = "\o{33}[0m";
inline constexpr const char *kRedColor = "\o{33}[31m";
inline constexpr const char *kGreenColor = "\o{33}[32m";
```

Файл program.h

```
#pragma once

#include "passenger_carrier.h"

class Program {
    PassengerCarrier **carriers = nullptr;
    int size = 0;
    int cap = 1;

    void addCarrier();
    void chooseCarrier();
    void expandCarries();
    void showInfo() const;
    void removeCarriers();

  public:
    Program();

    void run();
};
```

Файл program.cpp

```
#include "program.h"
#include "airplane.h"
#include "car.h"
#include "consts.h"
#include "menus.h"
#include "train.h"
#include "utils.h"
#include <iostream>

Program::Program() = default;

void Program::addCarrier() {
    expandCarries();
```

```
    chooseCarrier();

    std::cout << kGreenColor << "\nYou have successfully added carrier!" <<
kWhiteColor << std::endl;
}

void Program::expandCarries() {
    if (size < 0) {
        size = 0;
    }

    if (size + 1 == cap) {
        cap *= 2;
        auto **new_carriers = new PassengerCarrier *[cap];

        for (int i = 0; i < size; i++) {
            new_carriers[i] = carriers[i];
        }

        delete[] carriers;

        carriers = new_carriers;
    }

    size++;
}

void Program::chooseCarrier() {
    int opt = 0;

    showCarriersMenu();

    while (true) {
        opt = getNumber("\nPlease enter carriers menu option ", 1, 3);

        switch (opt) {
        case 1:
            carriers[size - 1] = new Car;
            return;
        case 2:
            carriers[size - 1] = new Train;
            return;
        case 3:
            carriers[size - 1] = new Airplane;
            return;
        default:
            std::cout << kRedColor << "\nError, you picked is an incorrect
carrier menu option. Please try again."
                      << kWhiteColor << std::endl;
        }
    }
}

void Program::showInfo() const {
    if (size == 0) {
        std::cout << kRedColor << "\nError, you have not added any carriers
yet. Please use option 1 to add a carrier."
                  << kWhiteColor << std::endl;
        return;
    }

    std::cout << "\n\t\t\t\tINFO" << std::endl;

    for (int i = 0; i < size; i++) {
```

```
        carriers[i]->printInfo();
        std::cout << std::endl;
    }

    std::cout << kGreenColor << "\nYou have successfully showed all info about
carriers!" << kWhiteColor << std::endl;
}

void Program::removeCarriers() {
    for (int i = 0; i < size; i++) {
        delete carriers[i];
    }

    delete[] carriers;
    carriers = nullptr;
    size = 0;
    cap = 0;
}

void Program::run() {
    int opt = 0;

    system("clear");
    showTaskMenu();

    while (true) {
        opt = getNumber("\nPlease enter menu option ", 1, 3);

        switch (opt) {
        case 1:
            addCarrier();
            break;
        case 2:
            showInfo();
            break;
        case 3:
            removeCarriers();
            std::cout << kGreenColor << "\nYou have successfully exited the
program." << kWhiteColor << std::endl;
            return;
        default:
            std::cout << kRedColor << "\nError, you picked is an incorrect menu
option. Please try again."
                      << kWhiteColor << std::endl;
        }
    }
}
```

4 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

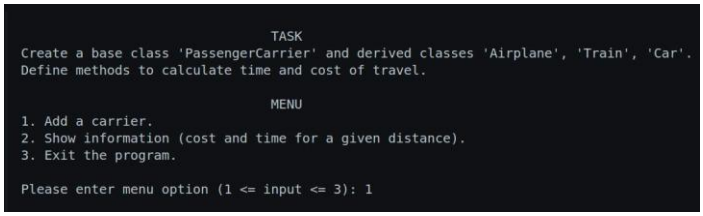

Рисунок 4.1 – Описание задания и главное меню программы

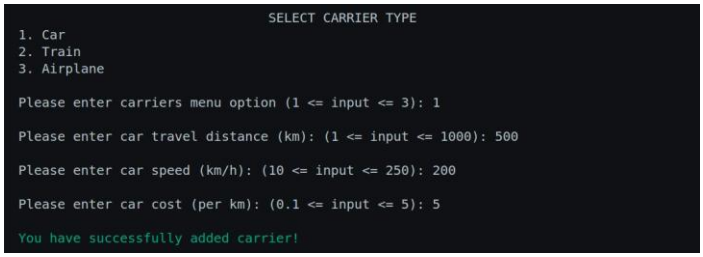

Рисунок 4.2 – Меню выбора пассажироперевозчика и добавление машины

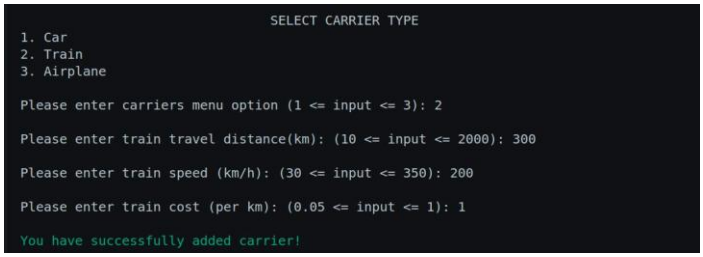

Рисунок 4.3 – Меню выбора пассажироперевозчика и добавление поезда

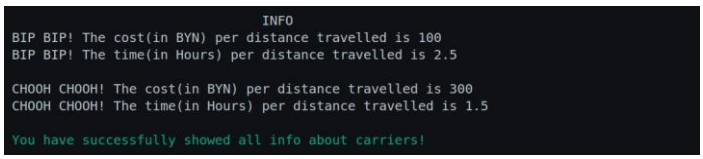

Рисунок 4.4 – Вывод всей информации о пассажироперевозчиках

Рисунок 4.5 – Завершение программы

5 ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы были закреплены навыки объектно-ориентированного программирования в C++ с использованием принципов наследования и полиморфизма. Был разработан базовый класс «Пассажироперевозчик», а также производные классы «Самолёт», «Поезд» и «Автомобиль», реализующие общие и специфические характеристики различных видов транспорта. В классах были определены методы для расчёта времени и стоимости передвижения в зависимости от заданных параметров. Проведённое тестирование подтвердило корректность работы иерархии классов, правильность переопределения виртуальных функций и корректное взаимодействие базового и производных классов.