**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И.Раззакова**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №4**

Выполнил: студент группы ПИ-3-21

Сагыналиев Таалайбек

Проверил: Мусабаев Э.Б.

**Бишкек 2024**

# **Задание №1**

**Постановка задачи:**

Для участия в ежегодном трансконтинентальном Ралли-марафоне «Дакар» по усложненной трассе в Южной Америке выбраны грузовики КАМАЗ и Tatra, которые после 2000 года лидируют в зачете грузовиков.

Решите задачу сравнения скоростей движения грузовиков по разным по проходимости участкам трассы, а именно: по равнине, горам, пустыне. Создайте и выдайте на экран таблицу результатов ралли - марафона. Определите победителя.

Для решения задачи используйте классы **Kamaz** и **Tatra,** а такжефункциюсравнения скоростей **FrCreater (**). Функция  **FrCreater (**) возвращает число +1, если объект **kamaz** движется быстрее объекта **tatra**; нуль, если их скорости одинаковы; число -1, если объект **kamaz** движется медленнее объекта **tatra** .

Оба класса содержат поля: «скорость» и «наименование» грузовой машины, а также методы: инициализация и отображение полей на экране. Определитесь с идентификаторами доступа к членам класса, не нарушая принципа инкапсуляции.

**Исходные данные:**

**Исходный код программы на C++:**

#include <iostream>

#include <string>

// Класс для представления грузовика Камаз

class Kamaz {

private:

int speed; // Скорость грузовика

std::string name; // Наименование грузовика

public:

// Конструктор класса Kamaz

Kamaz(int spd, std::string n) : speed(spd), name(n) {}

// Метод для получения скорости грузовика

int getSpeed() const {

return speed;

}

// Метод для вывода информации о грузовике на экран

void display() const {

std::cout << "Камаз '" << name << "': скорость " << speed << " км/ч" << std::endl;

}

};

// Класс для представления грузовика Tatra

class Tatra {

private:

int speed; // Скорость грузовика

std::string name; // Наименование грузовика

public:

// Конструктор класса Tatra

Tatra(int spd, std::string n) : speed(spd), name(n) {}

// Метод для получения скорости грузовика

int getSpeed() const {

return speed;

}

// Метод для вывода информации о грузовике на экран

void display() const {

std::cout << "Татра '" << name << "': скорость " << speed << " км/ч" << std::endl;

}

};

// Функция для сравнения скоростей грузовиков Камаз и Tatra

int FrCreater(const Kamaz& kamaz, const Tatra& tatra) {

if (kamaz.getSpeed() > tatra.getSpeed()) {

return 1; // Камаз быстрее

} else if (kamaz.getSpeed() < tatra.getSpeed()) {

return -1; // Татра быстрее

} else {

return 0; // Скорости равны

}

}

int main() {

// Создание объектов грузовиков для различных участков трассы

setlocale(LC\_ALL, "ru");

Kamaz kamazPlain(120, "Камаз равнина 1");

Tatra tatraPlain(110, "Татра равнина 2");

Kamaz kamazMountains(90, "Камаз горы 1");

Tatra tatraMountains(80, "Татра горы 2");

Kamaz kamazDesert(120, "Камаз пустыня 1");

Tatra tatraDesert(115, "Татра пустыня 2");

// Вывод результатов сравнения для каждого участка трассы

std::cout << "Результаты ралли-марафона:" << std::endl;

std::cout << "-------------------------------------" << std::endl;

// Равнина

std::cout << "Равнина:" << std::endl;

kamazPlain.display();

tatraPlain.display();

int plainResult = FrCreater(kamazPlain, tatraPlain);

if (plainResult > 0) {

std::cout << "Победитель: Камаз" << std::endl;

} else if (plainResult < 0) {

std::cout << "Победитель: Татра" << std::endl;

} else {

std::cout << "Ничья" << std::endl;

}

std::cout << std::endl;

// Горы

std::cout << "Горы:" << std::endl;

kamazMountains.display();

tatraMountains.display();

int mountainsResult = FrCreater(kamazMountains, tatraMountains);

if (mountainsResult > 0) {

std::cout << "Победитель: Камаз" << std::endl;

} else if (mountainsResult < 0) {

std::cout << "Победитель: Татра" << std::endl;

} else {

std::cout << "Ничья" << std::endl;

}

std::cout << std::endl;

// Пустыня

std::cout << "Пустыня:" << std::endl;

kamazDesert.display();

tatraDesert.display();

int desertResult = FrCreater(kamazDesert, tatraDesert);

if (desertResult > 0) {

std::cout << "Победитель: Камаз" << std::endl;

} else if (desertResult < 0) {

std::cout << "Победитель: Татра" << std::endl;

} else {

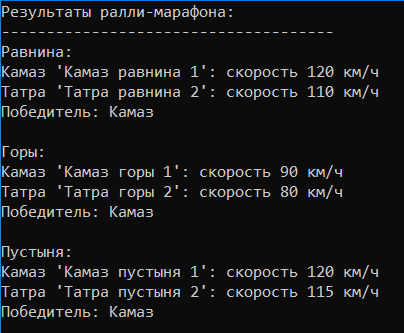
std::cout << "Ничья" << std::endl;

}

return 0;

}

Тесты:



# **Задание №2**

**Постановка задачи:**

Сделайте функцию **FrCreater(**) дружественной: классу **Kamaz,** обоим классам **Kamaz** и **Tatra.**

**Исходные данные:**

**Исходный код программы на C++:**

#include <iostream>

#include <string>

using namespace std;

class Tatra; // Предварительное объявление класса Tatra, чтобы FrCreater() мог использовать его

// Класс для представления грузовика Камаз

class Kamaz {

private:

int speed;

string name;

public:

// Конструктор класса Kamaz

Kamaz(int spd, string n) : speed(spd), name(n) {}

// Метод для получения скорости грузовика

int getSpeed() const {

return speed;

}

// Метод для получения названия модели грузовика

string getName() const {

return name;

}

// Метод для вывода информации о грузовике на экран

void display() const {

cout << "Камаз: Модель " << name << ", скорость: " << speed << " км/ч" << std::endl;

}

// Объявление FrCreater() в качестве дружественной функции класса Kamaz

friend string FrCreater(const Kamaz& kamaz, const Tatra& tatra);

};

// Класс для представления грузовика Tatra

class Tatra {

private:

int speed;

string name;

public:

// Конструктор класса Tatra

Tatra(int spd, string n) : speed(spd), name(n) {}

// Метод для получения скорости грузовика

int getSpeed() const {

return speed;

}

// Метод для получения названия модели грузовика

string getName() const {

return name;

}

// Метод для вывода информации о грузовике на экран

void display() const {

cout << "Татра: Модель " << name << ", скорость: " << speed << " км/ч" << std::endl;

}

// Объявление FrCreater() в качестве дружественной функции класса Tatra

friend string FrCreater(const Kamaz& kamaz, const Tatra& tatra);

};

// Функция для сравнения скоростей грузовиков Камаз и Tatra

string FrCreater(const Kamaz& kamaz, const Tatra& tatra) {

if (kamaz.getSpeed() > tatra.getSpeed()) {

return "Камаз быстрее";

} else if (kamaz.getSpeed() < tatra.getSpeed()) {

return "Татра быстрее";

} else {

return "Скорости равны";

}

}

int main() {

setlocale(LC\_ALL, "ru");

// Создание объектов грузовиков для различных участков трассы

Kamaz kamaz(120, "Камаз 005");

Tatra tatra(110, "Tатра 003");

// Вывод результатов сравнения скоростей

cout << "Результат: " << FrCreater(kamaz, tatra) << std::endl;

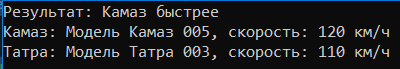
kamaz.display();

tatra.display();

return 0;

}

Тесты:



# **Задание №3**

**Постановка задачи:**

**Опишите класс, реализующий стек (Stack).**

Сделайте класс **Tatra** дружественным классу **Kamaz.**

**Исходные данные:**

**Исходный код программы на C++:**

#include <iostream>

#include <string>

class Kamaz; // Предварительное объявление класса Kamaz, чтобы Tatra могла использовать его

// Класс для представления грузовика Tatra

class Tatra {

private:

int speed; // Скорость грузовика

std::string name; // Наименование грузовика

public:

// Конструктор класса Tatra

Tatra(int spd, std::string n) : speed(spd), name(n) {}

// Метод для получения скорости грузовика

int getSpeed() const {

return speed;

}

// Метод для получения наименования грузовика

std::string getName() const {

return name;

}

// Метод для вывода информации о грузовике на экран

void display() const {

std::cout << "Татра '" << name << "': скорость " << speed << " км/ч" << std::endl;

}

// Объявление дружественности с классом Kamaz

friend class Kamaz;

};

// Класс для представления грузовика Kamaz

class Kamaz {

private:

int speed; // Скорость грузовика

std::string name; // Наименование грузовика

public:

// Конструктор класса Kamaz

Kamaz(int spd, std::string n) : speed(spd), name(n) {}

// Метод для получения скорости грузовика

int getSpeed() const {

return speed;

}

// Метод для получения наименования грузовика

std::string getName() const {

return name;

}

// Метод для вывода информации о грузовике на экран

void display() const {

std::cout << "Камаз '" << name << "': скорость " << speed << " км/ч" << std::endl;

}

};

int main() {

setlocale(LC\_ALL, "ru");

// Создание объектов грузовиков

Kamaz kamaz(120, "Какмаз 005");

Tatra tatra(110, "TATRA 003");

// Вывод информации о грузовиках

kamaz.display();

tatra.display();

return 0;

}

Тесты:



# **Ответы на вопросы:**

**1. Что такое дружественная функция, для чего она нужна?**

**Дружественная функция**—это функция, не являющаяся членом класса, но обеспечивающая доступ ко всем членам класса, для которого она дружественна. Независима от класса **дружественная функция** находится вне класса. Но она объявляется внутри класса как обычная функция и перед ней указывается ключевое слово **friend**. Вне класса перед описанием дружественной функции это ключевое слово не ставится**.**

**2. Как она объявляется, каков её синтаксис?**

Синтаксис прототипа дружественной функции

***friend*** тип имя ([список\_ параметров])

**3. Через что дружественная функция может получить доступ к закрытому полю класса?**

Так как дружественная функция не является методом класса, для которого она дружественна, нельзя вызвать дружественную функцию обычным способом в точечном формате с помощью команды **a.show()**, а можно только командой **show(a).** Так как, хотя дружественная функция "знает" о закрытом поле класса, для которого она является дружественной, доступ к нему она может получить только через объект этого класса, который объявлен внутри функции и передан ей в качестве аргумента. ДФ вызывается для объекта класса.

**4. Какой вариант вызова ДФ правильный:**

a.show() или show(a)?

Если функция дружественная, то второй вариант.

**5. Наследуется ли дружественная функция?**

Дружественная функция не наследуется. То есть, если в базовый класс дружественная функция включается, то она не является таковой для производных классов.

**6. Может ли дружественная функция быть дружественной более чем к одному классу?**

Дружественная функция может быть дружественной более чем к одному классу.

**7. Для решения каких задачах есть смысл использовать функцию, дружественную двум и более классам?**

Такая дружественная функция полезна тогда, когда у двух и более разных классов имеется нечто общее, с чем необходимо произвести какие-то действия, например, сравнение или арифметические действия закрытых полей объектов, принадлежащих разным классам.

**8. Когда нужно использовать дружественные классы?**

ДК используются тогда, когда классу необходим доступ к закрытым данным другого класса, чтобы работать с ними своими методами.

**9. Какому принципу ООП не соответствуют дружественные функции?**

Инкапсуляция.

**10. Как можно ограничить использование ДФ?**

Не использовать их, использовать открытые методы класса. Или каждый раз наследуя один класс от дружественного, в таком случае, ДФ не наследуется.

**11. В каких крайних случаях нужно использовать ДФ?**

Если требуется для базового класса сделать доступ к приватным данным, а для наследуемых нет. В этом случае играет роль то, что дружественные функции не наследуются.