Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им. И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление: 710400 «Программная инженерия»

Дисциплина: « Объектно-ориентрованное проектирование / Объектно-ориентрованное программирование»

**ОТЧЕТ**

По лабораторной работе №4

Тема: «Дружественные функции и дружественные классы»

Выполнила: студентка группы

ПИ(б)-3-21 Пак Ксения

Проверил: Мусабаев Э. Б.

Бишкек – 2024

Задания

1. Для участия в ежегодном трансконтинентальном Ралли-марафоне «Дакар» по усложненной трассе в Южной Америке выбраны грузовики [КАМАЗ](https://ru.wikipedia.org/wiki/%D0%9A%D0%90%D0%9C%D0%90%D0%97) и [Tatra](https://ru.wikipedia.org/wiki/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.

#include <iostream>

#include <string>

#include <Windows.h>

class Kamaz {

private:

std::string name;

double speed;

public:

Kamaz(const std::string& truckName, double truckSpeed) : name(truckName), speed(truckSpeed) {}

void displayInfo() const {

std::cout << "Kamaz " << name << ": Speed - " << speed << " km/h\n";

}

double getSpeed() const {

return speed;

}

};

class Tatra {

private:

std::string name;

double speed;

public:

Tatra(const std::string& truckName, double truckSpeed) : name(truckName), speed(truckSpeed) {}

void displayInfo() const {

std::cout << "Tatra " << name << ": Speed - " << speed << " km/h\n";

}

double getSpeed() const {

return speed;

}

};

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

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

return 1;

}

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

return -1;

}

else {

return 0;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Kamaz kamaz("Master", 80.0);

Tatra tatra("Phoenix", 75.0);

kamaz.displayInfo();

tatra.displayInfo();

int result = FrCreater(kamaz, tatra);

if (result == 1) {

std::cout << "Kamaz победил!\n";

}

else if (result == -1) {

std::cout << "Tatra победил!\n";

}

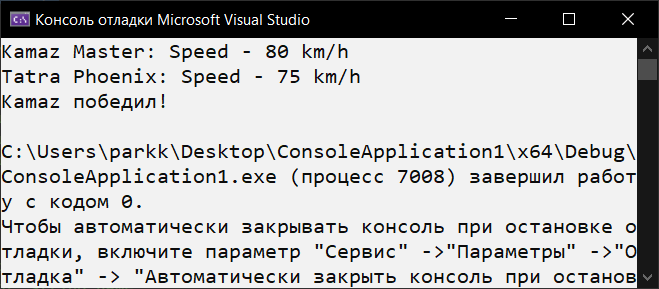
else {

std::cout << "Ничья! Скорости равны.\n";

}

return 0;

}



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

#include <iostream>

#include <string>

#include <Windows.h>

class Tatra;

class Kamaz {

private:

std::string name;

double speed;

public:

Kamaz(const std::string& truckName, double truckSpeed) : name(truckName), speed(truckSpeed) {}

void displayInfo() const {

std::cout << "Kamaz " << name << ": Speed - " << speed << " km/h\n";

}

double getSpeed() const {

return speed;

}

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

};

class Tatra {

private:

std::string name;

double speed;

private:

double getSpeed() const {

return speed;

}

public:

Tatra(const std::string& truckName, double truckSpeed) : name(truckName), speed(truckSpeed) {}

void displayInfo() const {

std::cout << "Tatra " << name << ": Speed - " << speed << " km/h\n";

}

/\*double getSpeed() const {

return speed;

}\*/

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

};

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

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

return 1;

}

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

return -1;

}

else {

return 0;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Kamaz kamaz("Master", 80.0);

Tatra tatra("Phoenix", 75.0);

kamaz.displayInfo();

tatra.displayInfo();

int result = FrCreater(kamaz, tatra);

if (result == 1) {

std::cout << "Kamaz победил!\n";

}

else if (result == -1) {

std::cout << "Tatra победил!\n";

}

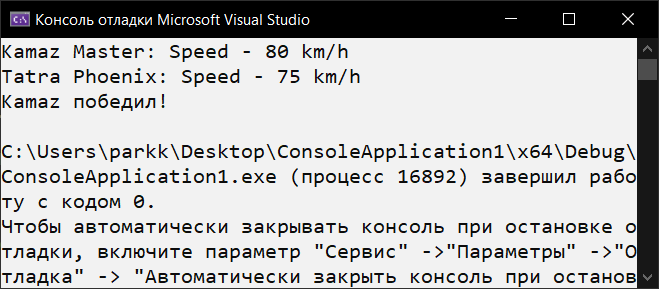
else {

std::cout << "Ничья! Скорости равны.\n";

}

return 0;

}



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

#include <iostream>

#include <string>

#include <Windows.h>

class Tatra;

class Kamaz {

private:

std::string name;

double speed;

public:

Kamaz(const std::string& truckName, double truckSpeed) : name(truckName), speed(truckSpeed) {}

void displayInfo() const {

std::cout << "Kamaz " << name << ": Speed - " << speed << " km/h\n";

}

double getSpeed() const {

return speed;

}

friend class Tatra;

};

class Tatra {

private:

std::string name;

double speed;

private:

double getSpeed() const {

return speed;

}

public:

Tatra(const std::string& truckName, double truckSpeed) : name(truckName), speed(truckSpeed) {}

void displayInfo() const {

std::cout << "Tatra " << name << ": Speed - " << speed << " km/h\n";

}

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

};

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

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

return 1;

}

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

return -1;

}

else {

return 0;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Kamaz kamaz("Master", 80.0);

Tatra tatra("Phoenix", 75.0);

kamaz.displayInfo();

tatra.displayInfo();

int result = FrCreater(kamaz, tatra);

if (result == 1) {

std::cout << "Kamaz победил!\n";

}

else if (result == -1) {

std::cout << "Tatra победил!\n";

}

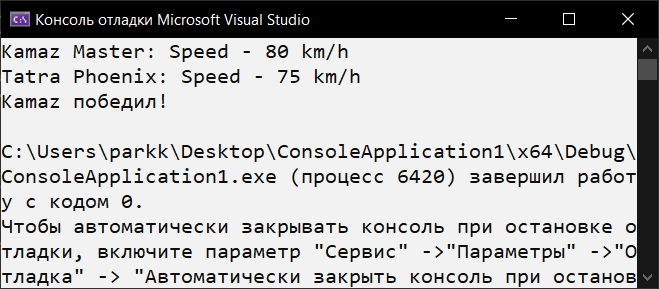
else {

std::cout << "Ничья! Скорости равны.\n";

}

return 0;

}



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.

1. Как она объявляется, каков её синтаксис?

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**.**

**Синтаксис прототипа дружественной функции**

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

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

Чтобы внешняя функция имела доступ к закрытому полю, она объявлена, как дружественная. Для этого в описании класса добавлен её прототип friend.

1. Какой вариант вызова ДФ правильный: a.show() или show(a)?

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).**

1. 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.

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

Дружественная функция может быть дружественной более чем к одному классу.

1. 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.

1. Когда нужно использовать дружественные классы?

Часто дружественные классы используются тогда, когда классу необходим доступ к закрытым данным другого класса.

1. Какому принципу ООП не соответствуют дружественные функции?

С одной стороны, дружественные функции повышают гибкость языка, но, с другой стороны, они не соответствуют принципу ограничения доступа к данным, в соответствии с которым только методы класса могут иметь доступ к сокрытым данным класса.

1. Как можно ограничить использование ДФ?

Определение только внутри класса: Дружественные функции могут быть определены только внутри класса. Это ограничит доступ к ним извне.

Декларация в публичном интерфейсе: Дружественные функции должны быть объявлены в публичной части интерфейса класса. Это позволит контролировать, какие функции могут быть дружественными.

Ограничение доступа: В классе можно использовать модификаторы доступа (public, private, protected) для контроля доступа к функциям.

1. В каких крайних случаях нужно использовать ДФ?

Доступ к закрытым данным: Когда необходимо предоставить ограниченный доступ к закрытым членам класса другой функции или классу.

Перегрузка операторов: В случае перегрузки операторов, когда нужно предоставить доступ к закрытым членам класса для выполнения операции.

Функции, не являющиеся методами класса: Когда функция логически связана с классом, но не является его методом.

Обеспечение интерфейса с другими классами: В некоторых случаях, чтобы предоставить специфический доступ к классу другим классам без использования наследования.