**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 №5**

Выполнил: студент группы ПИ-3-21

Урманбетов Султан

Проверил: Мусабаев Э.Б.

**Бишкек 2024**

# **Задание №1-3**

**Постановка задачи:**

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.

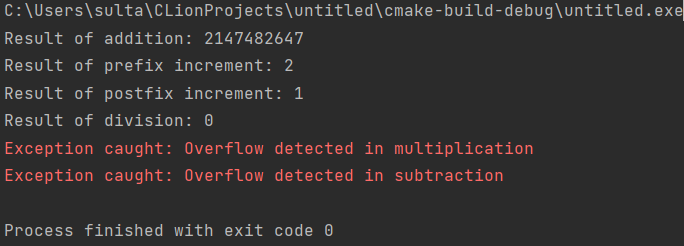
Для решения задачи используйте классы **Kamaz** и **Tatra,** а такжефункциюсравнения скоростей **FrCreater (**). Функция  **FrCreater (**) возвращает число +1, если объект **kamaz** движется быстрее объекта **tatra**; нуль, если их скорости одинаковы; число -1, если объект **kamaz** движется медленнее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

**Исходный код программы на C++:**

#include <iostream>  
#include <limits>  
  
using namespace std;  
class Int {  
private:  
 int value;  
  
public:  
 Int(int val = 0) : value(val) {}  
  
 // Overloaded arithmetic operators  
 Int operator+(const Int& other) const {  
 if ((long double)value + other.value > numeric\_limits<int>::max() ||  
 (long double)value + other.value < numeric\_limits<int>::min()) {  
 throw overflow\_error("Overflow detected in addition");  
 }  
 return Int(value + other.value);  
 }  
  
 Int operator-(const Int& other) const {  
 if ((long double)value - other.value > numeric\_limits<int>::max() ||  
 (long double)value - other.value < numeric\_limits<int>::min()) {  
 throw overflow\_error("Overflow detected in subtraction");  
 }  
 return Int(value - other.value);  
 }  
  
 Int operator\*(const Int& other) const {  
 if ((long double)value \* other.value > numeric\_limits<int>::max() ||  
 (long double)value \* other.value < numeric\_limits<int>::min()) {  
 throw overflow\_error("Overflow detected in multiplication");  
 }  
 return Int(value \* other.value);  
 }  
  
 Int operator/(const Int& other) const {  
 if (other.value == 0) {  
 throw invalid\_argument("Division by zero");  
 }  
 return Int(value / other.value);  
 }  
  
 // Prefix increment operator  
 Int& operator++() {  
 if (value == numeric\_limits<int>::max()) {  
 throw overflow\_error("Overflow detected in prefix increment");  
 }  
 ++value;  
 return \*this;  
 }  
  
 // Postfix increment operator  
 Int operator++(int) {  
 if (value == numeric\_limits<int>::max()) {  
 throw overflow\_error("Overflow detected in postfix increment");  
 }  
 Int temp(\*this);  
 ++value;  
 return temp;  
 }  
  
 // Output operator  
 friend ostream& operator<<(ostream& os, const Int& obj) {  
 return os << obj.value;  
 }  
};  
  
int main() {  
 Int a(-1000); // max value of int = 2147483647  
 Int a1(1);  
 Int a2(1);  
 Int b(2147483647);  
  
 try {  
 Int c = a + b;  
 cout << "Result of addition: " << c << endl;  
 } catch (const exception& e) {  
 cerr << "Exception caught: " << e.what() << endl;  
 }  
  
 try {  
 Int d = a \* b;  
 cout << "Result of multiplication: " << d << endl;  
 } catch (const exception& e) {  
 cerr << "Exception caught: " << e.what() << endl;  
 }  
  
 try {  
 Int e = ++a1;  
 cout << "Result of prefix increment: " << e << endl;  
 } catch (const exception& e) {  
 cerr << "Exception caught: " << e.what() << endl;  
 }  
  
 try {  
 Int f = a2++;  
 cout << "Result of postfix increment: " << f << endl;  
 } catch (const exception& e) {  
 cerr << "Exception caught: " << e.what() << endl;  
 }  
 try {  
 Int f = a-b;  
 cout << "Result of subtraction: " << f << endl;  
 } catch (const exception& e) {  
 cerr << "Exception caught: " << e.what() << endl;  
 }  
 try {  
 Int f = a/b;  
 cout << "Result of division: " << f << endl;  
 } catch (const exception& e) {  
 cerr << "Exception caught: " << e.what() << endl;  
 }  
 return 0;  
}

**Тест:**

****

# **Ответы на вопросы:**