МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

Кафедра «Информационные системы»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине

«Технологии программирования. Объектно-ориентированный подход»

Вариант 8

Выполнил:

Донец Н.О.

Проверил:

Сметанина Т. И.

Севастополь

2024 г.

**1 ЛАБОРАТОРНАЯ РАБОТА №1**

**ИССЛЕДОВАНИЕ ОСОБЕННОСТЕЙ ОБЪЕКТНОГО**

**ПОДХОДА ПРИ ПРОГРАММИРОВАНИИ СЛОЖНЫХ**

**СТРУКТУР ДАННЫХ С ДИНАМИЧЕСКИМИ ПОЛЯМИ**

**Цель работы:**

Исследование основных средств определения класса, создания объектов класса, приобретение навыков разработки и отладки программ, использующих динамическую память. Исследование особенностей использования конструкторов копирования.

**Задание:**

Разработать программу на языке C++. Работа программы должна

быть реализована в виде меню со следующими пунктами:

- создание объекта;

- создание копии объекта (используя конструктор копирования);

- просмотр полей оригинала и копии;

- изменение одного из полей объекта (копии или оригинала);

- выполнение задания согласно варианту.

- описать класс — циклическая очередь. Хранит информацию

о штрафах: номер автомобиля, марка и величина штрафа. Предусмотреть функции добавления элементов в очередь и удаления из нее, а также функцию вычисления величины суммы штрафов со всех авто.

**Разработка программы:**

Был разработан класс циклической очереди (Листинг 1).

Листинг 1 – Класс циклической очереди

#include <iostream>

template <typename T>

class CircularQueue {

private:

std::size\_t size;

std::size\_t head;

std::size\_t tail;

T\*\* queue;

public:

void enqueue(const T& element);

void dequeue();

T\* front();

T\* back();

void print();

CircularQueue(size\_t \_size): size(\_size) {

std::cout<<"Standart constructor queue"<<std::endl;

head = 0;

tail = -1;

queue = new T\*[size];

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

queue[i] = NULL;

}

}

CircularQueue(const CircularQueue& \_queue): size(\_queue.size), head(\_queue.head), tail(\_queue.tail) {

std::cout<<"Copy constructor queue"<<std::endl;

queue = new T\*[size];

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

queue[i] = new T(\*(\_queue.queue[i]));

}

}

~CircularQueue() {

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

delete queue[i];

delete[] queue;

std::cout<<"Queue destructor"<<std::endl;

}

};

template <typename T>

void CircularQueue<T>::enqueue(const T& element) {

if (tail == size-1) {

tail = 0;

delete queue[tail];

}

else {

tail++;

}

queue[tail] = new T(element);

}

template <typename T>

void CircularQueue<T>::dequeue() {

if (tail < 0) return;

if (queue[head] == NULL) return;

delete queue[head];

queue[head++] = NULL;

if (head == size) {

head = 0;

}

}

template <typename T>

T\* CircularQueue<T>::front() {

if (head < 0) return NULL;

return queue[head];

}

template <typename T>

T\* CircularQueue<T>::back() {

if (tail < 0) return NULL;

return queue[tail];

}

template <typename T>

void CircularQueue<T>::print() {

for (int i = head, j = 0; queue[i] != NULL && j < size; i < size-1 ? i++ : i=0, j++) {

if (queue[i] != NULL) queue[i]->print();

}

}

Также был разработан класс данных согласно варианту (Листинг 2).

Листинг 2 – Класс штрафа

#include <iostream>

class Penalty {

public:

std::string number;

std::string brand;

int amount;

void print() {

std::cout<<"number: "<<number<<" brand: "<<brand<<" amount: "<<amount<<std::endl;

}

Penalty(int \_amount = 0, std::string \_number = "h666el", std::string \_brand = "bugatti"): amount(\_amount), number(\_number), brand(\_brand) {

std::cout<<"Standart constructor penalty"<<std::endl;

}

Penalty(const Penalty& \_penalty): number(\_penalty.number), brand(\_penalty.brand), amount(\_penalty.amount) {

std::cout<<"Copy constructor penalty"<<std::endl;

}

~Penalty() {

std::cout<<"Penalty destructor"<<std::endl;

}

static Penalty createPenalty() {

std::string \_number;

std::string \_brand;

int \_amount;

std::cout<<"number: "; std::cin>>\_number;

std::cout<<"brand: "; std::cin>>\_brand;

std::cout<<"amount: "; std::cin>>\_amount;

Penalty newPenalty(\_amount, \_number, \_brand);

return newPenalty;

}

};

Была разработана программа, решающая поставленную задачу с использованием написанных ранее классов (Листинг 3).

Листинг 3 – Основная программа

#include "CircularQueue.cpp"

#include "Penalty.cpp"

#include <string>

int summarizePenaltys(CircularQueue<Penalty>& queue);

void checkQueue();

void printMenu();

int getChoice();

void callFunctions(int choice);

void changeField();

const int size = 9;

CircularQueue<Penalty> queueMain(size);

int main() {

int choice = 0;

do {

printMenu();

choice = getChoice();

callFunctions(choice);

} while(choice);

}

int summarizePenaltys(CircularQueue<Penalty>& queue) {

int sum = 0;

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

sum += queue.front()->amount;

std::cout<<"sum: "<<sum<<" amount: "<<queue.front()->amount<<std::endl;

queue.dequeue();

}

return sum;

}

void printMenu() {

std::cout<<"\n";

std::cout<<"0 - Exit"<<std::endl;

std::cout<<"1 - Create obj"<<std::endl;

std::cout<<"2 - Copy obj"<<std::endl;

std::cout<<"3 - Print fields"<<std::endl;

std::cout<<"4 - Change field"<<std::endl;

std::cout<<"5 - Check queue"<<std::endl;

std::cout<<"\n";

}

int getChoice() {

std::string input = "0";

int choice;

try {

std::cin>>input;

}

catch (...) {

choice = 0;

}

try {

choice = stoi(input);

}

catch (...) {

choice = 0;

}

return choice;

}

void callFunctions(int choice) {

switch(choice) {

case 1:

queueMain.enqueue(Penalty::createPenalty());

break;

case 2:

queueMain.enqueue(\*(queueMain.front()));

break;

case 3:

std::cout<<"Queue main"<<std::endl;

queueMain.print();

break;

case 4:

changeField();

break;

case 5:

checkQueue();

break;

}

}

void changeField() {

std::cout<<"1 - Change first element amount 2 - change last element amount"<<std::endl;

int choice = getChoice();

switch (choice) {

case 1:

std::cin>>queueMain.front()->amount;

break;

case 2:

std::cin>>queueMain.back()->amount;

}

}

void checkQueue() {

CircularQueue<Penalty> queue1(size);

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

queue1.enqueue(i);

}

CircularQueue<Penalty> queue2 = queue1;

queue1.print();

queue2.print();

std::cout<<summarizePenaltys(queue1)<<std::endl;

std::cout<<summarizePenaltys(queue2)<<std::endl;

}

Были проведены тесты разработанной программы (рисунки 1 – 3).

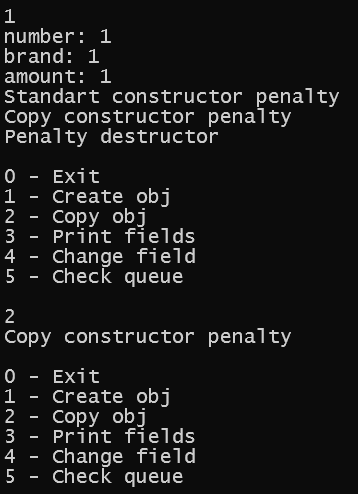


Рисунок 1- Первый тест

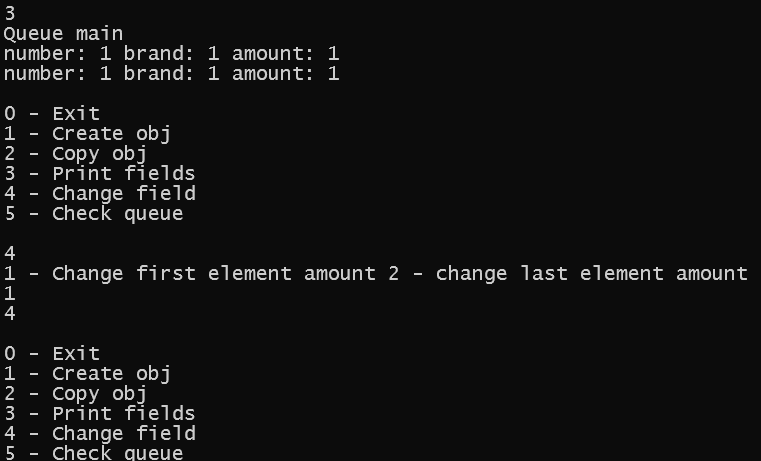


Рисунок 2 – Тест второй

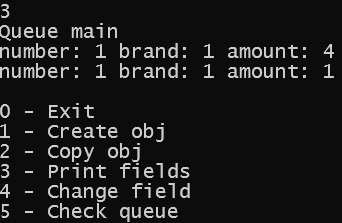


Рисунок 3 – Третий тест

**Вывод:**

В ходе лабораторной работы были исследованы основные средства определения класса, создания объектов класса, были приобретены навыки разработки и отладки программ, использующих динамическую память. Были исследованы особенности использования конструкторов копирования.