Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3
По дисциплине «Надежность программного обеспечения»
Тема: «Реализация и анализ отказоустойчивости»

Выполнил:
Студент 3 курса
Группы ПО-11
Хведорец В. С.
Проверил:
Козик И. Д.

Брест 2025

Цель работы: Разработка отказоустойчивой системы и анализ ее поведения при сбоях.

Вариант 21

Резервирование данных в оперативной памяти: Реализуйте программу, которая сохраняет данные в оперативной памяти и на диске. При сбое данные восстанавливаются с диска.

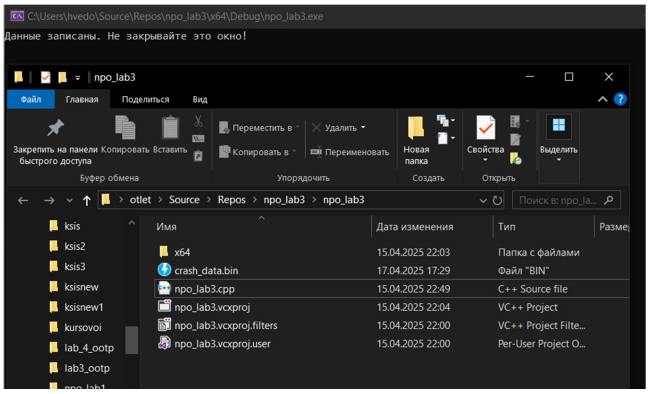
Код программы:

```
#include <iostream>
#include <fstream>
#include <string>
#include <unordered_map>
#include <stdexcept>
#include <mutex>

class FaultTolerantStorage {
private:
    std::unordered_map<std::string, std::string> memoryCache;
    std::string backupFilename;
    std::mutex mtx;
    // внутренняя функция без блокировки
    void unsafeSaveToDisk() {
        std::ofstream outFile(backupFilename, std::ios::binary);
        if (!outFile) throw std::runtime_error("File write error");
        for (const auto& [key, value] : memoryCache) {
            size_t keySize = key.size();
            size_t valueSize = value.size();
            outFile.write(reinterpret_cast<const char*>(&keySize), sizeof(keySize));
            outFile.write(key.data(), keySize);
            outFile.write(reinterpret_cast<const char*>(&valueSize), sizeof(valueSize));
            outFile.write(value.data(), valueSize);
        }
    }
    // внутренняя функция без блокировки
    void unsafeRestoreFromDisk() {
        memoryCache.clear();
        std::ifstream inFile(backupFilename, std::ios::binary);
        if (!inFile) return;
        while (inFile.peek() != EOF) {
            size_t keySize, valueSize;
            inFile.read(reinterpret_cast<char*>(&keySize), sizeof(keySize));
            std::string key(keySize, '\0');
            inFile.read(&key[0], keySize);
            inFile.read(reinterpret_cast<char*>(&valueSize), sizeof(valueSize));
            std::string value(valueSize, '\0');
            inFile.read(&value[0], valueSize);
            memoryCache[key] = value;
        }
    }
public:
    FaultTolerantStorage(const std::string& filename) : backupFilename(filename) {
        std::lock_guard<std::mutex> lock(mtx);
        unsafeRestoreFromDisk();
    }
    ~FaultTolerantStorage() {
        try {
            std::lock_guard<std::mutex> lock(mtx);
            unsafeSaveToDisk();
        }
        catch (...) {}
    }
    void set(const std::string& key, const std::string& value) {
        std::lock_guard<std::mutex> lock(mtx);
        memoryCache[key] = value;
        unsafeSaveToDisk();  // используем неблокирующую версию
    }
    std::string get(const std::string& key) {
        std::lock_guard<std::mutex> lock(mtx);
        return memoryCache.at(key);
```

```
    }
    void remove(const std::string& key) {
        std::lock_guard<std::mutex> lock(mtx);
        memoryCache.erase(key);
        unsafeSaveToDisk();  // используем неблокирующую версию
    }
    bool contains(const std::string& key) {
        std::lock_guard<std::mutex> lock(mtx);
        return memoryCache.find(key) != memoryCache.end();
    }
};
int main() {
    setlocale(LC_ALL, "rus");
    //FaultTolerantStorage storage("crash_data.bin");
    //std::cout << storage.get("test"); // должно вывести "123"
    FaultTolerantStorage storage("crash_data.bin");
    storage.set("test", "123");
    std::cout << "Данные записаны. Не закрывайте это окно!" << std::endl;
    std::cin.get();
    return 0;
}
```

Результат работы:



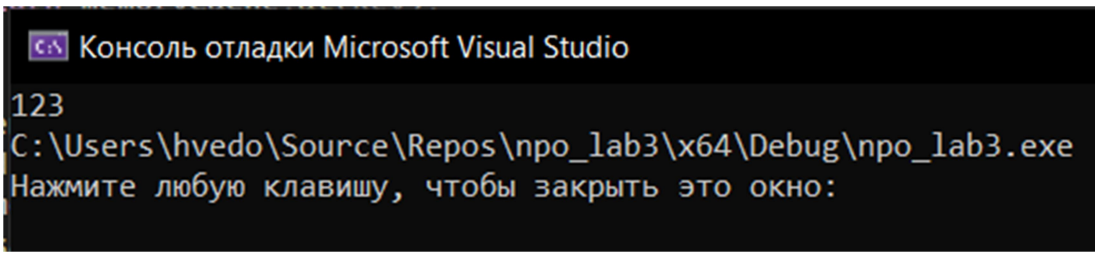
После чего мы через деспетчер зада закрываем нашу программу и комментируем данные строки

```
FaultTolerantStorage storage("crash_data.bin");
    storage.set("test", "123");
    std::cout << "Данные записаны. Не закрывайте это окно!" << std::endl;
    std::cin.get();
```

Вместо них мы добавляем

```
FaultTolerantStorage storage("crash_data.bin");
 std::cout << storage.get("test"); // должно вывести "123"
```

и нам восстановятся данные которые были в нашем файле до сбоя



Вывод: реализовал программу, которая при возникновения сбоя может восстановить данные, которые заранее записала.