МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения

вычислительной техники и автоматизированных

систем

Лабораторная работа №2

по дисциплине: ООП

тема: «Модульное программирование. Интерфейсы.»

Выполнил: студент группы ПВ-233
Мороз Роман Алексеевич

Проверили:

Белгород 2025

Лабораторная работа №2

Модульное программирование. Интерфейсы.

Цель работы: Получение навыков модульной декомпозиции предметной области, создания модулей. Разработка интерфейсов.

Задание: разработать программу, состоящую из трех модулей в соответствии с указанным вариантом задания. Первый модуль – основной код программы; второй содержит интерфейсы; третий модуль – реализацию этих интерфейсов. Количество структур данных ("объектов") не менее пяти.

Содержание отчета:

1. Тема, цель работы, вариант задания.
2. Реализация задачи на языке C++.

Контрольные вопросы:

1. Что такое модуль?
2. Какие бывают типы модулей в C++?
3. Структура типа «запись» (Struct).
4. Что такое интерфейс?
5. Принципы ООП.

Пример:

Пусть требуется разработать программу «Часы». Время хранится в структуре *Time* (часы, минуты, секунды), которая в свою очередь входит в состав структуры *Date*. Методы позволяют изменять текущее время и выводить время на экран.

```
struct Time {
	int hours; int minute; int sec;
};

struct Date {
	Time time;
	void settime(int, int, int);
	void printtime();
};
```

```
#include <stdio.h>
#include "date.h"

void Date::settime(int h, int m, int s)
{
```

```
	time.hours = h;
	time.minute = m;
	time.sec= s;
}

void Date::printtime()
{
	printf("%i:%i:%i", time.hours, time.minute, time.sec);
}
```

```
#include <stdio.h>
#include "date.h"

int main()
{
	Date a;
	a.settime(23,43,10);
	a.printtime();
	getchar();
	return 0;
}
```

1. Разработать программу согласно своему варианту.

№	Задание
1	Программа «Домашняя библиотека»
2	Программа «Домашняя аудио-коллекция»
3	Программа «Домашняя фильмотека»
4	Программа «Адресная книга»
5	Программа «Справочник нумизмата»
6	Программа «Игра в крестики нолики»
7	Программа «Автодиспетчер»
8	Программа «Файловый чат»
9	Программа «Будильник»
10	Программа «Органайзер» (учет и планирование личного времени)
11	Программа «Дневник погоды»
12	Программа «Домашний бюджет»
13	Программа «Электронный журнал»
14	Программа «Расстановка предметов мебели в помещении»
15	Программа «Реализовать стек на массиве»
16	Программа «Определения времени в различных часовых поясах»
17	Программа «Футбольный симулятор»
18	Программа «Турнир единоборств»
19	Программа «Турнирная таблица»
20	Программа «Планирование учебного расписания»

Вариант 8

Программа «Файловый чат»

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <chrono>
#include <iomanip>

#define CHAT_FILE "chat.txt"

#define RESET   "\033[0m"
#define RED     "\033[31m"
#define GREEN   "\033[32m"
#define YELLOW  "\033[33m"
#define BLUE    "\033[34m"
#define CYAN    "\033[36m"

struct Message {
private:
    std::string username;
    std::string message;
    long long timestamp;

public:
    Message(const std::string& user, const std::string& msg)
    : username(user), message(msg), timestamp(generateTimestamp()) {}

    Message(const std::string& user, const std::string& msg, long long ts)
    : username(user), message(msg), timestamp(ts) {}

    static long long generateTimestamp() {
              return  std::chrono::system_clock::now().time_since_epoch()  /
std::chrono::seconds(1);
    }

    std::string formatTimestamp() const {
        std::time_t time = timestamp;
        std::tm tm = *std::localtime(&time);
        std::ostringstream oss;
        oss << std::put_time(&tm, "%d-%m-%Y %H:%M");
        return oss.str();
    }
```

```
    std::string serialize() const {
        return username + "|" + message + "|" + std::to_string(timestamp) +
"\n";
    }

    static Message deserialize(const std::string& line) {
        std::istringstream ss(line);
        std::string user, msg, timeStr;
        getline(ss, user, '|');
        getline(ss, msg, '|');
        getline(ss, timeStr, '|');
        return Message(user, msg, std::stoll(timeStr));
    }

    std::string getUsername() const { return username; }
    std::string getMessage() const { return message; }
    long long getTimestamp() const { return timestamp; }

    void print() const {
         std::cout << CYAN << "[" << formatTimestamp() << "]" << RESET << "
"
                   << GREEN << username << RESET << ": " << BLUE << message
<< RESET << std::endl;
    }
};

void sendMessage(const std::string& username, const std::string& message)
{
    std::ofstream outFile(CHAT_FILE, std::ios::app);

    if (!outFile) {
           std::cerr << RED << "Ошибка открытия файла чата!" << RESET <<
std::endl;
        return;
    }

    Message msg(username, message);
    outFile << msg.serialize();
    outFile.close();
}

std::vector<Message> readChat() {
    std::vector<Message> messages;
    std::ifstream inFile(CHAT_FILE);
```

```
    if (!inFile) {
            std::cerr << RED << "Ошибка открытия файла чата!" << RESET <<
std::endl;
        return messages;
    }

    std::string line;
    while (getline(inFile, line))
        messages.push_back(Message::deserialize(line));

    inFile.close();

    return messages;
}

void printMessages(const std::vector<Message>& messages) {
    if (messages.empty()) {
        std::cout << YELLOW << "Чат пуст!" << RESET << std::endl;
        return;
    }

    for (const auto& msg : messages)
        msg.print();
}

void clearChat() {
    std::ofstream outFile(CHAT_FILE, std::ios::trunc);

    if (!outFile) {
        std::cerr << RED << "Ошибка очистки чата!" << RESET << std::endl;
        return;
    }

    outFile.close();
    std::cout << GREEN << "Чат очищен!" << RESET << std::endl;
}

void menu() {
    int choice;

    while (true) {
         std::cout << "\n" << YELLOW << "=== Файловый чат ===" << RESET <<
"\n";
        std::cout << GREEN << "1. Отправить сообщение" << RESET << "\n";
        std::cout << GREEN << "2. Прочитать чат" << RESET << "\n";
```

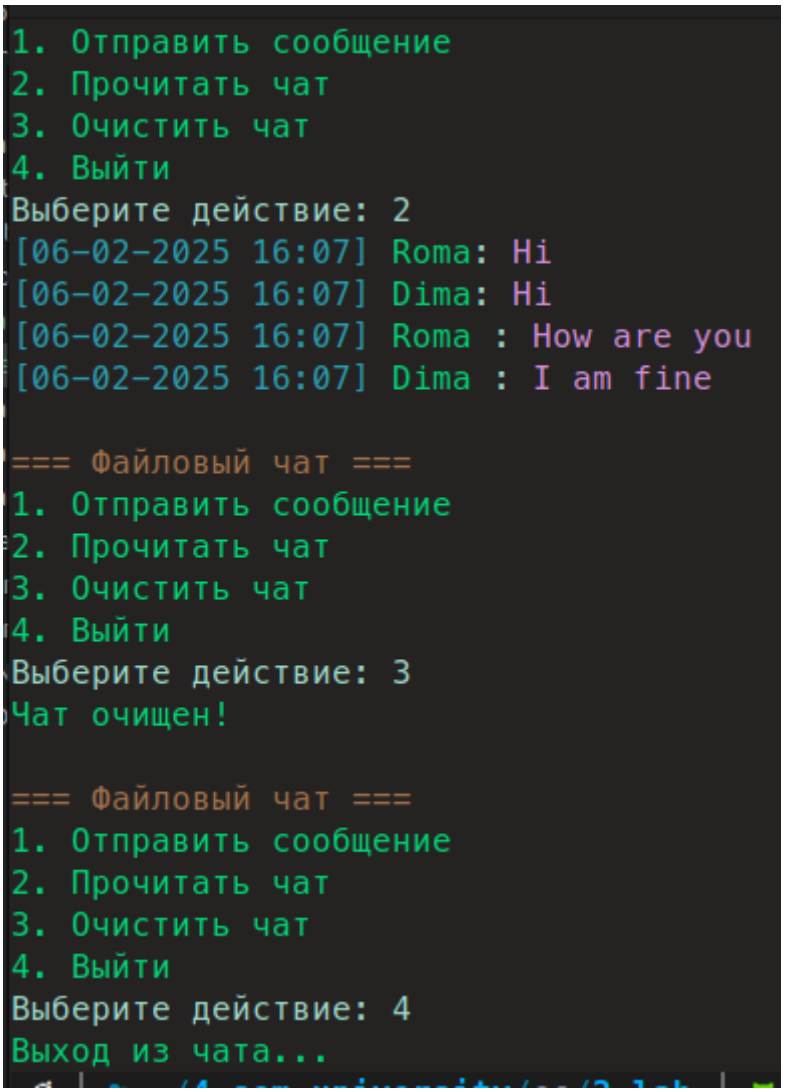
```
        std::cout << GREEN << "3. Очистить чат" << RESET << "\n";
        std::cout << GREEN << "4. Выйти" << RESET << "\n";
        std::cout << "Выберите действие: ";

        std::cin >> choice;
        std::cin.ignore();

        switch (choice) {
            case 1: {
                std::string username, message;
                std::cout << "Введите ваше имя: ";
                getline(std::cin, username);
                std::cout << "Введите сообщение: ";
                getline(std::cin, message);
                sendMessage(username, message);
                break;
            }
            case 2: {
                std::vector<Message> messages = readChat();
                printMessages(messages);
                break;
            }
            case 3: {
                clearChat();
                break;
            }
            case 4: {
                std::cout << GREEN << "Выход из чата...\n" << RESET;
                return;
            }
            default: {
                        std::cout << RED << "Некорректный выбор. Попробуйте
снова.\n" << RESET;
                break;
            }
        }
    }
}

int main() {
    menu();

    return 0;
}
```



Вывод: Получили навыки модульной декомпозиции предметной области, создания модулей. Разработка интерфейсов.