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»

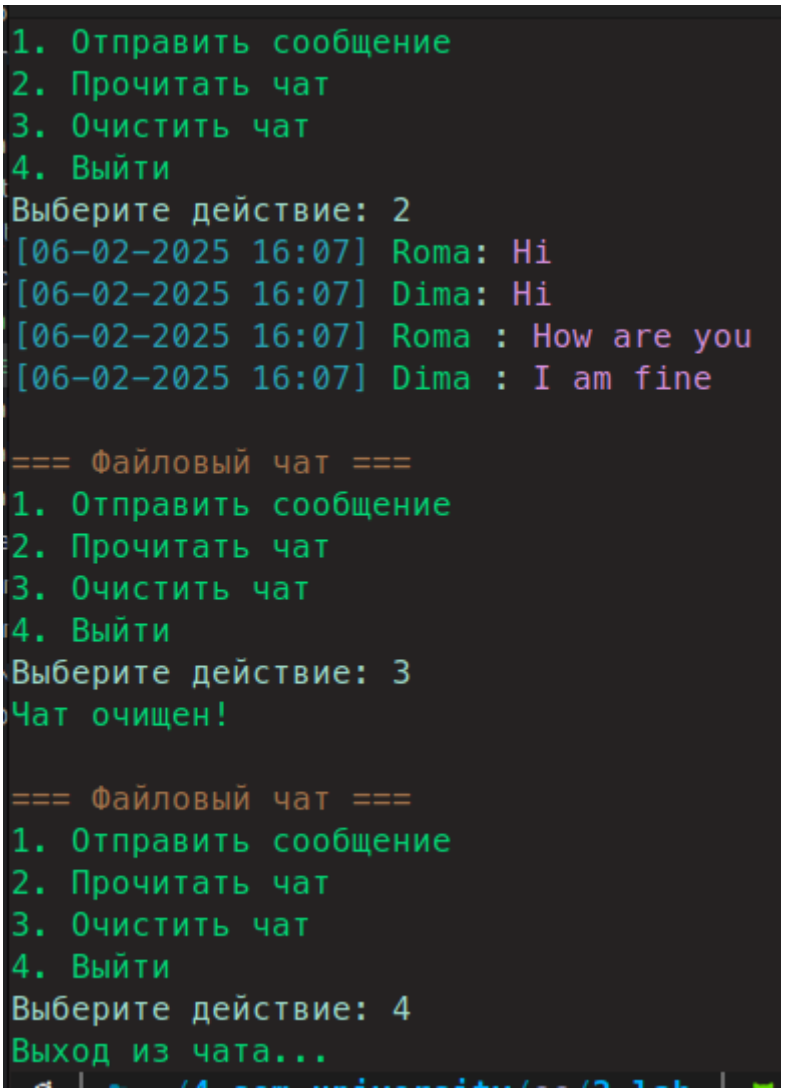
```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <chrono>
#include <iomanip>

#define RESET   "\033[0m"
#define RED     "\033[31m"
#define GREEN   "\033[32m"
#define YELLOW  "\033[33m"
#define BLUE    "\033[34m"
#define CYAN    "\033[36m"
#define MAGENTA "\033[35m"

class Message {
private:
   std::string username;
   std::string content;
   long long timestamp;

public:
   Message(const std::string& user, const std::string& msg, long long ts =
0): username(user), content(msg),
                                        timestamp(ts      ?      ts      :
std::chrono::system_clock::now().time_since_epoch()                       /
std::chrono::seconds(1)) {}

   const std::string getUsername() const { return username; }
   const std::string getContent() const { return content; }
   long long getTimestamp() const { return timestamp; }
   void setTimestamp(long long ts) { timestamp = ts; }
       void  updateContent(const  std::string&  newContent)  {  content  =
newContent; }


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

   static Message deserialize(const std::string& line) {
```

```
        std::istringstream ss(line);
        std::string parts[3];

        for(int i = 0; i < 3 && getline(ss, parts[i], '|'); i++);

        return Message(parts[0], parts[1], std::stoll(parts[2]));
    }

    inline void display() const {
        std::time_t time = timestamp;
        std::tm tm = *std::localtime(&time);
        std::cout << CYAN << "[" << std::put_time(&tm, "%d-%m-%Y %H:%M") <<
"] "
                  << GREEN << username << RESET << ": " << BLUE << content
<< RESET << std::endl;
    }
};

class Chat {
public:
    static const std::string CHAT_FILE_PATH;

    static void save(const Message& message) {
        std::ofstream file(CHAT_FILE_PATH, std::ios::app);
        if(file) file << message.serialize();
    }

    static std::vector<Message> load() {
        std::vector<Message> messages;
        std::ifstream file(CHAT_FILE_PATH);

        if(file) {
            std::string line;
            while(getline(file, line)) {
                messages.push_back(Message::deserialize(line));
            }
        }

        return messages;
    }

     inline static void clear() { std::ofstream file(CHAT_FILE_PATH,
std::ios::trunc); }
};
```

```
class ChatPrinter {
public:
    inline static void printMessages(const std::vector<Message>& messages)
{
        if (messages.empty()) {
            printWarning("Чат пуст");
            return;
        }

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

    inline static void printWarning(const std::string& message) {
        std::cout << YELLOW << message << RESET << std::endl;
    }

    inline static void printError(const std::string& message) {
        std::cerr << RED << "Ошибка: " << message << RESET << std::endl;
    }

    inline static void printSuccess(const std::string& message) {
        std::cout << GREEN << message << RESET << std::endl;
    }
};

class ChatManager {
public:
    static void sendMessage(const std::string& username, const std::string&
content) {
        if(username.empty() || content.empty()) {
            ChatPrinter::printError("Нельзя отправить пустое сообщение!");
            return;
        }

        Chat::save(Message(username, content));
        ChatPrinter::printSuccess("Сообщение отправлено!");
    }

                inline      static      void      showChatHistory()      {
ChatPrinter::printMessages(Chat::load()); }

    inline static void clearChatHistory() {
        Chat::clear();
        ChatPrinter::printSuccess("Чат успешно очищен!");
    }
```

```
};

class ChatInterface {
public:
    void run() {
        while(true) {
            displayMenu();

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

            handleChoice(choice);
        }
    }

private:
    ChatManager manager;

    std::string getInput(const std::string& prompt) {
        std::cout << prompt;
        std::string input;

        getline(std::cin, input);

        return input;
    }

    void displayMenu() {
         std::cout << "\n" << MAGENTA << "=== Файловый Чат ===" << RESET <<
"\n"
                   << "1. Отправить сообщение\n"
                   << "2. Просмотреть историю\n"
                   << "3. Очистить чат\n"
                   << "4. Выход\n"
                   << CYAN << "Выберите действие: " << RESET;
    }

    void handleChoice(int choice) {
        switch(choice) {
            case 1: {
                std::string username = getInput("Введите ваше имя: ");
                std::string content = getInput("Введите сообщение: ");
                manager.sendMessage(username, content);
                break;
```

```
            }

            case 2:
                manager.showChatHistory();
                break;

            case 3:
                manager.clearChatHistory();
                break;

            case 4:
                ChatPrinter::printSuccess("Выход из программы...");
                exit(0);

            default:
                ChatPrinter::printError("Некорректный выбор!");
        }
    }
};

int main() {

    ChatInterface chat;
    chat.run();

    return 0;
}
```



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