

Министерство цифрового развития, связи
и массовых коммуникаций Российской Федерации

Сибирский государственный университет
телекоммуникаций и информатики

Кафедра вычислительных систем

КУРСОВАЯ РАБОТА

По дисциплине: «Технологии разработки программного обеспечения»

По теме: «Разработка классической игры «100 спичек»»

Выполнил:

Студент 2 курса группы ИП-111

Попов М.И.

Проверил:

Старший преподаватель кафедры ВС

Токмашева Е.И.

Новосибирск, 2023

Оглавление

1. Введение и постановки задачи.....	3
2. Техническое задание	4
3. Описание выполненного проекта	5
4. Личный вклад в проект	10
5. Приложение. Листинги кода	11
5.1. Библиотеки.....	11
5.2. Функции	13
5.3. Файл исполнения.....	19

Введение и постановки задачи

Главная задача курсового проекта: написание кода классической игры «100 спичек», с использованием знаний, полученных при обучении дисциплине «Технологии разработки программного обеспечения».

Цель курсового проекта: предоставить работающую игру, в соответствии с принятыми соглашениями по поводу правил и технической части приложения.

«100 спичек» - это игра, предназначенная для одного игрока (в режиме игры с компьютером) или двух игроков (в режиме совместной игры). Суть игры заключается в том, что игроки поочередно выбирают и вводят с клавиатуры N-ое количество спичек ($1 \leq N \leq 10$), с изначальным количеством в 100 спичек, как это понятно из названия. Проиграет тот, кто возьмет последнюю спичку, то есть игрокам нужно придумать собственную стратегию победы, а также предугадывать ходы соперника.

Техническое задание

1. Описание задачи:

"Сто спичек" - это игра, где изначально имеется куча, состоящая из 100 спичек. Игроки берут поочередно из этой кучи от 1 до 10 спичек в один ход. Игра заканчивается, если один из игроков возьмет последнюю оставшуюся спичку.

2. Функциональность программы:

Реализация приложения для этой игры в терминале. Простое меню для навигации, чтобы можно было начинать игру, посмотреть правила и выйти из программы. Игра осуществляется либо с оппонентом, либо с компьютером. После выбора режима игры: ввод имени (при игре с оппонентом), затем отображение количества спичек, чей ход, а также запрос на ввод значения от 1 до 10. При окончании игры выводить проигравшего и дать возможность начать игру заново.

3. Формат входных данных:

- Выбор через ввод с клавиатуры пунктов в меню;
- Ввод с клавиатуры имени игрока или же имен игроков в зависимости от режима игры;
- Ввод с клавиатуры количества спичек от 1 до 10.

4. Интерфейс приложения:

Интерактивное консольное приложение с несколькими пунктами меню для навигации, состоящие из начала игры, правил и выхода. При начале игры пункты выбора режима игры, далее ввод имени и уже начало самой игры с выводом информации по игре и ввод определенного количества спичек. При выборе правил, выводится сам список правил.

Описание выполненного проекта

Этап I. Начало работы:

Перед планированием курсового проекта команда выбрала тему разработки игры «100 спичек». После недолгих дискуссий по поводу реализации данного приложения было решено следующее:

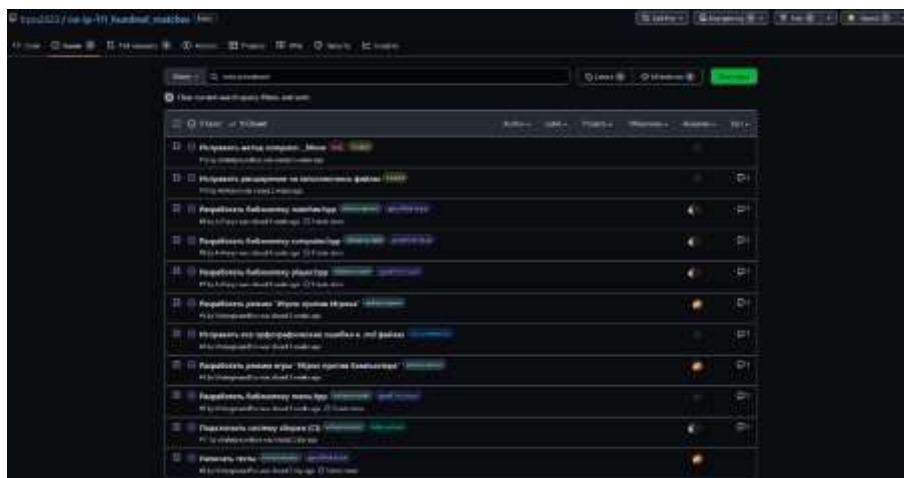
- В качестве языка программирования был выбран C++, поскольку с ним был знаком каждый член команды.
- В качестве операционной системы было решено выбрать ОС Linux.

Затем была составлена единая концепция проекта, написано представленное выше техническое задание.

Опираясь на ТЗ, удалось оптимизировать процесс разработки программы, поэтому завершить работу над проектом получилось в краткие сроки, при этом процесс разработки был весьма удобен и не вызывал много вопросов.

Этап II. Организация работы и распределение задач:

Следующим немаловажным шагом в разработке игры «100 спичек» стала декомпозиция Технического Задания. Задачи были распределены между членами команды разработки. Список задач имел следующий вид:



Скриншот 1 - Список выполненных задач после разработки

Задачи были распределены в соответствии с желаниями и знаниями каждого из участников команды.

Этап III. Подготовка рабочего каталога и начало разработки:

После распределения задач и выяснения всех необходимых вопросов, связанных с разработкой, было решено подготовить репозиторий.

В первую очередь было решено создать структуру проекта. Это и было самым первым изменением в репозитории.

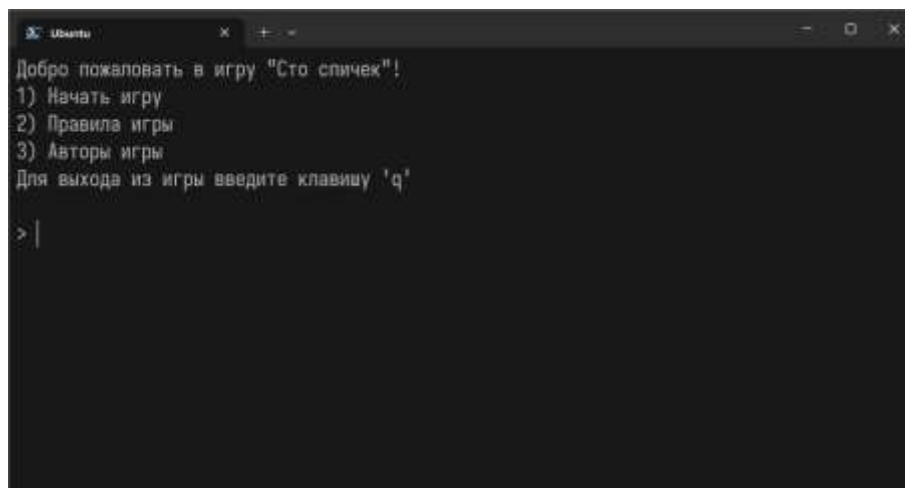
На ранних этапах были добавлены файлы формата *.md* (Файлы описания проекта и само ТЗ) и *.gitignore*, что необходимы для игнорирования файлов, полученных во время сборки приложения соответственно, чтобы предотвратить артефакты сборки.

Для сборки программы использовалась утилита *make*. На раннем этапе разработки приложения был написан *Makefile*. Он имел нужный и полезный на тот момент функционал: сборка приложения и очистка файлов, полученных в ходе компиляции. После того как проект представлял из себя рабочую версию, в *Makefile* были добавлены команды для сборки тестов и их ручного запуска сразу же после сборки.

Этап IV. Написание кода приложения:

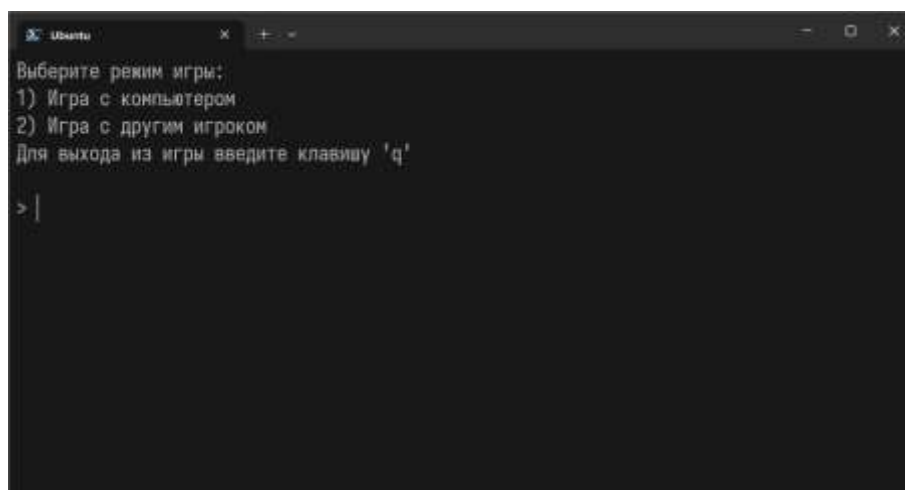
Перед стартом разработки приложения было понятно, как будут реализованы функции. Благодаря заранее спланированному названию параметров функций удалось избежать конфликтов в коде. Из-за этого организация процесса параллельной разработки была успешной.

Работа приложения:

A terminal window titled 'Ubuntu' showing the main menu of a game. The text displayed is: 'Добро пожаловать в игру "Сто спичек"!'. Below this is a numbered list: '1) Начать игру', '2) Правила игры', and '3) Авторы игры'. A line of text says 'Для выхода из игры введите клавишу 'q''. At the bottom, there is a prompt '> |' with a cursor.

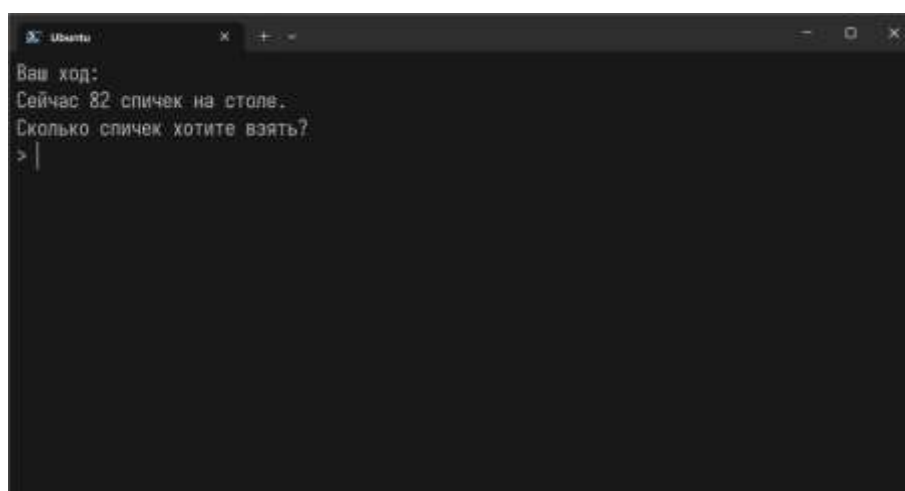
```
Ubuntu
Добро пожаловать в игру "Сто спичек"!
1) Начать игру
2) Правила игры
3) Авторы игры
Для выхода из игры введите клавишу 'q'
> |
```

Скриншот 2 - Главное меню

A terminal window titled 'Ubuntu' showing the game mode selection menu. The text displayed is: 'Выберите режим игры:'. Below this is a numbered list: '1) Игра с компьютером' and '2) Игра с другим игроком'. A line of text says 'Для выхода из игры введите клавишу 'q''. At the bottom, there is a prompt '> |' with a cursor.

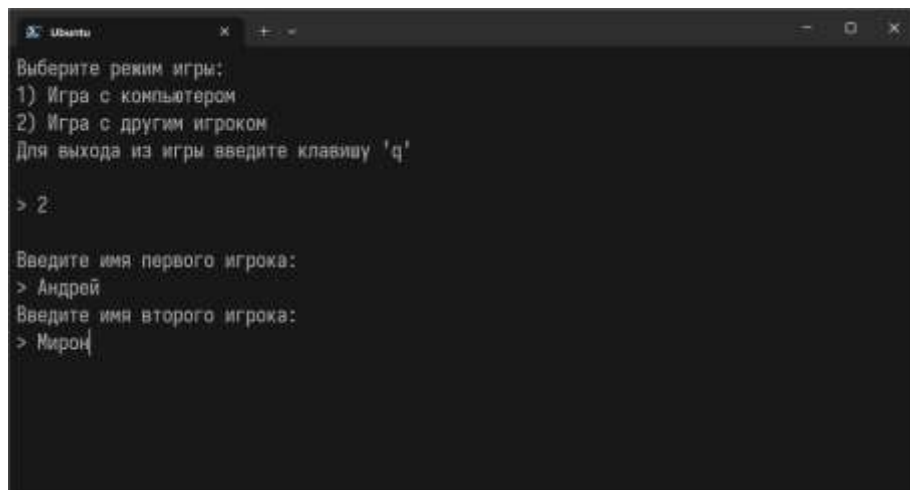
```
Ubuntu
Выберите режим игры:
1) Игра с компьютером
2) Игра с другим игроком
Для выхода из игры введите клавишу 'q'
> |
```

Скриншот 3 - Меню начала игры

A terminal window titled 'Ubuntu' showing the game interface during a match with a computer. The text displayed is: 'Ваш ход:', 'Сейчас 82 спичек на столе.', and 'Сколько спичек хотите взять?'. At the bottom, there is a prompt '> |' with a cursor.

```
Ubuntu
Ваш ход:
Сейчас 82 спичек на столе.
Сколько спичек хотите взять?
> |
```

Скриншот 4 – Игра с компьютером



```
Выберите режим игры:
1) Игра с компьютером
2) Игра с другим игроком
Для выхода из игры введите клавишу 'q'

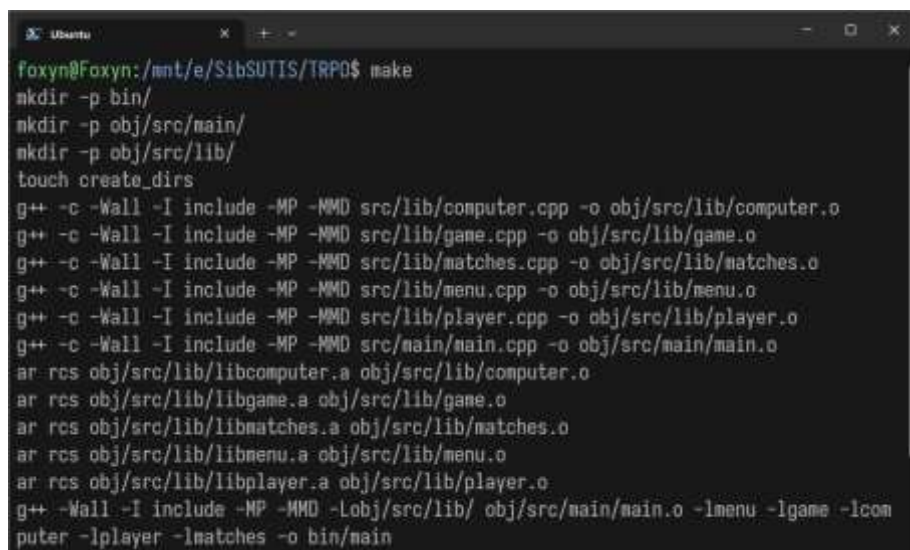
> 2:

Введите имя первого игрока:
> Андрей
Введите имя второго игрока:
> Мирон
```

Скриншот 5 – Игра с другим игроком

Этап V. Написание Make файла:

Правила преобразования задаются в скрипте с именем *Makefile*, который должен находиться в корне рабочей директории проекта. После того как разработка всех основных функций, необходимых для правильного функционирования приложения, была закончена, оставалось только скомпилировать файл.

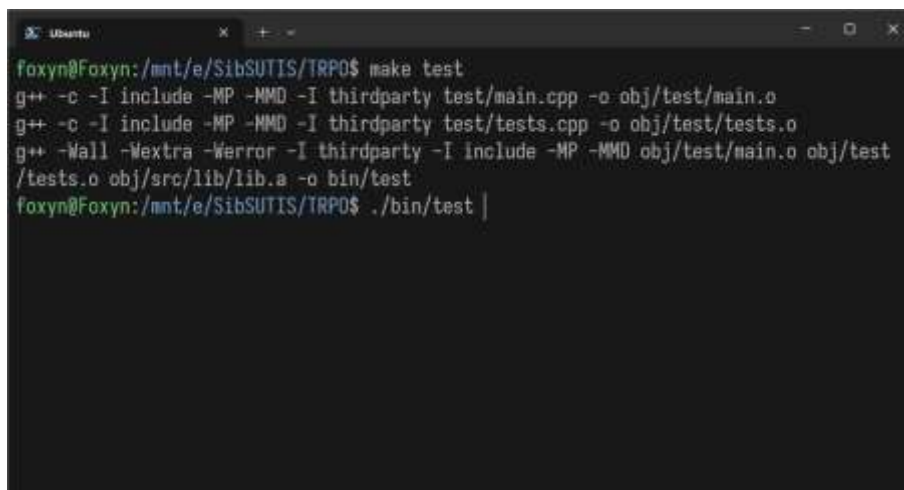


```
foxynt@foxynt:/mnt/e/SibSUTIS/TRPO$ make
mkdir -p bin/
mkdir -p obj/src/main/
mkdir -p obj/src/lib/
touch create_dirs
g++ -c -Wall -I include -MP -MMD src/lib/computer.cpp -o obj/src/lib/computer.o
g++ -c -Wall -I include -MP -MMD src/lib/game.cpp -o obj/src/lib/game.o
g++ -c -Wall -I include -MP -MMD src/lib/matches.cpp -o obj/src/lib/matches.o
g++ -c -Wall -I include -MP -MMD src/lib/menu.cpp -o obj/src/lib/menu.o
g++ -c -Wall -I include -MP -MMD src/lib/player.cpp -o obj/src/lib/player.o
g++ -c -Wall -I include -MP -MMD src/main/main.cpp -o obj/src/main/main.o
ar rcs obj/src/lib/libcomputer.a obj/src/lib/computer.o
ar rcs obj/src/lib/libgame.a obj/src/lib/game.o
ar rcs obj/src/lib/libmatches.a obj/src/lib/matches.o
ar rcs obj/src/lib/libmenu.a obj/src/lib/menu.o
ar rcs obj/src/lib/libplayer.a obj/src/lib/player.o
g++ -Wall -I include -MP -MMD -Lobj/src/lib/ obj/src/main/main.o -lmenu -lgame -lcomputer -lplayer -lmatches -o bin/main
```

Скриншот 6 – Сборка через *Makefile* по команде *make*

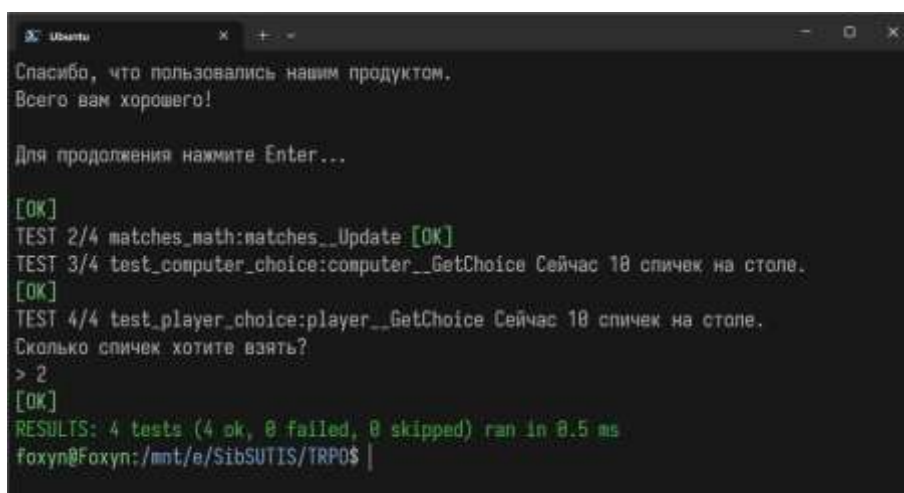
Этап VI. Покрытие тестами:

Нужно было убедиться, что функции работают правильно, поэтому были написаны тесты на важные функции. В ходе реализации тестов нашлось пару ошибок и неполадок функций, а затем, исправив их, программа теперь работает успешно. Следовательно, из этого можно сделать вывод, что если составляющие успешно функционируют, то правильно работает и сама функция.



```
foxy@foxy:/mnt/e/SibSUTIS/TRPO$ make test
g++ -c -I include -MP -MMD -I thirdparty test/main.cpp -o obj/test/main.o
g++ -c -I include -MP -MMD -I thirdparty test/tests.cpp -o obj/test/tests.o
g++ -Wall -Wextra -Werror -I thirdparty -I include -MP -MMD obj/test/main.o obj/test
/tests.o obj/src/lib/lib.a -o bin/test
foxy@foxy:/mnt/e/SibSUTIS/TRPO$ ./bin/test |
```

Скриншот 7 – Сборка тестов через make файл



```
Спасибо, что пользовались нашим продуктом.
Всего вам хорошего!

Для продолжения нажмите Enter...

[OK]
TEST 2/4 matches_math:matches__Update [OK]
TEST 3/4 test_computer_choice:computer__GetChoice Сейчас 18 спичек на столе.
[OK]
TEST 4/4 test_player_choice:player__GetChoice Сейчас 18 спичек на столе.
Сколько спичек хотите взять?
> 2
[OK]
RESULTS: 4 tests (4 ok, 0 failed, 0 skipped) ran in 0.5 ms
foxy@foxy:/mnt/e/SibSUTIS/TRPO$ |
```

Скриншот 8 – Выполнение Unit и UI тестов

Личный вклад в проект

Мой личный вклад начался с того что я выслушал то как представляет наш проект Андрей, проговорив некоторые моменты мы начали разделять задачи проекта.

Моя задача предстояла в разработке библиотеки `player` и `game`, а также организации всей системы сборки и конвейера с тестирование проекта.

Я реализовал функции в следующих файлах для игры:

- «`player`» - многопользовательский режим игры на двух игроков;
- «`game`» - главная библиотека процесса игры;

`Makefile` был подготовлен максимально универсальный для возможности дальней правки кода или подключение новых библиотек и минимальный изменения в `make`. Тесты были разработаны для тестирования приложения на наличии внезапно появившихся ошибок и дальнейших их отладки исключая дальнейшее попадание бага в репозиторий.

Приложение. Листинги кода

Библиотеки

computer.hpp

```
#include "include.hpp"
#ifndef _COMPUTER_HPP_
#define _COMPUTER_HPP_

int computer__GetChoice(int matches);

int computer__Move(int matches);
void computer__Check(int matches);

int computer__Move(int matches, int computerNumber);
void computer__Check(int matches, int computerNumber);

#endif
```

game.hpp

```
#include "include.hpp"
#ifndef _GAME_HPP_
#define _GAME_HPP_

void game__Computer();
void game__Players();
void game__Secret();
void game__End();

#endif
```

include.hpp

```
#ifndef _INCLUDE_HPP_
#define _INCLUDE_HPP_

// Стандартные библиотеки
#include <iostream>
#include <string>
#include <ctime>

// Кастомные библиотеки
#include "computer.hpp"
#include "matches.hpp"
#include "player.hpp"
#include "game.hpp"
#include "menu.hpp"

#endif
```

matches.hpp

```
#include "include.hpp"
#ifndef _MATCHES_HPP_
#define _MATCHES_HPP_

const int MATCHES = 100;
const int MIN_MATCHES = 1;
const int MAX_MATCHES = 10;

void matches__Get(int matches);
int matches__Update(int matches, int choice);

#endif
```

menu.hpp

```
#include "include.hpp"
#ifndef _MENU_HPP_
#define _MENU_HPP_

std::string input(const std::string &str);

void menu();
void menu__Game();
void menu__Rules();
void menu__Authors();
void menu__Quit();

#endif
```

player.hpp

```
#include "include.hpp"
#ifndef _PLAYER_HPP_
#define _PLAYER_HPP_

int player__GetChoice(int matches);

int player__Move(int matches);
void player__Check(int matches);

int player__Move(int matches, std::string playerName);
void player__Check(int matches, std::string playerName);
void player__Change();

#endif
```

Функции

computer.cpp

```
#include "computer.hpp"
using namespace std;

int computer__GetChoice(int matches)
{
    matches__Get(matches);
    return rand() % min(MAX_MATCHES, matches) + 1;
}

int computer__Move(int matches)
{
    system("clear");
    std::cout << "Ход компьютера:\n";
    int computerChoice = computer__GetChoice(matches);
    std::cout << "Компьютер взял "
        << computerChoice << " спичек\n";
    return matches__Update(matches, computerChoice);
}

void computer__Check(int matches)
{
    if (matches == 0)
    {
        std::cout << "Победа! Компьютер взял последнюю спичку на столе!\n"
            << "\nДля продолжения нажмите Enter...\n";
        std::cin.ignore().get();
        game__End();
    }
}

int computer__Move(int matches, int computerNumber)
{
    system("clear");
    std::cout << "Ход компьютера №" << computerNumber << ":\n";
    int computerChoice = computer__GetChoice(matches);
    std::cout << "Компьютер №" << computerNumber << " взял "
        << computerChoice << " спичек\n";
    return matches__Update(matches, computerChoice);
}

void computer__Check(int matches, int computerNumber)
{
    if (matches == 0)
    {
        std::cout << "Поражение! Компьютер №" << computerNumber
            << " взял последнюю спичку на столе!\n"
            << "\nДля продолжения нажмите Enter...\n";
        std::cin.ignore().get();
        game__End();
    }
}
```

game.cpp

```
#include "game.hpp"
void game__Computer()
{
    int matches = MATCHES;
    while (matches > 0)
    {
        matches = player__Move(matches);
        player__Check(matches);
        matches = computer__Move(matches);
        computer__Check(matches);
    }
}

void game__Players()
{
    std::string firstPlayerName = input(
        "\nВведите имя первого игрока:\t");
    std::string secondPlayerName = input(
        "Введите имя второго игрока:\t");
    system("clear");

    int matches = MATCHES;
    while (matches > 0)
    {
        matches = player__Move(matches, firstPlayerName);
        player__Check(matches, firstPlayerName);
        player__Change();
        matches = player__Move(matches, secondPlayerName);
        player__Check(matches, secondPlayerName);
        player__Change();
    }
}

void game__Secret()
{
    int matches = MATCHES;
    while (matches > 0)
    {
        matches = computer__Move(matches, 1);
        computer__Check(matches);
        std::cin.ignore().get();
        matches = computer__Move(matches, 2);
        computer__Check(matches);
        std::cin.ignore().get();
    }
}

void game__End()
{
    system("clear");
    while (true)
    {
        std::string choose = input("\nЖелаете начать новую игру? (y|n)");
        switch (choose[0])
        {
            case 'y':
                menu__Game();
                break;
            case 'n':
                return;
            default:
                continue;
        }
    }
}
```

```

        break;
    case 'n':
        menu();
        break;
    default:
        std::cout << "\nОшибка! Такого выбора нет в списке.\n"
            << "Для продолжения нажмите Enter...\n";
        std::cin.ignore().get();
    }
}
}

```

matches.cpp

```

#include "matches.hpp"

void matches__Get(int matches)
{
    std::cout << "Сейчас " << matches << " спичек на столе.\n";
}

int matches__Update(int matches, int choice)
{
    return matches - choice;
}

```

menu.cpp

```

#include "menu.hpp"
std::string input(const std::string &str)
{
    std::cout << str
        << "\n> ";
    std::string Answer;
    std::cin >> Answer;
    return Answer;
}

void menu()
{
    while (true)
    {
        system("clear");
        std::string choose = input(
            "Добро пожаловать в игру \n\"Сто спичек\n\"!\n"
            "1) Начать игру\n"
            "2) Правила игры\n"
            "3) Авторы игры\n"
            "Для выхода из игры введите клавишу \n'q'\n");
        switch (choose[0])
        {
            case '1':
                menu__Game();
                break;
            case '2':
                menu__Rules();
                break;
            case '3':

```

```

    menu__Authors();
    break;
case 'q':
case 'Q':
    menu__Quit();
    return;
default:
    std::cout << "\nОшибка! Такого выбора нет в списке.\n"
        << "Для продолжения нажмите Enter...\n";
    std::cin.ignore().get();
}
}
}
void menu__Game()
{
    while (true)
    {
        system("clear");
        std::string choose = input(
            "Выберите режим игры:\n"
            "1) Игра с компьютером\n"
            "2) Игра с другим игроком\n"
            "Для выхода из игры введите клавишу 'q'\n");
        switch (choose[0])
        {
            case '1':
                game__Computer();
                break;
            case '2':
                game__Players();
                break;
            case '3':
                game__Secret();
                break;
            case 'q':
            case 'Q':
                return;
            default:
                std::cout << "\nОшибка! Такого выбора нет в списке.\n"
                    << "Для продолжения нажмите Enter...\n";
                std::cin.ignore().get();
        }
    }
}
void menu__Rules()
{
    system("clear");
    std::cout << "\nСто спичек\n" - это игра, где изначально имеется куча, состоящая из 100 спичек.\n"
        << "Игроки берут поочередно из этой кучи от 1 до 10 спичек в один ход.\n"
        << "Игра заканчивается, если один из игроков возьмет последнюю оставшуюся спичку.\n"
        << "\nДля продолжения нажмите Enter...\n";
    std::cin.ignore().get();
}
void menu__Authors()
{
    system("clear");
    std::cout
        << "\bАвторы игры:\n"
        << "\tИП-111 Корнилов А.А. (-=Foxyn=-);\n"

```



```

        << "\tИП-111 Попов М.И (Underground_Evo)\n"
        << "\nДля продолжения нажмите Enter...\n";
    std::cin.ignore().get();
}
void menu__Quit()
{
    system("clear");
    std::cout << "Спасибо, что пользовались нашим продуктом.\n"
        << "Всего вам хорошего!\n"
        << "\nДля продолжения нажмите Enter...\n";
    std::cin.ignore().get();
}

```

player.cpp

```

#include "player.hpp"

int player__GetChoice(int matches)
{
    matches__Get(matches);

    int choice;
    std::cout << "Сколько спичек хотите взять?\n"
        << "> ";
    std::cin >> choice;

    while (choice < MIN_MATCHES ||
        choice > MAX_MATCHES ||
        choice > matches)
    {
        std::cout << "Ошибка! Пожалуйста возьмите спички от 1 до " << std::min(10, matches) << ":\n";
        std::cin >> choice;
    }

    return choice;
}

int player__Move(int matches)
{
    system("clear");
    std::cout << "Ваш ход:\n";
    int playerChoice = player__GetChoice(matches);
    return matches__Update(matches, playerChoice);
}

void player__Check(int matches)
{
    if (matches == 0)
    {
        std::cout << "Поражение! Вы взяли последнюю спичку на столе!\n"
            << "\nДля продолжения нажмите Enter...\n";
        std::cin.ignore().get();
        game__End();
    }
}

int player__Move(int matches, std::string playerName)
{

```

```

system("clear");
std::cout << "Ходит " << playerName << ":\n";
int playerChoice = player__GetChoice(matches);
return matches__Update(matches, playerChoice);
}

void player__Check(int matches, std::string playerName)
{
    if(matches == 0)
    {
        std::cout << "Поражение! Игрок " << playerName
            << " взял последнюю спичку на столе!\n"
            << "\nДля продолжения нажмите Enter...\n";
        std::cin.ignore().get();
        game__End();
    }
}

void player__Change()
{
    std::cout << "\nВаш ход закончен!\n"
        << "Передайте управление другому игроку.\n"
        << "\nДля продолжения нажмите Enter...\n";
    std::cin.ignore().get();
}

```

Файл исполнения

main.cpp

```
#include "include.hpp"

int main()
{
    srand(time(0));
    menu();
    return 0;
}
```