

Министерство цифрового развития, связи и массовых коммуникаций РФ
Федеральное государственное бюджетное образовательное учреждение
Высшего профессионального образования «Сибирский государственный
университет телекоммуникаций и информатики»

Кафедра вычислительных систем

КУРСОВАЯ РАБОТА

по дисциплине «Технологии разработки программного обеспечения»
на тему: «100 спичек»

Выполнил:

ст. гр. ИП-217

Павлова В. А.

Проверил:

ст. преподаватель Токмашева Е. И.

Новосибирск

2023

Содержание

Введение и постановка задачи	3
Техническое задание	4
Описание выполненного проекта	5
Меню	5
Ход пользователя	5
Ввод буквы.....	5
Ввод числа, не входящего в диапазон	6
Ввод числа, превышающего размер кучи.....	6
Ввод корректных данных	6
Ход бота	6
Конец игры.....	7
Победил игрок	7
Победил бот	7
Тестирование приложения	7
Личный вклад в проект	8
Приложение. Текст программы	9

Введение и постановка задачи

Целью курсового проекта является проектирование и реализация игры «100 спичек». Из кучки, первоначально содержащей 100 спичек, двое играющих поочередно берут по несколько спичек: не менее одной и не более десяти. Выигрывает взявший последнюю спичку.

Техническое задание

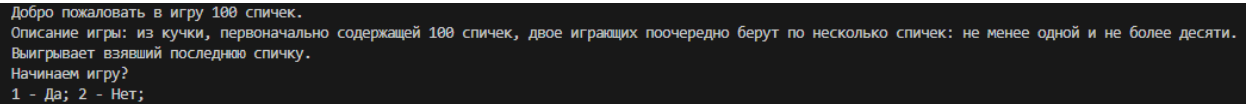
Разработать игру 100 спичек на языке С. Для выполнения поставленной цели были выделены следующие задачи:

1. разработка структуры проекта;
2. распределение обязанностей;
3. реализация функционала;
4. проверка работоспособности;

Описание выполненного проекта

Меню

При запуске собранного приложения, пользователя встречает небольшое меню с кратким пояснением цели игры:



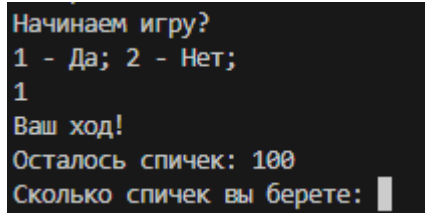
```
Добро пожаловать в игру 100 спичек.  
Описание игры: из кучки, первоначально содержащей 100 спичек, двое играющих поочередно берут по несколько спичек: не менее одной и не более десяти.  
Выигрывает взявший последнюю спичку.  
Начинаем игру?  
1 - Да; 2 - Нет;
```

Рисунок 1 – Основное меню ([main](#) и [start](#))

По умолчанию пользователь начинает первый. Затем идет проверка на количество оставшихся спичек в куче и, если их больше нуля, то наступает очередь Бота. Следом идёт такая же проверка на количество спичек и ход переходит к игроку. Очередность прописана в функции [queue](#).

Ход пользователя

Если пользователь вводит 2, то программа закрывается, а если вводят 1, то наступает ход игрока:



```
Начинаем игру?  
1 - Да; 2 - Нет;  
1  
Ваш ход!  
Осталось спичек: 100  
Сколько спичек вы берете: █
```

Рисунок 2 – Ход игрока ([xod_playera](#))

Далее есть несколько возможных вариаций вводимых данных, которые проверяются в отдельной функции [checks](#) и, если ввод некорректный, то выводится соответствующее предупреждение, каждый случай которого описан в функции [info](#) и рассмотрен далее:

Ввод буквы

Вполне может случиться такое, что пользователь случайно нажал не на ту кнопку и в итоге ввел какой-то символ. На этот случай выходит предупреждение о том, что вводимые данные не подходят и игрока просят ввести число.

```
Ваш ход!  
Осталось спичек: 100  
Сколько спичек вы берете: а  
Это не число!  
Введите число: █
```

Рисунок 3 – Некорректный ввод: символы ([digit or not](#))

Ввод числа, не входящего в диапазон

Также может случиться и такое, что пользователь случайно добавил лишнюю цифру и конечное число превышает указанный в правилах диапазон. Тогда выходит предупреждение с повторением нужного диапазона и повторный запрос ввода.

```
Введите число: 90  
Число не входит в диапазон от 1 до 10!  
Ваш ход!  
Осталось спичек: 100  
Сколько спичек вы берете: █
```

Рисунок 4 – Некорректный ввод: число не входит в заданный диапазон ([check diapazon](#))

Ввод числа, превышающего размер кучи

Ближе к окончанию игры, в том случае, если в куче осталось меньше 10 спичек, пользователь может ввести большее число, которое всё ещё будет входить в заданный диапазон. Однако в куче не может получиться отрицательного числа оставшихся спичек, поэтому пользователю будет предложено снова ввести число.

```
Ваш ход!  
Осталось спичек: 6  
Сколько спичек вы берете: 10  
Спичек осталось меньше, чем вы ввели!  
Ваш ход!  
Осталось спичек: 6  
Сколько спичек вы берете: █
```

Рисунок 5 – Некорректный ввод: число не входит в заданный диапазон ([check kol vo](#))

Ввод корректных данных

Если вводимые данные проходят все проверки, то ход переходит к боту.

Ход бота

Ход бота заключается в выборе случайного числа в диапазоне.

```
Ходит Бот!  
Осталось спичек: 90  
Он взял: 4  
Ваш ход!  
Осталось спичек: 86  
Сколько спичек вы берете: 10
```

Рисунок 6 – ход бота ([xod bota](#))

Конец игры

Существует два варианта окончания, для каждого из которых выводится соответствующая надпись.

Победил игрок

```
Ваш ход!  
Осталось спичек: 6  
Сколько спичек вы берете: 6  
Вы победили!  
Начинаем игру?  
1 - Да; 2 - Нет;
```

Рисунок 7 – вывод надписи о победе игрока ([winner](#))

Победил бот

```
Ходит Бот!  
Осталось спичек: 6  
Он взял: 6  
Бот победил!  
Начинаем игру?  
1 - Да; 2 - Нет;
```

Рисунок 8 – вывод надписи о победе бота ([winner](#))

Тестирование приложения

Все случаи с ошибками, которые были показаны ранее, проверяются посредством тестов:

```
TEST 1/6 digit:check_digit [OK]  
TEST 2/6 good_diapazon:check_good_diapazon [OK]  
TEST 3/6 bad_diapazon:check_bad_diapazon [OK]  
TEST 4/6 digit_or_not:check_digit_or_not [OK]  
TEST 5/6 kol_vo:check_kol_vo [OK]  
TEST 6/6 bad_kol_vo:check_bad_kol_vo [OK]  
RESULTS: 6 tests (6 ok, 0 failed, 0 skipped) ran in 0 ms
```

Рисунок 9 – проверка работоспособности ([ctest](#))

Личный вклад в проект

Непосредственное участие было принято:

1. в реализации структуры проекта – создание директорий для файлов, библиотек и объектных файлов;
2. в написании основного функционала приложения, представленного в `lib100matches.c`;
3. в исправлении разных мелких неточностей в других частях проекта – написание `readme`, добавление `clang-format` и тому подобное;

Приложение. Текст программы

// file 100matches.c

```
#include <lib100matches.h>

int main()
{
    printf("Добро пожаловать в игру 100 спичек.\n");
    printf("Описание игры: из кучки, первоначально содержащей 100 спичек, двое "
           "играющих поочередно берут по несколько спичек: не менее одной и не "
           "более десяти. Выигрывает взявший последнюю спичку.\n");
    while (1) {
        if (start() == 1)
            break;
    }
}
```

// file lib100matches.c

```
#include <ctype.h>
#include <lib100matches.h>
#include <stdio.h>
#include <stdlib.h>

enum Errors {
    er_not_number,
    er_not_diapazon,
    er_too_much,
};

int info(int error)
{
    switch (error) {
        case er_not_number:
            printf("Это не число!\n");
            return 1;
        case er_not_diapazon:
            printf("Число не входит в диапазон от 1 до 10!\n");
            return 2;
        case er_too_much:
            printf("Спичек осталось меньше, чем вы ввели!\n");
            return 3;
    }
    return 0;
}
```

```

int winner(int* kucha, int queue)
{
    if (*kucha == 0) {
        if (queue == 1) {
            printf("\nПобедил игрок!\n\n");
            return 1;
        } else {
            printf("\nПобедил Бот!\n\n");
            return 2;
        }
    }
    return 0;
}

void queue(int* kucha)
{
    xod_playera(kucha);
    if (winner(kucha, 1) == 0) {
        xod_bota(kucha);
        winner(kucha, 2);
    }
}

int xod_bota(int* kucha)
{
    int bot;
    printf("\nХодит Бот!\n");
    printf("Осталось спичек: %d\n", *kucha);
    if (*kucha <= 10) {
        bot = *kucha;
    } else {
        bot = rand() % 10 + 1;
    }
    printf("Он взял: %d\n", bot);
    *kucha -= bot;
    return 0;
}

int digit_or_not(char* xod)
{
    if (isdigit(xod[0]) == 0 && isdigit(xod[1]) == 0) {
        return 1;
    }
    return 0;
}

int check_diapazon(int xod)
{
    if (xod >= 1 && xod <= 10) {
        return 0;
    }
}

```

```

        return 1;
    }

int check_kol_vo(int xod, int* kucha)
{
    if (*kucha >= xod) {
        *kucha -= xod;
        return 0;
    }
    return 1;
}

int checks(int* kucha, char* xod)
{
    if (digit_or_not(xod) != 0) {
        info(er_not_number);
        return 1;
    }
    int player;
    player = atoi(xod);
    if (check_diapazon(player) == 1) {
        info(er_not_diapazon);
        return 2;
    }
    if (check_kol_vo(player, kucha) == 1) {
        info(er_too_much);
        return 3;
    }
    return 0;
}

int xod_playera(int* kucha)
{
    fflush(stdin);
    char xod[20];
    printf("\nВаш ход!\n");
    printf("Осталось спичек: %d\n", *kucha);
    printf("Сколько спичек вы берете: ");
    fgets(xod, 20, stdin);
    if (xod[0] == '\n') {
        fgets(xod, 20, stdin);
    }
    if (checks(kucha, xod) != 0) {
        xod_playera(kucha);
    }
    return 0;
}

int start()
{
    int kucha;

```

```

char quit;
printf("Начинаем игру?\n1 - Да; 2 - Нет;\n");
scanf("%c", &quit);
while (1) {
    switch (quit) {
        case '2':
            return 1;
        case '1':
            kucha = 100;
            while (kucha > 0) {
                queue(&kucha);
            }
            printf("Начинаем игру?\n1 - Да; 2 - Нет;\n");
            scanf("\n%c", &quit);
            break;
        default:
            printf("Нет такого выбора!\nВаше действие: ");
            scanf(" %c", &quit);
            break;
    }
}
return 0;
}

```

// file parser_test.c

```

#include <ctest.h>
#include <lib100matches.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#define SQR(x) (x) * (x)

CTEST(digit, check_digit)
{
    char* xod = "5";
    const int expect = 0;
    int res = digit_or_not(xod);
    ASSERT_EQUAL(expect, res);
}

CTEST(good_diapazon, check_good_diapazon)
{
    int xod = 8;
    const int expect = 0;
    int res = check_diapazon(xod);
    ASSERT_EQUAL(expect, res);
}

CTEST(bad_diapazon, check_bad_diapazon)
{

```

```
    int xod = -1;
    const int expect = 1;
    int res = check_diapazon(xod);
    ASSERT_EQUAL(expect, res);
}
CTEST(digit_or_not, check_digit_or_not)
{
    char* xod = "h";
    const int expect = 1;
    int res = digit_or_not(xod);
    ASSERT_EQUAL(expect, res);
}
CTEST(kol_vo, check_kol_vo)
{
    int kucha = 15;
    int xod = 9;
    const int expect = 0;
    int res = check_kol_vo(xod, &kucha);
    ASSERT_EQUAL(expect, res);
}
CTEST(bad_kol_vo, check_bad_kol_vo)
{
    int kucha = 8;
    int xod = 9;
    const int expect = 1;
    int res = check_kol_vo(xod, &kucha);
    ASSERT_EQUAL(expect, res);
}
```