

Министерство образования и науки Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
по курсу «Программирование»
на тему «Лабиринт»

Выполнил студент группы 22BOA1:

Токарев А.А.

Принял:

Митрохин М.А.

удовл.

29.12.23



Пенза 2024

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра «Вычислительная техника»

«УТВЕРЖДАЮ»

Зав. кафедрой ВТ

Митрохин М. А.

«___» _____ 2023 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Токареву Артёму Александровичу Группа 22BOA1

Тема проекта: Разработка программы сложной структуры методом
нисходящего программирования

Исходные данные (технические требования) на проектирование

Наименование программы: игра «Лабиринт».

Задание. Программа должна выполнять следующие функции:

- Построение лабиринта с несвязанными стенами
- Переход к новому лабиринту, после прохождения прежнего
- Запись результата последней игры в файл
- Обновление и вывод на экран результата лучших игр

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных — списков, структур, файлов.
3. Режим работы видеосистемы — текстовый/графический.
4. Устройство ввода информации — клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система — Windows.
8. Система программирования — C/C++ на базе среды Microsoft Visual Studio 2019.
9. Язык программирования — C++.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Структура модулей программы, схема алгоритма программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

1. В соответствии с графиком выполнения курсового проекта.

Дата выдачи задания « 9 » сентября 2023 г

Дата защиты проекта « ____ » _____ 2023 г

Руководитель Митрохин Максим Александрович

Задание получил « 9 » сентября 2023 г

Студент Покарев Артём Александрович

Содержание

Введение	5
2 Основная часть	6
2.1 Постановка задачи	6
2.2 Выбор решения	7
2.2.1. Определение необходимых модулей программы	7
2.2.2 Логика построения игрового поля	8
2.3. Описание разработки программы	11
2.4. Отладка и тестирование	13
2.5. Описание программы	13
2.5.1 Разработка приложения Kurs.exe	13
2.6. Руководство пользователя	14
Заключение	18
Список используемых источников	19
Приложение А.1.	20
Приложение А.2.	20
Приложение А.2.	46

Введение

При создании средних по размеру приложений используется структурное программирование, идея которого заключается в том, что структура программы должна отражать структуру решаемой задачи, чтобы алгоритм решения был ясно виден из исходного текста. Для этого надо иметь средства для создания программы не только с помощью трех простых операторов, но и с помощью средств, более точно отражающих конкретную структуру алгоритма. С этой целью в программирование введено понятие подпрограммы — набора операторов, выполняющих нужное действие и не зависящих от других частей исходного кода. Программа разбивается на множество мелких подпрограмм (занимающих до 50 операторов — критический порог для быстрого понимания цели подпрограммы), каждая из которых выполняет одно из действий, предусмотренных исходным заданием. Комбинируя эти подпрограммы, удастся формировать итоговый алгоритм уже не из простых операторов, а из законченных блоков кода, имеющих определенную смысловую нагрузку, причем обращаться к таким блокам можно по названиям. Получается, что подпрограммы — это новые операторы или операции языка, определяемые программистом.

Лабиринт (др.-греч. λαβύρινθος) — структура (в нашем случае в двухмерном пространстве), состоящая из запутанных путей к выходу (и/или путей, ведущих в тупик).

Под лабиринтом у древних греков и римлян подразумевалось более или менее обширное пространство, состоящее из многочисленных залов, камер, дворов и переходов, расположенных по сложному и запутанному плану, с целью запутать и не дать выхода несведущему в плане лабиринта человеку.

Считается, что если проходить лабиринт, касаясь только одного из краёв его

стенок, то лабиринт обязательно будет пройден, хотя это не всегда верно: в лабиринте с несвязанными стенами этот способ может не сработать.

В процессе выполнения курсовой работы была разработана программа на языке C/C++, для игры в «Лабиринт».

2 Основная часть

2.1 Постановка задачи

Необходимо разработать программу «Лабиринт»

Программа должна предоставить возможности: Построение игрового поля, запись последнего результата в файл, загрузку последнего результата из файла, запись 10 лучших игр в файл, загрузку 10 лучших игр из файла

Необходимо использовать сложные типы данных – массивы, структуры, файлы. Необходимо определиться с элементами управления. Необходимо изучить приемы и механизмы псевдографического интерфейса. Это необходимо для понимания, как и почему на консоли появляются псевдографические символы линии и т.д.

Устройство ввода – клавиатура. Необходимо научиться анализировать и различать события, возникающие от работы с клавиатурой. Необходимо однозначно идентифицировать и выполнять те или иные действия в зависимости от действий пользователя. Это необходимо для легкого использования программы.

Пользовательский интерфейс должен быть построен на основе меню. Необходимо изучить приемы программирования меню, выбрать подходящий вид меню. Это необходимо для создания интуитивно понятного интерфейса.

Многомодульность программы. Необходимо разделить программу на логические модули. Это повысит локализацию ошибок при отладке и тестировании программы.

2.2 Выбор решения

2.2.1. Определение необходимых модулей программы

Разработанная программа состоит из следующих модулей



Рисунок 1 - Структура модулей программы.

Файлы: «Kurs.cpp», «program.cpp», «Header.h», - необходимы для включения функций обработки данных. Данные файлы содержат функции построения игрового поля, сохранения и загрузки результатов последней и 10 лучших игр.

Приложение «Kurs.exe» представляет собой совокупность данных модулей. Приложение использует данные, содержащиеся в файле «task1». Все вышеперечисленное удовлетворяет требованию модульности программы данной курсовой работы. Управление приложением осуществляется с

помощью клавиатуры. При запуске программы строится игровое поле с возможностью начать прохождение лабиринта.

2.2.2 Логика построения игрового поля

Начиная с середины игрового поля и с каждым последующим разом задаётся случайное направление (верх, низ, право, лево). Далее идёт проверка, возможно ли построение в эту сторону (нет ли с этой стороны прохода или конца игрового поля). Если эта сторона недоступна, идёт проверка, какие стороны для построения доступны и задаётся случайное из них. Если такого направления нет, то есть мы попали в тупик, возвращаемся по тому же маршруту, по которому пришли до момента, откуда можно продолжить построение. Если в результате возвращения мы вернулись в положение, из которого всё начиналось, значит всё поле с заданными ему высотой и шириной было отведено под проходы лабиринта.

После случайным образом задаётся расположение выхода из лабиринта, а так же ломаются случайные стены, чтобы получился лабиринт с несвязанными стенами.



Рисунок 2 - Схемы данных и ресурсов

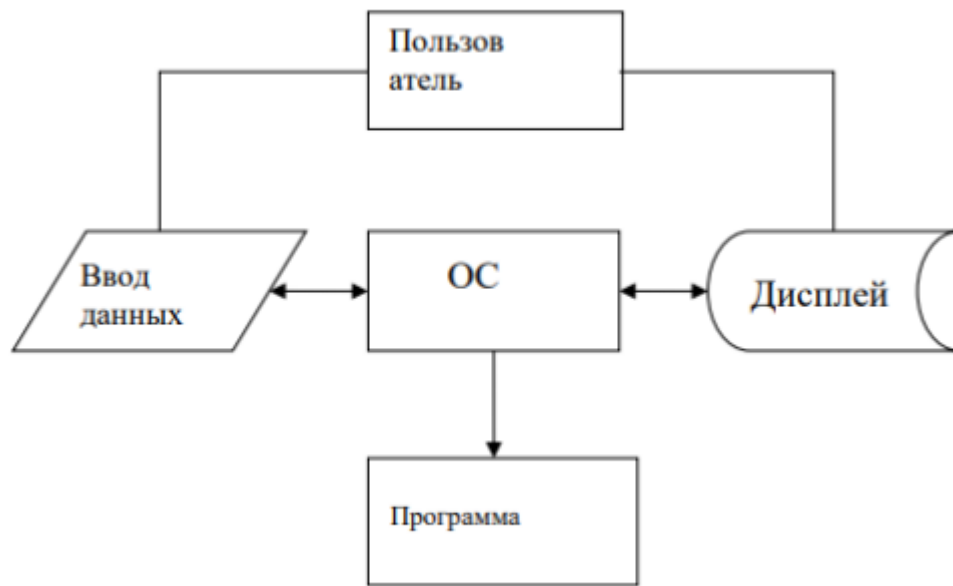


Рисунок 3 - Схема данных

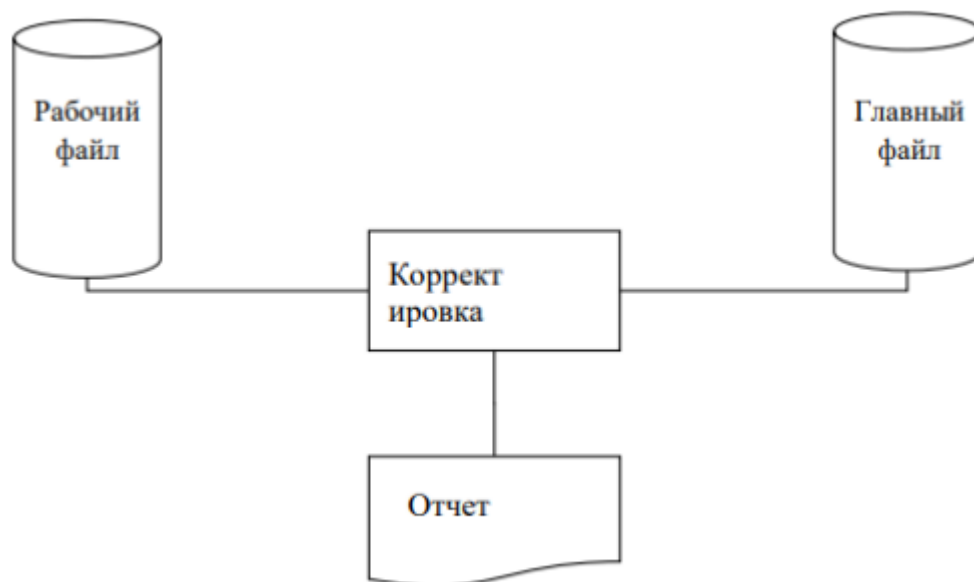


Рисунок 4 - Схема ресурсов

2.3. Описание разработки программы

Для написания данной программы были использованы языки программирования Си, C++.

В качестве среды программирования был выбран программный продукт VisualStudio 2019.

Интерфейс информационной системы представлен в виде меню, т.к. он является наиболее удобным. Исполнение навигации программы по пунктам меню будет осуществляться с помощью клавиатуры.

Разработка программы началась с реализации набора функций, необходимых для построения игрового поля, работы с файлами. Используются:

void Setup () – функция, вызываемая для построения игрового поля

void Input () – Основная логика игры, завязанная на нажатии клавиши

void Logic () – Оставшаяся логика игры, не завязанная на нажатии клавиш

Основная программа выполняется функцией main(). В функции main реализован цикл обработки событий. Цикл обработки событий реагирует на события, вызванные клавиатурой, выполняет дешифрацию события и вызывает функции для обработки того или иного события. Был разработан алгоритм функции main. Блок-схема функции представлена на рисунке.

Был разработан алгоритм функции main. Блок-схема функции представлена на рисунке. Функция отвечает за инициализацию консоли и за передачу управления в функции обработки меню. Цикл обработки событий главного окна обеспечивает активизацию меню и выход из него. Программа получает идентификатор выбранного пункта меню. В зависимости от идентификатора происходит перенаправление в саму игру, на экран выводится таблица лидеров или выводится результат последней игры.

В зависимости от введённого символа с клавиатуры, главная функция выполняет то или иное действие. При нахождении в меню и нажатии соответствующего символа или при переходе на следующий уровень, посредством нажатия клавиши вызывается функция по построению игрового поля.



Рисунок 5 - схема обработка событий

2.4. Отладка и тестирование

В качестве среды разработки была выбрана программа VisualStudio2019. Программа предоставляет все средства необходимые при разработке и отладке разрабатываемых модулей и программ. Для отладки использовались такие инструменты как точки останова, выполнение кода по шагам, анализ содержимого локальных и глобальных переменных, анализ содержимого памяти. Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с перерисовкой содержимого таблицы, с перерисовкой меню и многое другое. Также были добавлены проверки, связанные с недопустимостью обработки некоторых пунктов меню при определенных условиях.

2.5. Описание программы

2.5.1 Разработка приложения Kurs.exe

Приложение «Kurs.exe» является основным модулем программы. При запуске происходит построение игрового поля, после чего можно начать игру. На игру отводится заданное в программе время. Идёт подсчёт пройденных за это время уровней. Так же показывается оставшееся на игру время.

По завершении игры проситься ввести имя для записи результата в файл «Последней Игры» и возможно в файл «Рейтинг», где находится результат 10 лучших игр, после чего идёт перенаправление в меню.

Меню реагирует на нажатие клавиш: цифр от 1 до 3. При этом на выбранном пункте меню управление остаётся в этой функции.

В меню при выборе пункта 1 создаётся новая игра. По прошествии отведённого на игру времени, пользователя просят ввести имя от 3 до 15 символов, после чего результат записывается в файл «Последней игры», и

если этот результат лучше минимального в таблице лидеров – этот результат попадает в список «10 Лучших игр»

В меню при выборе пункта 2 выводится результат 10 лучших игр.

В меню при выборе пункта 3 выводится результат последней игры.

В любой момент можно переместиться в меню. При этом, находясь в лабиринте результат не сохраняется.

2.6. Руководство пользователя

Программа «Kurs.exe» предназначена для игры в «Лабиринт». Программа имеет интуитивно понятный интерфейс и поддерживает такие операции как игра в сам лабиринт, запись количества пройденных уровней с введённым именем в файл, перезапись результатов 10 лучших игр, выход в меню непосредственно во время игры, начала новой игры из меню, просмотр последней игры и просмотр 10 лучших игр из файла.



Сразу по началу нас кидает в саму игру, где перемещение осуществляется клавишами «w» - вверх, «a» - влево, «s» - вниз, «d» - вправо. Для прохождения уровня необходимо используя клавиши перемещения добраться до выхода, который представляет из себя «пролом» в стене игрового поля и соответствующей буквой «E».

Так же снизу выводится информация о том, за сколько секунд был завершён предыдущий уровень, сколько уровней было пройдено за эту игру и сколько времени (в секундах) на игру осталось. Из игры можно выйти непосредственно в меню, нажатием клавиши «m».

По истечении имени пользователя просят ввести имя, для записи результата, после чего требуется нажать ENTER

```
Game over  
Enter your name (3-15 symbols) >
```

При этом игра не позволяет ввести имя меньше 3х или больше 15 символов

```
Name is too short or too long, try another(3 - 15 symbols) >
```

Вводится любое имя

```
New_Name
```

После чего необходимо нажать «M», чтобы продолжить

```
Press 'M' to continue_
```

Сразу после нас кидает в меню игры



Для дальнейших действий требуется выбрать следующее действие «Новая игра», «10 лучших игр» или «Результат последней игры». Вводом в консоль «1», «2» или «3» соответственно, после чего требуется нажать ENTER.

При вводе «2» выводиться результат 10 лучших игр

```
Pepe - 9
NW - 8
MLZ - 7
name - 7
King - 6
AllStar - 5
smt - 5
NFWMB - 4
Rookie - 2
newbie - 1

'M' - return to Main Menu
```

Вернуться в меню можно нажатием «М»

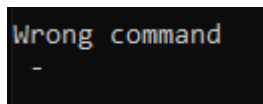
При вводе «3», выводиться результат последней игры

```
Last game result:
0 - New_Name

'M' - return to Main Menu
```

Вернуться в меню можно нажатием «М»

При этом при вводе в меню другого или нескольких символов вылезает предупреждение



При вводе «1», начинается новая игра с новым полем и отведённым под неё времени

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Были освоены приемы создания псевдографического интерфейса, изучены функции работы с консолью, способы обработки событий от клавиатуры. Усвоены механизмы реализации меню. Изучены функции работы с файлами. Получены базовые навыки программирования на языке C. Изучены основные возможности среды программирования VisualStudio2010. Получены навыки отладки и тестирования программ.

В рамках выполнения курсовой работы была написана программа, представляющая из себя игру «Лабиринт», предоставляющая возможность просмотра результатов 10 лучших игр, а также результата последней игры. Не требует большого количества системных ресурсов, построение игрового поля и отклик программы занимают мало времени.

Список используемых источников

1. Герберт Шилдт. Полный справочник по С.
2. Н. Культин. Основы программирования в Microsoft Visual C++ 2010, 2010г

Приложение А.1.

Файл Header.h

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>
void Setup();
void Input();
void Logic();
```

Приложение А.2.

Файл program.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <conio.h>
#include <windows.h>
using namespace std;

bool levelComplete, gameOver = false, c = false, block = false;
const int width = 21, height = 21; // 13x13 (+-4 - разница в две клетки)
const int buildDirUp = 1, buildDirRight = 2, buildDirDown = 3, buildDirLeft = 4;
int x, y, exitX, exitY, buildX, buildY, buildDir, buildDirPrev, buildDir2, i, buildComplete
= 0, stop = 0, exitCoords, score, timeForGame = 10, uScore, s, breakX, breakY;
char field[height][width], cameFrom[height][width], ratNames[10], ratScore[10],
currentName[15], a, name[15], fileScore[2], fileName[15];
time_t secondsLevel, secondsLevelEnd, secondsGame = time(NULL), currentTime;
char strScanf[] = "%d";

typedef struct { char Name[15], Score[2]; } ITEM;
ITEM Unit[10];

FILE* Rating;
FILE* RatingBackup;
FILE* LastGame;

//при запуске
// построение поля и его отображение
```

```

void Setup() {
    levelComplete = false;
    x = width / 2;
    y = height / 2;
    buildX = x;
    buildY = y;
    stop = 0;
    buildComplete = 0;
    secondsLevel = time(NULL);
    srand(time(0));

    exitCoords = rand() % ((width - 3) + (height - 3)) + 1;
    for (int i = 0; i < width; i++) {
        (*(field + 0) + i) = static_cast<char>(32);
        (*(field + 1) + i) = static_cast<char>(219);
    }
    (*(field + 1) + 0) = static_cast<char>(32);
    (*(field + 1) + (width - 1)) = static_cast<char>(32);
    for (int i = 2; i < height - 2; i++) {
        for (int j = 0; j < width; j++) {
            (*(cameFrom + i) + j) = static_cast<char>(32);
            if (j == 1 || j == width - 2)
                (*(field + i) + j) = static_cast<char>(219);
            else if (j == 0 || j == width - 1)
                (*(field + i) + j) = static_cast<char>(32);
            else if (i == y && j == x)
                (*(field + i) + j) = static_cast<char>(32);
            else if (i == exitY && j == exitX)
                (*(field + i) + j) = static_cast<char>(69);
            else {
                cout << static_cast<char>(219);
                (*(field + i) + j) = static_cast<char>(219);
            }
        }
    }
    for (int i = 0; i < width; i++) {
        (*(field + (height - 2)) + i) = static_cast<char>(219);
        (*(field + (height - 1)) + i) = static_cast<char>(32);
    }
}

```

```

*(*(field + (height - 2)) + 0) = static_cast<char>(32);
*(*(field + (height - 2)) + (width - 1)) = static_cast<char>(32);
//-----
    if (exitCoords <= ((width - 3) / 2)) {
        *(*(field + 1) + (exitCoords * 2)) = static_cast<char>(32);
        exitX = exitCoords * 2;
        exitY = 0;
    }
    else if (exitCoords <= ((width - 3) / 2) + ((height - 3) / 2)) {
        exitCoords -= ((width - 3) / 2);
        *(*(field + (exitCoords * 2)) + (width - 2)) = static_cast<char>(32);
        exitX = width - 1;
        exitY = exitCoords * 2;
    }
    else if (exitCoords <= (width - 3 + ((height - 3) / 2))) {
        exitCoords -= (((width - 3) / 2) + ((height - 3) / 2));
        *(*(field + (height - 2)) + (exitCoords * 2)) = static_cast<char>(32);
        exitX = exitCoords * 2;
        exitY = height - 1;
    }
    else if (exitCoords <= ((width - 3) + (height - 3))) {
        exitCoords -= ((width - 3) + ((height - 3) / 2));
        *(*(field + (exitCoords * 2)) + 1) = static_cast<char>(32);
        exitX = 0;
        exitY = exitCoords * 2;
    }
//-----
while (buildComplete != 1) {
    i++;
    srand(time(0) - i);
    buildDir = rand() % 4;
    switch (buildDir) {
        //верх
    case 0:
        //если верх недоступен
        if (*(*(field + (buildY - 2)) + buildX) == static_cast<char>(32)) {
            srand(time(0) - i);
            buildDir2 = rand() % 2;
            //если низ недоступен

```

```

        if (*(field + (buildY + 2)) + buildX) == static_cast<char>(32))
    {
        //если лево недоступно
        if (*(field + buildY) + (buildX - 2)) ==
static_cast<char>(32)) {
            //если право недоступно
            if (*(field + buildY) + (buildX + 2)) ==
static_cast<char>(32)) {
                stop = 1;
            }
            //иначе построить право
            else {
                buildX = buildX + 2;
                (*(field + buildY) + buildX) =
static_cast<char>(32);
                (*(field + buildY) + (buildX - 1)) =
static_cast<char>(32);
                (*(cameFrom + buildY) + buildX) =
buildDirLeft;
            }
        }
        // если право недоступно построить лево
        else if (*(field + buildY) + (buildX + 2)) ==
static_cast<char>(32)) {
            buildX = buildX - 2;
            (*(field + buildY) + buildX) =
static_cast<char>(32);
            (*(field + buildY) + (buildX + 1)) =
static_cast<char>(32);
            (*(cameFrom + buildY) + buildX) = buildDirRight;
        }
        // иначе либо право либо лево
        else {
            if (buildDir2 == 0) {
                buildX = buildX + 2;
                (*(field + buildY) + buildX) =
static_cast<char>(32);
                (*(field + buildY) + (buildX - 1)) =
static_cast<char>(32);
            }
        }
    }
}

```

```

buildDirLeft;
    }
    else {
        buildX = buildX - 2;
        *((field + buildY) + buildX) =
static_cast<char>(32);
        *((field + buildY) + (buildX + 1)) =
static_cast<char>(32);
        *((cameFrom + buildY) + buildX) =
buildDirRight;
    }
}
// Если низ все таки доступен, но недоступно лево
else if ((* (field + buildY) + (buildX - 2)) ==
static_cast<char>(32)) {
    //если право недоступно
    if ((* (field + buildY) + (buildX + 2)) ==
static_cast<char>(32)) {
        buildY = buildY + 2;
        *((field + buildY) + buildX) =
static_cast<char>(32);
        *((field + (buildY - 1)) + buildX) =
static_cast<char>(32);
        *((cameFrom + buildY) + buildX) = buildDirUp;
    }
    // иначе либо право либо низ
    else {
        if (buildDir2 == 0) {
            buildX = buildX + 2;
            *((field + buildY) + buildX) =
static_cast<char>(32);
            *((field + buildY) + (buildX - 1)) =
static_cast<char>(32);
            *((cameFrom + buildY) + buildX) =
buildDirLeft;
        }
        else {

```

```

        buildY = buildY + 2;
        *((field + buildY) + buildX) =
static_cast<char>(32);
        *((field + (buildY - 1)) + buildX) =
static_cast<char>(32);
        *((cameFrom + buildY) + buildX) =
buildDirUp;
    }
}
// низ доступен, недоступно право
// либо низ, либо лево
else {
    if (buildDir2 == 0) {
        buildX = buildX - 2;
        *((field + buildY) + buildX) =
static_cast<char>(32);
        *((field + buildY) + (buildX + 1)) =
static_cast<char>(32);
        *((cameFrom + buildY) + buildX) = buildDirRight;
    }
    else {
        buildY = buildY + 2;
        *((field + buildY) + buildX) =
static_cast<char>(32);
        *((field + (buildY - 1)) + buildX) =
static_cast<char>(32);
        *((cameFrom + buildY) + buildX) = buildDirUp;
    }
}
}
//иначе построить путь вверх
else {
    buildY = buildY - 2;
    *((field + buildY) + buildX) = static_cast<char>(32);
    *((field + (buildY + 1)) + buildX) = static_cast<char>(32);
    *((cameFrom + buildY) + buildX) = buildDirDown;
}
break;

```

```

//-----
//право
case 1:
//если право недоступно
if (*(field + buildY) + (buildX + 2)) == static_cast<char>(32) {
    srand(time(0) - i);
    buildDir2 = rand() % 2;
//если лево недоступно
if (*(field + buildY) + (buildX - 2)) == static_cast<char>(32))
{
    //если верх недоступен
    if (*(field + (buildY - 2)) + buildX) ==
static_cast<char>(32)) {
        //если низ недоступен
        if (*(field + (buildY + 2)) + buildX) ==
static_cast<char>(32)) {
            stop = 1;
        }
        //иначе построить низ
        else {
            buildY = buildY + 2;
            (*(field + buildY) + buildX) =
static_cast<char>(32);
            (*(field + (buildY - 1)) + buildX) =
static_cast<char>(32);
            (*(cameFrom + buildY) + buildX) =
buildDirUp;
        }
    }
    // если низ недоступен построить верх
    else if (*(field + (buildY - 2)) + buildX) ==
static_cast<char>(32)) {
        buildY = buildY - 2;
        (*(field + buildY) + buildX) =
static_cast<char>(32);
        (*(field + (buildY + 1)) + buildX) =
static_cast<char>(32);
        (*(cameFrom + buildY) + buildX) =
buildDirDown;
    }
}

```

```

    }
    // иначе либо низ либо верх
    else {
        if (buildDir2 == 0) {
            buildY = buildY + 2;
            *((field + buildY) + buildX) =
static_cast<char>(32);
            *((field + (buildY - 1)) + buildX) =
static_cast<char>(32);
            *((cameFrom + buildY) + buildX) =
buildDirUp;
        }
        else {
            buildY = buildY - 2;
            *((field + buildY) + buildX) =
static_cast<char>(32);
            *((field + (buildY + 1)) + buildX) =
static_cast<char>(32);
            *((cameFrom + buildY) + buildX) =
buildDirDown;
        }
    }
}
// Если лево все таки доступно, но недоступен верх
else if ((* (field + (buildY - 2)) + buildX) ==
static_cast<char>(32)) {
    //если низ недоступен
    if ((* (field + (buildY + 2)) + buildX) ==
static_cast<char>(32)) {
        buildX = buildX - 2;
        *((field + buildY) + buildX) =
static_cast<char>(32);
        *((field + buildY) + (buildX + 1)) =
static_cast<char>(32);
        *((cameFrom + buildY) + buildX) = buildDirRight;
    }
    // иначе либо низ либо лево
    else {
        if (buildDir2 == 0) {

```

```

static_cast<char>(32);
static_cast<char>(32);
buildDirRight;

static_cast<char>(32);
static_cast<char>(32);
buildDirUp;

        }
    }
    // лево доступно, недоступен низ
    // либо лево, либо верх
    else {
        if (buildDir2 == 0) {
            buildX = buildX - 2;
            (*(field + buildY) + buildX) =
            (*(field + buildY) + (buildX + 1)) =
            (*(cameFrom + buildY) + buildX) =
        }
        else {
            buildY = buildY - 2;
            (*(field + buildY) + buildX) =
            (*(field + (buildY + 1)) + buildX) =
            (*(cameFrom + buildY) + buildX) =
        }
    }
    buildDirDown;

```

```

        }
    }
}
else {
    buildX = buildX + 2;
    (*(field + buildY) + buildX) = static_cast<char>(32);
    (*(field + buildY) + (buildX - 1)) = static_cast<char>(32);
    (*(cameFrom + buildY) + buildX) = buildDirLeft;
}
break;
//-----
//низ
case 2:
    //если низ недоступен
    if (*(field + (buildY + 2)) + buildX == static_cast<char>(32)) {
        srand(time(0) - i);
        buildDir2 = rand() % 2;
        //если верх недоступен
        if (*(field + (buildY - 2)) + buildX ==
static_cast<char>(32)) {
            //если лево недоступно
            if (*(field + buildY) + (buildX - 2) ==
static_cast<char>(32)) {
                //если право недоступно
                if (*(field + buildY) + (buildX + 2) ==
static_cast<char>(32)) {
                    stop = 1;
                }
                //иначе построить право
            } else {
                buildX = buildX + 2;
                (*(field + buildY) + buildX) =
static_cast<char>(32);
                (*(field + buildY) + (buildX - 1)) =
static_cast<char>(32);
                (*(cameFrom + buildY) + buildX) =
buildDirLeft;
            }
        }
    }
}

```

```

static_cast<char>(32)) {
    static_cast<char>(32);
    static_cast<char>(32);
    buildDirRight;

    // если право недоступно построить лево
    else if (*(field + buildY) + (buildX + 2)) ==
        buildX = buildX - 2;
        (*(field + buildY) + buildX) =
        (*(field + buildY) + (buildX + 1)) =
        (*(cameFrom + buildY) + buildX) =
    }
    // иначе либо право либо лево
    else {
        if (buildDir2 == 0) {
            buildX = buildX + 2;
            (*(field + buildY) + buildX) =
            (*(field + buildY) + (buildX - 1)) =
            (*(cameFrom + buildY) + buildX) =
        }
        else {
            buildX = buildX - 2;
            (*(field + buildY) + buildX) =
            (*(field + buildY) + (buildX + 1)) =
            (*(cameFrom + buildY) + buildX) =
        }
    }
}

// Если верх все таки доступен, но недоступно лево
else if (*(field + buildY) + (buildX - 2)) ==
static_cast<char>(32)) {
    //если право недоступно

```

```

static_cast<char>(32)) {
    static_cast<char>(32);
    static_cast<char>(32);
    buildDirDown;

    if (*(field + buildY) + (buildX + 2)) ==
        buildY = buildY - 2;
        (*(field + buildY) + buildX) =
            (*(field + (buildY + 1)) + buildX) =
                (*(cameFrom + buildY) + buildX) =
                    }
                    // иначе либо право либо верх
                    else {
                        if (buildDir2 == 0) {
                            buildX = buildX + 2;
                            (*(field + buildY) + buildX) =
                                (*(field + buildY) + (buildX - 1)) =
                                    (*(cameFrom + buildY) + buildX) =
                                        }
                                        else {
                                            buildY = buildY - 2;
                                            (*(field + buildY) + buildX) =
                                                (*(field + (buildY + 1)) + buildX) =
                                                    (*(cameFrom + buildY) + buildX) =
                                                        }
                                                        }
                                                        }
                                                        // верх доступен, недоступно право
                                                        // либо верх, либо лево
                                                        else {
                                                            if (buildDir2 == 0) {
                                                                buildX = buildX - 2;
                                                                (*(field + buildY) + buildX) =
                                                                    static_cast<char>(32);

```

```

static_cast<char>(32);
buildDirRight;

        }
        else {
            buildY = buildY - 2;
            (*(field + buildY) + buildX) =
static_cast<char>(32);
            (*(field + (buildY + 1)) + buildX) =
static_cast<char>(32);
            (*(cameFrom + buildY) + buildX) =
buildDirDown;
        }
    }
}
else {
    buildY = buildY + 2;
    (*(field + buildY) + buildX) = static_cast<char>(32);
    (*(field + (buildY - 1)) + buildX) = static_cast<char>(32);
    (*(cameFrom + buildY) + buildX) = buildDirUp;
}
break;
//-----
//лево
case 3:
    //если лево недоступно
    if (*(field + buildY) + (buildX - 2)) == static_cast<char>(32)) {
        srand(time(0) - i);
        buildDir2 = rand() % 2;
        //если право недоступно
        if (*(field + buildY) + (buildX + 2)) == static_cast<char>(32))
{
            //если верх недоступен
            if (*(field + (buildY - 2)) + buildX) ==
static_cast<char>(32)) {
                //если низ недоступен
                if (*(field + (buildY + 2)) + buildX) ==
static_cast<char>(32)) {

```

```

        stop = 1;
    }
    // иначе построить низ
    else {
        buildY = buildY + 2;
        *((field + buildY) + buildX) =
static_cast<char>(32);
        *((field + (buildY - 1)) + buildX) =
static_cast<char>(32);
        *((cameFrom + buildY) + buildX) =
buildDirUp;
    }
}
// если низ недоступен построить верх
else if (*(field + (buildY + 2)) + buildX) ==
static_cast<char>(32)) {
    buildY = buildY - 2;
    *((field + buildY) + buildX) =
static_cast<char>(32);
    *((field + (buildY + 1)) + buildX) =
static_cast<char>(32);
    *((cameFrom + buildY) + buildX) =
buildDirDown;
}
// иначе либо низ либо верх
else {
    if (buildDir2 == 0) {
        buildY = buildY + 2;
        *((field + buildY) + buildX) =
static_cast<char>(32);
        *((field + (buildY - 1)) + buildX) =
static_cast<char>(32);
        *((cameFrom + buildY) + buildX) =
buildDirUp;
    }
    else {
        buildY = buildY - 2;
        *((field + buildY) + buildX) =
static_cast<char>(32);

```

```

static_cast<char>(32);
buildDirDown;

    }
}

// Если право все таки доступно, но недоступен верх
else if (*(field + (buildY - 2)) + buildX) ==
static_cast<char>(32)) {
    //если низ недоступен
    if (*(field + (buildY + 2)) + buildX) ==
static_cast<char>(32)) {
        buildX = buildX + 2;
        (*(field + buildY) + buildX) =
static_cast<char>(32);
        (*(field + buildY) + (buildX - 1)) =
static_cast<char>(32);
        (*(cameFrom + buildY) + buildX) = buildDirLeft;
    }
    // иначе либо низ либо право
    else {
        if (buildDir2 == 0) {
            buildX = buildX + 2;
            (*(field + buildY) + buildX) =
static_cast<char>(32);
            (*(field + buildY) + (buildX - 1)) =
static_cast<char>(32);
            (*(cameFrom + buildY) + buildX) =
buildDirLeft;
        }
        else {
            buildY = buildY + 2;
            (*(field + buildY) + buildX) =
static_cast<char>(32);
            (*(field + (buildY - 1)) + buildX) =
static_cast<char>(32);
            (*(cameFrom + buildY) + buildX) =
buildDirUp;
        }
    }
}

```

```

        }
    }
}
// право доступно, недоступен низ
// либо право, либо верх
else {
    if (buildDir2 == 0) {
        buildX = buildX + 2;
        (*(field + buildY) + buildX) =
static_cast<char>(32);
        (*(field + buildY) + (buildX - 1)) =
static_cast<char>(32);
        (*(cameFrom + buildY) + buildX) = buildDirLeft;
    }
    else {
        buildY = buildY - 2;
        (*(field + buildY) + buildX) =
static_cast<char>(32);
        (*(field + (buildY + 1)) + buildX) =
static_cast<char>(32);
        (*(cameFrom + buildY) + buildX) =
buildDirDown;
    }
}
}
else {
    buildX = buildX - 2;
    (*(field + buildY) + buildX) = static_cast<char>(32);
    (*(field + buildY) + (buildX + 1)) = static_cast<char>(32);
    (*(cameFrom + buildY) + buildX) = buildDirRight;
}
break;
default:
    break;
}
if (stop == 1) {
    //если пришёл сверху
    if (*(cameFrom + buildY) + buildX) == buildDirUp)
        buildY = buildY - 2;
}

```

```

        //если пришёл справа
        else if (*(cameFrom + buildY) + buildX) == buildDirRight)
            buildX = buildX + 2;
        //если пришёл снизу
        else if (*(cameFrom + buildY) + buildX) == buildDirDown)
            buildY = buildY + 2;
        //если пришёл слева
        else
            buildX = buildX - 2;
        if (*(field + (buildY - 2)) + buildX) == static_cast<char>(219) ||
            (*(field + (buildY + 2)) + buildX) == static_cast<char>(219) ||
            (*(field + buildY) + (buildX - 2)) == static_cast<char>(219) ||
            (*(field + buildY) + (buildX + 2)) == static_cast<char>(219))
            stop = 0;
    }
    if (buildX == (width / 2) && buildY == (height / 2))
        buildComplete = 1;
}
for (int k = 0; k < (((width - 3) / 2 + (height - 3) / 2) / 2); k++) {
    srand(time(0) + k);
    breakX = rand() % ((width - 3) / 2) + 1;
    breakY = rand() % (((height - 3) / 2) - 1) + 1;
    (*(field + (breakY * 2 + 1)) + (breakX * 2)) = static_cast<char>(32);
}
system("cls");
for (int i = 0; i < height; i++) {
    for (int j = 0; j < width; j++) {
        if (i == y && j == x)
            (*(field + i) + j) = static_cast<char>(48);
        if (i == exitY && j == exitX)
            (*(field + i) + j) = static_cast<char>(69);
        cout << (*(field + i) + j);
    }
    cout << endl;
}
printf(" Previous level completed in %lld seconds", secondsLevelEnd);
printf("\n Levels completed - %d", score);
printf("\n Time left - %lld", (timeForGame - (currentTime - secondsGame)));
printf("\n\n 'M' - return to Main Menu");

```

```

}
// Считывание действий пользователя
void Input() {
    if (_kbhit()) {
        switch (_getch()) {
            case 'a':
                if ((timeForGame - (currentTime - secondsGame)) > 0 && !block) {
                    if (*(field + y) + (x - 1)) == static_cast<char>(32)) {
                        (*(field + y) + x) = static_cast<char>(32);
                        x = x - 2;
                        (*(field + y) + x) = static_cast<char>(48);
                    }
                    system("cls");
                    for (int i = 0; i < height; i++) {
                        for (int j = 0; j < width; j++)
                            cout << (*(field + i) + j);
                        cout << endl;
                    }
                    printf(" Previous level completed in %lld seconds",
secondsLevelEnd);

                    printf("\n Levels completed - %d", score);
                    printf("\n Time left - %lld", (timeForGame - (currentTime -
secondsGame)));

                    printf("\n\n 'M' - return to Main Menu");
                }
            else if (block && a == static_cast<char>(50)) {
                system("cls");
                for (i = 9; i >= 0; i--) {
                    printf("%s - %s\n", Unit[i].Name, Unit[i].Score);
                }
                printf("\n\n 'M' - return to Main Menu");
            }
            else if (block && a == static_cast<char>(51)) {
                system("cls");
                LastGame = fopen("LastGame.txt", "r");
                fscanf(LastGame, "%s%s", fileScore, fileName);
                printf(" Last game result:\n %s - %s", fileScore, fileName);
                fclose(LastGame);
                printf("\n\n 'M' - return to Main Menu");
            }
        }
    }
}

```

```

    }
    else {
        system("cls");
        printf("Game over\n");
        printf("Enter your name (3-15 symbols) > ");
        scanf("%s", &currentName);
        while (strlen(currentName) > 15 ||
                strlen(currentName) < 3) {
            system("cls");
            printf("Name is too short or too long, try another(3 - 15
symbols) > \n");

            scanf("%s", &currentName);
        }
        LastGame = fopen("LastGame.txt", "w");
        *fileScore = score + '0';
        fprintf(LastGame, "%s\n%s", fileScore, currentName);
        fclose(LastGame);
        system("cls");
        printf("Press 'M' to continue");
    }
    break;
    //-----
case 'd':
    if ((timeForGame - (currentTime - secondsGame)) > 0
&& !block) {
        if (*(field + y) + (x + 1)) == static_cast<char>(32)) {
            (*(field + y) + x) = static_cast<char>(32);
            x = x + 2;
            (*(field + y) + x) = static_cast<char>(48);
        }
        system("cls");
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++)
                cout << (*(field + i) + j);
            cout << endl;
        }
        printf(" Previous level completed in %lld seconds",
secondsLevelEnd);

        printf("\n Levels completed - %d", score);
    }
}

```

```

        printf("\n Time left - %lld", (timeForGame - (currentTime
- secondsGame)));

        printf("\n\n 'M' - return to Main Menu");
    }
    else if (block && a == static_cast<char>(50)) {
        system("cls");
        for (i = 9; i >= 0; i--) {
            printf("%s - %s\n", Unit[i].Name, Unit[i].Score);
        }
        printf("\n\n 'M' - return to Main Menu");
    }
    else if (block && a == static_cast<char>(51)) {
        system("cls");
        LastGame = fopen("LastGame.txt", "r");
        fscanf(LastGame, "%s%s", fileScore, fileName);
        printf(" Last game result:\n %s - %s", fileScore, fileName);
        fclose(LastGame);
        printf("\n\n 'M' - return to Main Menu");
    }
    else {
        system("cls");
        printf("Game over\n");
        printf("Enter your name (3-15 symbols) > ");
        scanf("%s", &currentName);
        while (strlen(currentName) > 15 ||
            strlen(currentName) < 3) {
            system("cls");
            printf("Name is too short or too long, try another(3
- 15 symbols) > \n");

            scanf("%s", &currentName);
        }
        LastGame = fopen("LastGame.txt", "w");
        *fileScore = score + '0';
        fprintf(LastGame, "%s\n%s", fileScore, currentName);
        fclose(LastGame);
        system("cls");
        printf("Press 'M' to continue");
    }
    break;

```

```

//-----
case 'w':
    if ((timeForGame - (currentTime - secondsGame)) > 0 && !block) {
        if (*(field + (y - 1)) + x) == static_cast<char>(32)) {
            (*(field + y) + x) = static_cast<char>(32);
            y = y - 2;
            (*(field + y) + x) = static_cast<char>(48);
        }
        system("cls");
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++)
                cout << (*(field + i) + j);
            cout << endl;
        }
        printf(" Previous level completed in %lld seconds",
secondsLevelEnd);

        printf("\n Levels completed - %d", score);
        printf("\n Time left - %lld", (timeForGame - (currentTime -
secondsGame)));

        printf("\n\n 'M' - return to Main Menu");
    }
    else if (block && a == static_cast<char>(50)) {
        system("cls");
        for (i = 9; i >= 0; i--) {
            printf("%s - %s\n", Unit[i].Name, Unit[i].Score);
        }
        printf("\n\n 'M' - return to Main Menu");
    }
    else if (block && a == static_cast<char>(51)) {
        system("cls");
        LastGame = fopen("LastGame.txt", "r");
        fscanf(LastGame, "%s%s", fileScore, fileName);
        printf(" Last game result:\n %s - %s", fileScore, fileName);
        fclose(LastGame);
        printf("\n\n 'M' - return to Main Menu");
    }
    else {
        system("cls");
        printf("Game over\n");
    }

```

```

printf("Enter your name (3-15 symbols) > ");
scanf("%s", &currentName);
while (strlen(currentName) > 15 ||
      strlen(currentName) < 3) {
    system("cls");
    printf("Name is too short or too long, try another(3 - 15
symbols) > \n");

    scanf("%s", &currentName);
}
LastGame = fopen("LastGame.txt", "w");
*fileScore = score + '0';
fprintf(LastGame, "%s\n%s", fileScore, currentName);
fclose(LastGame);
system("cls");
printf("Press 'M' to continue");
}
break;
//-----
case 's':
    if ((timeForGame - (currentTime - secondsGame)) > 0 && !block) {
        if (*(field + (y + 1)) + x) == static_cast<char>(32)) {
            (*(field + y) + x) = static_cast<char>(32);
            y = y + 2;
            (*(field + y) + x) = static_cast<char>(48);
        }
        system("cls");
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++)
                cout << (*(field + i) + j);
            cout << endl;
        }
        printf(" Previous level completed in %lld seconds",
secondsLevelEnd);

        printf("\n Levels completed - %d", score);
        printf("\n Time left - %lld", (timeForGame - (currentTime -
secondsGame)));

        printf("\n\n 'M' - return to Main Menu");
    }
    else if (block && a == static_cast<char>(50)) {

```

```

        system("cls");
        for (i = 9; i >= 0; i--) {
            printf("%s - %s\n", Unit[i].Name,
                Unit[i].Score);
        }
        printf("\n\n 'M' - return to Main Menu");
    }
    else if (block && a == static_cast<char>(51)) {
        system("cls");
        LastGame = fopen("LastGame.txt", "r");
        fscanf(LastGame, "%s%s", fileScore, fileName);
        printf(" Last game result:\n %s - %s", fileScore, fileName);
        fclose(LastGame);
        printf("\n\n 'M' - return to Main Menu");
    }
    else {
        system("cls");
        printf("Game over\n");
        printf("Enter your name (3-15 symbols) > ");
        scanf("%s", &currentName);
        while (strlen(currentName) > 15 ||
            strlen(currentName) < 3) {
            system("cls");
            printf("Name is too short or too long, try another(3 - 15
symbols) > \n");

            scanf("%s", &currentName);
        }
        LastGame = fopen("LastGame.txt", "w");
        *fileScore = score + '0';
        fprintf(LastGame, "%s\n%s", fileScore, currentName);
        fclose(LastGame);
        system("cls");
        printf("Press 'M' to continue");
    }
    break;
case 'm':
    block = false;
    a = '0';
    //если счёт больше минимального в файле - начать перезапись

```

```
// полностью перезаписать файл имеющимся массивом
Rating = fopen("Rating.txt", "r+");
for (i = 0; i < 10; i++) {
    fscanf(Rating, "%s%s", Unit[i].Score, Unit[i].Name);
    uScore = *Unit[i].Score - '0';
    if (uScore < score && i != 0) {
        strcpy(Unit[i - 1].Score, Unit[i].Score);
        *Unit[i].Score = score + '0';
        strcpy(name, Unit[i - 1].Name);
        strcpy(Unit[i - 1].Name, Unit[i].Name);
        strcpy(Unit[i].Name, name);
    }
    else if (uScore < score && i == 0) {
        *Unit[i].Score = score + '0';
        strcpy(Unit[i].Name, currentName);
    }
}
fclose(Rating);
Rating = fopen("Rating.txt", "w");
for (i = 0; i < 10; i++)
    fprintf(Rating, "%s\n%s\n", Unit[i].Score, Unit[i].Name);
```

[illegible]

[illegible]

```

        printf("      |____|____|____|____|____|      (\\.-./)
|____|____|____|____|____|_\\n");
        printf("____|____|____|____|____|_      =  (^   Y   ^)  =
|____|____|____|____|____|_\\n");
        printf("|____|____|____|____|____|____|      / ^ --- `\\
____|____|____|____|____|____|_\\n");
        printf("
____|____|____|____|____|____|_U____|_U|____|____|____|____|____|_\\n");
        printf("
|____|____|____|____|____|____|____|____|____|____|____|____|____|____|____|_\\n");

        printf("____|____|____|____|____|____|____|____|____|____|____|____|____|____|____|_\\n"
);

        printf("\\n\\n");
        printf(" +---+      +---+      +---+\\n");
        printf(" | 1 | - New game   | 2 | - Top 10 runs   | 3 | - Last game
result\\n");

        printf(" +---+      +---+      +---+\\n");
        printf("\\nWrong command");
        printf("\\n - ");

        scanf("%s", &a);
    }
    if (a == '1') {
        a = '0';
        secondsLevelEnd = 0;
        score = 0;
        secondsGame = time(NULL);
        Setup();
    }
    if (a == '2') {
        block = true;
        system("cls");
        Rating = fopen("Rating.txt", "r");
        for (i = 0; i < 10; i++) {
            fscanf(Rating, "%s%s", Unit[i].Score, Unit[i].Name);
        }
        for (i = 9; i >= 0; i--) {
            printf("%s - %s\\n", Unit[i].Name, Unit[i].Score);

```

```

        }
        fclose(Rating);
        printf("\n\n 'M' - return to Main Menu");
    }
    if (a == '3') {
        block = true;
        system("cls");
        LastGame = fopen("LastGame.txt", "r");
        fscanf(LastGame, "%s%s", fileScore, fileName);
        printf(" Last game result:\n %s - %s", fileScore, fileName);
        fclose(LastGame);
        printf("\n\n 'M' - return to Main Menu");
    }
    break;
}
}
}
//Логика игры
void Logic() {
    currentTime = time(NULL);
    if (x == exitX && y == exitY) {
        levelComplete = true;
        secondsLevelEnd = time(NULL);
        secondsLevelEnd -= secondsLevel;
        score++;
        Setup();
    }
}

```

Приложение А.2.

Файл Kurs.cpp

```

#include "Header.h"
int c = 0;
int main() {
    Setup();
    while (c != 1) {
        Input();
        Logic();
    }
}

```

```
        return 0;
    }
    //при старте указать нужны ли случайно сломанные стены
```