

Оглавление

[**Введение** 1](#_Toc90283082)

[**1.** **Постановка задачи** 2](#_Toc90283083)

[**2.** **Теоретическая часть задания** 3](#_Toc90283084)

[**3.** **Описание алгоритма поставленной задачи** 4](#_Toc90283085)

[**4.** **Пример ручного расчёта задачи и вычислений** 6](#_Toc90283086)

[**5.** **Описание самой программы** 8](#_Toc90283087)

[**6.** **Тесты** 10](#_Toc90283088)

[**Заключение. Вывод о работе программы** 12](#_Toc90283089)

[**Список использованных источников** 13](#_Toc90283090)

[**Приложение А. Листинг программы.** 14](#_Toc90283091)

# 

# **Введение**

Основная задача графов – это отображать связи между разными сущностями, то есть вершинами. К примеру, это может использоваться в карте метро, на которой станции являются вершинами графа, а связывающие их перегоны – ребрами.

С помощью графов можно представить связи между пользователями социальной сети или ссылки между разными страницами одного сайта. В таких графах рёбра будут иметь направление – пользователь А подписан на пользователя Б, а не наоборот. Это ориентированные графы.

В настоящее время это очень востребованная структура данных, так как она позволяет работать с большими объёмами плохо структурированной информации. На графах, например, основываются разнообразные системы рекомендаций и ранжирования контента.

Самый популярный способ для представления графов в языках программирования – это матрицы смежности.

Основные алгоритмы на графах – это обход графа и нахождение кратчайшего пути. Есть два пути обхода вершин в графе: поиск в ширину и глубину. Существует множество алгоритмов нахождения кратчайшего пути, например, алгоритм Дейкстры.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio2019, язык программирования – С. Целью данной курсовой работы является разработка программы на языке С, который является широко используемым. Именно с его помощью в данном курсовом проекте реализуется алгоритм нахождения кратчайшего пути.

# **Постановка задачи**

Требуется разработать программу, которая будет обладать функционалом необходимым для прохождения игры “Лабиринт”.

Исходный граф в программе должен задаваться матрицей смежности, причём при генерации данных должны быть предусмотрены определённые условия, формирующие лабиринт. Программа должна работать так, чтобы пользователь мог ввести количество вершин для генерации матрицы смежности, загрузить ранее сгенерированную матрицу смежности. Пользователь должен иметь возможность выбора сложности. После обработки этих данных на экран должен выводиться лабиринт основанный на графе, где каждая вершина лабиринта соответствует вершине графа. Необходимо предусмотреть различные исходы игры, чтобы программа не выдавала ошибок и работала правильно. Устройство ввода – клавиатура и мышь.

# **Теоретическая часть задания**

Лабиринт задаётся графом, сгенерированным следующим образом: каждой вершине графа соответствует вершина лабиринта, смежными могут быть только те вершины, которые соответствуют соседним вершинам лабиринта.

Алгоритм генерации гарантирует наличие пути от входа к выходу лабиринта.

Для игры лабиринт необходимо реализовать простейшего “Противника”, который будет проходить лабиринт одновременно с игроком. Для реализации прохождения лабиринта “Противником” необходимо найти кратчайший путь от начала к выходу лабиринта.

# **Описание алгоритма поставленной задачи**

Алгоритм генерации лабиринта работает на графе, отражающем лабиринт.

Алгоритм работает пошагово — на каждом шаге он записывает текущую вершину в стек и “Пробивает стену” к следующей случайной, соседней, не посещенной вершине, если такой вершины нет, то алгоритм возвращается к предыдущей.

Когда алгоритм вернётся в начальную вершину, это будет означать, что все вершины графа уже посещены и алгоритм закончил генерацию лабиринта.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

Рисунок 1 - Пример работы

Алгоритм нахождения пути реализован двумя циклами и работает следующим образом: первый цикл — обход графа в ширину, начиная с начала лабиринта, с сохранением расстояний в вектор, второй цикл — анализ вектора расстояний и построение пути на его основе.

Алгоритм, при постройке пути, начинает с конца лабиринта, являющегося наиболее удалённой вершиной от начала чем все вершины пути.

На первом шаге вершина соответствующая концу лабиринта заносится в стек. На втором и последующих шагах, алгоритм заносит в стек соседнюю вершину, расстояние до которой от начала меньше на один. Таким образом, в стеке окажется очередь из последовательно идущих к выходу лабиринта вершин.

# **Пример ручного расчёта задачи и вычислений**

Проведу проверку алгоритма на данном размере поля.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Рисунок 2 - Пример работы

Возникла тупиковая ситуация. Алгоритм вернётся к ближайшей вершине из которой возможно продолжение.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Рисунок 3 - Пример работы

После генерации лабиринта программа находит расстояние до выхода.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  | 0 | 1 | 2 |  | 8 |  |
|  |  |  | 3 |  | 7 |  |
|  |  |  | 4 | 5 | 6 |  |
|  |  |  |  |  | 7 |  |
|  |  |  |  |  | 8 |  |
|  |  |  |  |  |  |  |

Рисунок 4 - Пример работы

Теперь, когда известны расстояния, можно построить путь к выходу.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Рисунок 5 - Пример работы

# **Описание самой программы**

Для написания данной программы использован язык программирования Си. Язык программирования Си - универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

В начале работы программы показывается графическое меню, с помощью которого можно изменить параметры игры, начать игру, выйти.

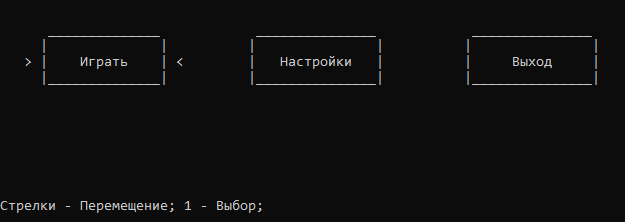


Рисунок 6 - Описание интерфейса

Если выбрать первый пункт, то программа перейдёт к игре.

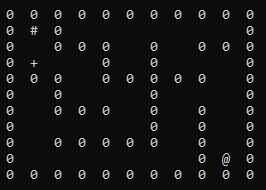


Рисунок 7 - Описание интерфейса

После того, как пользователь закончит игру, будет выведен результат.

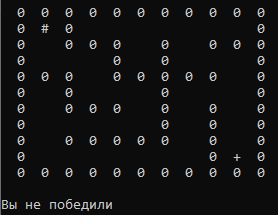


Рисунок 8 - Описание интерфейса

Если выбрать второй пункт, то программа перейдёт к настройкам, где пользователь может задать размер для случайной генерации лабиринта, загрузить готовый лабиринт, выбрать сложность игры или сбросить настройки к стандартным.

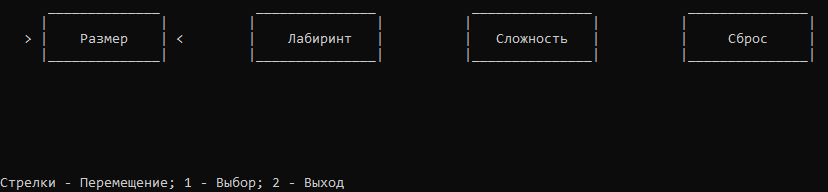


Рисунок 9 - Описание интерфейса

# **Тесты**

Среда разработки Microsoft Visual Studio 2019 предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

Ниже продемонстрирован результат тестирования программы при вводе пользователем различных размеров лабиринта и вывод результата.



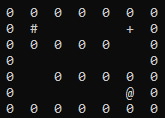


Рисунок 10 - Тестирование



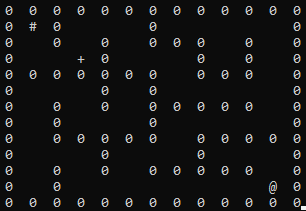


Рисунок 11 - Тестирование

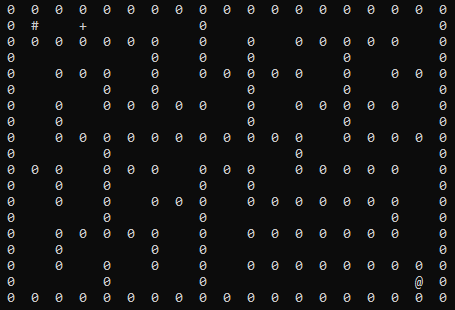


Рисунок 12 - Тестирование

С разными размерами матрицы смежности алгоритм работает верно.

# **Заключение. Вывод о работе программы**

Таким образом, в процессе создания данного проекта разработана программа, реализующая игровой агент Microsoft Visual Studio 2019.

При выполнении данной курсовой работы были получены навыки разработки программ и освоен алгоритм поиска путей в лабиринте. Приобретены навыки по осуществлению алгоритма поиска кратчайших расстояний. Углублены знания языка программирования Cи.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса.

Программа имеет большой и достаточный для использования функционал возможностей.

# **Список использованных источников**

1. Керниган Б. Ритчи Д. Язык программирования Си. 1985 г.
2. Константин Поляков. Хранение и обработка данных.
3. Дейтел и Пол Харви. Как программировать на Си.
4. А.А. Тюгашев. Языки программирования. Учебное пособие. 2018 г.

# **Приложение А. Листинг программы**

**LogicKurs.cpp**

#include "Sys.h"

void main()

{

setlocale(LC\_ALL, "Rus");

dvigat\_1();

}

void start()

{

int\*\* M = NULL;

char\*\* L = NULL;

int n;

float t;

El\* Q = NULL;

FILE\* F;

char name[] = "Settings.txt", lab[31];

if ((F = fopen(name, "r")) == NULL)

{

stand();

}

F = fopen(name, "r");

fseek(F, 43, SEEK\_SET);

if (fgetc(F) == '-')

{

fseek(F, 19, SEEK\_SET);

fscanf(F, "%d", &n);

M = gen(n);

}

else

{

fseek(F, 43, SEEK\_SET);

fgets(lab, 30, F);

fclose(F);

F = fopen(lab, "r");

fseek(F, 0, SEEK\_SET);

int i, j, N;

fscanf(F, "%d", &N);

M = (int\*\*)malloc(N \* sizeof(int\*));

for (i = 0; i < N; i++)

M[i] = (int\*)malloc(N \* sizeof(int));

fseek(F, 12, SEEK\_SET);

for (i = 0; i < N; i++)

{

for (j = 0; j < N; j++)

{

fscanf(F, "%d", &M[i][j]);

fseek(F, 1, SEEK\_CUR);

}

fseek(F, 1, SEEK\_CUR);

}

n = sqrt(N);

}

fclose(F);

F = fopen(name, "r");

fseek(F, -1, SEEK\_END);

fscanf(F, "%f", &t);

fclose(F);

L = Lgen(M, n);

Q = put(M, n \* n, Q);

print\_L(L, n \* 2 + 1);

HZ(L, Q, n, M, t/2);

}

**Lab.cpp**

#include "Sys.h"

Node\* add(Node\* Q, int k, int l)

{

Node\* p;

p = (struct Node\*)malloc(sizeof(struct Node));

p->x = k;

p->y = l;

p->Next = Q;

Q = p;

return Q;

}

Node\* del(Node\* Q)

{

Node\* h;

h = Q;

Q = Q->Next;

free(h);

return Q;

}

void print\_L(char\*\* M, int n)

{

int i, j;

for (i = 0; i < n; i++)

{

printf("\n");

for (j = 0; j < n; j++)

{

printf("%\*C", 3, M[i][j]);

}

}

}

int\*\* gen(int n)

{

int i, j, k, z = 0, N = n\*n;

Node\* Q = NULL;

int\*\* M;

bool\* m;

m = (bool\*)malloc(N \* sizeof(bool));

for (i = 0; i < N; i++)

{

m[i] = false;

}

M = (int\*\*)malloc(N \* sizeof(int\*));

for (i = 0; i < N; i++)

M[i] = (int\*)malloc(N \* sizeof(int));

for (i = 0; i < N; i++)

{

for (j = 0; j < N; j++)

{

M[i][j] = 0;

}

}

i = 0;

j = 0;

m[0] = true;

Q = add(Q, i, j);

srand(time(NULL));

//srand(12);

while (Q != NULL)

{

k = 0;

if (i != n - 1)

if (m[j \* n + i + 1] == false) k++;

if (i != 0)

if (m[j \* n + i - 1] == false) k++;

if (j != n - 1)

if (m[(j + 1) \* n + i] == false) k++;

if (j != 0)

if (m[(j - 1) \* n + i] == false) k++;

if (k > 0)

{

k = rand() % 4;

while (z != 4)

{

switch (k)

{

case 0:

if (i != n - 1)

if (m[j \* n + i + 1] == false)

{

M[j \* n + i + 1][j \* n + i] = 1;

M[j \* n + i][j \* n + i + 1] = 1;

m[j \* n + i + 1] = true;

i++;

Q = add(Q, i, j);

k = -1;

}

break;

case 1:

if (i != 0)

if (m[j \* n + i - 1] == false)

{

M[j \* n + i - 1][j \* n + i] = 1;

M[j \* n + i][j \* n + i - 1] = 1;

m[j \* n + i - 1] = true;

i--;

Q = add(Q, i, j);

k = -1;

}

break;

case 2:

if (j != n - 1)

if (m[(j + 1) \* n + i] == false)

{

M[j \* n + i][(j + 1) \* n + i] = 1;

M[(j + 1) \* n + i][j \* n + i] = 1;

m[(j + 1) \* n + i] = 1;

j++;

Q = add(Q, i, j);

k = -1;

}

break;

case 3:

if (j != 0)

if (m[(j - 1) \* n + i] == false)

{

M[j \* n + i][(j - 1) \* n + i] = 1;

M[(j - 1) \* n + i][j \* n + i] = 1;

m[(j - 1) \* n + i] = 1;

j--;

Q = add(Q, i, j);

k = -1;

}

break;

}

if (k == -1) break;

if (k == 3) k = 0;

else k++;

z++;

}

z = 0;

}

else

{

if (Q->Next == NULL)

{

Q = del(Q);

break;

}

i = Q->Next->x;

j = Q->Next->y;

Q = del(Q);

}

}

free(m);

return M;

}

char\*\* Lgen(int\*\* M, int n)

{

int i, j, N=n\*2+1, x = 0, y = 0, s = 1;

Node\* Q = NULL;

char\*\* L;

bool\* m;

m = (bool\*)malloc(n\*n \* sizeof(bool));

for (i = 0; i < n\*n; i++)

{

m[i] = false;

}

L = (char\*\*)malloc(N \* sizeof(char\*));

for (i = 0; i < N; i++)

L[i] = (char\*)malloc(N \* sizeof(char));

for (i = 0; i < N; i++)

{

for (j = 0; j < N; j++)

{

L[i][j] = '0';

}

}

i = 1;

j = 1;

L[i][j] = ' ';

m[0] = true;

Q = add(Q, i, j);

while (s != n\*n)

{

while(true)

{

if (i != N - 2)

if (M[((j - 1) / 2) \* n + (i - 1) / 2 + 1][((j - 1) / 2) \* n + (i - 1) / 2] == 1 && m[((j - 1) / 2) \* n + (i - 1) / 2 + 1] == false)

{

m[((j - 1) / 2) \* n + (i - 1) / 2 + 1] = true;

L[j][i + 1] = ' ';

L[j][i + 2] = ' ';

i = i + 2;

Q = add(Q, i, j);

s++;

break;

}

if (i != 1)

if (M[((j - 1) / 2) \* n + (i - 1) / 2 - 1][((j - 1) / 2) \* n + (i - 1) / 2] == 1 && m[((j - 1) / 2) \* n + (i - 1) / 2 - 1] == false)

{

m[((j - 1) / 2) \* n + (i - 1) / 2 - 1] = true;

L[j][i - 1] = ' ';

L[j][i - 2] = ' ';

i = i - 2;

Q = add(Q, i, j);

s++;

break;

}

if (j != N - 2)

if (M[(((j - 1) / 2) + 1) \* n + (i - 1) / 2][((j - 1) / 2) \* n + (i - 1) / 2] == 1 && m[(((j - 1) / 2) + 1) \* n + (i - 1) / 2] == false)

{

m[(((j - 1) / 2) + 1) \* n + (i - 1) / 2] = true;

L[j + 1][i] = ' ';

L[j + 2][i] = ' ';

j = j + 2;

Q = add(Q, i, j);

s++;

break;

}

if (j != 1)

if (M[(((j - 1) / 2) - 1) \* n + (i - 1) / 2][((j - 1) / 2) \* n + (i - 1) / 2] == 1 && m[(((j - 1) / 2) - 1) \* n + (i - 1) / 2] == false)

{

m[(((j - 1) / 2) - 1) \* n + (i - 1) / 2] = true;

L[j - 1][i] = ' ';

L[j - 2][i] = ' ';

j = j - 2;

Q = add(Q, i, j);

s++;

break;

}

if (Q->Next == NULL)

{

Q = del(Q);

break;

}

i = Q->Next->x;

j = Q->Next->y;

Q = del(Q);

break;

}

}

free(m);

return L;

}

**Hod.cpp**

#include "Sys.h"

El\* add(El\* Q, int k)

{

El\* p;

p = (struct El\*)malloc(sizeof(struct El));

p->v = k;

p->Next = Q;

Q = p;

return Q;

}

El\* del(El\* Q)

{

El\* h;

h = Q;

Q = Q->Next;

free(h);

return Q;

}

Och\* plus(Och\* O, int k)

{

El\* p;

p = (struct El\*)malloc(sizeof(struct El));

p->v = k;

p->Next = NULL;

O->End->Next = p;

O->End = p;

return O;

}

Och\* min(Och\* O)

{

El\* h;

h = O->Head;

O->Head = O->Head->Next;

free(h);

return O;

}

El\* put(int\*\* M, int n, El\* Q)

{

Och\* O;

int i, j;

int\* m;

m = (int\*)malloc(n \* sizeof(int));

for (i = 0; i < n; i++)

{

m[i] = -1;

}

O = (struct Och\*)malloc(sizeof(struct Och));

O->Head = (struct El\*)malloc(sizeof(struct El));

O->End = O->Head;

plus(O, 0);

m[0] = 0;

while (O->End != O->Head)

{

j = O->Head->Next->v;

min(O);

for (i = 0; i < n; i++)

{

if (M[j][i] == 1 && m[i] == -1)

{

m[i] = m[j] + 1;

plus(O, i);

}

}

if (m[n - 1] != -1) break;

}

free(O->Head);

free(O);

int v = n-1;

Q = add(Q, v);

while (v != 0)

{

for (i = 0; i < n; i++)

{

if (M[i][v] == 1 && m[i] == m[v] - 1)

{

v = i;

Q = add(Q, v);

}

}

}

return Q;

}

void HZ(char\*\* L, El\* Q, int n, int\*\* M, float hod)

{

int i = 0, j = 0, x = 1, y = 1, N = n\*2+1, win = 0;

clock\_t S, E = 0;

char name[31];

FILE\* F;

Q = del(Q);

L[N - 2][N - 2] = '@';

while (Q != NULL)

{

system("cls");

L[y][x] = '#';

print\_L(L, n \* 2 + 1);

L[(Q->v / n)\*2+1][(Q->v % n)\*2+1] = '+';

if (L[j \* 2 + 1][i \* 2 + 1] != '#') L[j \* 2 + 1][i \* 2 + 1] = ' ';

j = Q->v / n;

i = Q->v % n;

Q = del(Q);

S = clock();

while (float(E-S)/CLK\_TCK < hod)

{

E = clock();

if (\_kbhit() == 1)

switch (\_getch())

{

case 72: if (L[y - 1][x] != '0')

{

if (L[y][x] != '+') L[y][x] = ' ';

y = y - 2;

L[y][x] = '#';

system("cls");

print\_L(L, n \* 2 + 1);

if (L[N - 2][N - 2] == '#')

{

win = 1;

break;

}

} break;

case 75: if (L[y][x - 1] != '0')

{

if (L[y][x] != '+') L[y][x] = ' ';

x = x - 2;

L[y][x] = '#';

system("cls");

print\_L(L, n \* 2 + 1);

if (L[N - 2][N - 2] == '#')

{

win = 1;

break;

}

} break;

case 80: if (L[y + 1][x] != '0')

{

if (L[y][x] != '+') L[y][x] = ' ';

y = y + 2;

L[y][x] = '#';

system("cls");

print\_L(L, n \* 2 + 1);

if (L[N - 2][N - 2] == '#')

{

win = 1;

break;

}

} break;

case 77: if (L[y][x + 1] != '0')

{

if (L[y][x] != '+') L[y][x] = ' ';

x = x + 2;

L[y][x] = '#';

system("cls");

print\_L(L, n \* 2 + 1);

if (L[N - 2][N - 2] == '#')

{

win = 1;

break;

}

} break;

case 49:

{

system("cls");

printf("Введите название лабиринта:");

scanf("%30s", name);

save(name, n, M);

}break;

}

}

if (win == 1)

{

break;

}

}

system("cls");

print\_L(L, n \* 2 + 1);

printf("\n\n");

if (win == 1) printf("Вы победили");

else printf("Вы не победили");

}

**Menu.cpp**

#include "Sys.h"

void menu(int i, char\* m)

{

system("cls");

m[i] = '>';

m[i + 1] = '<';

printf("\n\n\n\n");

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

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

printf(" %c | Играть | %c %c | Настройки | %c %c | Выход | %c \n", m[0], m[1], m[2], m[3], m[4], m[5]);

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

printf("\n\n\n\n\n\n\n");

printf("Стрелки - Перемещение; 1 - Выбор;");

m[i] = ' ';

m[i + 1] = ' ';

}

void set(int i, char\* m)

{

system("cls");

m[i] = '>';

m[i + 1] = '<';

printf("\n\n\n\n");

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

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

printf(" %c | Размер | %c %c | Лабиринт | %c %c | Сложность | %c %c | Сброс | %c\n", m[0], m[1], m[2], m[3], m[4], m[5], m[6], m[7]);

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

printf("\n\n\n\n\n\n\n");

printf("Стрелки - Перемещение; 1 - Выбор; 2 - Выход");

m[i] = ' ';

m[i + 1] = ' ';

}

void seti(int i, char\* m)

{

system("cls");

m[i] = '>';

m[i + 1] = '<';

printf("\n\n\n\n");

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

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

printf(" %c | Легко | %c %c | Нормально | %c %c | Тяжело | %c \n", m[0], m[1], m[2], m[3], m[4], m[5]);

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

printf("\n\n\n\n\n\n\n");

printf("Стрелки - Перемещение; 1 - Выбор; 2 - Выход");

m[i] = ' ';

m[i + 1] = ' ';

}

void raz()

{

int n;

FILE\* F;

char name[] = "Settings.txt";

system("cls");

printf("Введите размер лабиринта:");

scanf("%d", &n);

if ((F = fopen(name, "r+")) == NULL)

{

F = fopen(name, "w+");

stand();

}

fseek(F, 19, SEEK\_SET);

fprintf(F, "%d ", n);

fclose(F);

}

void lab()

{

int n;

FILE\* F;

char lab[31];

char name[] = "Settings.txt";

system("cls");

printf("Введите название лабиринта:");

scanf("%30s", lab);

if ((F = fopen(lab, "r+")) == NULL)

{

printf("Нет такого");

\_getch();

return;

}

fclose(F);

if ((F = fopen(name, "r+")) == NULL)

{

F = fopen(name, "w+");

stand();

}

fseek(F, 43, SEEK\_SET);

fputs(lab, F);

fclose(F);

}

void dvigat\_2();

void dvigat\_3()

{

char m[6] = { ' ',' ',' ',' ',' ',' ' };

int i = 0, k = 0, n;

seti(i, m);

while (k == 0)

{

switch (\_getch())

{

case 75: if (i != 0) i = i - 2; seti(i, m); break;

case 77: if (i != 4) i = i + 2; seti(i, m); break;

case 49: if (i == 0) sloj(3); if (i == 2) sloj(2); if (i == 4) sloj(1); break;

case 50: dvigat\_2();

}

}

}

void dvigat\_2()

{

char m[8] = { ' ',' ',' ',' ',' ',' ',' ',' ' };

int i = 0, k = 0, n;

set(i, m);

while (k == 0)

{

switch (\_getch())

{

case 75: if (i != 0) i = i - 2; set(i, m); break;

case 77: if (i != 6) i = i + 2; set(i, m); break;

case 49: if (i == 0) raz(); if (i == 2) lab(); if (i == 6) stand(); if (i == 4) dvigat\_3(); break;

case 50: dvigat\_1();

}

}

}

void dvigat\_1()

{

char m[6] = { ' ',' ',' ',' ',' ',' ' };

int i = 0, k = 0;

menu(i, m);

while (k == 0)

{

switch (\_getch())

{

case 75: if (i != 0) i = i - 2; menu(i, m); break;

case 77: if (i != 4) i = i + 2; menu(i, m); break;

case 49: if (i == 0) start(); if (i == 2) dvigat\_2(); if (i == 4) exit(0); break;

}

}

}

**File.cpp**

#include "Sys.h"

void stand()

{

FILE\* F;

char name[] = "Settings.txt";

F = fopen(name, "w+");

fprintf(F, "Размер лабиринта = %d \n", 5);

fprintf(F, "Лабиринт = - \n");

fprintf(F, "Сложность = %d", 1);

fclose(F);

}

void save(char\* lab, int N, int\*\* M)

{

FILE\* F;

int i, j, n = N\*N;

char name[] = "Settings.txt";

F = fopen(lab, "w+");

fprintf(F, " \n ");

fseek(F, 0, SEEK\_SET);

fprintf(F, "%d", n);

fseek(F, 12, SEEK\_SET);

for (i = 0; i < n; i++)

{

for (j = 0; j < n; j++)

{

fprintf(F, "%d ", M[i][j]);

}

fprintf(F, "\n");

}

fclose(F);

F = fopen(name, "r+");

fseek(F, 43, SEEK\_SET);

fputs(lab, F);

fclose(F);

}

void sloj(int k)

{

FILE\* F;

char name[] = "Settings.txt";

if ((F = fopen(name, "r")) == NULL)

{

stand();

}

F = fopen(name, "r+");

fseek(F, -1, SEEK\_END);

fprintf(F, "%d", k);

fclose(F);

}

**Sys.h**

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <conio.h>

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include <locale.h>

#include <math.h>

struct Node

{

int x, y;

Node\* Next;

};

struct El

{

int v;

El\* Next;

};

struct Och

{

El\* Head;

El\* End;

};

int\*\* gen(int n);

char\*\* Lgen(int\*\* M, int n);

void print\_L(char\*\* M, int n);

El\* put(int\*\* M, int n, El\* Q);

void HZ(char\*\* L, El\* Q, int n, int\*\* M, float hod);

void dvigat\_1();

void stand();

void start();

void save(char\* lab, int n, int\*\* M);

void sloj(int k);