Министерство образования и науки РФ

ФГАОУ ВПО «Уральский федеральный университет имени   
первого президента России Б.Н.Ельцина»  
Институт радиоэлектроники и информационных технологий – РТФ  
Департамент информационных технологий и автоматики

Оценка работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**АЛГОРИТМЫ РАЗМЕТКИ ЛАБИРИНТА**

Отчет по лабораторной работе №2  
по дисциплине «Методы исследования и моделирования  
информационных процессов и технологий»

Подпись Дата Ф.И.О.

Преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Добряк П.В.  
Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тимошенкова Ю.С.

Группа РИМ-161203

Екатеринбург

2017

**Цель работы:** Реализовать различные типы алгоритмов поиска выхода из лабиринта.

**Задание:**

1. Запрограммировать алгоритмы разметки лабиринта в сторону выхода:

1.1. Наивным методом;

1.2. Поиском в длину с бэктрекингом;

1.3. Поиском в длину рекурсией;

1.4. Поиском в ширину.

2. Написать алгоритм разметки лабиринта на секторы

3. Написать алгоритм выделения в лабиринте залов (минимальный размер зала – 2x2)

**СОДЕРЖАНИЕ**

[Задание 1 алгоритм разметки лабиринта в сторну выхода 3](#_Toc475914712)

[1.1 Наивный метод 3](#_Toc475914713)

[1.2 Поиск в длину с бэктрекингом 4](#_Toc475914714)

[1.3 Поиск в длину рекурсией 6](#_Toc475914715)

[1.4 Поиск в ширину 7](#_Toc475914716)

[Задание 2 алгоритм разметки лабиринта на секторы 9](#_Toc475914717)

[Задание 3 алгоритм выделения в лабиринте залов (минимальный размер зала – 2x2) 10](#_Toc475914718)

[ВЫВОДЫ 11](#_Toc475914719)

[ПРИЛОЖЕНИЕ A КОД ПРОГРАММЫ 12](#_Toc475914720)

# Задание 1 алгоритм разметки лабиринта в сторну выхода

## Наивный метод

Код реализации на языке C++

bool isCheck(int i, int j)

{

if (lab[i][j] == 9 || lab[i][j] == 0)

return 0;

else

return 1;

}

void naiveExit()

{

for (int i = 0; i < 7; i++)

if (lab[i][0] == 0)

lab[i][0] = 3;

for (int i = 0; i < 7; i++)

if (lab[i][7] == 0)

lab[i][7] = 1;

for (int j = 0; j < 8; j++)

if (lab[0][j] == 0)

lab[0][j] = 2;

for (int j = 0; j < 8; j++)

if (lab[6][j] == 0)

lab[6][j] = 4;

int flag = 0;

for (int c = 1; c < 6; c++)

{

if (flag == 1)

break;

else flag = 1;

for (int i = 1; i < 6; i++)

{

for (int j = 1; j < 7; j++)

{

if (isCheck(i, j - 1) && lab[i][j] == 0)

{

lab[i][j] = 3;

flag = 0;

}

else if (isCheck(i, j + 1) && lab[i][j] == 0)

{

lab[i][j] = 1;

flag = 0;

}

else if (isCheck(i - 1, j) && lab[i][j] == 0)

{

lab[i][j] = 2;

flag = 0;

}

else if (isCheck(i + 1, j) && lab[i][j] == 0)

{

lab[i][j] = 4;

flag = 0;

}

}

}

}

}

Результат работы алгоритма

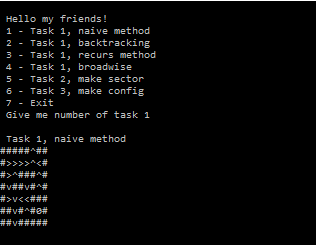


Рисунок 1 – Результат работы программы

## Поиск в длину с бэктрекингом

Код реализации на языке C++

void longway(int i, int j, int c, int z)

{

if (lab[i][j + 1] == 0 && j + 1 != 8) //cмотрим вправо

{

lab[i][j + 1] = 3;

longway(i, j + 1, c, z);

}

else if (lab[i - 1][j] == 0 && i - 1 != -1)//cмотрим вверх

{

lab[i - 1][j] = 4;

longway(i - 1, j, c, z);

}

else if (lab[i][j - 1] == 0 && j - 1 != -1)//cмотрим влево

{

lab[i][j - 1] = 1;

longway(i, j - 1, c, z);

}

else if (lab[i + 1][j] == 0 && i + 1 != 7)//cмотрим вниз

{

lab[i + 1][j] = 2;

longway(i + 1, j, c, z);

}

else if (lab[i][j] == 4 && !(i == c && j == z))

longway(i + 1, j, c, z);

else if (lab[i][j] == 3 && !(i == c && j == z))

longway(i, j - 1, c, z);

else if (lab[i][j] == 2 && !(i == c && j == z))

longway(i - 1, j, c, z);

else if (lab[i][j] == 1 && !(i == c && j == z))

longway(i, j + 1, c, z);

}

void backtracking()

{

int c, z;

int flag = 1;

while (flag == 1)

{

flag = 0;

for (int i = 0; i < 7; i++)

{

if (lab[i][0] == 0 && flag == 0)

{

lab[i][0] = 3;

flag = 1;

c = i;

z = 0;

}

if (lab[i][7] == 0 && flag == 0)

{

lab[i][7] = 1;

flag = 1;

c = i;

z = 7;

}

}

for (int j = 0; j < 8; j++)

{

if (lab[0][j] == 0 && flag == 0)

{

lab[0][j] = 2;

flag = 1;

c = 0;

z = j;

}

if (lab[6][j] == 0 && flag == 0)

{

lab[6][j] = 4;

flag = 1;

c = 6;

z = j;

}

}

longway(c, z, c, z);

}

}

Результат работы алгоритма

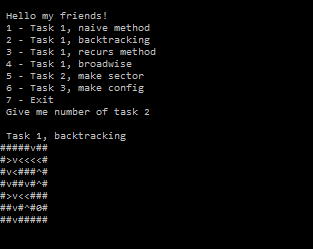


Рисунок 2 – Результат работы программы

## Поиск в длину рекурсией

Код реализации на языке C++

void recurs(int i, int j)

{

if (lab[i][j + 1] == 0 && j + 1 != 8) //cмотрим вправо

{

lab[i][j + 1] = 3;

recurs(i, j + 1);

}

if (lab[i - 1][j] == 0 && i - 1 != -1)//cмотрим вверх

{

lab[i - 1][j] = 4;

recurs(i - 1, j);

}

if (lab[i][j - 1] == 0 && j - 1 != -1)//cмотрим влево

{

lab[i][j - 1] = 1;

recurs(i, j - 1);

}

if (lab[i + 1][j] == 0 && i + 1 != 7)//cмотрим вниз

{

lab[i + 1][j] = 2;

recurs(i + 1, j);

}

}

void recursion()

{

int c, z;

int flag = 1;

while (flag == 1)

{

flag = 0;

for (int i = 0; i < 7; i++)

{

if (lab[i][0] == 0 && flag == 0)

{

lab[i][0] = 3;

flag = 1;

c = i;

z = 0;

}

if (lab[i][7] == 0 && flag == 0)

{

lab[i][7] = 1;

flag = 1;

c = i;

z = 7;

}

}

for (int j = 0; j < 8; j++)

{

if (lab[0][j] == 0 && flag == 0)

{

lab[0][j] = 2;

flag = 1;

c = 0;

z = j;

}

if (lab[6][j] == 0 && flag == 0)

{

lab[6][j] = 4;

flag = 1;

c = 6;

z = j;

}

}

recurs(c, z);

}

}

Результат работы алгоритма

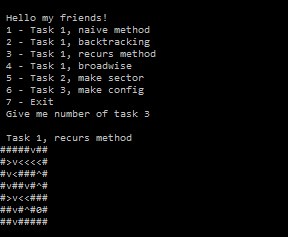


Рисунок 3 – Результат работы программы

## Поиск в ширину

Код реализации на языке C++

struct A

{

int array[2];

};

A mass;

list<A> myList;

void broadwise()

{

list<int> ilist, jlist;

for (int i = 0; i < 7; i++)

if (lab[i][0] == 0)

{

lab[i][0] = 3;

mass.array[0] = i;

mass.array[1] = 0;

myList.push\_back(mass);

}

for (int i = 0; i < 7; i++)

if (lab[i][7] == 0)

{

lab[i][7] = 1;

mass.array[0] = i;

mass.array[1] = 7;

myList.push\_back(mass);

}

for (int j = 0; j < 8; j++)

if (lab[0][j] == 0)

{

lab[0][j] = 2;

mass.array[0] = 0;

mass.array[1] = j;

myList.push\_back(mass);

}

for (int j = 0; j < 8; j++)

if (lab[6][j] == 0)

{

lab[6][j] = 4;

mass.array[0] = 6;

mass.array[1] = j;

myList.push\_back(mass);

}

for (auto c : myList)

{

if (lab[c.array[0]][c.array[1] + 1] == 0 && c.array[1] + 1 != 8) //cмотрим вправо

{

lab[c.array[0]][c.array[1] + 1] = 3;

mass.array[0] = c.array[0];

mass.array[1] = c.array[1] + 1;

myList.push\_back(mass);

}

if (lab[c.array[0] - 1][c.array[1]] == 0 && c.array[0] - 1 != -1)//cмотрим вверх

{

lab[c.array[0] - 1][c.array[1]] = 4;

mass.array[0] = c.array[0] - 1;

mass.array[1] = c.array[1];

myList.push\_back(mass);

}

if (lab[c.array[0]][c.array[1] - 1] == 0 && c.array[1] - 1 != -1)//cмотрим влево

{

lab[c.array[0]][c.array[1] - 1] = 1;

mass.array[0] = c.array[0];

mass.array[1] = c.array[1] - 1;

myList.push\_back(mass);

}

if (lab[c.array[0] + 1][c.array[1]] == 0 && c.array[0] + 1 != 7)//cмотрим вниз

{

lab[c.array[0] + 1][c.array[1]] = 2;

mass.array[0] = c.array[0] + 1;

mass.array[1] = c.array[1];

myList.push\_back(mass);

}

}

}

Результат работы алгоритма

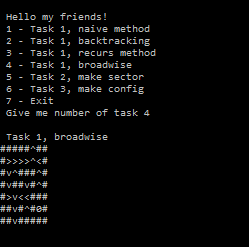


Рисунок 4 – Результат работы программы

# Задание 2 алгоритм разметки лабиринта на секторы

Код реализации на языке C++

void swap(int c)

{

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 8; j++)

{

if (lab[i][j] == 1 || lab[i][j] == 2 || lab[i][j] == 3 || lab[i][j] == 4)

lab[i][j] = c;

}

}

}

void sectorize()

{

int c = 10;

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 8; j++)

{

if (lab[i][j] != 8)

recurs(i, j);

if (lab[i][j] == 1 || lab[i][j] == 2 || lab[i][j] == 3 || lab[i][j] == 4)

{

swap(c);

c++;

}

}

}

}

Результат работы алгоритма

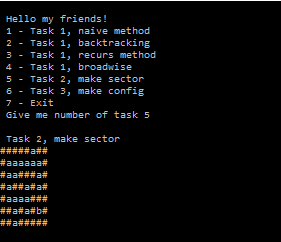


Рисунок 6 – Результат работы программы

# Задание 3 алгоритм выделения в лабиринте залов (минимальный размер зала – 2x2)

Код реализации на языке C++

void config()

{

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 8; j++)

{

if ((lab[i][j] == 0 || lab[i][j] == 1) && (lab[i + 1][j] == 0 || lab[i + 1][j] == 1) && (lab[i][j + 1] == 0 || lab[i][j + 1] == 1) && (lab[i + 1][j + 1] == 0 || lab[i + 1][j + 1] == 1))

{

lab[i][j] = 1;

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

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

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

}

}

}

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 8; j++)

{

if (lab[i][j] == 0)

lab[i][j] = 8;

if (lab[i][j] == 1)

lab[i][j] = 0;

}

}

sectorize();

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 8; j++)

if (lab[i][j] == 8)

lab[i][j] = 0;

}

}

Результат работы алгоритма

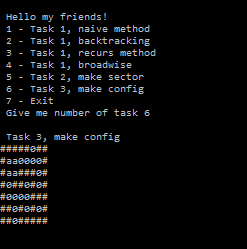


Рисунок 7 – Результат работы программы

# ВЫВОДЫ

В ходе выполнения лабораторной работы были реализованы различные методы поиска выхода из лабиринта:

1.1. Наивный метод;

1.2. Поиск в длину с бэктрекингом;

1.3. Поиск в длину рекурсией;

1.4. Поиск в ширину.

Входе выполнения заданий были реализованы алгоритм разметки лабиринта на секторы и алгоритм выделения в лабиринте зала размером 2 на 2.

При выполнении работы было написано полностью работоспособное приложение, код представлен в приложении. Результаты работы приложения представлены в соответствующих разделах отчета.

# ПРИЛОЖЕНИЕ A КОД ПРОГРАММЫ

// MIiMIPiT\_lab2.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <iostream>

#include <list>

#include <vector>

#include <Array>

using namespace std;

int lab[7][8] = { { 9,9,9,9,9,0,9,9 },

{ 9,0,0,0,0,0,0,9 },

{ 9,0,0,9,9,9,0,9 },

{ 9,0,9,9,0,9,0,9 },

{ 9,0,0,0,0,9,9,9 },

{ 9,9,0,9,0,9,0,9 },

{ 9,9,0,9,9,9,9,9 } };

void ShowExit()

{

for (int i = 0; i<7; i++)

{

for (int j = 0; j<8; j++)

{

if (lab[i][j] == 0)

cout << '0';

else if (lab[i][j] == 9)

cout << '#';

else if (lab[i][j] == 1)

cout << '>';

else if (lab[i][j] == 2)

cout << '^';

else if (lab[i][j] == 3)

cout << '<';

else if (lab[i][j] == 4)

cout << 'v';

else if (lab[i][j] == 10)

cout << 'a';

else if (lab[i][j] == 11)

cout << 'b';

else if (lab[i][j] == 12)

cout << 'c';

else if (lab[i][j] == 13)

cout << 'd';

else if (lab[i][j] == 8)

cout << '8';

}

cout << endl;

}

cout << endl;

}

//-----------------------------------Наивный

bool isCheck(int i, int j)

{

if (lab[i][j] == 9 || lab[i][j] == 0)

return 0;

else

return 1;

}

void naiveExit()

{

for (int i = 0; i < 7; i++)

if (lab[i][0] == 0)

lab[i][0] = 3;

for (int i = 0; i < 7; i++)

if (lab[i][7] == 0)

lab[i][7] = 1;

for (int j = 0; j < 8; j++)

if (lab[0][j] == 0)

lab[0][j] = 2;

for (int j = 0; j < 8; j++)

if (lab[6][j] == 0)

lab[6][j] = 4;

int flag = 0;

for (int c = 1; c < 6; c++)

{

if (flag == 1)

break;

else flag = 1;

for (int i = 1; i < 6; i++)

{

for (int j = 1; j < 7; j++)

{

if (isCheck(i, j - 1) && lab[i][j] == 0)

{

lab[i][j] = 3;

flag = 0;

}

else if (isCheck(i, j + 1) && lab[i][j] == 0)

{

lab[i][j] = 1;

flag = 0;

}

else if (isCheck(i - 1, j) && lab[i][j] == 0)

{

lab[i][j] = 2;

flag = 0;

}

else if (isCheck(i + 1, j) && lab[i][j] == 0)

{

lab[i][j] = 4;

flag = 0;

}

}

}

}

}

//-----------------------------------В глубину с бэктрэкингом

void longway(int i, int j, int c, int z)

{

if (lab[i][j + 1] == 0 && j + 1 != 8) //cмотрим вправо

{

lab[i][j + 1] = 3;

longway(i, j + 1, c, z);

}

else if (lab[i - 1][j] == 0 && i - 1 != -1)//cмотрим вверх

{

lab[i - 1][j] = 4;

longway(i - 1, j, c, z);

}

else if (lab[i][j - 1] == 0 && j - 1 != -1)//cмотрим влево

{

lab[i][j - 1] = 1;

longway(i, j - 1, c, z);

}

else if (lab[i + 1][j] == 0 && i + 1 != 7)//cмотрим вниз

{

lab[i + 1][j] = 2;

longway(i + 1, j, c, z);

}

else if (lab[i][j] == 4 && !(i == c && j == z))

longway(i + 1, j, c, z);

else if (lab[i][j] == 3 && !(i == c && j == z))

longway(i, j - 1, c, z);

else if (lab[i][j] == 2 && !(i == c && j == z))

longway(i - 1, j, c, z);

else if (lab[i][j] == 1 && !(i == c && j == z))

longway(i, j + 1, c, z);

}

void backtracking()

{

int c, z;

int flag = 1;

while (flag == 1)

{

flag = 0;

for (int i = 0; i < 7; i++)

{

if (lab[i][0] == 0 && flag == 0)

{

lab[i][0] = 3;

flag = 1;

c = i;

z = 0;

}

if (lab[i][7] == 0 && flag == 0)

{

lab[i][7] = 1;

flag = 1;

c = i;

z = 7;

}

}

for (int j = 0; j < 8; j++)

{

if (lab[0][j] == 0 && flag == 0)

{

lab[0][j] = 2;

flag = 1;

c = 0;

z = j;

}

if (lab[6][j] == 0 && flag == 0)

{

lab[6][j] = 4;

flag = 1;

c = 6;

z = j;

}

}

longway(c, z, c, z);

}

}

//-----------------------------------В глубину с рекурсией

void recurs(int i, int j)

{

if (lab[i][j + 1] == 0 && j + 1 != 8) //cмотрим вправо

{

lab[i][j + 1] = 3;

recurs(i, j + 1);

}

if (lab[i - 1][j] == 0 && i - 1 != -1)//cмотрим вверх

{

lab[i - 1][j] = 4;

recurs(i - 1, j);

}

if (lab[i][j - 1] == 0 && j - 1 != -1)//cмотрим влево

{

lab[i][j - 1] = 1;

recurs(i, j - 1);

}

if (lab[i + 1][j] == 0 && i + 1 != 7)//cмотрим вниз

{

lab[i + 1][j] = 2;

recurs(i + 1, j);

}

}

void recursion()

{

int c, z;

int flag = 1;

while (flag == 1)

{

flag = 0;

for (int i = 0; i < 7; i++)

{

if (lab[i][0] == 0 && flag == 0)

{

lab[i][0] = 3;

flag = 1;

c = i;

z = 0;

}

if (lab[i][7] == 0 && flag == 0)

{

lab[i][7] = 1;

flag = 1;

c = i;

z = 7;

}

}

for (int j = 0; j < 8; j++)

{

if (lab[0][j] == 0 && flag == 0)

{

lab[0][j] = 2;

flag = 1;

c = 0;

z = j;

}

if (lab[6][j] == 0 && flag == 0)

{

lab[6][j] = 4;

flag = 1;

c = 6;

z = j;

}

}

recurs(c, z);

}

}

//-----------------------------------В ширину

struct A

{

int array[2];

};

A mass;

list<A> myList;

void broadwise()

{

list<int> ilist, jlist;

for (int i = 0; i < 7; i++)

if (lab[i][0] == 0)

{

lab[i][0] = 3;

mass.array[0] = i;

mass.array[1] = 0;

myList.push\_back(mass);

}

for (int i = 0; i < 7; i++)

if (lab[i][7] == 0)

{

lab[i][7] = 1;

mass.array[0] = i;

mass.array[1] = 7;

myList.push\_back(mass);

}

for (int j = 0; j < 8; j++)

if (lab[0][j] == 0)

{

lab[0][j] = 2;

mass.array[0] = 0;

mass.array[1] = j;

myList.push\_back(mass);

}

for (int j = 0; j < 8; j++)

if (lab[6][j] == 0)

{

lab[6][j] = 4;

mass.array[0] = 6;

mass.array[1] = j;

myList.push\_back(mass);

}

for (auto c : myList)

{

if (lab[c.array[0]][c.array[1] + 1] == 0 && c.array[1] + 1 != 8) //cмотрим вправо

{

lab[c.array[0]][c.array[1] + 1] = 3;

mass.array[0] = c.array[0];

mass.array[1] = c.array[1] + 1;

myList.push\_back(mass);

}

if (lab[c.array[0] - 1][c.array[1]] == 0 && c.array[0] - 1 != -1)//cмотрим вверх

{

lab[c.array[0] - 1][c.array[1]] = 4;

mass.array[0] = c.array[0] - 1;

mass.array[1] = c.array[1];

myList.push\_back(mass);

}

if (lab[c.array[0]][c.array[1] - 1] == 0 && c.array[1] - 1 != -1)//cмотрим влево

{

lab[c.array[0]][c.array[1] - 1] = 1;

mass.array[0] = c.array[0];

mass.array[1] = c.array[1] - 1;

myList.push\_back(mass);

}

if (lab[c.array[0] + 1][c.array[1]] == 0 && c.array[0] + 1 != 7)//cмотрим вниз

{

lab[c.array[0] + 1][c.array[1]] = 2;

mass.array[0] = c.array[0] + 1;

mass.array[1] = c.array[1];

myList.push\_back(mass);

}

}

}

//-----------------------------------Поиск области достижимости

void swap(int c)

{

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 8; j++)

{

if (lab[i][j] == 1 || lab[i][j] == 2 || lab[i][j] == 3 || lab[i][j] == 4)

lab[i][j] = c;

}

}

}

void sectorize()

{

int c = 10;

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 8; j++)

{

if (lab[i][j] != 8)

recurs(i, j);

if (lab[i][j] == 1 || lab[i][j] == 2 || lab[i][j] == 3 || lab[i][j] == 4)

{

swap(c);

c++;

}

}

}

}

//-----------------------------------Поиск нужной конфигурации

void config()

{

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 8; j++)

{

if ((lab[i][j] == 0 || lab[i][j] == 1) && (lab[i + 1][j] == 0 || lab[i + 1][j] == 1) && (lab[i][j + 1] == 0 || lab[i][j + 1] == 1) && (lab[i + 1][j + 1] == 0 || lab[i + 1][j + 1] == 1))

{

lab[i][j] = 1;

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

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

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

}

}

}

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 8; j++)

{

if (lab[i][j] == 0)

lab[i][j] = 8;

if (lab[i][j] == 1)

lab[i][j] = 0;

}

}

sectorize();

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 8; j++)

if (lab[i][j] == 8)

lab[i][j] = 0;

}

}

void main()

{

int n;

cout << "\n" << " Hello my friends! " << "\n" << " 1 - Task 1, naive method " << "\n" << " 2 - Task 1, backtracking ";

cout << "\n" << " 3 - Task 1, recurs method " << "\n" << " 4 - Task 1, broadwise " << "\n" << " 5 - Task 2, make sector ";

cout << "\n" << " 6 - Task 3, make config " << "\n" << " 7 - Exit ";

cout << "\n" << " Give me number of task ";

cin >> n;

if (n == 1)

{

cout << "\n" << " Task 1, naive method " << "\n";

naiveExit();

ShowExit();

main();

}

else if (n == 2)

{

cout << "\n" << " Task 1, backtracking " << "\n";

backtracking();

ShowExit();

main();

}

else if (n == 3)

{

cout << "\n" << " Task 1, recurs method " << "\n";

recursion();

ShowExit();

main();

}

else if (n == 4)

{

cout << "\n" << " Task 1, broadwise " << "\n";

broadwise();

ShowExit();

main();

}

else if (n == 5)

{

cout << "\n" << " Task 2, make sector " << "\n";

sectorize();

ShowExit();

main();

}

else if (n == 6)

{

cout << "\n" << " Task 3, make config" << "\n";

config();

ShowExit();

main();

}

else if (n == 7)

{

exit(1);

}

system("pause");

}