Лабораторная работа № 4. Рекурсия и головоломки.

(наименование темы проекта или работы)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

ОТЧЕТ по лабораторной работе

(курсовому проекту, ОТЧЕТ по лабораторной работе)

по дисциплине

**Технологии Программирования**

(наименование дисциплины)

РУКОВОДИТЕЛЬ:

**\_\_\_\_\_\_\_\_\_** Капранов С.Н

(подпись) (ФИО)

СТУДЕНТ:

**\_\_\_\_\_\_\_\_\_**Полежаева А.С

(подпись) (ФИО)

18-ИСТ-3

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

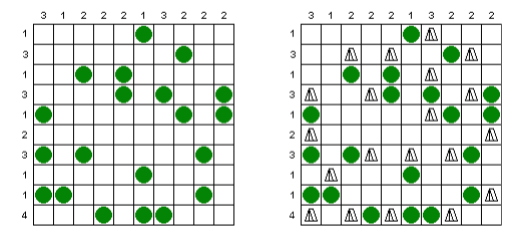
С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020

Задание (вариант 14):

Лагерь ("Tents", "TentsandTrees") представляет собой прямоугольную сетку, некоторые клетки которой содержат "деревья". Необходимо разместить рядом с деревьями "палатки", соблюдая следующие правила:

* Число палаток равняется числу деревьев.
* Каждая палатка располагается рядом со "своим" деревом по горизонтали или вертикали, но не по диагонали. Если это условие выполнено, расположение по отношению к "чужим" деревьям значения не имеет.
* Две палатки не могут располагаться в соседних клетках, в том числе и по диагонали.
* Числа сбоку и сверху означают, сколько палаток находится в этой строке или столбце.



Ходработы:

#include<iostream>

#include<iomanip>

intmain()

{

srand(time(0));

int H, W, tree, count;

bool flag = true;

std::cout<<"Enter Height: "; std::cin>> H;//вводвысоты

std::cout<<"Enter Width: "; std::cin>> W;//вводширины

std::cout<<"Enter trees quantity: "; std::cin>> tree;//вводдеревьев

std::cout<<"\n\n";

char\*\* M = newchar \* [H]; //создание 2-мерного массива чар

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

M[i] = newchar[W];

count = H \* W;

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

{

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

{

M[i][j] = '.';

}

}

int\* numOfTree = newint(1);

while (count >= 0)

{

int i\_index, j\_index;

std::cout <<"Tree number "<< \*numOfTree <<"\n\n";

std::cout <<"Enter the X coordinate for the tree ";

std::cin >> j\_index;

std::cin.ignore(1024, '\n');

std::cout <<"Enter the Y coordinate for the tree ";

std::cin >> i\_index;

std::cin.ignore(1024, '\n');

std::cout <<"\n";

flag = false;

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

{

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

{

if (i == i\_index && j == j\_index)

{

if (M[i][j] != 'O')

{

flag = true;

M[i][j] = 'O';

tree--;

\*numOfTree += 1;

break;

}

else

{

std::cout <<"There is already a tree at this place"<< std::endl;

count++;

break;

}

}

}

}

if (tree == 0) {

delete numOfTree;

break;

}

count--;

}

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

{

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

{

if (M[i][j] == 'O')

{

count = 0;

if (i - 1 >= 0)

{

if (i - 2 >= 0)

{

if (M[i - 2][j] == '#')

count++;

if (j - 1 >= 0)

{

if (M[i - 2][j - 1] == '#')

count++;

}

if (j + 1 < W)

{

if (M[i - 2][j + 1] == '#')

count++;

}

}

if (j + 1 < W)

{

if (M[i - 1][j + 1] == '#' || M[i][j + 1] == '#')

count++;

}

if (j - 1 >= 0)

{

if (M[i - 1][j - 1] == '#' || M[i][j - 1] == '#')

count++;

}

if (count == 0)

{

M[i - 1][j] = '#';

j++;

if (j == W)

break;

}

}

}

if (M[i][j] == 'O')

{

count = 0;

if (j + 1 < W)

{

if (j + 2 < W)

{

if (M[i][j + 2] == '#')

count++;

if (i - 1 >= 0)

{

if (M[i - 1][j + 2] == '#')

count++;

}

if (i + 1 < H)

{

if (M[i + 1][j + 2] == '#')

count++;

}

}

if (i - 1 >= 0)

{

if (M[i - 1][j + 1] == '#' || M[i - 1][j] == '#')

count++;

}

if (i + 1 < H)

{

if (M[i + 1][j + 1] == '#' || M[i + 1][j] == '#')

count++;

}

if (count == 0)

{

M[i][j + 1] = '#';

j++;

if (j == W)

break;

}

}

}

if (M[i][j] == 'O')

{

count = 0;

if (i + 1 < H)

{

if (i + 2 < H)

{

if (M[i + 2][j] == '#')

count++;

if (j - 1 >= 0)

{

if (M[i + 2][j - 1] == '#')

count++;

}

if (j + 1 < W)

{

if (M[i + 2][j + 1] == '#')

count++;

}

}

if (j - 1 >= 0)

{

if (M[i + 1][j - 1] == '#' || M[i][j - 1] == '#')

count++;

}

if (j + 1 < W)

{

if (M[i + 1][j + 1] == '#' || M[i][j + 1] == '#')

count++;

}

if (count == 0)

{

M[i + 1][j] = '#';

j++;

if (j == W)

break;

}

}

}

if (M[i][j] == 'O')

{

count = 0;

if (j - 1 >= 0)

{

if (j - 2 >= 0)

{

if (M[i][j - 2] == '#')

count++;

if (i - 1 >= 0)

{

if (M[i - 1][j - 2] == '#')

count++;

}

if (i + 1 < H)

{

if (M[i + 1][j - 2] == '#')

count++;

}

}

if (i - 1 >= 0)

{

if (M[i - 1][j - 1] == '#' || M[i - 1][j] == '#')

count++;

}

if (i + 1 < H)

{

if (M[i + 1][j - 1] == '#' || M[i + 1][j] == '#')

count++;

}

if (count == 0)

{

M[i][j - 1] = '#';

j++;

if (j == W)

break;

}

}

}

}

}

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

{

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

{

if (M[j][i] == '#')

count++;

}

std::cout << std::setw(4) << count;

count = 0;

}

std::cout <<"\n";

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

std::cout << std::setw(4) <<"====";

std::cout <<"\n";

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

{

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

{

std::cout << std::setw(4) << M[i][j];

}

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

{

if (M[i][j] == '#')

count++;

}

std::cout <<" -- "<< count;

count = 0;

std::cout <<"\n\n";

}

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

delete[] M[i];

delete[] M;

system("pause");

return 0;

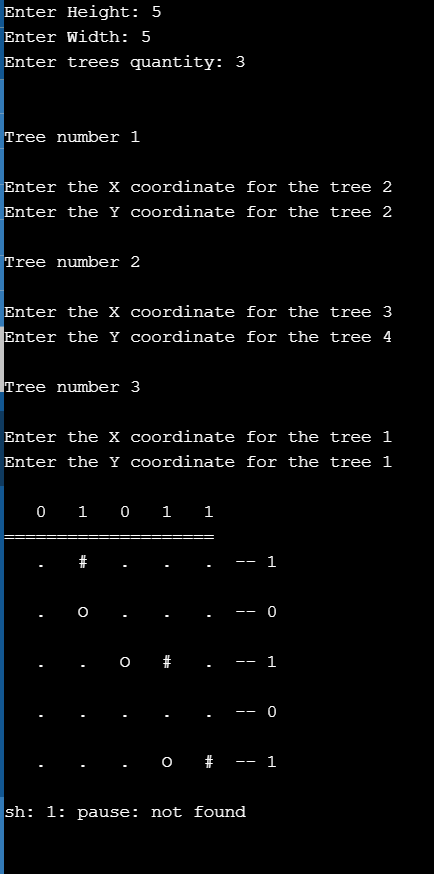
}

Основные функции:

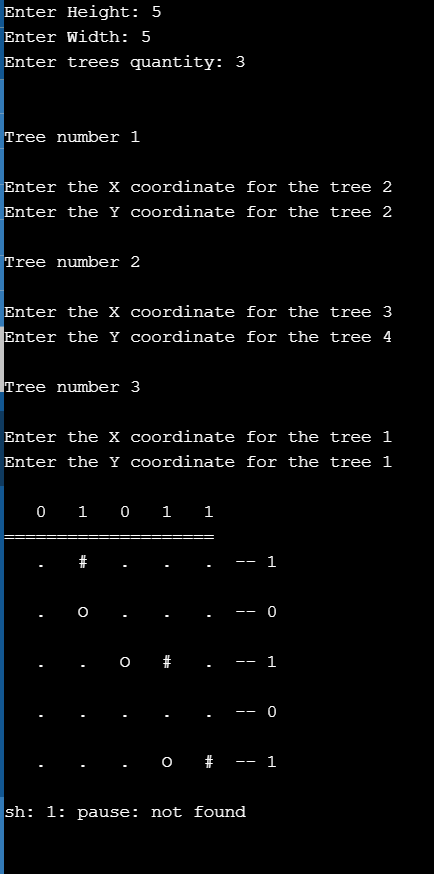
Setw – устанавливает ширину вывода, чтобы все вывелось аккуратно

Принтскрины экранов:

1)Вводим высоту и ширину игрового поля, а также количество деревьев.



2)Вводим координаты наших деревьев:



3)Результат:

