**РУСЕНСКИ УНИВЕРСИТЕТ “АНГЕЛ КЪНЧЕВ”**

**КУРСОВА РАБОТА**

**№1**

**по Изкуствен интелект**

Студент:

Факултетен номер:

Група:

Специалност:

Курс:

**Изготвил:**

**Дата: Проверил:**

# Задание.

„Ход на коня“ (Knight’s tour problem)

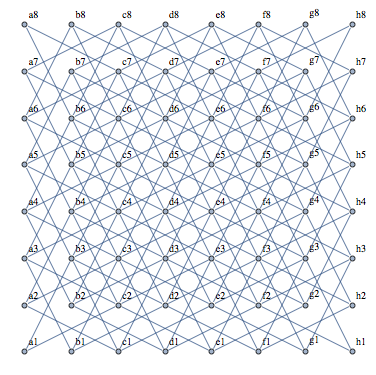
Да се открие последователност от премествания на кон върху шахматна дъска до зададена позиция от нея.

Алгоритъм: ограничен в дълбочина backtracking [*Backtracking with depth bound*].

# Представяне на задачата като задача за търсене в ПС.

* **Състояние** – Нека състоянието на тази задача да се представи чрез наредена двойка (α1, α2), където α1 и α2 са координатите (позицията) на „коня“.
* **Начално състояние** – задава се от клавиатурата.
* **Финално състояние** – задава се от клавиатурата.

# Граф на ПС.



Фигура 1 Граф на ПС при 8х8 шахматна дъска

Източник: <http://rosettacode.org/mw/images/1/12/KnightsTour-3.png>

# Програмна реализация.

## Псевдо-код на алгоритъма.

/\* board[][] е дъската с размери nn \*/

void следващХод(x, y, i)

{

if (текущата\_позиция == GOAL) { /\* Печатаме решението и прекратяваме търсенето \*/}

if (i >= max\_depth) { return; }

board[x][y] = i;

for (всеки\_възможен\_ход\_(u,v)\_на\_коня\_от\_клетка\_(x,y))

if (0 == board[u][v]) /\* ако полето е свободно \*/

path.Add(позицията);

следващХод(u, v, i+1);

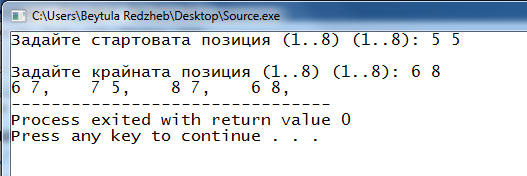
board[x][y] = 0; /\* връщане \*/

}

## Описание на разработениете модули.

* **Вид на функцията**: void nextMove (unsigned X, unsigned Y, unsigned i)
* **Действие**: Функцията служи за намиране на решението на задачата „Ход на коня“, чрез *Backtracking with depth bound* алгоритъма;
* **Параметри**: X, Y – текущата позиция на „коня“; i – текущото ниво;
* **Използвани функции**: рекурсия
* **Вид на функцията**: int main(void)
* **Действие**: Главна функция
* **Параметри**: няма;
* **Използвани функции**: nextMove ().

## Тестови примери.



## Код на програмата.

**#include <stdio.h> // printf(); scanf();**

**#include <stdlib.h> // exit();**

**#include <locale.h> // setlocale(); - кирилица**

**#include <vector> // vector<Node> vertex;**

**using namespace std;**

**/\* Максимален размер на дъската \*/**

**#define MAXN 10**

**/\* Максимален брой правила за движение на коня \*/**

**#define MAXD 10**

**/\* Максимална дълбочина \*/**

**#define MAX\_DEPTH 30**

**/\* Размер на дъската \*/**

**const unsigned n = 8;**

**/\* Стартова позиция \*/**

**unsigned startX;**

**unsigned startY;**

**/\* Крайна позиция \*/**

**unsigned endX;**

**unsigned endY;**

**/\* Правила за движение на коня \*/**

**const unsigned maxDiff = 8;**

**const int diffX[MAXD] = { 1, 1, -1, -1, 2, -2, 2, -2 };**

**const int diffY[MAXD] = { 2, -2, 2, -2, 1, 1, -1, -1 };**

**unsigned board[MAXN][MAXN];**

**unsigned newX, newY;**

**struct Node {**

**int x;**

**int y;**

**};**

**vector<Node> vertex;**

**void nextMove(unsigned X, unsigned Y, unsigned i)**

**{**

**unsigned k;**

**board[X][Y] = i;**

**if (X == endX && Y == endY) {**

**for (int v = 0; v != vertex.size(); ++v) {**

**printf("%d %d,\t", vertex[v].x + 1, vertex[v].y + 1);**

**}**

**exit(0);**

**}**

**if (i >= MAX\_DEPTH) { return; }**

**for (k = 0; k < maxDiff; k++) {**

**newX = X + diffX[k]; newY = Y + diffY[k];**

**if ((newX >= 0 && newX < n && newY >= 0 && newY < n) &&**

**(0 == board[newX][newY])) {**

**Node n;**

**n.x = newX;**

**n.y = newY;**

**vertex.push\_back(n);**

**nextMove(newX, newY, i + 1);**

**}**

**}**

**board[X][Y] = 0;**

**}**

**int main(void)**

**{**

**setlocale(LC\_ALL, "BGR");**

**unsigned i, j;**

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

**for (j = 0; j < n; j++) board[i][j] = 0;**

**printf("Задайте стартовата позиция (1..%d) (1..%d): ", n, n);**

**scanf("%d %d", &startX, &startY);**

**printf("\nЗадайте крайната позиция (1..%d) (1..%d): ", n, n);**

**scanf("%d %d", &endX, &endY);**

**endX--;**

**endY--;**

**nextMove(startX - 1, startY - 1, 1);**

**printf("Безуспешен опит! \n");**

**return 0;**

**}**

# Творческа задача.

Достатъчно е да се премахне оператора за прекратяване на търсенето, когато решението е намерено – exit(0).