**Ткаченко Кирилл Станиславович**
*Инженер 1-й категории, аспирант, ассистент Севастопольского государственного университета.*

# Интересное решение задачи о восьми ферзях

Рассматривается итеративное решение задачи о восьми ферзях, приводятся исходные тексты программ.

Одной из известнейших задач, использующихся при обучении информатике и программированию, является задача о восьми ферзях. Эта задача рассмотрена во многих учебниках и монографиях по программированию, в том числе и в [1–4], различных интернет-ресурсах. При этом видно, что использование различных подходов к составлению алгоритма её решения и последующего программирования накладывает свой отпечаток на результирующий программный код и эффективность функционирования.

Одним из интересных решений, на взгляд автора, является приведённое в учебнике по программированию на языке программирования высокого уровня Фортран [4]. Особенностью этого решения является следующее:

- используются из структур данных и структур представления данных только массивы;
- тип всех переменных – целочисленный;
- программа и алгоритм реализованы в процедурном стиле, без использования рекурсии;
- уникальный в своём роде способ построения перестановок с использованием нескольких вложенных циклов.

Поскольку во многих современных источниках в открытой печати для решения задач комбинаторики используются не столь ограниченные выразительные средства, любые такие решения могут иметь некоторый интерес как для подготовки программистов, так и для использования для самотренировки.

Начинать необходимо с постановки задачи. В наиболее простой и ясной формулировке это звучит следующим образом:

Расставить восемь ферзей на шахматной доске 8 на 8, при этом ни один из ферзей не должен бить любого другого.

Речь идёт о том, что на доске должно находиться ровно восемь

ферзей, на каждой горизонтали, вертикали и диагонали должен находиться ровно один ферзь (речь идёт о любых диагоналях доски).

Необходимо разработать алгоритм и компьютерную программу, с помощью которой возможно найти все решения, удовлетворяющие поставленной задаче.

Полным перебором эту задачу можно решить, получив 8! (читатель, я уверен, легко найдёт это число) расстановок ферзей (очевидно, что каждый ферзь находится только на одной горизонтали или вертикали), и оставить из них корректные. Но гораздо более эффективным в практическом плане является схема перебора с возвратом, когда последовательно, в порядке возрастания номеров горизонталей и вертикалей происходит заполнение шахматной доски. Наверное, на ум сразу приходит рекурсивное решение, но такое решение не является достаточно хорошим в виду существенных расходов на оперативную память и стек. Попытки переписать его в итеративной форме приводят к необходимости использования либо достаточно сложных структур данных для сохранения состояния вычислительного процесса, либо для хранения соотношений на шахматной доске. Зато при реализации подхода [4] используется только один-единственный массив целочисленных компонентов и несколько целочисленных переменных.

Пусть на каждой вертикали может находиться только один ферзь. Это означает, что для описания состояния шахматной доски необходим массив из 8 элементов (назовём его D), причём относительное положение ферзя в вертикальном ряду с номером $i$ будет храниться в элементе массива d[$i$].

Будут последовательно рассматриваться вертикали с первой по восьмую. Устанавливается первая вертикаль. Затем начинается процесс решения для получения всех возможных корректных результатов.

– У ферзя, который находится на первой вертикали, устанавливается первая горизонталь.

– После начинается цикл для формирования одного, текущего решения.

– – Полагается, что ни один ферзь не находится под атакой со стороны любого другого ферзя.

– – После этого, если рассматриваемый ферзь на $i$-й вертикали не находится на первой вертикали:

– – – Проверяется в цикле, что ферзь на $i$-й вертикали не бьёт ферзей, находящихся на вертикалях с первой по ($i$-1)-ю, то есть $j$ изменяется от 1 до $i$-1 и рассматриваются ферзи d[1], ..., d[$i$-1], d[$i$].

Два ферзя не бьют друг друга, если:

* Ферзи не находятся на одной вертикали (достигается использованием структуры данных «массив»);

* Ферзи не находятся на одной горизонтали (бьют, если d[$i$] равно d[$j$]);

* Ферзи находятся на одной диагонали с углом наклона 45 градусов и одной диагонали с углом наклона – 45 градусов (бьют, если $|d[i]-d[j]|$ равно $|i-j|$).

Если бьющиеся ферзи найдены, то это отмечается и прерывается цикл.

– – Иначе, если d[$i$] $\geq 5$, то получены все решения, в которых ферзь располагается с первой по четвёртую горизонтали. С помощью симметрии можно достаточно просто получить другие, в данном случае избыточные, решения.

– – Если в результате предыдущих шагов выяснено, что ферзь d[$i$] не бьёт всех других ферзей:

– – – Продвигается вертикаль $i$.

– – – Если не рассмотрены ($i \leq 8$)

все вертикали, то цикл прерывается и они рассматриваются.

– – – Выводится найденное решение простым и очевидным отображением одномерного массива d.

– – – Устанавливается восьмая вертикаль.

– – Текущее значение $i$ уменьшается до тех пор, пока не выполнится $d[i] < 8$. Тогда ферзь на $i$-й вертикали продвигается с увеличением $d[i]$.

– Конец цикла

Конец цикла

Данное решение реализовано на языках программирования высокого уровня Си (листинг 1), Java (листинг 2), Python (листинг 3), результаты вычислений приводятся в листинге 4.

Листинг 1 – Программный код на языке Си

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int d[8], i = 1, j, k, underAttack;

    while (1) {
        d[i - 1] = 1;
        while (1) {
            underAttack = 0;
            if (i != 1) {
                for (j = 1; j < i; ++j) {
                    if (d[i - 1] == d[j - 1]
                            || abs(d[i - 1] - d[j - 1]) == abs(i
- j)) {
                        underAttack = 1;
                        break;
                    }
                }
            } else if (d[i - 1] >= 5) {
                return 0;
            }
            if (!underAttack) {
                ++i;
                if (i <= 8) {
                    break;
                }
                for (k = 0; k < 8; ++k) {
                    printf("%2d", d[k]);
                }
                printf("\n");
                i = 8;
            }
            while (d[i - 1] >= 8) {
                --i;
            }
            ++d[i - 1];
        }
    }
```

```
    return 0;
}
```

Листинг 2 – Программный код на языке Java

```
public class Main {
    public Main() {
        int[] d = new int[8];
        int i = 1;

        while (true) {
            d[i - 1] = 1;
            while (true) {
                boolean underAttack = false;
                if (i != 1) {
                    for (int j = 1; j < i; ++j) {
                        if (d[i - 1] == d[j - 1]
                                || Math.abs(d[i - 1] - d[j - 1])
== Math.abs(i - j)) {
                            underAttack = true;
                            break;
                        }
                    }
                } else if (d[i - 1] >= 5) {
                    return;
                }
                if (!underAttack) {
                    ++i;
                    if (i <= 8) {
                        break;
                    }
                    for (int k = 0; k < 8; ++k) {
                        System.out.print(" " + d[k]);
                    }
                    System.out.println();
                    i = 8;
                }
                while (d[i - 1] >= 8) {
                    --i;
                }
                ++d[i - 1];
            }
        }
    }

    public static void main(String[] args) {
        new Main();
    }
}
```

Листинг 3 – Программный код на языке Python

```
# -*- coding: utf-8 -*-


def main():
    d = [0 for x in range(8)]
    i = 1
    while True:
        d[i - 1] = 1
        while True:
            underAttack = False
            if i != 1:
                for j in range(1, i):
                    if d[i - 1] == d[j - 1] or abs(d[i - 1] -
d[j - 1]) == abs(i - j):
                        underAttack = True
                        break
            elif d[i - 1] >= 5:
                return 0
            if not underAttack:
                i += 1
                if i <= 8:
                    break
                print(' ' + ' '.join([str(x) for x in d]))
                i = 8
            while d[i - 1] >= 8:
                i -= 1
            d[i - 1] += 1
    return 0


if __name__ == '__main__':
    main()
```

Листинг 4 – Результаты вычислений

```
1 5 8 6 3 7 2 4
1 6 8 3 7 4 2 5
1 7 4 6 8 2 5 3
1 7 5 8 2 4 6 3
2 4 6 8 3 1 7 5
2 5 7 1 3 8 6 4
2 5 7 4 1 8 6 3
2 6 1 7 4 8 3 5
2 6 8 3 1 4 7 5
2 7 3 6 8 5 1 4
2 7 5 8 1 4 6 3
2 8 6 1 3 5 7 4
3 1 7 5 8 2 4 6
3 5 2 8 1 7 4 6
3 5 2 8 6 4 7 1
3 5 7 1 4 2 8 6
3 5 8 4 1 7 2 6
3 6 2 5 8 1 7 4
3 6 2 7 1 4 8 5
3 6 2 7 5 1 8 4
```

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

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

После трансляции и выполнения любой из приведённых выше программ будут получены все уникальные решения. Конечно, можно исключить и некоторые другие симметричные решения, существуют способы сделать это и получить небольшое количество оставшихся «фундаментальных».

Подобный подход может быть использован и при решении некоторых классических комбинаторных задач и задач теории расписаний, например, при планировании работ вычислительных систем, что позволит некоторым образом сэкономить ресурсы без использования рекурсии.

## Литература

1. *Арсак Ж.* Программирование игр и головоломок / *Ж. Арсак.* – М.: Наука, 1990. – 224 с.

2. *Корнилов Е.Н.* Программирование шахмат и других логических игр / *Е.Н. Корнилов.* – СПб: БХВ-Петербург, 2005. – 272 с.: ил.

3. *Дагене В.А.* 100 задач по программированию: Кн. для учащихся / *В.А. Дагене, Г.К. Григас, К.Ф. Аугутис.* – М.: Просвещение, 1993. – 255 с.

4. *Ламуатье Ж.-П.* Упражнения по программированию на Фортране-IV / *Ж.-П. Ламуатье.* – М.: Мир, 1978. – 168 с.

## Юмор Юмор Юмор Юмор Юмор Юмор

### Хвала умеренному суеверию

Прочитав в своём гороскопе, что ему опасно сегодня выходить из дома, человек, испугавшись предсказанной угрозы, наглухо закрылся в квартире, чтобы не было соблазна выйти на улицу. А в его доме в тот день вспыхнул сильный пожар.

Очнулся он на носилках, на которых пожарные выносили его из огня. «Как хорошо, – подумал приверженец советов гороскопа, – что пожарные не так суеверны, как я!»