

Ткаченко Кирилл Станиславович
Старший преподаватель

Разработка программ для построения латинских квадратов на нескольких языках программирования высокого уровня

В настоящей публикации рассматривается разработка очень простых программ для построения латинских квадратов на нескольких языках программирования высокого уровня — C#, Java, Python. Решения на двух языках используют объектно-ориентированный подход. Приводятся полные исходные тексты программ, которые станут полезны школьникам, изучающим программирование.

Школьникам полезно решать простые задачи, связанные с программированием. Решение простых задач является для школьников хорошей тренировкой. Одной из таких простых задач является построение латинских квадратов [1]. Латинским квадратом n-го порядка называют квадратную таблицу размером n на n, которая заполнена n различными элементами таким образом, чтобы в каждой строке и в каждом столбце каждый элемент встречался однократно. Латинские квадраты часто заполняют целыми положительными числами 1, 2, ..., n, либо буквами латинского алфавита. Пример латинского квадрата 9-го порядка приводится ниже:

1 2 3 4 5 6 7 8 9
2 3 4 5 6 7 8 9 1
3 4 5 6 7 8 9 1 2
4 5 6 7 8 9 1 2 3
5 6 7 8 9 1 2 3 4
6 7 8 9 1 2 3 4 5
7 8 9 1 2 3 4 5 6
8 9 1 2 3 4 5 6 7
9 1 2 3 4 5 6 7 8

Для построения латинского квадрата n-го порядка известна формула [1]:

$$lij = (i + j - 1) mod n. \quad (1)$$

В формуле (1) l_{ij} — значение элемента латинского квадрата, 0, 1, ..., $n - 1$, i — номер строки, 0, 1, ..., $n - 1$, j — номер столбца, 0, 1, ..., $n - 1$, mod — операция нахождения остатка от деления.

Чтобы значениями l_{ij} были целые положительные числа, а в левом верхнем углу была единица, формулу (1) можно модифицировать, подобрав смещение:

$$l_{ij} = ((i + j)\text{mod}n) + 1. \quad (2)$$

Схема алгоритма построения латинского квадрата приводится на рисунке 1.

Эту схему алгоритма можно реализовать на различных языках программирования высокого уровня.

Программа на языке C#

Программа на языке C# начинается с расширения области видимости:

```
using System;
```

Определяется головной класс, в нем определяется головная функция:

```
public class Potentsial {
    public static void Main(string[] args) {
```

Вводится порядок латинского квадрата:

```
        Console.Write("N: ");
        int n = Convert.ToInt32(Console.ReadLine());
```

Определяется двумерный массив для хранения элементов латинского квадрата:

```
        int[,] lkva = new int[n, n];
```

По формуле (2) рассчитываются значения элементов латинского квадрата:

```
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                lkva[i, j] = ((i + j) % n) + 1;
            }
        }
```

Выводятся значения элементов латинского квадрата:

```
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                Console.Write(" " + lkva[i, j]);
            }
            Console.WriteLine();
        }
    }
}
```

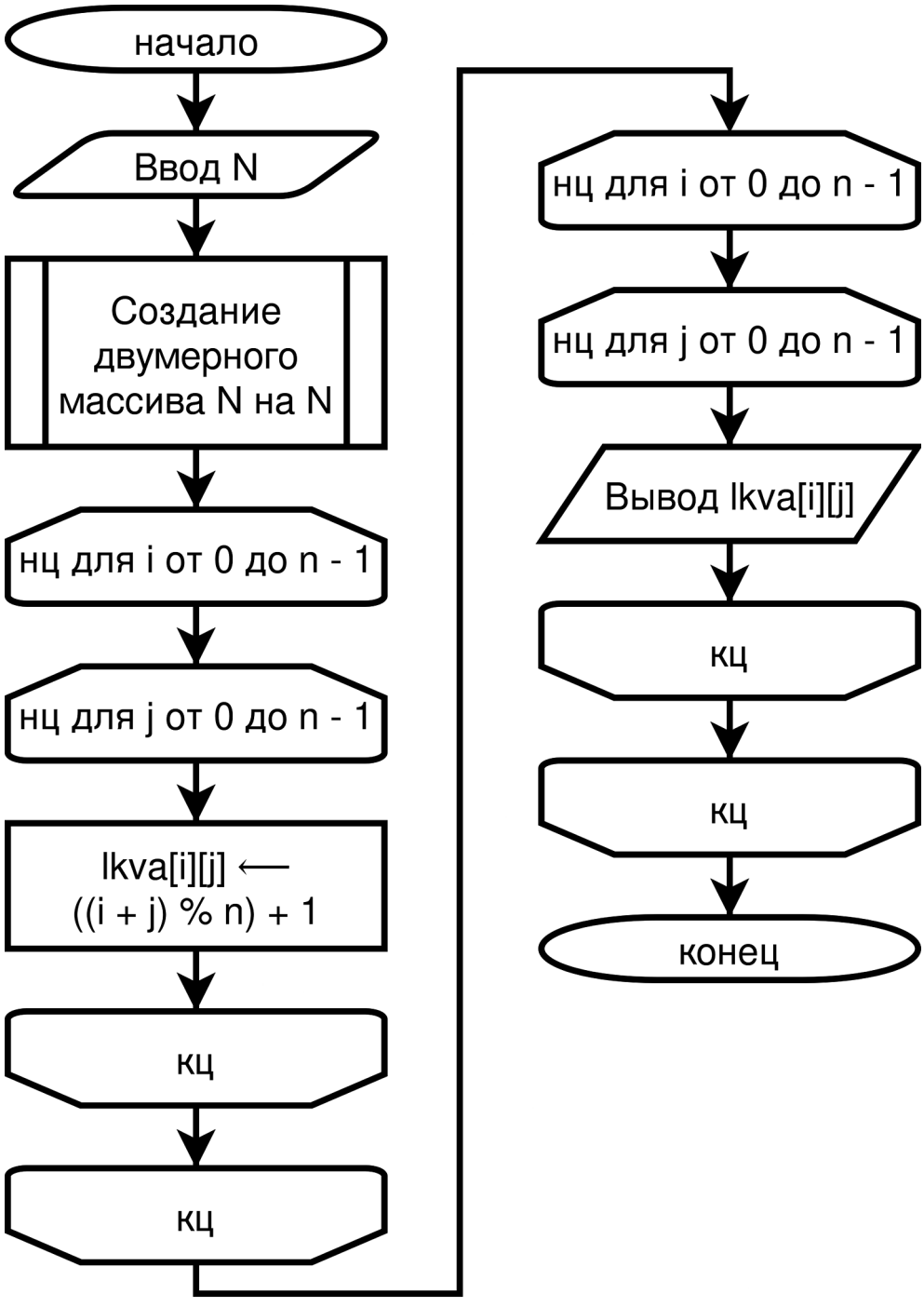


Рис. 1. Схема алгоритма построения латинского квадрата

Программа на языке Java

Программа на языке *Java* начинается с определения головного класса, в нем — головной функции, в ней — объекта для разбора вводимых значений:

```
public class Potentsial {
    public static void main(String[] args) {
        java.util.Scanner sc = new java.util.Scanner(System.in);
```

Вводится порядок латинского квадрата:

```
        System.out.print("N: ");
        int n = sc.nextInt();
```

Определяется двумерный массив для хранения элементов латинского квадрата:

```
        int[][] lkva = new int[n][n];
```

По формуле (2) рассчитываются значения элементов латинского квадрата:

```
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                lkva[i][j] = ((i + j) % n) + 1;
            }
        }
```

Выводятся значения элементов латинского квадрата:

```
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                System.out.print(" " + lkva[i][j]);
            }
            System.out.println();
        }
    }
}
```

Программа на языке Python

Программа на языке *Python* начинается со ввода порядка латинского квадрата:

```
n = int(input("N: "))
```

Определяется двумерный массив для хранения элементов латинского квадрата:

```
lkva = [[0 for _ in range(n)] for _ in range(n)]
```

По формуле (2) рассчитываются значения элементов латинского квадрата:

```
for i in range(n):
    for j in range(n):
        lkva[i][j] = ((i + j) % n) + 1
```

Выводятся значения элементов латинского квадрата:

```
for i in range(n):
    for j in range(n):
        print(f" {lkva[i][j]}", end="")
    print()
```

Полный исходный текст программы на языке C# приводится в приложении А, на языке *Java* — приложении Б, на языке *Python* — приложении В, результаты работы программ — приложении Г. Эти разработки станут полезны для школьников, изучающих программирование.

Литература

1. Латинский квадрат. Википедия. Режим доступа: URL: https://ru.wikipedia.org/wiki/Латинский_квадрат (дата доступа: 18.06.2026).

Приложение А

Полный исходный текст программы на языке C#

```
using System;

public class Potentsial {
 public static void Main(string[] args) {
   Console.Write("N: ");
   int n = Convert.ToInt32(Console.ReadLine());

   int[,] lkva = new int[n, n];

   for (int i = 0; i < n; ++i) {
     for (int j = 0; j < n; ++j) {
       lkva[i, j] = ((i + j) % n) + 1;
     }
   }

   for (int i = 0; i < n; ++i) {
     for (int j = 0; j < n; ++j) {
       Console.Write(" " + lkva[i, j]);
     }
     Console.WriteLine();
   }
 }
}
```

Приложение Б

Полный исходный текст программы на языке Java

```
public class Potentsial {
 public static void main(String[] args) {
   java.util.Scanner sc = new java.util.Scanner(System.in);
   System.out.print("N: ");
   int n = sc.nextInt();

   int[][] lkva = new int[n][n];

   for (int i = 0; i < n; ++i) {
     for (int j = 0; j < n; ++j) {
       lkva[i][j] = ((i + j) % n) + 1;
     }
   }

   for (int i = 0; i < n; ++i) {
     for (int j = 0; j < n; ++j) {
       System.out.print(" " + lkva[i][j]);
     }
     System.out.println();
   }
 }
}
```

Приложение В

Полный исходный текст программы на языке Python

```
n = int(input("N: "))

lkva = [[0 for _ in range(n)] for _ in range(n)]

for i in range(n):
 for j in range(n):
   lkva[i][j] = ((i + j) % n) + 1

for i in range(n):
 for j in range(n):
   print(f" {lkva[i][j]}", end="")
 print()
```

Приложение Г

Результаты работы программ

```
N: 9
   1 2 3 4 5 6 7 8 9
   2 3 4 5 6 7 8 9 1
   3 4 5 6 7 8 9 1 2
   4 5 6 7 8 9 1 2 3
   5 6 7 8 9 1 2 3 4
   6 7 8 9 1 2 3 4 5
   7 8 9 1 2 3 4 5 6
   8 9 1 2 3 4 5 6 7
   9 1 2 3 4 5 6 7 8
```

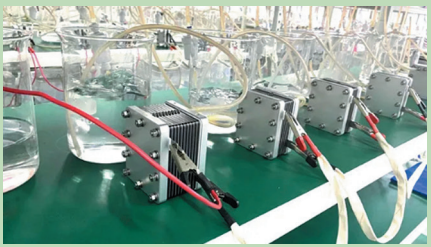
НОВОСТИ

Учёные МФТИ предложили новый материал для более эффективного получения «зелёного» водорода

В МФТИ разработали перспективный катализатор, способный сделать производство водорода из воды более доступным и экономически эффективным. Результаты исследования, посвящённого созданию материала на основе соединений марганца, селена и неодима, опубликованы в научном журнале Fuel.

Водород рассматривается как одно из ключевых экологичных видов топлива будущего, так как продуктом его сгорания является обычная вода. Однако широкому внедрению «зелёного» водорода препятствует высокая стоимость его получения методом электролиза. Для эффективного расщепления воды требуются катализаторы. Традиционно лучшими из них считаются благородные металлы, прежде всего платина, однако их высокая цена и ограниченность запасов заставляют исследователей искать доступные альтернативы.

Учёные Физтеха предложили использовать гетероструктуру, состоящую из селенида марганца и оксида неодима. В таком материале компоненты соприкасаются на атомном уровне, взаимно усиливая каталитические свойства системы. Особое межфазное взаимодействие облегчает перенос электронов и повышает активность поверхности, а развитая пористая структура существенно увеличивает площадь контакта с электролитом.



Лабораторные испытания показали высокий потенциал разработки в щелочной среде. Для достижения стандартной плотности тока 10 мА/см^2 катализатору потребовалось лишь 83 мВ перенапряжения. Чем ниже этот показатель, тем меньше энергии расходуется на реакцию. Кроме того, материал продемонстрировал устойчивую работу при высокой плотности тока 100 мА/см^2 (при 152 мВ) и сохранял стабильность на протяжении 80 часов непрерывного тестирования.

Как отмечает соавтор работы, профессор кафедры молекулярной и клеточной биологии ФБМФ МФТИ, член-корреспондент РАН Сулейман Аллахвердиев, практическая ценность исследования заключается в открытии пути к созданию доступных каталитических систем. В случае успешного подтверждения долговечности и масштабируемости в промышленных условиях подобные гетероструктуры смогут стать базой для нового поколения электролизёров и ускорить переход к низкоуглеродной энергетике.

Источник: https://mipt.ru/news