Московский государственный технический университет
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию

Выполнил:
студент группы ИУ5-36Б
Бойко Артем Андреевич
Проверил:
Нардид Анатолий Николаевич

Москва, 2025

Цель домашнего задания: изучение ранее неизвестного языка программирования.

Задание:

1. Выберите язык программирования (который Вы ранее не изучали) и (1) напишите по нему реферат с примерами кода или (2) реализуйте на нем небольшой проект (с детальным текстовым описанием).
2. Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка или содержать решение какой-либо задачи на этом языке.
3. Необходимо установить на свой компьютер компилятор (интерпретатор, транспилятор) этого языка и произвольную среду разработки.
4. В случае написания реферата необходимо разработать и откомпилировать примеры кода (или модифицировать стандартные примеры).
5. В случае создания проекта необходимо детально комментировать код.
6. При написании реферата (создании проекта) необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.
7. Приветствуется написание черновика статьи по результатам выполнения ДЗ. Черновик статьи может быть подготовлен группой студентов, которые исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

matrix_calculator.nim

```
# Основная программа - Калькулятор матриц на Nim
# Консольное приложение для выполнения операций с матрицами

# Импорт необходимых модулей:
# - matrix_operations: наш собственный модуль с матричными операциями
# - strutils: для работы со строками
# - random: для генерации случайных чисел
# - math: для математических операций (round и др.)
import matrix_operations
import std/[strutils, random, math]

# Константы для информации о программе
const
  VERSION = "1.2"                    # Версия программы
  AUTHOR = "Nim Matrix Calculator"    # Автор/название программы


proc showMenu() =
  ## Отображает главное меню программы

  # Верхняя граница меню
  echo "=".repeat(60)  # Создаем строку из 60 знаков "="

  # Заголовок программы с версией
  echo "Калькулятор матриц v" & VERSION

  # Нижняя граница меню
  echo "=".repeat(60)

  # Список доступных операций
  echo "1. Сложение матриц"
  echo "2. Умножение матриц"
  echo "3. Транспонирование матрицы"
  echo "4. Вычисление определителя"
  echo "5. Создать случайную матрицу"
  echo "6. Примеры матриц (для тестирования)"
  echo "7. Создать единичную матрицу"
  echo "8. Умножение матрицы на скаляр"
  echo "0. Выход"

  # Разделитель
  echo "-".repeat(60)

proc createRandomMatrix(): Matrix =
  ## Создает случайную матрицу по параметрам от пользователя
  ## Возвращает случайно сгенерированную матрицу

  echo "=== Создание случайной матрицы ==="

  # Запрос количества строк
  echo "Введите количество строк:"
  let rowsStr = readLine(stdin).strip()  # Читаем ввод пользователя

  # Преобразуем в число с обработкой ошибок
```

```
  # Если преобразование не удалось, используем значение по умолчанию (2)
  let rows = try:
    rowsStr.parseInt()  # Пытаемся преобразовать строку в целое число
  except:
    2  # Значение по умолчанию в случае ошибки

  # Запрос количества столбцов (аналогично строкам)
  echo "Введите количество столбцов:"
  let colsStr = readLine(stdin).strip()
  let cols = try:
    colsStr.parseInt()
  except:
    2  # Значение по умолчанию

  # Проверка корректности размеров
  if rows <= 0 or cols <= 0:
    echo "Используются значения по умолчанию: 2x2"
    return createRandomMatrix()  # Рекурсивный вызов с правильными параметрами

  # Инициализация генератора случайных чисел
  randomize()

  # Создаем матрицу нужного размера
  result = createMatrix(rows, cols)

  # Заполняем матрицу случайными числами от 1.0 до 10.0
  for i in 0..<rows:
    for j in 0..<cols:
      # Генерируем случайное число от 1.0 до 10.0
      let randomValue = rand(1.0..10.0)

      # Округляем до 2 знаков после запятой:
      # 1. Умножаем на 100: 1.2345 -> 123.45
      # 2. Округляем: 123.45 -> 123.0
      # 3. Делим на 100: 123.0 -> 1.23
      result[i][j] = round(randomValue * 100) / 100

proc showExamples() =
  ## Показывает примеры матриц для тестирования программы

  echo "=== Примеры матриц для тестирования ==="
  echo ""  # Пустая строка для читаемости

  # Пример 1: Матрица 2x2
  echo "1. Матрица 2x2:"
  let m1 = createMatrixFromArray(@[@[1.0, 2.0], @[3.0, 4.0]])
  printMatrix(m1)

  # Пример 2: Матрица 3x3
  echo "\n2. Матрица 3x3:"
  let m2 = createMatrixFromArray(@[
    @[1.0, 2.0, 3.0],
    @[4.0, 5.0, 6.0],
    @[7.0, 8.0, 9.0]
  ])
  printMatrix(m2)
```

```
  # Пример 3: Единичная матрица 3x3
  echo "\n3. Единичная матрица 3x3:"
  let m3 = getIdentityMatrix(3)
  printMatrix(m3)

  # Пауза для просмотра примеров
  echo "\nНажмите Enter для продолжения..."
  discard readLine(stdin)  # Игнорируем введенную строку

proc showHelp() =
  ## Показывает справку по формату ввода матриц

  echo "=== Справка по вводу матриц ==="

  # Основные правила ввода
  echo "1. Сначала введите количество строк"
  echo "2. Затем введите количество столбцов"
  echo "3. Вводите матрицу построчно"
  echo "4. В каждой строке введите числа через пробел"

  echo ""  # Пустая строка

  # Конкретный пример
  echo "Пример ввода матрицы 2x3:"
  echo "Количество строк: 2"
  echo "Количество столбцов: 3"
  echo "Строка 1: 1 2 3"
  echo "Строка 2: 4 5 6"

  echo ""  # Пустая строка

  # Пауза для чтения справки
  echo "Нажмите Enter для продолжения..."
  discard readLine(stdin)

proc createIdentityMatrix(): Matrix =
  ## Создает единичную матрицу по размеру от пользователя
  ## Возвращает: единичную матрицу указанного размера

  echo "=== Создание единичной матрицы ==="

  # Запрос размера матрицы
  echo "Введите размер матрицы (n x n):"
  let nStr = readLine(stdin).strip()

  # Преобразуем в число с обработкой ошибок
  let n = try:
    nStr.parseInt()
  except:
    3  # Значение по умолчанию

  # Проверка корректности размера
  if n <= 0:
    echo "Размер должен быть положительным. Используется 3."
    result = getIdentityMatrix(3)  # Используем значение по умолчанию
  else:
    result = getIdentityMatrix(n)  # Создаем матрицу указанного размера
```

```
  # Показываем результат
  echo "Создана единичная матрица:"
  printMatrix(result)

proc multiplyByScalar(): Matrix =
  ## Умножает матрицу на скаляр (число)
  ## Возвращает: результат умножения

  echo "=== Умножение матрицы на скаляр ==="

  # Ввод матрицы
  echo "Введите матрицу:"
  let m = inputMatrix()

  # Показываем введенную матрицу
  echo "Введенная матрица:"
  printMatrix(m)

  # Ввод скаляра (числа-множителя)
  echo "Введите скаляр (число):"
  let scalarStr = readLine(stdin).strip()

  # Преобразуем в число с плавающей точкой
  let scalar = try:
    scalarStr.parseFloat()
  except:
    2.0  # Значение по умолчанию

  # Выполняем умножение
  result = scalarMultiply(m, scalar)

  # Показываем результат
  echo "Результат умножения на " & formatFloat(scalar, ffDecimal, precision = 2) & ":"
  printMatrix(result)


proc main() =
  ## Главная функция программы - точка входа

  # Приветствие
  echo "Добро пожаловать в калькулятор матриц! (автор: " & AUTHOR & ")"
  echo "Для справки по вводу нажмите 'h' в главном меню"
  echo ""  # Пустая строка

  # Основной цикл программы
  while true:
    try:  # Блок try для обработки исключений
      # Показываем меню
      showMenu()

      # Запрос выбора операции
      echo "Выберите операцию (или 'h' для справки):"

      # Читаем выбор пользователя
      var choice = readLine(stdin).strip().toLowerAscii()  # Приводим к нижнему регистру
```

```
# Обработка выбора пользователя (конструкция case)
case choice

# Вариант 0: Выход из программы
of "0", "выход", "exit", "quit":
  echo "До свидания! Спасибо за использование калькулятора."
  break  # Выход из цикла while, завершение программы

# Вариант h: Показать справку
of "h", "help", "?", "справка":
  showHelp()
  continue  # Переходим к следующей итерации цикла

# Вариант 1: Сложение матриц
of "1":
  echo "=== Сложение матриц ==="

  # Ввод первой матрицы
  echo "Введите первую матрицу:"
  let m1 = inputMatrix()
  echo "Первая матрица:"
  printMatrix(m1)

  # Ввод второй матрицы
  echo "\nВведите вторую матрицу:"
  let m2 = inputMatrix()
  echo "Вторая матрица:"
  printMatrix(m2)

  # Выполнение сложения с обработкой ошибок
  try:
    let result = addMatrices(m1, m2)
    echo "\n✓ Результат сложения:"
    printMatrix(result)
  except ValueError as e:
    echo "\n✗ Ошибка: " & e.msg  # Вывод сообщения об ошибке

# Вариант 2: Умножение матриц (аналогично сложению)
of "2":
  echo "=== Умножение матриц ==="
  echo "Введите первую матрицу:"
  let m1 = inputMatrix()
  echo "Первая матрица:"
  printMatrix(m1)

  echo "\nВведите вторую матрицу:"
  let m2 = inputMatrix()
  echo "Вторая матрица:"
  printMatrix(m2)

  try:
    let result = multiplyMatrices(m1, m2)
    echo "\n✓ Результат умножения:"
    printMatrix(result)
  except ValueError as e:
    echo "\n✗ Ошибка: " & e.msg
```

```
# Вариант 3: Транспонирование матрицы
of "3":
  echo "=== Транспонирование матрицы ==="
  echo "Введите матрицу:"
  let m = inputMatrix()
  echo "\nИсходная матрица:"
  printMatrix(m)

  # Транспонирование (не требует обработки ошибок для корректных матриц)
  let result = transposeMatrix(m)
  echo "\n✓ Транспонированная матрица:"
  printMatrix(result)

# Вариант 4: Вычисление определителя
of "4":
  echo "=== Вычисление определителя ==="
  echo "Введите квадратную матрицу:"
  let m = inputMatrix()
  echo "\nИсходная матрица:"
  printMatrix(m)

  try:
    # Вычисление определителя
    let det = determinant(m)

    # Вывод результата с 6 знаками после запятой
    echo "\n✓ Определитель матрицы: " & formatFloat(det, ffDecimal, precision = 6)

    # Дополнительная информация о матрице
    if abs(det) < 1e-10:  # Если определитель очень близок к нулю
      echo "⚠ Матрица вырожденная (определитель близок к нулю)"
    elif det == 0:  # Если определитель равен нулю
      echo "⚠ Матрица вырожденная (определитель равен нулю)"

  except ValueError as e:
    echo "\n✗ Ошибка: " & e.msg

# Вариант 5: Создание случайной матрицы
of "5":
  echo "=== Создание случайной матрицы ==="
  try:
    let m = createRandomMatrix()
    echo "\n✓ Случайная матрица:"
    printMatrix(m)
  except Exception as e:
    echo "\n✗ Ошибка: " & e.msg

# Вариант 6: Примеры матриц
of "6":
  showExamples()

# Вариант 7: Создание единичной матрицы
of "7":
  let m = createIdentityMatrix()
  # Результат уже выведен внутри функции
```

```
      # Вариант 8: Умножение на скаляр
      of "8":
        let m = multiplyByScalar()
        # Результат уже выведен внутри функции

      # Неизвестный выбор
      else:
        echo "Неверный выбор. Попробуйте снова."

    # Обработка исключений
    except ValueError as e:  # Ошибки ввода данных
      echo "\n❌ Ошибка ввода: " & e.msg
    except Exception as e:  # Все остальные ошибки
      echo "\n❌ Неожиданная ошибка: " & e.msg

    # Пауза между операциями
    echo "\n" & "-".repeat(40)  # Разделитель
    echo "Нажмите Enter для продолжения..."
    discard readLine(stdin)  # Ожидание нажатия Enter


when isMainModule:
  ## Условная компиляция: этот код выполняется только если файл
  ## компилируется как основная программа, а не импортируется как модуль

  main()  # Запуск главной функции
```

matrix_operations.nim

```
# Модуль операций с матрицами на языке Nim
# Реализация основных матричных операций


# Импорт необходимых модулей из стандартной библиотеки Nim
import std/[strutils, math]  # strutils для работы со строками, math для математических операций


# Определяем тип Matrix как последовательность последовательностей float
# Звездочка (*) означает, что тип экспортируется (публичный) из модуля
type Matrix* = seq[seq[float]]  # Двумерный массив чисел с плавающей точкой


proc createMatrix*(rows, cols: int): Matrix =
  ## Создает новую матрицу заданного размера, заполненную нулями
  ## rows: количество строк (целое число, больше 0)
  ## cols: количество столбцов (целое число, больше 0)
  ## Возвращает: матрицу размера rows x cols, заполненную нулями

  # Создаем внешнюю последовательность (строки) длиной rows
  result = newSeq[seq[float]](rows)  # result - специальная переменная в Nim для возвращаемого
значения

  # Для каждой строки создаем внутреннюю последовательность (столбцы) длиной cols
  for i in 0..<rows:  # Диапазон 0..<rows означает от 0 до rows-1 включительно
```

```
    result[i] = newSeq[float](cols)  # Инициализируем каждую строку последовательностью нулей

proc createMatrixFromArray*(data: seq[seq[float]]): Matrix =
  ## Создает матрицу из двумерной последовательности чисел
  ## data: двумерная последовательность чисел
  ## Возвращает: матрицу с данными из последовательности

  # Получаем количество строк
  let rows = data.len

  # Если данных нет, возвращаем пустую матрицу
  if rows == 0:
    return createMatrix(0, 0)

  # Получаем количество столбцов из первой строки
  let cols = data[0].len

  # Создаем матрицу нужного размера
  result = createMatrix(rows, cols)

  # Копируем данные из последовательности в матрицу
  for i in 0..<rows:
    # Проверяем, что все строки имеют одинаковую длину
    if data[i].len != cols:
      raise newException(ValueError, "Все строки должны иметь одинаковую длину")

    # Копируем элементы
    for j in 0..<cols:
      result[i][j] = data[i][j]


proc inputMatrix*(): Matrix =
  ## Запрашивает у пользователя ввод матрицы через консоль
  ## Возвращает: введенную пользователем матрицу

  # Шаг 1: Ввод количества строк
  echo "Введите количество строк:"
  var rowsStr = readLine(stdin).strip()  # Читаем строку, удаляем пробелы по краям

  # Проверяем, что строка не пустая
  while rowsStr == "":
    echo "Пожалуйста, введите количество строк:"
    rowsStr = readLine(stdin).strip()

  # Преобразуем строку в целое число с обработкой ошибок
  let rows = try:
    rowsStr.parseInt()  # Пытаемся преобразовать строку в целое число
  except ValueError:  # Если не удалось (пользователь ввел не число)
    echo "Ошибка: введите целое число"
    return inputMatrix()  # Рекурсивно вызываем функцию снова

  # Проверяем, что количество строк положительное
  if rows <= 0:
    echo "Ошибка: количество строк должно быть положительным"
    return inputMatrix()

  # Шаг 2: Ввод количества столбцов (аналогично строкам)
```

```
  echo "Введите количество столбцов:"
  var colsStr = readLine(stdin).strip()

  while colsStr == "":
    echo "Пожалуйста, введите количество столбцов:"
    colsStr = readLine(stdin).strip()

  let cols = try:
    colsStr.parseInt()
  except ValueError:
    echo "Ошибка: введите целое число"
    return inputMatrix()

  if cols <= 0:
    echo "Ошибка: количество столбцов должно быть положительным"
    return inputMatrix()

  # Создаем матрицу нужного размера
  var matrix = createMatrix(rows, cols)

  # Инструкции для пользователя
  echo "Введите матрицу " & $rows & "x" & $cols & " построчно (в каждой строке " & $cols & "
чисел, разделенных пробелами):"
  echo "Пример строки: 1 2.5 3.14"

  # Шаг 3: Ввод элементов построчно
  for i in 0..<rows:
    while true:  # Цикл продолжается до тех пор, пока пользователь не введет корректную строку
      echo "Строка " & $(i+1) & ":"  # Нумерация строк для пользователя начинается с 1
      let line = readLine(stdin).strip()

      # Проверка на пустую строку
      if line == "":
        echo "Строка не может быть пустой. Попробуйте снова."
        continue  # Переходим к следующей итерации цикла

      # Разбиваем строку на элементы по пробелам
      let elements = line.splitWhitespace()

      # Проверяем количество элементов
      if elements.len != cols:
        echo "Ошибка: ожидается " & $cols & " элементов, получено " & $elements.len & ".
Попробуйте снова."

        # Создаем пример правильного ввода для пользователя
        var example = ""
        for k in 0..<cols:
          example.add("1.0")
          if k < cols - 1:
            example.add(" ")
        echo "Пример правильного ввода: " & example

        continue

      # Проверяем, что все элементы - корректные числа
      var allValid = true  # Флаг корректности всех элементов
      for j in 0..<cols:
```

```
        try:
          # Пытаемся преобразовать строку в число с плавающей точкой
          matrix[i][j] = elements[j].parseFloat()
        except ValueError:  # Если не удалось
          echo "Ошибка: '" & elements[j] & "' не является числом. Попробуйте снова."
          allValid = false  # Устанавливаем флаг в false
          break  # Прерываем внутренний цикл

      # Если все элементы корректны, выходим из цикла while
      if allValid:
        break

  # Возвращаем введенную матрицу
  return matrix

proc printMatrix*(m: Matrix) =
  ## Выводит матрицу в консоль в красивом формате
  ## m: матрица для вывода

  # Проверка на пустую матрицу
  if m.len == 0:
    echo "[Пустая матрица]"
    return  # Завершаем выполнение функции

  # Получаем размеры матрицы
  let rows = m.len        # Количество строк
  let cols = m[0].len     # Количество столбцов (берем длину первой строки)

  # Шаг 1: Находим максимальную длину строкового представления числа
  # Это нужно для красивого выравнивания при выводе
  var maxLen = 0  # Начальное значение максимальной длины

  for i in 0..<rows:
    for j in 0..<cols:
      # Форматируем число с 2 знаками после запятой и получаем длину строки
      let strLen = formatFloat(m[i][j], ffDecimal, precision = 2).len

      # Обновляем максимальную длину если текущая больше
      if strLen > maxLen:
        maxLen = strLen

  # Шаг 2: Выводим матрицу с выравниванием
  echo "Матрица " & $rows & "x" & $cols & ":"  # Заголовок с размерами

  for i in 0..<rows:
    stdout.write "[ "  # Начало строки матрицы

    for j in 0..<cols:
      # Форматируем число
      let numStr = formatFloat(m[i][j], ffDecimal, precision = 2)

      # Выводим число с выравниванием по правому краю
      stdout.write align(numStr, maxLen)

      # Добавляем пробелы между элементами (кроме последнего)
      if j < cols - 1:
        stdout.write "  "
```

```
    echo " ]"  # Конец строки матрицы и переход на новую строку


proc addMatrices*(a, b: Matrix): Matrix =
  ## Сложение двух матриц
  ## a, b: матрицы для сложения
  ## Возвращает: сумму матриц a + b

  # Проверка на пустые матрицы
  if a.len == 0 or b.len == 0:
    raise newException(ValueError, "Матрицы не могут быть пустыми")

  # Проверка совпадения размеров (необходимое условие для сложения)
  if a.len != b.len or a[0].len != b[0].len:
    # Создаем информативное сообщение об ошибке
    let errorMsg = "Матрицы должны быть одного размера для сложения. " &
                   "Первая: " & $a.len & "x" & $a[0].len & ", вторая: " & $b.len & "x" & $b[0].len
    raise newException(ValueError, errorMsg)

  # Создаем матрицу для результата того же размера, что и входные матрицы
  result = createMatrix(a.len, a[0].len)

  # Поэлементное сложение
  for i in 0..<a.len:
    for j in 0..<a[0].len:
      result[i][j] = a[i][j] + b[i][j]

proc multiplyMatrices*(a, b: Matrix): Matrix =
  ## Умножение двух матриц
  ## a, b: матрицы для умножения (a * b)
  ## Возвращает: произведение матриц a * b

  # Проверка на пустые матрицы
  if a.len == 0 or b.len == 0:
    raise newException(ValueError, "Матрицы не могут быть пустыми")

  # Проверка условия умножения матриц:
  # Количество столбцов первой матрицы должно равняться количеству строк второй
  if a[0].len != b.len:
    # Создаем информативное сообщение об ошибке
    let errorMsg = "Невозможно умножить матрицы: количество столбцов первой (" & $a[0].len & ")
" &
                   "должно равняться количеству строк второй (" & $b.len & ")"
    raise newException(ValueError, errorMsg)

  # Создаем матрицу для результата:
  # Количество строк как у первой матрицы, столбцов как у второй
  result = createMatrix(a.len, b[0].len)

  # Алгоритм умножения матриц (строка на столбец)
  for i in 0..<a.len:             # Для каждой строки первой матрицы
    for j in 0..<b[0].len:        # Для каждого столбца второй матрицы
      var sum = 0.0               # Накопитель для суммы

      for k in 0..<a[0].len:      # Для каждого элемента строки/столбца
        # Умножаем элемент строки i первой матрицы на элемент столбца j второй матрицы
```

```
        sum += a[i][k] * b[k][j]

      # Сохраняем результат в результирующую матрицу
      result[i][j] = sum

proc transposeMatrix*(m: Matrix): Matrix =
  ## Транспонирование матрицы (строки становятся столбцами и наоборот)
  ## m: исходная матрица
  ## Возвращает: транспонированную матрицу

  # Проверка на пустую матрицу
  if m.len == 0:
    return createMatrix(0, 0)  # Транспонирование пустой матрицы - пустая матрица

  # Получаем размеры исходной матрицы
  let rows = m.len      # Количество строк исходной матрицы
  let cols = m[0].len   # Количество столбцов исходной матрицы

  # Создаем матрицу с обратными размерами (столбцы становятся строками и наоборот)
  result = createMatrix(cols, rows)

  # Заполняем транспонированную матрицу
  for i in 0..<rows:
    for j in 0..<cols:
      result[j][i] = m[i][j]  # Элемент [i,j] исходной становится [j,i] в транспонированной

proc determinant*(m: Matrix): float =
  ## Вычисление определителя квадратной матрицы (рекурсивный алгоритм)
  ## m: квадратная матрица
  ## Возвращает: определитель матрицы

  # Проверка на пустую матрицу
  if m.len == 0:
    raise newException(ValueError, "Матрица не может быть пустой")

  # Проверка, что матрица квадратная (необходимое условие для определителя)
  if m.len != m[0].len:
    let errorMsg = "Матрица должна быть квадратной для вычисления определителя. " &
                   "Получена матрица " & $m.len & "x" & $m[0].len
    raise newException(ValueError, errorMsg)

  let n = m.len  # Размер квадратной матрицы (n x n)

  # Базовые случаи рекурсии:

  # Матрица 1x1: определитель равен единственному элементу
  if n == 1:
    return m[0][0]

  # Матрица 2x2: вычисляем по формуле ad - bc
  if n == 2:
    return m[0][0] * m[1][1] - m[0][1] * m[1][0]

  # Матрица 3x3: вычисляем по правилу Саррюса (треугольников)
  if n == 3:
    # Правило Саррюса:
    # Сумма произведений элементов главных диагоналей минус
```

```
    # сумма произведений элементов побочных диагоналей
    return m[0][0]*m[1][1]*m[2][2] +
           m[0][1]*m[1][2]*m[2][0] +
           m[0][2]*m[1][0]*m[2][1] -
           m[0][2]*m[1][1]*m[2][0] -
           m[0][0]*m[1][2]*m[2][1] -
           m[0][1]*m[1][0]*m[2][2]

  # Рекурсивный случай для матриц большего размера (n > 3):
  # Разложение определителя по первой строке (метод Лапласа)

  var det = 0.0  # Накопитель для определителя

  # Проходим по всем элементам первой строки
  for j in 0..<n:
    # Создаем минор (n-1)x(n-1) - матрица без первой строки и j-го столбца
    var minor = createMatrix(n-1, n-1)

    # Заполняем минор
    for i in 1..<n:             # Начинаем со второй строки (индекс 1)
      var colIndex = 0          # Индекс столбца в миноре

      for k in 0..<n:           # Проходим по всем столбцам исходной матрицы
        if k != j:              # Пропускаем j-й столбец
          minor[i-1][colIndex] = m[i][k]  # Копируем элемент в минор
          colIndex += 1         # Увеличиваем индекс столбца в миноре

    # Определяем знак для текущего элемента: (-1)^(1+j)
    let sign = if j mod 2 == 0: 1.0 else: -1.0  # Четные столбцы: +, нечетные: -

    # Рекурсивно вычисляем определитель минора и добавляем к общей сумме
    det += sign * m[0][j] * determinant(minor)

  # Возвращаем вычисленный определитель
  return det



proc isSquare*(m: Matrix): bool =
  ## Проверяет, является ли матрица квадратной
  ## m: матрица для проверки
  ## Возвращает: true если матрица квадратная, false в противном случае

  # Матрица квадратная если:
  # 1. Она не пустая (m.len > 0)
  # 2. Количество строк равно количеству столбцов
  m.len > 0 and m.len == m[0].len

proc getIdentityMatrix*(n: int): Matrix =
  ## Создает единичную матрицу размера n x n
  ## n: размер матрицы (целое число > 0)
  ## Возвращает: единичную матрицу размера n x n

  # Проверка корректности размера
  if n <= 0:
    raise newException(ValueError, "Размер должен быть положительным")
```

```
  # Создаем матрицу размера n x n, заполненную нулями
  result = createMatrix(n, n)

  # Заполняем главную диагональ единицами
  for i in 0..<n:
    result[i][i] = 1.0  # Элементы с одинаковыми индексами (i,i) = 1

proc scalarMultiply*(m: Matrix, scalar: float): Matrix =
  ## Умножает матрицу на скаляр (каждый элемент умножается на число)
  ## m: исходная матрица
  ## scalar: число-множитель
  ## Возвращает: матрицу, умноженную на скаляр

  # Проверка на пустую матрицу
  if m.len == 0:
    return createMatrix(0, 0)

  # Создаем матрицу того же размера
  result = createMatrix(m.len, m[0].len)

  # Умножаем каждый элемент на скаляр
  for i in 0..<m.len:
    for j in 0..<m[0].len:
      result[i][j] = m[i][j] * scalar
```

Результат выполнения программы

```
● @artem-hedgehog →/workspaces/ProgLangParadigms_BoikoA_2025/DZ (main) $ nim c -r matrix_calculator.nim
  Hint: used config file '/home/codespace/.choosenim/toolchains/nim-2.2.6/config/nim.cfg' [Conf]
  Hint: used config file '/home/codespace/.choosenim/toolchains/nim-2.2.6/config/config.nims' [Conf]
  Hint: mm: orc; threads: on; opt: none (DEBUG BUILD, `-d:release` generates faster code)
  10687 lines; 0.477s; 10.625MiB peakmem; proj: /workspaces/ProgLangParadigms_BoikoA_2025/DZ/matrix_calculator.nim; out: /workspaces/ProgLangParadigms_BoikoA_2025
  /DZ/matrix_calculator [SuccessX]
  Hint: /workspaces/ProgLangParadigms_BoikoA_2025/DZ/matrix_calculator [Exec]
  Добро пожаловать в калькулятор матриц! (автор: Nim Matrix Calculator)
  Для справки по вводу нажмите 'h' в главном меню

  ============================================================
  Калькулятор матриц v1.2
  ============================================================
  1. Сложение матриц
  2. Умножение матриц
  3. Транспонирование матрицы
  4. Вычисление определителя
  5. Создать случайную матрицу
  6. Примеры матриц (для тестирования)
  7. Создать единичную матрицу
  8. Умножение матрицы на скаляр
  0. Выход
  ------------------------------------------------------------
  Выберите операцию (или 'h' для справки):
1
=== Сложение матриц ===
Введите первую матрицу:
Введите количество строк:
2
Введите количество столбцов:
2
Введите матрицу 2x2 построчно (в каждой строке 2 чисел, разделенных пробелами):
Пример строки: 1 2.5 3.14
Строка 1:
1 2
Строка 2:
3 4
Первая матрица:
Матрица 2x2:
[ 1.00  2.00 ]
[ 3.00  4.00 ]

Введите вторую матрицу:
Введите количество строк:
2
Введите количество столбцов:
2
Введите матрицу 2x2 построчно (в каждой строке 2 чисел, разделенных пробелами):
Пример строки: 1 2.5 3.14
Строка 1:
-5 6
Строка 2:
7 -8
Вторая матрица:
Матрица 2x2:
[ -5.00   6.00 ]
[  7.00  -8.00 ]

✅ Результат сложения:
Матрица 2x2:
[ -4.00   8.00 ]
[ 10.00  -4.00 ]


------------------------------------------
Нажмите Enter для продолжения...
```

```
============================================================
Калькулятор матриц v1.2
============================================================
1. Сложение матриц
2. Умножение матриц
3. Транспонирование матрицы
4. Вычисление определителя
5. Создать случайную матрицу
6. Примеры матриц (для тестирования)
7. Создать единичную матрицу
8. Умножение матрицы на скаляр
0. Выход
------------------------------------------------------------
Выберите операцию (или 'h' для справки):
2
=== Умножение матриц ===
Введите первую матрицу:
Введите количество строк:
2
Введите количество столбцов:
2
Введите матрицу 2x2 построчно (в каждой строке 2 чисел, разделенных пробелами):
Пример строки: 1 2.5 3.14
Строка 1:
0 1
Строка 2:
2 3
Первая матрица:
Матрица 2x2:
[ 0.00  1.00 ]
[ 2.00  3.00 ]

Введите вторую матрицу:
Введите количество строк:
2
Введите количество столбцов:
2
Введите матрицу 2x2 построчно (в каждой строке 2 чисел, разделенных пробелами):
Пример строки: 1 2.5 3.14
Строка 1:
4 -5
Строка 2:
6 0
Вторая матрица:
Матрица 2x2:
[  4.00  -5.00 ]
[  6.00   0.00 ]

✅ Результат умножения:
Матрица 2x2:
[   6.00    0.00 ]
[  26.00  -10.00 ]

----------------------------------------
Нажмите Enter для продолжения...
```

```
============================================================
Калькулятор матриц v1.2
============================================================
1. Сложение матриц
2. Умножение матриц
3. Транспонирование матрицы
4. Вычисление определителя
5. Создать случайную матрицу
6. Примеры матриц (для тестирования)
7. Создать единичную матрицу
8. Умножение матрицы на скаляр
0. Выход
------------------------------------------------------------
Выберите операцию (или 'h' для справки):
3
=== Транспонирование матрицы ===
Введите матрицу:
Введите количество строк:
3
Введите количество столбцов:
3
Введите матрицу 3x3 построчно (в каждой строке 3 чисел, разделенных пробелами):
Пример строки: 1 2.5 3.14
Строка 1:
2 5 1
Строка 2:
0 -4 2
Строка 3:
3 -10 3

Исходная матрица:
Матрица 3x3:
[   2.00    5.00    1.00 ]
[   0.00   -4.00    2.00 ]
[   3.00  -10.00    3.00 ]

✅ Транспонированная матрица:
Матрица 3x3:
[   2.00    0.00    3.00 ]
[   5.00   -4.00  -10.00 ]
[   1.00    2.00    3.00 ]

--------------------------------------
Нажмите Enter для продолжения...
```

```
============================================================
Калькулятор матриц v1.2
============================================================
1. Сложение матриц
2. Умножение матриц
3. Транспонирование матрицы
4. Вычисление определителя
5. Создать случайную матрицу
6. Примеры матриц (для тестирования)
7. Создать единичную матрицу
8. Умножение матрицы на скаляр
0. Выход
------------------------------------------------------------
Выберите операцию (или 'h' для справки):
4
=== Вычисление определителя ===
Введите квадратную матрицу:
Введите количество строк:
2
Введите количество столбцов:
2
Введите матрицу 2x2 построчно (в каждой строке 2 чисел, разделенных пробелами):
Пример строки: 1 2.5 3.14
Строка 1:
2 4
Строка 2:
1 6

Исходная матрица:
Матрица 2x2:
[ 2.00  4.00 ]
[ 1.00  6.00 ]

✅ Определитель матрицы: 8.000000

----------------------------------------
Нажмите Enter для продолжения...
============================================================
Калькулятор матриц v1.2
============================================================
1. Сложение матриц
2. Умножение матриц
3. Транспонирование матрицы
4. Вычисление определителя
5. Создать случайную матрицу
6. Примеры матриц (для тестирования)
7. Создать единичную матрицу
8. Умножение матрицы на скаляр
0. Выход
------------------------------------------------------------
Выберите операцию (или 'h' для справки):
5
=== Создание случайной матрицы ===
=== Создание случайной матрицы ===
Введите количество строк:
3
Введите количество столбцов:
2

✅ Случайная матрица:
Матрица 3x2:
[ 3.76  5.84 ]
[ 8.85  6.59 ]
[ 8.55  9.15 ]

----------------------------------------
Нажмите Enter для продолжения...
```

```
============================================================
Калькулятор матриц v1.2
============================================================
1. Сложение матриц
2. Умножение матриц
3. Транспонирование матрицы
4. Вычисление определителя
5. Создать случайную матрицу
6. Примеры матриц (для тестирования)
7. Создать единичную матрицу
8. Умножение матрицы на скаляр
0. Выход
------------------------------------------------------------
Выберите операцию (или 'h' для справки):
6
=== Примеры матриц для тестирования ===

1. Матрица 2x2:
Матрица 2x2:
[ 1.00  2.00 ]
[ 3.00  4.00 ]

2. Матрица 3x3:
Матрица 3x3:
[ 1.00  2.00  3.00 ]
[ 4.00  5.00  6.00 ]
[ 7.00  8.00  9.00 ]

3. Единичная матрица 3x3:
Матрица 3x3:
[ 1.00  0.00  0.00 ]
[ 0.00  1.00  0.00 ]
[ 0.00  0.00  1.00 ]

Нажмите Enter для продолжения...
```

```
============================================================
Калькулятор матриц v1.2
============================================================
1. Сложение матриц
2. Умножение матриц
3. Транспонирование матрицы
4. Вычисление определителя
5. Создать случайную матрицу
6. Примеры матриц (для тестирования)
7. Создать единичную матрицу
8. Умножение матрицы на скаляр
0. Выход
------------------------------------------------------------
Выберите операцию (или 'h' для справки):
7
=== Создание единичной матрицы ===
Введите размер матрицы (n x n):
3
Создана единичная матрица:
Матрица 3x3:
[ 1.00  0.00  0.00 ]
[ 0.00  1.00  0.00 ]
[ 0.00  0.00  1.00 ]

----------------------------------------
Нажмите Enter для продолжения...
```

```
============================================================
Калькулятор матриц v1.2
============================================================
1. Сложение матриц
2. Умножение матриц
3. Транспонирование матрицы
4. Вычисление определителя
5. Создать случайную матрицу
6. Примеры матриц (для тестирования)
7. Создать единичную матрицу
8. Умножение матрицы на скаляр
0. Выход
------------------------------------------------------------
Выберите операцию (или 'h' для справки):
8
=== Умножение матрицы на скаляр ===
Введите матрицу:
Введите количество строк:
1
Введите количество столбцов:
4
Введите матрицу 1x4 построчно (в каждой строке 4 чисел, разделенных пробелами):
Пример строки: 1 2.5 3.14
Строка 1:
5 -9 3.7 2.4
Введенная матрица:
Матрица 1x4:
[  5.00  -9.00   3.70   2.40 ]
Введите скаляр (число):
4
Результат умножения на 4.00:
Матрица 1x4:
[  20.00  -36.00   14.80    9.60 ]

----------------------------------------
Нажмите Enter для продолжения...
```

```
============================================================
Калькулятор матриц v1.2
============================================================
1. Сложение матриц
2. Умножение матриц
3. Транспонирование матрицы
4. Вычисление определителя
5. Создать случайную матрицу
6. Примеры матриц (для тестирования)
7. Создать единичную матрицу
8. Умножение матрицы на скаляр
0. Выход
------------------------------------------------------------
Выберите операцию (или 'h' для справки):
h
=== Справка по вводу матриц ===
1. Сначала введите количество строк
2. Затем введите количество столбцов
3. Вводите матрицу построчно
4. В каждой строке введите числа через пробел

Пример ввода матрицы 2x3:
Количество строк: 2
Количество столбцов: 3
Строка 1: 1 2 3
Строка 2: 4 5 6

Нажмите Enter для продолжения...

============================================================
Калькулятор матриц v1.2
============================================================
1. Сложение матриц
2. Умножение матриц
3. Транспонирование матрицы
4. Вычисление определителя
5. Создать случайную матрицу
6. Примеры матриц (для тестирования)
7. Создать единичную матрицу
8. Умножение матрицы на скаляр
0. Выход
------------------------------------------------------------
Выберите операцию (или 'h' для справки):
0
До свидания! Спасибо за использование калькулятора.
```