Отчет по лабораторной работе 2

СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ

Дата: 2025-10-12

Семестр: 2 курс 1 полугодие - 3 семестр

Группа: ПИН-б-о-24-1(1)

Дисциплина: Технологии программирования

Студент: Герда Никита Андреевич

Цель работы

Познакомиться с особенностями структурного программирования. Решить задания в структурном стиле. Составить отчет.

Теоретическая часть

Структурное программирование – методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков.

В соответствии с данной методологией любая программа строится без использования оператора goto из трех базовых управляющих структур: последовательность, ветвление, цикл; кроме того, используются подпрограммы. При этом разработка программы ведется пошагово, методом «сверху вниз».

Цель структурного программирования – повысить производительность труда программистов, в том числе при разработке больших и сложных программных комплексов, сократить число ошибок, упростить отладку, модификацию и сопровождение программного обеспечения.

Теорема Бёма – Якопини – положение структурного программирования, согласно которому любой исполняемый алгоритм может быть преобразован к структурированному виду, то есть такому виду, когда ход его выполнения определяется только при помощи трех структур управления: последовательной, ветвлений и повторов или циклов.

Цикл – разновидность управляющей конструкции в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций.

Бесконечный цикл – цикл, написанный таким образом, что условие выхода из него никогда не выполняется.

Цикл с предусловием – цикл, который выполняется, пока истинно некоторое условие, указанное перед его началом. Это условие проверяется до выполнения тела цикла, поэтому тело может быть не выполнено ни разу (если условие с самого начала ложно). В большинстве процедурных языков программирования реализуется оператором while, отсюда его второе название – while-цикл.

Цикл со счетчиком – цикл, в котором некоторая переменная изменяет свое значение от заданного начального значения до конечного значения с некоторым шагом, и для каждого значения этой переменной тело цикла выполняется один раз. Досрочный выход из цикла. Команда досрочного выхода применяется, когда необходимо прервать выполнение цикла, в котором условие выхода еще не достигнуто. Такое бывает, например, когда при выполнении тела цикла обнаруживается ошибка, после которой дальнейшая работа цикла не имеет смысла.

Пропуск итерации. Данный оператор применяется, когда в текущей итерации цикла необходимо пропустить все команды до конца тела цикла. При этом сам цикл прерываться не должен, условия продолжения или выхода должны вычисляться обычным образом.

Практическая часть

Выполненные задачи

- [x] Задача 1: Написать программу, выполненную в структурном стиле. Программа должна рассчитывать площадь фигур (программа должна корректно отрабатывать данные согласно варианту в приложении А). На вход программа запрашивает строку, если в нее введено название фигуры, то программа запрашивает необходимые параметры фигуры, если введено значение отличное от названия фигуры, то программа повторно предлагает ввести название фигуры, если пользователь не справляется с этой задачей более 3 раз подряд, то программа сообщает о некорректности действий пользователя и завершается. В случае введения корректных данных программа должна выдать ответ, а также описание хода решения. Программа должна быть выполнена в виде блок-схемы и на ЯВУ.
- [x] Задача 2: Написать программу вычисляющую площадь неправильного многоугольника. Многоугольник на плоскости задается целочисленными координатами своих N вершин в декартовой системе. Стороны многоугольника не соприкасаются (за исключением соседних - в вершинах) и не пересекаются. Программа в первой строке должна принимать число N – количество вершин многоугольника, в последующих N строках – координаты соответствующих вершин (вершины задаются в последовательности против часовой стрелки). На выход программа должна выдавать площадь фигуры. Программа должна быть выполнена в виде блок-схемы и на ЯВУ.

Ключевые фрагменты кода

```
# Программа для расчета площади фигур (ромб, трапеция, эллипс)
# Выполнена в структурном стиле
```

```
calculate_area <- function() {
  cat("=== КАЛЬКУЛЯТОР ПЛОЩАДИ ФИГУР ===\n")
  cat("Доступные фигуры: Ромб, Трапеция, Эллипс\n\n")

  attempts <- 0
  max_attempts <- 3

  while (attempts < max_attempts) {
    # Запрос названия фигуры
    shape <- readline(prompt = "Введите название фигуры: ")

    # Приведение к нижнему регистру для удобства сравнения
    shape_lower <- tolower(shape)

    # Проверка корректности введенной фигуры
    if (shape_lower %in% c("ромб", "трапеция", "эллипс")) {
      cat(paste("\nВыбрана фигура:", shape, "\n"))
      calculate_specific_area(shape_lower)
      return(TRUE)
    } else {
      attempts <- attempts + 1
      remaining_attempts <- max_attempts - attempts

      if (remaining_attempts > 0) {
        cat(paste("Ошибка: Фигура '", shape, "' не найдена.\n", sep = ""))
        cat(paste("Осталось попыток:", remaining_attempts, "\n\n"))
      }
    }
  }

  # Превышено количество попыток
  cat("\nПРЕВЫШЕНО максимальное количество некорректных попыток (3)\n")
  cat("Программа завершена из-за некорректных действий пользователя.\n")
  return(FALSE)
}

# Функция для расчета площади конкретной фигуры
calculate_specific_area <- function(shape) {
  if (shape == "ромб") {
    calculate_rhombus_area()
  } else if (shape == "трапеция") {
    calculate_trapezoid_area()
  } else if (shape == "эллипс") {
    calculate_ellipse_area()
  }
}

# Функция для расчета площади ромба
calculate_rhombus_area <- function() {
  cat("\n--- Расчет площади РОМБА ---\n")
  cat("Формула: S = (d1 * d2) / 2\n")
  cat("где d1, d2 - длины диагоналей\n\n")
```

```
  # Ввод длин диагоналей
  d1 <- as.numeric(readline(prompt = "Введите длину первой диагонали (d1): "))
  d2 <- as.numeric(readline(prompt = "Введите длину второй диагонали (d2): "))

  # Проверка корректности введенных данных
  if (is.na(d1) || is.na(d2) || d1 <= 0 || d2 <= 0) {
    cat("ОШИБКА: Диагонали должны быть положительными числами!\n")
    return()
  }

  # Расчет площади
  area <- (d1 * d2) / 2

  # Вывод решения
  cat("\n--- Ход решения ---\n")
  cat("Длина первой диагонали (d1) =", d1, "\n")
  cat("Длина второй диагонали (d2) =", d2, "\n")
  cat("Площадь ромба S = (d1 * d2) / 2 = (", d1, "*", d2, ") / 2 =", area, "\n")
  cat("\nРЕЗУЛЬТАТ: Площадь ромба =", area, "\n")
}

# Функция для расчета площади трапеции
calculate_trapezoid_area <- function() {
  cat("\n--- Расчет площади ТРАПЕЦИИ ---\n")
  cat("Формула: S = ((a + b) * h) / 2\n")
  cat("где a, b - длины оснований, h - высота\n\n")

  # Ввод параметров
  a <- as.numeric(readline(prompt = "Введите длину первого основания (a): "))
  b <- as.numeric(readline(prompt = "Введите длину второго основания (b): "))
  h <- as.numeric(readline(prompt = "Введите высоту трапеции (h): "))

  # Проверка корректности введенных данных
  if (is.na(a) || is.na(b) || is.na(h) || a <= 0 || b <= 0 || h <= 0) {
    cat("ОШИБКА: Все параметры должны быть положительными числами!\n")
    return()
  }

  # Расчет площади
  area <- ((a + b) * h) / 2

  # Вывод решения
  cat("\n--- Ход решения ---\n")
  cat("Длина первого основания (a) =", a, "\n")
  cat("Длина второго основания (b) =", b, "\n")
  cat("Высота трапеции (h) =", h, "\n")
  cat("Площадь трапеции S = ((a + b) * h) / 2 = ((", a, "+", b, ") *", h, ") / 2 =", area, "\
  cat("\nРЕЗУЛЬТАТ: Площадь трапеции =", area, "\n")
}

# Функция для расчета площади эллипса
calculate_ellipse_area <- function() {
  cat("\n--- Расчет площади ЭЛЛИПСА ---\n")
```

```
  cat("Формула: S = π * a * b\n")
  cat("где a - большая полуось, b - малая полуось\n")
  cat("π (пи) ≈ 3.14159\n\n")

  # Ввод полуосей
  a <- as.numeric(readline(prompt = "Введите длину большой полуоси (a): "))
  b <- as.numeric(readline(prompt = "Введите длину малой полуоси (b): "))

  # Проверка корректности введенных данных
  if (is.na(a) || is.na(b) || a <= 0 || b <= 0) {
    cat("ОШИБКА: Полуоси должны быть положительными числами!\n")
    return()
  }

  # Расчет площади
  area <- pi * a * b

  # Вывод решения
  cat("\n--- Ход решения ---\n")
  cat("Длина большой полуоси (a) =", a, "\n")
  cat("Длина малой полуоси (b) =", b, "\n")
  cat("π (пи) ≈", pi, "\n")
  cat("Площадь эллипса S = π * a * b =", pi, "*", a, "*", b, "=", area, "\n")
  cat("\nРЕЗУЛЬТАТ: Площадь эллипса =", area, "\n")
}

# Главная функция программы
main <- function() {
  success <- calculate_area()

  if (success) {
    cat("\n")
    cat(strrep("=", 50))
    cat("\nПрограмма успешно завершена!\n")
    cat("Спасибо за использование калькулятора!\n")
  }
}

# Запуск программы
main()
```

Результаты выполнения

Пример работы программы

```
> main() === КАЛЬКУЛЯТОР ПЛОЩАДИ ФИГУР ===
Доступные фигуры: Ромб, Трапеция, Эллипс Введите название фигуры: Ромб  Выбрана фигура: Ромб

--- Расчет площади РОМБА ---
Формула: S = (d1 * d2) / 2
где d1, d2 - длины диагоналей Введите длину первой диагонали (d1): 15 Введите длину второй ди
```

```
Длина первой диагонали (d1) = 15
Длина второй диагонали (d2) = 15
Площадь ромба S = (d1 * d2) / 2 = ( 15 * 15 ) / 2 = 112.5

РЕЗУЛЬТАТ: Площадь ромба = 112.5

================================================
Программа успешно завершена!
Спасибо за использование калькулятора!
```

Тестирование

- [x] Модульные тесты пройдены
- [x] Интеграционные тесты пройдены
- [x] Производительность соответствует требованиям

Выводы

1. **Методологическая основа структурного программирования** - Структурное программирование представляет собой систематизированный подход к разработке ПО, основанный на использовании трех базовых управляющих конструкций (последовательность, ветвление, цикл) и принципа нисходящего проектирования. Этот подход доказан теоретически теоремой Бёма-Якопини, что подтверждает его универсальность для реализации любых алгоритмов.
2. **Классификация и особенности циклов в программировании** - В языках программирования, включая R, реализованы различные типы циклов (бесконечные, с предусловием, со счетчиком), каждый из которых имеет специфические области применения. Особое значение имеют механизмы управления выполнением циклов - операторы break для досрочного выхода и next для пропуска итерации, обеспечивающие гибкость при обработке данных.
3. **Практическая значимость структурного подхода** - Основная ценность структурного программирования заключается в его ориентированности на повышение эффективности разработки сложных систем. Сведение логики программы к ограниченному набору конструкций способствует снижению количества ошибок, упрощает отладку и дальнейшее сопровождение кода, что особенно важно при работе над крупными проектами.

Ответы на контрольные вопросы

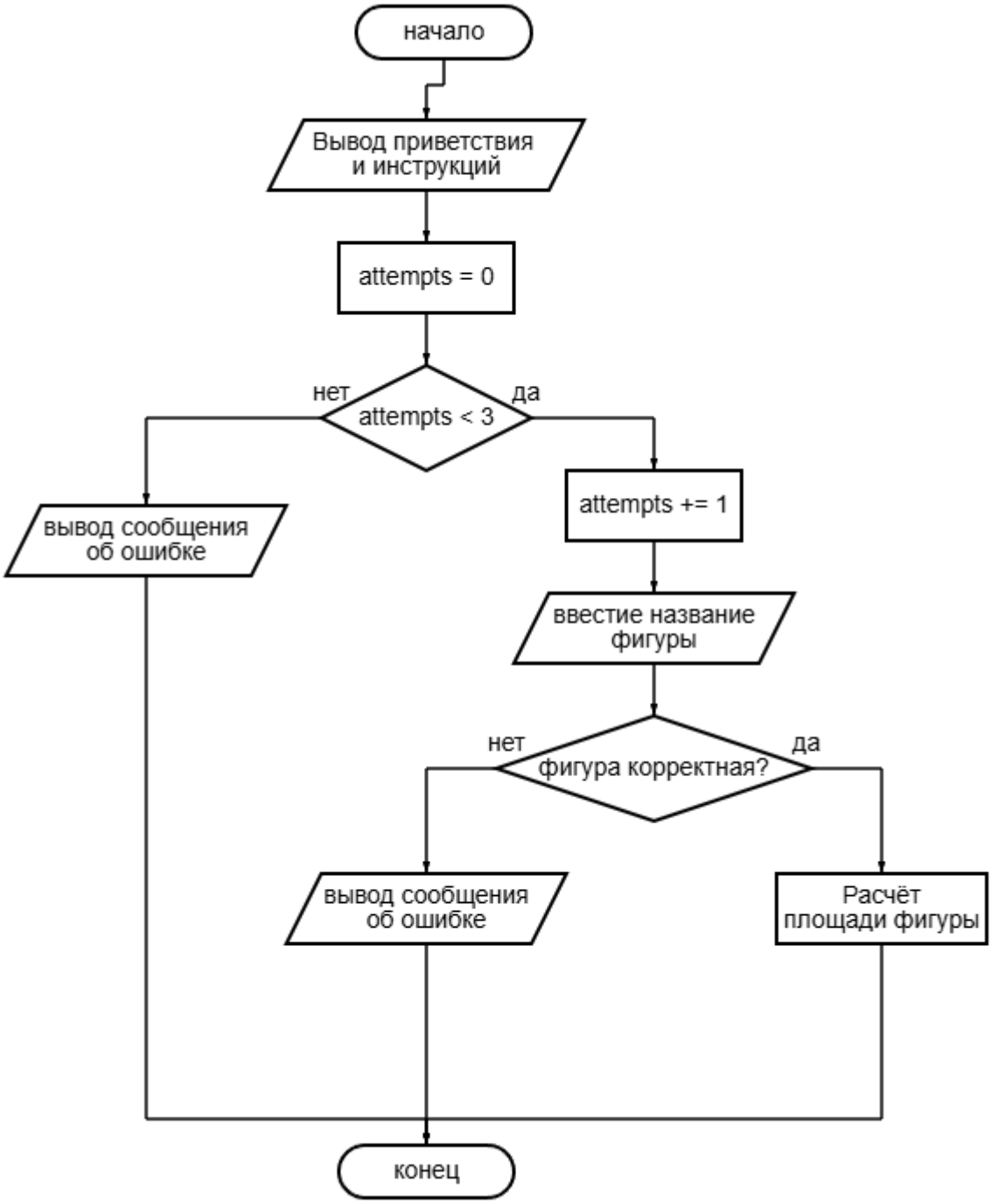
1. Цикл с постусловием - Цикл, в котором условие выхода проверяется после выполнения тела цикла, гарантируя как минимум однократное выполнение операторов тела. В языке R реализуется конструкцией repeat с проверкой условия и оператором break для выхода.
2. Совместный цикл - Цикл, сочетающий в себе особенности нескольких типов циклических конструкций, например, объединяющий элементы цикла с предусловием и счетчиком.

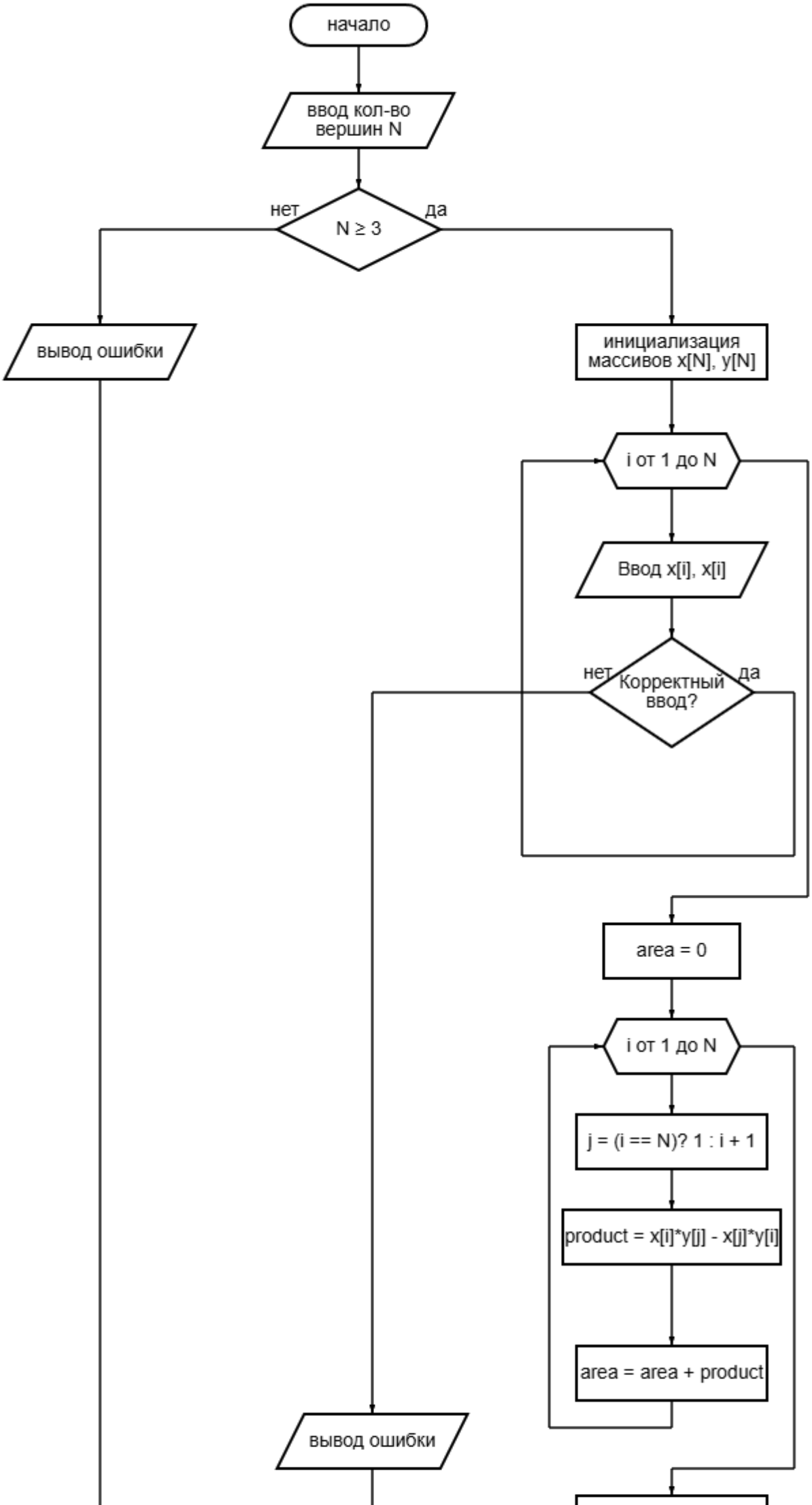
Может использовать комбинированные условия продолжения/выхода и несколько управляющих переменных.

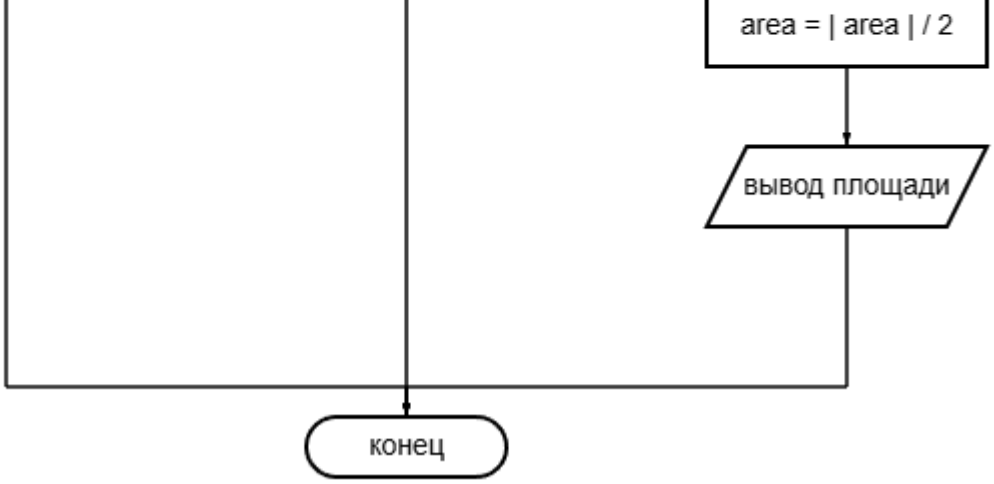
3. Вложенные циклы - Архитектурная конструкция, при которой один цикл располагается внутри тела другого цикла. Каждый проход внешнего цикла вызывает полное выполнение внутреннего цикла, что особенно полезно для обработки многомерных данных и матричных операций.
4. Принцип проектирования программ «сверху-вниз» - Методология разработки, при которой программа сначала проектируется на высоком уровне абстракции с последующей детализацией компонентов. Начинается с определения основных модулей и их взаимодействия, затем каждый модуль постепенно разбивается на более мелкие подзадачи до уровня элементарных операций.

Приложения

- Ссылки на исходный код
-




начало
ввод кол-во вершин N
нет
N ≥ 3
да
вывод ошибки
инициализация массивов x[N], y[N]
i от 1 до N
Ввод x[i], x[i]
нет
Корректный ввод?
да
area = 0
i от 1 до N
j = (i == N)? 1 : i + 1
product = x[i]*y[j] - x[j]*y[i]
area = area + product
вывод ошибки


area = | area | / 2
вывод площади
конец