Отчет по лабораторным работам

Дата: 03-12-2025
Семестр: 2 курс, 1 семестр
Группа: ПИН-Б-О-24-2
Дисциплина: Технологии программирования **Студент:** Осипов Александр Сергеевич

Цель работы

Изучить различные парадигмы программирования на языке R: 1. Императивное (процедурное) программирование 2. Структурное программирование 3. Объектно-ориентированное программирование 4. Векторное программирование 5. Функциональное программирование 6. Грамотное программирование 7. Параллельное программирование 8. Визуальное программирование

Теоретическая часть

Лабораторная работа №1: Императивное (процедурное) программирование

Процедурное программирование - парадигма, в которой выполнение программы сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти в заключительное состояние.

Основные особенности: - Последовательное выполнение операторов - Использование переменных для хранения данных - Организация кода в процедуры и функции - Явное управление памятью - Декомпозиция задач на шаги

Основные понятия: - **Переменная** - состоит из имени и выделенной области памяти - **Функция** - подпрограмма, принимающая параметры и возвращающая

результат - **Процедура** - именованная часть программы для выполнения действий

Лабораторная работа №2: Структурное программирование

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

Основные принципы: - Три базовые управляющие структуры: последовательность, ветвление, цикл - Отказ от оператора goto - Использование подпрограмм - Метод разработки "сверху вниз"

Теорема Бёма - Якопини: Любой исполняемый алгоритм может быть преобразован к структурированному виду, используя только три структуры управления.

Лабораторная работа №3: Объектно-ориентированное программирование

ООП - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса.

Механизмы ООП: - **Абстракция** - определение характеристик объекта - **Инкапсуляция** - сокрытие внутренних деталей - **Наследование** - создание специализированных классов - **Полиморфизм** - одна операция в разных объектах - **Композиция** - объекты включают другие объекты

R6 - пакет для создания классов в R с поддержкой приватных и публичных членов.

Лабораторная работа №4: Векторное программирование

Векторизация - поэлементное одновременное выполнение действий над всеми элементами вектора или массива.

Основные объекты R: - **Вектор** - основной объект, содержит однотипные данные - **Лист (list)** - может содержать разные по размеру и типу векторы - **Дата фрейм (data.frame)** - таблица данных - **Матрица (matrix)** - двумерная совокупность данных - **Массив (array)** - многомерная совокупность данных

Функции семейства apply: apply(), lapply(), sapply(), tapply(), mapply()

Лабораторная работа №5: Функциональное программирование

Функциональное программирование - парадигма, где вычисления рассматриваются как вычисление значений функций.

Основные концепции: - **Итерации** - через функции высшего порядка (map, reduce) - **Конвейеры (pipeline)** - оператор `%>%` для передачи значений между функциями - **Пакет purrr** - набор функций для функционального программирования

Функции семейства map_*: map(), map_lgl(), map_int(), map_dbl(), map_chr(), map_dfr(), map_dfc(), map_if(), map_at(), walk()

Лабораторная работа №6: Грамотное программирование

Грамотное программирование (Literate Programming) - методология, предложенная Дональдом Кнутом, где код и документация объединены в единый документ.

R Markdown - технология для создания динамических документов, объединяющих: - YAML заголовок - метаданные документа - Markdown текст - форматированный текст - Code chunks - блоки кода R - Inline code - встроенный код в тексте

Лабораторная работа №7: Параллельное программирование

Параллельное программирование - методология, позволяющая выполнять несколько задач одновременно на разных процессорах или ядрах.

Методы разделения задачи: - Разделение на задачи - каждая задача выполняется независимо - Разделение по данным - одна задача разбивается на части

Модели памяти: - Общая память (Shared Memory) - Разделенная память (Distributed Memory)

Пакет parallel - стандартный пакет R для параллельных вычислений.

Лабораторная работа №8: Визуальное программирование

Визуальное программирование - парадигма, где программы создаются путем манипулирования графическими элементами.

Scratch - визуальный язык программирования для обучения.

Основные концепции: - Спрайты - объекты, выполняющие действия - Скрипты - блоки команд - Сцены (фоны) - фоновые изображения - События - триггеры для запуска скриптов - Переменные - хранилища данных - Сообщения - способ взаимодействия между спрайтами

Практическая часть

Выполненные задачи

Лабораторная работа №1

- Задача 1: Написать программу расчета площадей трех фигур (квадрат, прямоугольник, круг) - выполнено
- Задача 2: Разобрать ассемблерный код и реализовать алгоритм факториала на R - выполнено

Лабораторная работа №2

- Задача 1: Программа расчета площади фигур с проверкой ввода - выполнено
- Задача 2: Программа расчета площади многоугольника по формуле Гаусса - выполнено

Лабораторная работа №3

- Задача 1: Создание дженерика для вычисления площади фигур - выполнено
- Задача 2: Создание класса Микроволновая печь (R6) - выполнено
- Задача 3: Создание класса Копилка (R6) - выполнено

Лабораторная работа №4

- Задача 1: Предобработка данных - отбор наблюдений - выполнено
- Задача 2: Функция поиска отрицательных значений в dataframe - выполнено

Лабораторная работа №5

- Задача 1: Создание именованного списка из данных repurrrsive - выполнено
- Задача 2: Описание функций семейства map_* с примерами - выполнено

Лабораторная работа №6

- Задача 1: Создание динамического документа R Markdown - выполнено

Лабораторная работа №7

- Задача 1: Визуализация слов из книг Джейн Остин - выполнено
- Задача 2: Распараллеливание кода с использованием кластера - выполнено

Лабораторная работа №8

- Задача 1: Создание игры "Космический защитник" в Scratch - выполнено

Ключевые фрагменты кода

Лабораторная работа №1: Процедурное программирование

Задание 1: Расчет площадей фигур

```
 # Запрос данных о квадрате
square_side <- as.numeric(readline("Введите сторону квадрата: "))
square_area <- square_side * square_side

# Запрос данных о круге
circle_radius <- as.numeric(readline("Введите радиус круга: "))
circle_area <- pi * circle_radius * circle_radius

# Запрос данных о прямоугольнике
rectangle_side1 <- as.numeric(readline("Введите первую сторону прямоугольника:
"))
rectangle_side2 <- as.numeric(readline("Введите вторую сторону прямоугольника:
"))
rectangle_area <- rectangle_side1 * rectangle_side2

# Вычисление общей площади
total_area <- square_area + circle_area + rectangle_area
```

```
# Вывод результатов
cat("Площадь квадрата:", square_area, "\n")
cat("Площадь круга:", circle_area, "\n")
cat("Площадь прямоугольника:", rectangle_area, "\n")
cat("Общая площадь:", total_area, "\n")
```

Задание 2: Функция факториала

```
 foo <- function(n) {
  if (n > 0) {
    result <- foo(n - 1)
    return(n * result)
  } else {
    return(1)
  }
}

# Проверка: foo(7) = 5040
```

Лабораторная работа №2: Структурное программирование

Задание 1: Программа с проверкой ввода

```
 valid_figures <- c("треугольник", "квадрат", "прямоугольник", "параллелограмм")
attempts <- 0
max_attempts <- 3
figure_found <- FALSE

while (!figure_found && attempts < max_attempts) {
  figure_name <- tolower(readline("Введите название фигуры: "))

  if (figure_name %in% valid_figures) {
    figure_found <- TRUE
    # Обработка фигуры...
  } else {
    attempts <- attempts + 1
    if (attempts >= max_attempts) {
      cat("Некорректность действий пользователя.\n")
    }
  }
}
```

Задание 2: Площадь многоугольника (формула Гаусса)

```
 # Вычисление площади по формуле шнурков Гаусса
sum_area <- 0
for (i in 1:N) {
  next_i <- ifelse(i == N, 1, i + 1)
  sum_area <- sum_area + (x[i] * y[next_i] - x[next_i] * y[i])
}
area <- abs(sum_area) / 2
```

Лабораторная работа №3: Объектно-ориентированное программирование

Задание 1: Дженерик для площади фигур

```
 get_area <- function(x, ...) {
  UseMethod("get_area")
}

get_area.default <- function(x, ...) {
  return("Невозможно обработать данные.")
}

# Класс для треугольника
triangle_vec <- c(10, 5)
class(triangle_vec) <- "triangle"
get_area.triangle <- function(x, ...) {
  return(0.5 * x[1] * x[2])
}
```

Задание 2: Класс Микроволновая печь

```
 library(R6)

MicrowaveOven <- R6Class(
  "MicrowaveOven",
  private = list(
    power = 800,
    door_open = FALSE
  ),
  public = list(
    initialize = function(power = 800, door_open = FALSE) {
      private$power <- power
      private$door_open <- door_open
```

```
    },
    cook = function() {
      if (private$door_open) {
        cat("ОШИБКА: Дверца открыта!\n")
        return(invisible(NULL))
      }
      cooking_time <- 60 * (800 / private$power)
      Sys.sleep(cooking_time)
      cat("Пища готова!\n")
    }
  )
)
```

Лабораторная работа №4: Векторное программирование

Задание 1: Предобработка данных

```
 my_vector <- c(21, 18, 21, 19, 25, 20, 17, 17, 18, 22, ...)
mean_value <- mean(my_vector)
sd_value <- sd(my_vector)
my_vector2 <- my_vector[abs(my_vector - mean_value) < sd_value]
```

Задание 2: Функция get_negative_values

```
get_negative_values <- function(df) {
  negative_list <- list()
  for (col_name in names(df)) {
    column <- df[[col_name]]
    negative_values <- column[!is.na(column) & column < 0]
    if (length(negative_values) > 0) {
      negative_list[[col_name]] <- negative_values
    }
  }
  # Преобразование в матрицу, если возможно
  lengths <- sapply(negative_list, length)
  if (length(unique(lengths)) == 1 && length(negative_list) > 1) {
    return(as.matrix(as.data.frame(negative_list)))
  }
  return(negative_list)
}
```

Лабораторная работа №5: Функциональное программирование

Задание 1: Именованный список

```
 library(purrr)
library(repurrrsive)

data(sw_films, package = "repurrrsive")
film_names <- map_chr(sw_films, ~ .x$title)
named_films_list <- sw_films %>% set_names(film_names)
```

Задание 2: Функции map_*

```
 # Пример с map_dbl
iris_list <- as.list(iris[1:4])
iris_means <- map_dbl(iris_list, ~ mean(.x))

# Пример с map_if
mixed_list <- list(1:3, "text", 4:6, "another")
result <- map_if(mixed_list, is.numeric, ~ sum(.x))
```

Лабораторная работа №6: Грамотное программирование

R Markdown документ:

````
 ---
title: "Лабораторная работа №6"
output: html_document
---

## Данные

Среднее значение: `r mean(x)`

```{r}
hist(data, main = "Распределение данных")
```
````

````
#### Лабораторная работа №7: Параллельное программирование

**Задание 1: Визуализация слов**

```r
library(janeaustenr)
````
````

```
library(stringr)

words_vector <- janeausten_words()
max_freq_words <- sapply(letters, function(letter) {
  max_frequency(letter, words = words_vector, min_length = 5)
})
barplot(max_freq_words, las = 2)
```

Задание 2: Параллельное выполнение

```
 library(parallel)

ncores <- detectCores(logical = FALSE)
cl <- makeCluster(ncores)
clusterExport(cl, "mean_of_rnorm")

result_par <- parSapply(cl, seq_len(50), function(iter) {
  mean_of_rnorm(10000)
})

stopCluster(cl)
```

Лабораторная работа №8: Визуальное программирование

Игра "Космический защитник" в Scratch:

- Спрайт 1: Космический корабль (управление стрелками, стрельба пробелом)
- Спрайт 2: Астероид (падает сверху независимо от игрока)
- Спрайт 3: Пуля (создается при действиях игрока, уничтожает астероиды)
- Система уровней: 2 уровня с разными фонами и скоростями

Результаты выполнения

Пример работы программ

Лабораторная работа №1

```
 Введите сторону квадрата: 5
Введите радиус круга: 3
```

```
Введите первую сторону прямоугольника: 4
Введите вторую сторону прямоугольника: 6
Площадь квадрата: 25
Площадь круга: 28.27433
Площадь прямоугольника: 24
Общая площадь: 77.27433

foo(7) = 5040
```

Лабораторная работа №2

```
 Введите название фигуры: квадрат
Введите сторону квадрата: 5
=== РЕЗУЛЬТАТ ===
Площадь квадрата: 25

Введите количество вершин многоугольника: 4
Площадь многоугольника: 12
```

Лабораторная работа №3

```
 Треугольник (основание=10, высота=5): Площадь = 25
Квадрат (сторона=5): Площадь = 25
Круг (радиус=4): Площадь = 50.26548

Микроволновая печь создана. Мощность: 800 Вт
Приготовление на мощности 800 Вт: Время: 60 сек
Пища готова!
```

Лабораторная работа №4

```
 Исходный вектор: 49 элементов
Среднее: 19.41, Стандартное отклонение: 3.69
Новый вектор: 30 элементов

get_negative_values(test_data)
$V1: -9.7 -10.0 -10.5 -7.8 -8.9
$V2: -10.2 -10.1 -9.3 -12.2
```

Лабораторная работа №5

```
 Именованный список создан: 7 элементов
Доступ к фильму 'A New Hope':
Режиссер: George Lucas

iris_means:
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     3.758000     1.199333
```

Лабораторная работа №6

R Markdown документ успешно скомпилирован в HTML с динамическими вычислениями, визуализациями и примененными CSS стилями.

Лабораторная работа №7

```
 Количество физических ядер: 4
Время выполнения (последовательно): 2.5 сек
Время выполнения (параллельно): 0.8 сек
Ускорение: 3.125x
```

Создан график наиболее часто встречающихся слов из книг Джейн Остин по буквам алфавита.

Лабораторная работа №8

Игра "Космический защитник" создана в Scratch с тремя спрайтами, системой уровней и всеми требуемыми функциями.

Тестирование

Лабораторная работа №1

- Программа корректно вычисляет площади фигур - выполнено
- Функция факториала работает правильно для всех тестовых значений - выполнено
- Код соответствует процедурному стилю программирования - выполнено

Лабораторная работа №2

- Программа корректно обрабатывает ввод названий фигур - выполнено
- Программа ограничивает количество попыток ввода - выполнено
- Программа правильно вычисляет площади фигур - выполнено
- Программа корректно вычисляет площадь многоугольника по формуле Гаусса - выполнено

Лабораторная работа №3

- Дженерик корректно обрабатывает разные классы фигур - выполнено
- Класс Микроволновая печь работает с разными мощностями - выполнено
- Класс Копилка корректно управляет балансом и состоянием - выполнено

Лабораторная работа №4

- Векторизованные операции работают корректно - выполнено
- Функция правильно обрабатывает dataframe с отрицательными значениями - выполнено
- Функция возвращает список или матрицу в зависимости от размеров - выполнено

Лабораторная работа №5

- Именованный список создан корректно - выполнено
- Доступ к элементам работает по имени и индексу - выполнено
- Все функции семейства map_* протестированы - выполнено

Лабораторная работа №6

- Документ корректно компилируется в HTML - выполнено
- Inline код правильно подставляет результаты - выполнено
- Code chunks выполняются и выводят результаты - выполнено
- CSS стили применяются корректно - выполнено

Лабораторная работа №7

- Визуализация слов создана корректно - выполнено
- Параллельное выполнение ускоряет вычисления - выполнено
- Результаты последовательного и параллельного выполнения совпадают - выполнено

Лабораторная работа №8

- Игра создана в Scratch - выполнено
- Все три спрайта имеют прописанную логику - выполнено
- Система уровней реализована через смену фонов - выполнено
- Управление работает корректно - выполнено

Выводы

Общие выводы по всем лабораторным работам

1. **Многообразие парадигм программирования в R:** - R поддерживает различные парадигмы программирования: процедурную, структурную, объектно-ориентированную, функциональную, векторную - Каждая парадигма имеет свои преимущества и области применения - Выбор парадигмы зависит от конкретной задачи

2. **Процедурное и структурное программирование:** - Процедурный стиль обеспечивает простоту понимания и отладки - Структурное программирование повышает читаемость и поддерживаемость кода - Отказ от оператора goto улучшает структуру программы

3. **Объектно-ориентированное программирование:** - R6 предоставляет удобный способ создания классов с инкапсуляцией - Дженерики обеспечивают полиморфизм в R - ООП позволяет инкапсулировать данные и методы в классах

4. **Векторное программирование:** - Векторизация значительно ускоряет вычисления в R - Векторные операции более читаемы и компактны, чем циклы - Функции семейства apply упрощают работу с данными

5. **Функциональное программирование:** - Функциональный подход упрощает работу с коллекциями данных - Конвейеры улучшают читаемость кода - Пакет purrr расширяет возможности функционального программирования

6. **Грамотное программирование:** - R Markdown позволяет создавать воспроизводимые отчеты - Динамические документы автоматически обновляются при изменении данных - Грамотное программирование улучшает читаемость и поддерживаемость кода

7. **Параллельное программирование:** - Параллельное программирование значительно ускоряет вычисления - Пакет parallel предоставляет удобный интерфейс для создания кластеров - Распараллеливание эффективно для независимых задач

8. **Визуальное программирование:** - Визуальное программирование упрощает создание программ для начинающих - Scratch предоставляет интуитивный интерфейс для создания игр - Визуальная структура облегчает понимание логики программы

Ответы на контрольные вопросы

Лабораторная работа №1

1. **Особенности процедурного программирования:** - Последовательное выполнение операторов - Использование переменных для хранения данных - Организация кода в процедуры и функции - Явное управление памятью - Декомпозиция задач на шаги

2. **Линейная программа:** - Программа, в которой команды выполняются последовательно, без условных переходов и циклов - Все команды выполняются строго в порядке их написания

3. **Понятия: переменная, процедура, функция:** - **Переменная** - состоит из имени и выделенной области памяти - **Процедура** - именованная часть программы, не возвращающая значение - **Функция** - подпрограмма, принимающая параметры и возвращающая результат

4. **Безусловный оператор:** - Оператор перехода (goto, jmp), позволяющий перейти к метке без проверки условий - В структурном программировании не рекомендуется

Лабораторная работа №2

1. **Особенности структурного программирования:** - Иерархическая структура блоков - Три базовые структуры: последовательность, ветвление, цикл - Отказ от оператора goto - Использование подпрограмм - Метод разработки "сверху вниз"

2. **Теорема Бёма - Якопини:** - Любой исполняемый алгоритм может быть преобразован к структурированному виду - Используя только три структуры управления: последовательную, ветвлений и повторов

3. **Пропуск итерации и досрочный выход из цикла:** - **break** - досрочный выход из цикла - **next** - пропуск текущей итерации цикла

Лабораторная работа №3

1. **Принципы ООП по Алану Кею:** - Все является объектом - Вычисления через взаимодействие объектов - Взаимодействие через сообщения - Независимая память объектов - Объекты - экземпляры классов - Иерархия наследования - Разделение ответственности

2. **Механизмы ООП:** - Абстракция, Инкапсуляция, Наследование, Полиморфизм, Композиция

3. **Основные понятия ООП:** - Объект, Класс, Атрибут класса, Методы класса

4. **Создание и назначение дженериков:** - Дженерик - функция, способная принимать разные структуры данных - Создается через UseMethod()

5. **Создание класса в R6:** - Используется функция R6Class() с параметрами: имя класса, списки private, public, active

Лабораторная работа №4

1. **Векторизация:** - Поэлементное одновременное выполнение действий над всеми элементами - Значительно ускоряет вычисления

2. **Основные объекты языка R:** - Вектор, Лист, Дата фрейм, Матрица, Массив

3. **Создание собственных функций:** - `function_name <- function(arguments) { body; return(value) }`

4. **Векторизованные функции семейства apply:** - apply(), lapply(), sapply(), tapply(), mapply(), vapply()

Лабораторная работа №5

1. **Итерации в функциональном программировании:** - Реализуются через функции высшего порядка (map, reduce) - Более декларативный подход,

чем явные циклы

2. **Конвейеры (pipeline):** - Инструмент для передачи значения между функциями - В R используется оператор `%>%` - Улучшает читаемость кода

3. **Функции семейства map в пакете purrr:** - map(), map_lgl(), map_int(), map_dbl(), map_chr(), map_dfr(), map_dfc(), map_if(), map_at(), walk() - Все применяют функцию к каждому элементу коллекции, различаются типом возвращаемого значения

Лабораторная работа №6

1. **Грамотное программирование (Literate Programming):** - Методология, где код и документация объединены в единый документ - Улучшает читаемость и поддерживаемость кода

2. **R Markdown:** - Технология для создания динамических документов - Поддерживает различные форматы вывода (HTML, PDF, Word)

3. **Структура R Markdown документа:** - YAML заголовок, Markdown текст, Code chunks, Inline code

4. **CSS стилизация в R Markdown:** - CSS позволяет настраивать внешний вид HTML документа - Файл CSS указывается в YAML заголовке

Лабораторная работа №7

1. **Параллельное программирование:** - Методология, позволяющая выполнять несколько задач одновременно - Цели: ускорение выполнения, эффективное использование многоядерных процессоров

2. **Методы разделения задачи:** - Разделение на задачи - каждая задача выполняется независимо - Разделение по данным - одна задача разбивается на части

3. **Модели памяти:** - Общая память - все процессы имеют доступ к одной области памяти - Разделенная память - каждый процесс имеет свою память

4. **Модели параллельного программирования:** - Master-worker - один главный процесс управляет рабочими - Map-reduce - предварительная обработка и свертка данных

5. **Пакет parallel в R:** - Основные функции: detectCores(), makeCluster(), clusterApply(), parSapply(), clusterExport(), stopCluster()

Лабораторная работа №8

1. **Визуальное программирование:** - Парадигма, где программы создаются путем манипулирования графическими элементами - Преимущества: легкость изучения, визуальная структура - Недостатки: ограниченная выразительность, зависимость от редактора

2. **Scratch:** - Визуальный язык программирования для обучения - Основные концепции: спрайты, скрипты, сцены, события, переменные, сообщения

3. **Структура программы в Scratch:** - Блоки команд: Движение, Внешний вид, Звук, События, Управление, Сенсоры, Операторы, Переменные - Структура: событие → команды → условия → циклы

4. **Применение визуального программирования:** - Образование - обучение программированию детей - Прототипирование - быстрое создание прототипов - Специализированные области - робототехника, автоматизация

Структура проекта

```
.
├── lab01/              # Императивное (процедурное) программирование
│   ├── project/
│   │   ├── lab01_zadanie1.R
│   │   └── lab01_zadanie2.R
│   ├── report/
│   ├── report.md
│   └── task.md
├── lab02/              # Структурное программирование
│   ├── project/
│   │   ├── zadanie1.R
│   │   └── zadanie2.R
│   ├── report/
│   ├── report.md
│   └── task.md
├── lab03/              # Объектно-ориентированное программирование
│   ├── project/
│   │   ├── zadanie1.R
│   │   ├── zadanie2.R
```

```
│   │   └── zadanie3.R
│   ├── report/
│   ├── report.md
│   └── task.md
├── lab04/              # Векторное программирование
│   ├── project/
│   │   ├── zadanie1.R
│   │   └── zadanie2.R
│   ├── report/
│   ├── report.md
│   └── task.md
├── lab05/              # Функциональное программирование
│   ├── project/
│   │   ├── zadanie1.R
│   │   └── zadanie2.R
│   ├── report/
│   ├── report.md
│   └── task.md
├── lab06/              # Грамотное программирование
│   ├── project/
│   │   ├── report.Rmd
│   │   └── style.css
│   ├── report/
│   ├── report.md
│   └── task.md
├── lab07/              # Параллельное программирование
│   ├── project/
│   │   ├── zadanie1.R
│   │   └── zadanie2.R
│   ├── report/
│   ├── report.md
│   └── task.md
├── lab08/              # Визуальное программирование
│   ├── project/
│   │   └── zadanie1.md
│   ├── report/
│   ├── report.md
│   └── task.md
├── README.md
├── ОТЧЕТ.md            # Данный файл
```