МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный лесотехнический университет»
(УГЛТУ)

Д. Д. Стратонов

# ОСНОВЫ R И ПОБЕГ ИЗ ЦИФРОВОГО ЛАБИРИНТА

Учебно-методическое пособие

Екатеринбург
УГЛТУ
2025

УДК 04.43(075.8)
ББК 32.973.22я73
С83

Рецензенты:

кафедра информационных систем и технологий Института инженерно-педагогического образования ФГАОУ ВО «Российский государственный профессионально-педагогический университет», доцент, канд. пед. наук *А. А. Шайдуров*;

*Н. В. Ломовцева*, канд. пед. наук, доцент, проректор по образовательной деятельности и цифровизации, ФГБОУ ВО «Уральский государственный аграрный университет»

**Стратонов, Дмитрий Дмитриевич.**

С83 Основы R и побег из цифрового лабиринта : учебно-методическое пособие / Д. Д. Стратонов ; Министерство науки и высшего образования Российской Федерации, Уральский государственный лесотехнический университет. – Екатеринбург : УГЛТУ, 2025. – 107 с.

ISBN 978-5-94984-945-3

Учебно-методическое пособие представляет собой вводный курс по изучению языка программирования R, интегрированный в увлекательное повествование. Книга предназначена для детей и подростков, желающих освоить основы программирования и анализа данных. В форме захватывающей истории о путешествии героев по цифровому лабиринту излагаются фундаментальные понятия языка R: переменные, типы данных, операторы, функции, структуры данных (векторы, списки, матрицы).

Каждый раздел книги содержит «R-практикум» с примерами кода, упражнениями и заданиями для самостоятельной работы, что позволяет читателю не только усвоить теоретический материал, но и применить его на практике. Пособие способствует развитию логического мышления, алгоритмических навыков и творческого подхода к решению задач. Книга может быть использована как для самостоятельного изучения R, так и в качестве дополнительного материала на уроках информатики.

Предназначено для обучающихся, осваивающих образовательные программы по направлению «Прикладная информатика» всех форм обучения.

Издается по решению редакционно-издательского совета Уральского государственного лесотехнического университета.

УДК 004.43(075.8)
ББК 32.973.22я73

ISBN 978-5-94984-945-3

# ОГЛАВЛЕНИЕ

# ДОБРО ПОЖАЛОВАТЬ В МИР R!

Ты когда-нибудь мечтал о путешествиях в другие миры, полные невероятных приключений и загадок? Алиса, героиня этой книги, тоже об этом мечтала. И ее мечта сбылась, когда она обнаружила у себя дома странный R-терминал – устройство, способное открыть дверь в цифровой лабиринт, где реальность соткана из кода.

Но это путешествие оказалось не просто увлекательной прогулкой. Цифровой лабиринт полон опасностей и ловушек, а хаос, проникший в его код, грозит разрушить все миры мультивселенной. Чтобы спасти себя и других, Алисе предстоит освоить мощный язык программирования R и научиться изменять реальность с его помощью.

В этом ей помогут верный друг Кай – цифровой эльф, а также мудрый наставник Элрос – хранитель кода. Вместе они пройдут через множество испытаний, решат сложные головоломки и раскроют тайны цифрового мира.

Эта книга для тебя, если ты:

- мечтаешь научиться программировать и создавать свои собственные приложения;
- любишь фантастику, приключения и головоломки;
- хочешь познать секреты языка R и его безграничные возможности.

В этой книге ты научишься:

- основам программирования на R: переменным, типам данных, операторам, функциям;
- работать с основными структурами данных в R: векторами, списками, матрицами;
- использовать R для решения задач и анализа данных;
- мыслить логически и находить нестандартные решения;
- но главное – ты поймешь, что R – это не просто язык программирования, а ключ к пониманию и изменению мира вокруг нас.

Приготовься к увлекательному путешествию в мир R!

# *Глава 1*
# ВХОД В МАТРИЦУ

Алиса раздраженно отшвырнула клавиатуру.

– Ну почему ничего не получается?! – пробормотала она, глядя на экран монитора, где красными буквами горело сообщение об ошибке. Уже битый час она пыталась написать простенькую программу на Python, но код упорно не хотел работать.

– Может, ну его, это программирование? – подумала Алиса, готовясь закрыть ноутбук. В этот момент комната наполнилась странным гулом, а воздух заискрился зелеными и синими огоньками. Девушка в испуге отпрянула от стола, наблюдая, как прямо посреди комнаты материализуется высокая фигура в длинном плаще.

– Не бойся, Алиса, – сказала фигура механическим голосом. – Меня зовут Элрос.

Алиса с трудом проглотила комок в горле. Перед ней стоял… киборг? Или робот? Она не могла точно определить. Его тело было покрыто металлическими пластинами, из-под которых местами проглядывала органическая ткань. Глаза светились ярко-голубым светом, а голос звучал так, будто его производил синтезатор речи.

– Элрос? – переспросила Алиса, все еще не веря своим глазам. – А вы… кто?

– Я – хранитель кода, – ответил Элрос. – И я пришел за тобой.

– За мной? Зачем? – Алиса начала чувствовать себя героиней фантастического фильма.

– Тебе предстоит важная миссия, Алиса, – сказал Элрос. – Ты должна помочь мне спасти мультивселенную.

– Мультивселенную? – Алиса уже ничему не удивлялась. – А что с ней случилось?

– Хаос проник в код реальности, – объяснил Элрос. – Миры искажаются, законы физики нарушаются, а вселенная находится на грани разрушения. Только мы можем это остановить.

– Мы? – Алиса указала на себя. – Но я же просто… школьница.

– Ты – избранная, Алиса, – сказал Элрос. – В тебе есть потенциал, который нужно развить. Я научу тебя использовать великую силу – язык R.

– R? – Алиса припомнила скучные уроки информатики. – Это же… язык программирования?

– Не просто язык, – поправил ее Элрос. – Это ключ к пониманию и управлению реальностью. С его помощью мы сможем исправить ошибки в коде и восстановить порядок в мультивселенной.

– Звучит… захватывающе, – протянула Алиса. – Но я не уверена, что справлюсь.

– Я буду рядом, – успокоил ее Элрос. – А теперь… нам пора.

Он щелкнул пальцами, и комната вновь наполнилась гулом и вспышками света. Когда все стихло, Алиса обнаружила, что они находятся в совершенно другом месте.

Вокруг тянулись бесконечные коридоры, стены которых состояли из строчек кода. Над головой висели гигантские мониторы, на которых мелькали цифры и символы. В воздухе парили странные существа, похожие на эльфов, но с прозрачными телами, сквозь которые проглядывали схемы и алгоритмы.

– Добро пожаловать в цифровой лабиринт, – сказал Элрос. – Здесь начинается твое обучение.

В этот момент из-за угла выглянул юноша с бледной кожей и серебристыми волосами. Он улыбнулся Алисе и сказал:

– Привет! Я Кай. Похоже, тебе нужна помощь?

– Кай? – Алиса с интересом разглядывала незнакомца. – А ты кто?

– Я – цифровой эльф, – ответил Кай. – Я помогу тебе освоиться в этом мире.

– Цифровой эльф? – Алиса уже ничему не удивлялась. – А что это такое?

Прежде чем Кай успел ответить, Элрос вновь щелкнул пальцами и… исчез.

– Элрос? – Алиса оглянулась по сторонам. – Куда он делся?

– Не волнуйся, он скоро вернется, – успокоил ее Кай. – А пока… давай я тебе все расскажу.

# R-ПРАКТИКУМ

**R** – это язык программирования и свободная программная среда для статистических вычислений и графики. Он был создан в начале 1990-х годов Россом Ихакой и Робертом Джентльменом в Оклендском университете (Новая Зеландия). R является реализацией языка S, разработанного в Bell Labs.

**Зачем нужен R?**

R широко используется в различных областях, таких как:

- *статистика:* для анализа данных, построения моделей, проверки гипотез;
- *анализ данных:* для обработки и визуализации данных, поиска закономерностей и аномалий;
- *машинное обучение:* для создания алгоритмов, которые могут обучаться на данных и делать прогнозы;
- *визуализация данных:* для создания графиков, диаграмм и других визуальных представлений данных;
- *научные исследования:* в биологии, медицине, экономике, социологии и других науках.

**Преимущества R:**

- *свободный и открытый исходный код:* R можно бесплатно скачать и использовать в любых целях;
- *мощный и гибкий:* R предоставляет широкий набор инструментов для анализа данных и программирования;
- *большое сообщество:* R имеет активное сообщество пользователей и разработчиков, которые создают новые пакеты и делятся своим опытом;
- *кроссплатформенность:* R работает на разных операционных системах (Windows, macOS, Linux).

## Установка и запуск R

### Шаг 1. Установка R:

- откройте веб-браузер и перейдите на официальный сайт R: https://cran.r-project.org/;
- выберите ссылку для скачивания R, соответствующую вашей операционной системе (Windows, macOS или Linux);
- скачайте установочный файл и запустите его;
- следуйте инструкциям на экране, чтобы установить R на ваш компьютер.

**Шаг 2. Установка RStudio:**

- откройте веб-браузер и перейдите на страницу загрузки RStudio: https://www.rstudio.com/products/rstudio/download/;
- выберите установочный файл RStudio Desktop, соответствующий вашей операционной системе;
- скачайте установочный файл и запустите его;
- следуйте инструкциям на экране, чтобы установить RStudio на ваш компьютер.

**Шаг 3. Запуск RStudio**

После установки R и RStudio вы можете запустить RStudio, дважды щелкнув по значку RStudio на рабочем столе или в меню «Пуск».

**Шаг 4. Знакомство с интерфейсом RStudio**

RStudio имеет несколько основных окон:

- консоль (Console): в этом окне вы можете вводить команды R и видеть результаты их выполнения;
- редактор кода (Source): в этом окне вы можете писать и редактировать скрипты – файлы с кодом R;
- окно Environment: в этом окне отображаются переменные, которые вы создали в текущей сессии R;
- окно Plots: в этом окне отображаются графики и диаграммы, созданные в R.

**Шаг 5. Первые шаги в R**

Давайте попробуем выполнить несколько простых команд в консоли R:

- вычисление: введите `2 + 2` и нажмите “Enter”. RStudio выведет результат: `[1] 4`;
- вывод текста: введите `print("Привет, мир!")` и нажмите “Enter”. RStudio выведет текст «Привет, мир!».

# *Глава 2*
# ПЕРЕМЕННЫЕ И ТИПЫ ДАННЫХ

– …И вот так я оказался здесь, – закончил свой рассказ Кай, указав на прозрачную фигуру, проплывающую мимо. – Это мой друг, Финн. Он тоже цифровой эльф, но специализируется на отладке кода движущихся платформ.

Алиса с интересом наблюдала за Финном, который ловко перепрыгивал с одной платформы на другую, исправляя ошибки в их программном коде. Платформы, похожие на гигантские пазлы, беспорядочно двигались в разных направлениях, сталкиваясь и меняя свое положение.

– А почему они так странно себя ведут? – спросила Алиса.

– Это из-за хаоса, который проник в код лабиринта, – объяснил Кай. – Он нарушает логику программ, и платформы начинают двигаться непредсказуемо. Если мы не исправим ошибки, весь лабиринт может рухнуть.

– И что нам делать? – спросила Алиса, чувствуя, как ее охватывает легкая паника.

В этот момент в воздухе появился голографический экран, на котором возникло лицо Элроса.

– Ученики, – сказал он своим механическим голосом, – я вижу, вы уже познакомились. А теперь приступайте к первому заданию.

Алиса и Кай выпрямились, словно солдаты перед генералом.

– Вам необходимо освоить основы языка R, – продолжил Элрос. – Начните с понятия переменных и типов данных. Это фундамент, на котором строится весь код.

– Переменные? Типы данных? – пробормотала Алиса.

– Не волнуйся, – успокоил ее Кай. – Это совсем не сложно. Переменные – это как контейнеры для хранения информации. А типы данных определяют, какую именно информацию можно хранить в этих контейнерах.

– Например, – включился Элрос, – если мы хотим запомнить твое имя, Алиса, мы можем создать переменную с именем “name” и присвоить ей значение «Алиса».

На голографическом экране появилась строчка кода:

```
name <- "Алиса"
```

– Символ «<-» означает «присвоить», – объяснил Элрос. – Теперь переменная “name” содержит твое имя.

– А какие бывают типы данных? – спросила Алиса.

– Основные типы данных – это числовые, строковые и логические, – ответил Кай. – Числовые переменные хранят числа, строковые – текст, а логические – значения «истина» или «ложь».

– Например, – снова включился Элрос, – если мы хотим запомнить твой возраст, Алиса, мы можем создать числовую переменную “age” и присвоить ей значение 19.

На экране появилась новая строчка кода:

```
age <- 19
```

– А если мы хотим запомнить, являешься ли ты цифровым эльфом, – продолжил Элрос, – мы можем создать логическую переменную “is_elf” и присвоить ей значение “FALSE”, потому что ты человек.

```
is_elf <- FALSE
```

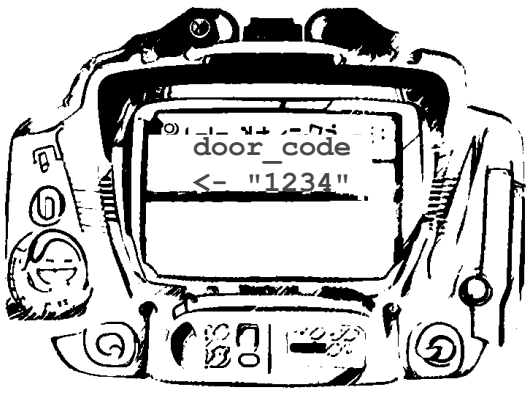
– А как это связано с лабиринтом? – спросила Алиса, которая уже начала скучать.

– Сейчас увидишь, – загадочно улыбнулся Кай. – Смотри!

Он подвел Алису к стене, на которой была написана странная надпись:

```
Если переменная "door_code" равна "1234",
        то дверь откроется.
```

– Это условие, – объяснил Кай. – Нам нужно создать переменную “door_code” и присвоить ей значение «1234».




Алиса сосредоточилась и вспомнила примеры Элроса. Она достала из кармана небольшой гаджет, похожий на смартфон, но с голографической клавиатурой. Это был ее личный R-терминал, который выдал ей Элрос перед путешествием.

– Так… – пробормотала Алиса, набирая код на терминале. – `door_code <- "1234"`. Готово!

В тот же момент стена перед ними распахнулась, открывая проход в следующий коридор.

– Вот видишь, – улыбнулся Кай. – R – это не просто язык программирования. Это магия!

– Или ключ к ней, – добавил голос Элроса, который вдруг появился рядом с ними. – А теперь продолжайте обучение. Впереди вас ждет еще много интересного.

И он снова исчез, оставив Алису и Кая в легком недоумении.

# R-ПРАКТИКУМ

**Переменная** – это именованный контейнер для хранения данных. Представьте себе коробку с ярлыком, на котором написано имя переменной. В эту коробку можно положить разные вещи: числа, текст, логические значения.

Правила именования переменных в R:

- имя переменной может состоять из букв, цифр и символов;
- имя переменной не может начинаться с цифры;
- R чувствителен к регистру, т. е. переменные `my_variable` и `My_Variable` – это разные переменные;
- желательно давать переменным осмысленные имена, которые отражают их содержание.

Примеры переменных:

```
name <- "Алиса"
age <- 19
pi <- 3.14159
is_raining <- TRUE
```

**Типы данных в R**

Каждый тип данных предназначен для хранения определенного вида информации:

- числовые (`numeric`) – для хранения чисел (целых или дробных). Например, `10`, `3.14`, `-5`;
- строковые (`character`) – для хранения текста. Строки заключаются в одинарные (`'`) или двойные (`"`) кавычки. Например,

```
"Привет, мир!", 'R - это круто!'.
```

- логические (`logical`) – для хранения логических значений TRUE (истина) или FALSE (ложь). Например, TRUE, FALSE.

**Определение типа данных**

Чтобы узнать, к какому типу данных относится переменная, можно использовать функцию `class()`.

Пример:

```
class(name)   # Выведет "character"
 class(age)    # Выведет "numeric"
```

**Преобразование типов данных**

Иногда нужно преобразовать данные из одного типа в другой. Для этого в R есть специальные функции:

- `as.numeric()` – преобразует данные в числовой тип;
- `as.character()` – преобразует данные в строковый тип;
- `as.logical()` – преобразует данные в логический тип.

Пример:

```
x <- "10"
class(x)  # Выведет "character"
y <- as.numeric(x)
class(y)  # Выведет "numeric"
```

**Задания для самостоятельной работы**

1. Создайте переменную `my_name` и присвойте ей значение вашего имени.
2. Создайте переменную `my_age` и присвойте ей значение вашего возраста.
3. Создайте переменную `is_programmer` и присвойте ей значение `TRUE`.
4. Проверьте типы данных созданных переменных с помощью функции `class()`.

# *Глава 3*
# ОПЕРАТОРЫ И ВЫРАЖЕНИЯ

Пройдя через открывшуюся дверь, Алиса и Кай оказались в огромном зале, похожем на пещеру. Стены мерцали, словно покрытые миллионами светлячков, а потолок терялся где-то в вышине. В центре зала возвышалась массивная дверь с замысловатым кодовым замком.

– Интересно, куда она ведет? – спросила Алиса, разглядывая дверь.

– Не знаю, – ответил Кай, приблизившись к замку. – Но, похоже, чтобы ее открыть, нам придется решить какую-то задачу.

На замке высветились строчки кода:

```
Если (5 + 3) * 2 > 15 И "password" != "qwerty",
                то дверь откроется.
```

– Хм, – пробормотал Кай, – похоже, нам нужно разобраться с операторами и выражениями.

– Операторы? Выражения? – Алиса почувствовала, как в голове начинается путаница.

– Не переживай, – успокоил ее Кай. – Операторы – это символы, которые выполняют какие-то действия с данными. Например, «+» – это оператор сложения, «*» – умножения, «>» – сравнения «больше». А выражения – это комбинации данных и операторов.

– То есть, вот это все – выражение? – Алиса указала на код на замке.

– Именно, – подтвердил Кай. – Давай разберемся по порядку. `(5 + 3) * 2` – это арифметическое выражение. Сначала выполняется сложение в скобках: 5 плюс 3 равно 8. Потом результат умножается на 2: 8 умножить на 2 равно 16.

– Получается, 16 больше 15, – сказала Алиса. – Значит первая часть условия верна.

– Правильно, – кивнул Кай. – Теперь смотрим дальше. `"password" != "qwerty"` – это выражение с оператором неравенства. Он проверяет, равны ли значения слева и справа. В данном случае “password” не равно “qwerty”, значит вторая часть условия тоже верна.

– А что значит «И»? – спросила Алиса.

– Это логический оператор, – ответил Кай. – Он объединяет два условия. В данном случае дверь откроется только если оба условия верны.

– Понятно, – сказала Алиса. – То есть, нам нужно просто ввести “password” в замок?

– Нет, – возразил Кай. – Замок принимает только логические значения: “TRUE” или “FALSE”. Нам нужно ввести “TRUE”, потому что оба условия верны.

Алиса ввела “TRUE” на панели замка, и дверь с грохотом отворилась.

– Ура! – воскликнула Алиса. – Мы справились!

– Да, – улыбнулся Кай. – Ты быстро учишься.

– А где же Элрос? – оглянулась Алиса.

Вместо ответа Кай показал на голографический экран, который появился в воздухе. На экране Элрос в компании двух киборгов, похожих на него, сидел за столом, уставленным бутылками и стаканами.

– … и говорю им, – вещал Элрос, – никакой паники! Все под контролем! Пусть малышня потренируется немного. А мы пока расслабимся…

– Похоже, он решил устроить себе небольшой перерыв, – усмехнулся Кай. – Ну что, идем дальше?

Алиса кивнула, и они вместе шагнули в неизвестность, скрывающуюся за дверью.

# R-ПРАКТИКУМ

**Операторы** – это символы, которые выполняют определенные действия над данными. Например, «+» – это оператор сложения, «−» – вычитания, «*» – умножения, «/» – деления.

Виды операторов в R:

- арифметические операторы (табл. 1): для выполнения математических операций;
- логические операторы (табл. 2): для выполнения логических операций;
- операторы сравнения (табл. 3): для сравнения значений;
- операторы присваивания («<-», «=»): для присваивания значений переменным.

*Таблица 1*

Арифметические операторы

| Оператор | Действие | Пример | Результат |
|---|---|---|---|
| + | Сложение | 2 + 2 | 4 |
| − | Вычитание | 5 − 3 | 2 |
| * | Умножение | 2 * 3 | 6 |
| / | Деление | 10 / 2 | 5 |
| ^ | Возведение в степень | 2 ^ 3 | 8 |
| %% | Остаток от деления | 7 %% 3 | 1 |
| %/% | Целая часть от деления | 7 %/% 3 | 2 |

*Таблица 2*

Логические операторы

| Оператор | Действие | Пример | Результат |
|---|---|---|---|
| & | Логическое «И» (AND) – истина, если оба операнда истинны | TRUE & TRUE | TRUE |
| ` | ` | Логическое «ИЛИ» (OR) – истина, если хотя бы один операнд истинен | `TRUE |
| ! | Логическое «НЕ» (NOT) – инвертирует значение операнда | !TRUE | FALSE |

*Таблица 3*

Операторы сравнения

| Оператор | Действие | Пример | Результат |
|---|---|---|---|
| > | Больше | 5 > 3 | TRUE |
| < | Меньше | 2 < 1 | FALSE |
| >= | Больше или равно | 5 >= 5 | TRUE |
| <= | Меньше или равно | 2 <= 3 | TRUE |
| == | Равно | 2 == 2 | TRUE |
| != | Не равно | “a” != “b” | TRUE |

**Выражение** – это комбинация значений, переменных, операторов и функций, которая вычисляется в одно значение.

Примеры выражений:

```
2 + 2 * 3        # Результат: 8
(10 - 5) / 2    # Результат: 2.5
"Hello" == "hello"  # Результат: FALSE
```

**Порядок выполнения операций**

В R, как и в математике, есть определенный порядок выполнения операций в выражениях. Сначала выполняются операции в скобках, затем возведение в степень, затем умножение и деление, и в конце – сложение и вычитание.

**Задания для самостоятельной работы**

1. Вычислите значение выражения `(10 + 5) * 2 - 8 / 4`.
2. Проверьте, верно ли утверждение `15 > 10 & 5 < 3`.
3. Проверьте, верно ли утверждение `"hello" != "world"`.

# *Глава 4*
# ФУНКЦИИ И СТРУКТУРЫ УПРАВЛЕНИЯ

За массивной дверью героев ожидал не коридор, а просторная площадка, напоминающая гигантскую детскую игровую комнату. Вместо стен – яркие конструкции из кубиков, пирамидок и шаров, которые постоянно меняли свою форму и цвет. В воздухе парили разноцветные платформы, соединенные лестницами и горками. А в центре площадки возвышался огромный лабиринт из лазерных лучей.

– Ух ты! – воскликнула Алиса, глаза которой разбегались от изобилия красок и форм. – Это же просто рай для геймера!

– Да, здесь можно весело провести время, – согласился Кай, но в его голосе звучала настороженность. – Но не забывай, мы все еще в цифровом лабиринте, и здесь могут быть скрытые опасности.

В этот момент над площадкой разнесся громогласный голос Элроса:

– Ученики! Я надеюсь, вы не забыли о своей миссии? Вам необходимо пройти через этот участок лабиринта и найти выход. Но будьте осторожны: здесь вас ждут новые испытания.

– Какие еще испытания? – пробормотала Алиса, с опаской поглядывая на лазерный лабиринт.

– Сейчас узнаете, – сказал Элрос. – Для начала вам необходимо освоить функции и структуры управления. Это позволит вам взаимодействовать с окружающей средой и контролировать ее.

– Функции? Структуры управления? – Алиса почувствовала, как ее энтузиазм постепенно угасает.

– Не пугайся, – успокоил ее Кай. – Функции – это просто блоки кода, которые выполняют определенные действия. А структуры управления позволяют нам определять, какие блоки кода и когда должны выполняться.

– Например, – включился Элрос, – если вы хотите переместить платформу вверх, вы можете использовать функцию `move_up()`. А если вы хотите проверить, достигла ли платформа определенной высоты, вы можете использовать условный оператор `if`.

На голографическом экране появился пример кода:

```
if (height < 10) {
  move_up()
}
```

– Этот код означает: «если высота платформы меньше 10, то переместить ее вверх», – объяснил Элрос.

– А как нам узнать высоту платформы? – спросила Алиса.

– Для этого существуют специальные функции, – ответил Кай. – Например, функция `get_height()` возвращает текущую высоту платформы.

– А как нам управлять лазерными лучами? – спросила Алиса, которая уже предвкушала захватывающую игру.

– Для этого тоже есть функции, – ответил Кай. – Например, `activate_laser()` активирует лазер, а `deactivate_laser()` деактивирует его.

– А еще есть циклы, – добавил Элрос. – Они позволяют повторять один и тот же блок кода несколько раз. Например, если вы хотите, чтобы платформа двигалась вверх до тех пор, пока не достигнет определенной высоты, вы можете использовать цикл `while`.

```
while (height < 10) {
  move_up()
}
```

– Этот код означает: «пока высота платформы меньше 10, повторять действие `move_up()`», – пояснил Элрос.

– Здорово! – воскликнула Алиса. – Теперь я понимаю, как это работает!

– Отлично, – сказал Элрос. – Тогда приступайте к выполнению задания. И не забудьте про осторожность! Лазерные лучи могут быть очень опасны.

И он вновь исчез, оставив Алису и Кая наедине с лазерным лабиринтом.

– Ну что, – сказал Кай, доставая свой R-терминал, – приступим?

Алиса кивнула, и они начали писать код, который поможет им преодолеть это испытание.

– Так, – сказала Алиса, вглядываясь в лазерный лабиринт, – нам нужно пройти через эти лучи, не задев их. Если мы дотронемся до луча, то…

– …превратимся в кучку цифрового пепла, – закончил за нее Кай. – Не самая приятная перспектива.

– Тогда давай хорошенько подумаем, – сказала Алиса, доставая R-терминал. – Элрос говорил, что есть функция `activate_laser()`

и `deactivate_laser()`. Может, мы можем как-то отключать лучи с их помощью?

– Хорошая идея, – поддержал ее Кай. – Но сначала нам нужно понять, как управлять платформами. Видишь те кнопки на полу? Наверное, они для этого.

Кай подошел к ближайшей кнопке и внимательно ее рассмотрел. На ней была надпись:

```
move_platform(platform_id, direction).
```

– Похоже, это функция для перемещения платформ, – сказал Кай. – `platform_id` – это идентификатор платформы, а `direction` – направление движения. Наверное, есть какие-то команды для указания направления.

– Давай попробуем! – воскликнула Алиса, уже набирая код на своем терминале. – `move_platform(1, "up")`.

В тот же момент одна из платформ начала медленно подниматься вверх.

– Получилось! – обрадовалась Алиса. – А как нам ее опустить?

– Наверное, нужно использовать команду `"down"`, – предположил Кай. – `move_platform(1, "down")`.

Платформа плавно опустилась вниз.

– Отлично! – сказала Алиса. – Теперь давай попробуем с лазерами.

Она направила терминал на один из лазеров и ввела команду: `deactivate_laser(1)`. Лазер мгновенно погас.

– Ура! – закричала Алиса. – Мы можем отключать лазеры!

– Не так быстро, – предупредил Кай. – Видишь, он снова включился.

Действительно, через несколько секунд лазер вновь загорелся.

– Что же делать? – растерялась Алиса.

– Помнишь, Элрос говорил про циклы? – напомнил Кай. – Может, нам нужно использовать цикл `while`, чтобы лазер оставался выключенным?

– Точно! – воскликнула Алиса. – `while (TRUE) { deactivate_laser(1) }`.

Она запустила код, и лазер погас. На этот раз он не включался снова.

– Получилось! – обрадовалась Алиса. – Теперь мы можем пройти через лабиринт!

– Не забывай про остальные лазеры, – напомнил Кай. – Нам нужно отключить их все.

– Да, конечно, – сказала Алиса, уже сосредоточившись на написании кода.

Вместе они начали продумывать стратегию прохождения лабиринта. Алиса управляла платформами, а Кай отключал лазеры, используя циклы и условные операторы. Они двигались медленно, но уверенно, преодолевая одно препятствие за другим.

Внезапно одна из платформ, на которой стояла Алиса, начала беспорядочно двигаться, словно сошла с ума.

– Что происходит? – испуганно воскликнула Алиса.

– Похоже, в коде платформы произошел сбой, – сказал Кай, пытаясь проанализировать ситуацию.

Платформа резко наклонилась, и Алиса чуть не свалилась в пропасть.

– Кай! Помоги! – закричала она.

Кай быстро среагировал и схватил ее за руку, удерживая от падения. В этот момент платформа резко поднялась вверх, подбросив их обоих в воздух.

– А-а-а! – закричали они в один голос.

В следующее мгновение они приземлились на другую платформу, которая находилась на значительной высоте.

– Фух, пронесло, – выдохнула Алиса, приходя в себя.

– Да, было близко, – согласился Кай. – Нам нужно быть осторожнее.

– Но что же случилось с платформой? – спросила Алиса.

– Не знаю, – ответил Кай. – Возможно, это тоже проделки хаоса. Нам нужно быстрее найти выход из этого лабиринта.

Они продолжили свой путь, но теперь двигались с еще большей осторожностью, внимательно отслеживая каждое свое действие.

– Смотри! – воскликнул Кай, указывая на светящийся символ в конце лабиринта. – Кажется, это выход!

Алиса, увлеченная написанием кода для очередной платформы, отвлеклась от своего R-терминала.

– Ура! – обрадовалась она. – Наконец-то! Я уже устала от этих лазеров и платформ.

– Подожди, – предупредил Кай, – нам нужно быть осторожными. Видишь те лучи, которые пересекаются перед выходом? Они двигаются очень быстро, и нам нужно точно рассчитать время, чтобы проскочить между ними.

– Хм, – задумалась Алиса, – кажется, для этого нам понадобится цикл `for` и функция `Sys.sleep()`, которая делает паузу в выполнении кода.

– Точно! – поддержал ее Кай. – Мы можем написать цикл, который будет последовательно отключать каждый луч на короткое время, а потом включать его обратно. В это время мы будем продвигаться вперед.

Алиса сосредоточилась и начала писать код:

```
for (i in 1:5) {  # Цикл для пяти лучей
  deactivate_laser(i)  # Отключаем i-й луч
  Sys.sleep(0.5)  # Пауза на полсекунды
  activate_laser(i)  # Включаем i-й луч обратно
  Sys.sleep(0.5)  # Пауза на полсекунды
}
```

– Готово! – сказала Алиса, запуская код.

Лучи начали по очереди включаться и выключаться, создавая узкие проходы между ними. Алиса и Кай, действуя слаженно, быстро проскочили через лабиринт и оказались перед выходом.

– Мы сделали это! – воскликнула Алиса, обнимая Кая от радости.

– Да, – улыбнулся Кай, – мы отличная команда!

В этот момент рядом с ними материализовался Элрос.

– Ну что, ученики, – сказал он своим обычным механическим голосом, – как продвигается ваше обучение?

– Мы прошли лабиринт! – с гордостью сообщила Алиса.

– Хм, – протянул Элрос, скептически оглядывая их, – неплохо. Но могли бы и быстрее. И код у вас… не самый элегантный.

– Но он же работает! – возразила Алиса.

– Работает – не значит хорошо, – отрезал Элрос. – В R важна не только функциональность, но и красота кода. Лаконичность, читаемость, эффективность – вот что отличает настоящего мастера R.

– Мы будем работать над этим, – пообещала Алиса, немного смутившись.

– Надеюсь на это, – сказал Элрос. – А теперь… мне пора. У меня еще много дел в других вселенных. Не скучайте!

И он исчез, оставив Алису и Кая в легком недоумении.

– «Красота кода»? – пробормотала Алиса. – Интересно, что он имел в виду?

– Не знаю, – пожал плечами Кай. – Но я думаю, что главное – чтобы код работал. А красота – это уже вторично.

– Наверное, ты прав, – согласилась Алиса. – Хотя… может быть, в этом что-то и есть. В конце концов, мы же не просто *кодим*, мы изменяем реальность. И, наверное, это нужно делать красиво.

– Философствуешь? – усмехнулся Кай.

– А что, нельзя? – парировала Алиса. – В конце концов, мы только что спасли цифровой мир от хаоса. Имеем право немного поразмышлять о жизни.

– Хорошо, философствуй, – сказал Кай. – А я пока поищу выход из этой игровой комнаты. Надеюсь, за следующей дверью нас не ждет парк аттракционов с ошибками в коде безопасности.

– О нет! – воскликнула Алиса. – Только не американские горки с бесконечным циклом!

Кай рассмеялся, и они вместе отправились на поиски новых приключений.

…Внезапно пространство вокруг них задрожало, и голографическое изображение Элроса начало мерцать и искажаться.

– Что происходит? – испуганно спросила Алиса.

– Не знаю, – ответил Кай, вглядываясь в изображение Элроса. – Похоже на какой-то сбой в системе.

В следующий момент изображение Элроса рассыпалось на тысячи светящихся частиц, которые закружились вокруг героев, а затем исчезли.

– Элрос! – позвала Алиса, но ответа не было.

– Он исчез, – сказал Кай, оглядываясь по сторонам. – Но… посмотри!

Он указал на место, где только что стоял Элрос. Там, на полу, лежала небольшая светящаяся сфера.

– Что это? – спросила Алиса, приближаясь к сфере.

– Не знаю, – ответил Кай, – но, кажется, это как-то связано с Элросом.

Алиса осторожно коснулась сферы, и в тот же момент ее R-терминал завибрировал. На экране появилось сообщение:

«Внимание! Критическая ошибка!
Целостность Мастера нарушена.
Необходима немедленная помощь!»

– Что это значит? – спросила Алиса, глядя на Кая.

– Я не знаю, – ответил Кай, – но, кажется, с Элросом случилось что-то серьезное. Нам нужно срочно его найти!

– Но как? – спросила Алиса. – Мы даже не знаем, где он!

В этот момент сфера на полу вспыхнула ярким светом, и перед героями появился голографический указатель, стрелка которого была направлена в глубину лабиринта.

– Похоже, это наш путь, – сказал Кай. – Идем!

**Функция** – это именованный блок кода, который выполняет определенное действие. Функции позволяют нам организовывать код, делать его более читаемым и многократно использовать один и тот же код для разных задач.

**Встроенные функции в R**

В R есть множество встроенных функций, которые выполняют различные действия:

- `print()`: выводит данные на экран;
- `sum()`: вычисляет сумму чисел;
- `mean()`: вычисляет среднее арифметическое;
- `sqrt()`: вычисляет квадратный корень;
- `length()`: возвращает длину вектора;
- `nchar()`: возвращает количество символов в строке.

**Создание собственных функций**

Вы также можете создавать свои собственные функции в R. Для этого используется следующий синтаксис:

```
my_function <- function(аргументы) {
  # Код  функции
}
```

Пример:

```
# Функция для вычисления площади прямоугольника
calculate_area <- function(length, width) {
  area <- length * width
  return(area)
}
```

**Структуры управления** – это конструкции в программе, которые определяют порядок выполнения команд. Они позволяют нам управлять потоком выполнения программы, выполнять код при определенных условиях или повторять его несколько раз.

**Виды структур управления в R:**

- условные операторы: `if`, `else if`, `else`;
- циклы: `for`, `while`.

Оператор `if` позволяет выполнить блок кода только при выполнении определенного условия.

Синтаксис:

```
if (условие) {
  # Код, который будет выполнен, если условие истинно
}
```

Пример:

```
if (age >= 18) {
  print("Вы совершеннолетний")
}
```

Цикл `for` позволяет повторять блок кода определенное количество раз.

Синтаксис:

```
for (переменная in последовательность) {
  # Код, который будет выполнен для каждого элемента последовательности
}
```

Пример:

```
for (i in 1:10) {
  print(i)
}
```

Цикл `while` позволяет повторять блок кода до тех пор, пока выполняется определенное условие.

Синтаксис:

```
while (условие) {
  # Код, который будет выполнен, пока условие истинно
}
```

Пример:

```
i <- 1
while (i <= 10) {
  print(i)
  i <- i + 1
}
```

**Задания для самостоятельной работы**

1. Напишите функцию `calculate_area()`, которая принимает два аргумента (длину и ширину) и возвращает площадь прямоугольника.
2. Напишите цикл `for`, который выводит на экран числа от 1 до 10.

# *Глава 5*
# В ЛАБИРИНТЕ ТАЙНЫХ ЗНАНИЙ

– Куда же он нас ведет? – спросила Алиса, следуя за голографической стрелкой, которая плыла в воздухе, указывая путь.

– Понятия не имею, – ответил Кай, оглядываясь по сторонам. – Этот участок лабиринта выглядит… иначе.

И действительно, атмосфера вокруг резко изменилась. Яркие цвета и игривые формы сменились строгими геометрическими фигурами и приглушенными тонами. Стены коридоров были покрыты сложными формулами и диаграммами, а в воздухе висели голографические модели различных алгоритмов.

– Похоже, мы попали в какую-то секретную зону, – предположила Алиса. – Может быть, это лаборатория Элроса?

– Возможно, – согласился Кай. – Но что он здесь делал? И куда он исчез?

В этот момент стрелка указателя резко повернула вниз, и герои оказались перед лестницей, ведущей в темноту.

– Ну что, спускаемся? – спросил Кай.

– Деваться некуда, – ответила Алиса, и они начали спускаться по лестнице.

Чем ниже они спускались, тем холоднее становился воздух. Стены коридоров покрылись инеем, а в воздухе замелькали снежинки. Наконец, лестница закончилась, и герои оказались в просторном зале, похожем на ледяную пещеру. В центре зала возвышалась гигантская конструкция, напоминающая маяк.

– Что это? – спросила Алиса, с восхищением разглядывая маяк.

– Не знаю, – ответил Кай, – но кажется, это и есть наша цель.

Внезапно R-терминал Алисы снова завибрировал. На экране появилось новое сообщение от Элроса:

«Это маяк межпространственной связи.
Используйте его, чтобы найти меня.
Но будьте осторожны: маяк охраняется
системой защиты. Вам нужно будет ввести
правильный код активации».

– Код активации? – переспросила Алиса. – А где нам его взять?

– Не паникуй, – успокоил ее Кай. – Мы что-нибудь придумаем.

Они подошли к маяку и начали изучать его устройство. На боковой панели маяка они обнаружили консоль управления с клавиатурой и небольшим экраном. На экране высвечивалась надпись:

«Введите код активации».

– Ну что, попробуем подобрать код? – спросила Алиса.

– Нет, это бессмысленно, – ответил Кай. – Код может быть любой длины и состоять из любых символов. Нам нужно найти подсказку.

Они начали внимательно осматривать зал, ища какие-либо надписи или символы, которые могли бы указывать на код. Внезапно Алиса заметила на одной из стен странный узор, состоящий из геометрических фигур.

– Кай, посмотри! – позвала она. – Что это может значить?

Кай подошел к стене и внимательно изучил узор.

– Хм, – пробормотал он, – интересно. Эти фигуры напоминают… векторы!

– Векторы? – переспросила Алиса. – А что это такое?

– Векторы – это один из основных типов данных в R, – объяснил Кай. – Они представляют собой упорядоченные наборы элементов. Например, вектор может содержать числа, символы или логические значения.

– А как это может нам помочь? – спросила Алиса.

– Подожди, я попробую что-нибудь придумать, – сказал Кай, доставая свой R-терминал.

Он начал вводить команды на терминале, анализируя узор на стене и преобразуя его в векторы.

– Так… – бормотал он, – первый вектор состоит из чисел 1, 3, 5… второй вектор – из символов “A”, “C”, “E”… третий вектор – из логических значений TRUE, FALSE, TRUE…

Внезапно Кай остановился и воскликнул:

– Я понял! Код активации – это конкатенация этих векторов!

– Конкатенация? – не поняла Алиса.

– Это объединение векторов в один, – объяснил Кай. – Смотри!

Он ввел следующую команду на терминале:

```
c(1, 3, 5, "A", "C", "E", TRUE, FALSE, TRUE)
```

На экране появился результат:

```
   [1] "1"       "3"       "5"       "A"       "C"
"E"       "TRUE"    "FALSE"   "TRUE"
```

– Вот наш код! – воскликнул Кай.

Он подбежал к консоли управления маяком и ввел полученный код. Маяк загудел, и его верхушка засияла ярким светом.

– Получилось! – обрадовалась Алиса. – Мы сможем найти Элроса!

В этот момент луч света из верхушки маяка устремился вверх, пробивая потолок пещеры и уходя в неизвестность.

– Держись! – крикнул Кай, хватая Алису за руку.

И они вместе исчезли в вихре света, отправляясь на поиски своего наставника.

# R-ПРАКТИКУМ

**Вектор** – это упорядоченная последовательность элементов одного типа. Представьте себе поезд, где каждый вагон – это элемент вектора, а весь состав – это вектор. В R векторы являются одним из основных типов данных и используются для хранения коллекций чисел, строк или логических значений.

**Создание векторов**

Для создания векторов в R используется функция `c()`, которая объединяет элементы в вектор.

Примеры:

```
# Вектор чисел
numbers <- c(1, 5, 2, 8, 3)
# Вектор строк
names <- c("Алиса", "Кай", "Элрос")
# Вектор логических значений
is_true <- c(TRUE, FALSE, TRUE)
```

**Доступ к элементам вектора**

Чтобы получить доступ к определенному элементу вектора, нужно указать его индекс в квадратных скобках. Индексы начинаются с 1.

Примеры:

```
numbers[1]  # Первый элемент вектора numbers (1)
names[3]    # Третий элемент вектора names ("Элрос")
```

**Операции над векторами**

Над векторами можно выполнять различные операции, например:

- арифметические операции: сложение, вычитание, умножение, деление. Операции выполняются поэлементно;
- сравнение: сравнение векторов на равенство или неравенство;
- логические операции: логическое «И», «ИЛИ», «НЕ».

Примеры:

```
# Сложение векторов
numbers + c(10, 10, 10, 10, 10)  # Результат: 11 15 12 18 13
```

```
   # Умножение вектора на число
   numbers * 2  # Результат: 2 10 4 16 6
   # Сравнение векторов
   numbers == c(1, 5, 3, 8, 3)  # Результат: TRUE
TRUE FALSE TRUE TRUE
```

**Полезные функции для работы с векторами:**

- `length()`: возвращает длину вектора (количество элементов);
- `sort()`: сортирует элементы вектора;
- `rev()`: разворачивает вектор (меняет порядок элементов на обратный);
- `unique()`: возвращает вектор с уникальными элементами.

Примеры:

```
length(numbers)  # Выведет 5
sort(numbers)  # Выведет 1 2 3 5 8
rev(numbers)  # Выведет 3 8 2 5 1
unique(c(1, 2, 2, 3, 3, 3))  # Выведет 1 2 3
```

**Задания для самостоятельной работы**

1. Создайте вектор `my_numbers`, содержащий числовые значения от 1 до 20.
2. Выведите на экран пятый элемент вектора `my_numbers`.
3. Умножьте все элементы вектора `my_numbers` на 3.
4. Создайте новый вектор, содержащий первые 10 элементов вектора `my_numbers`.

## *Глава 6*
# ВЕКТОРЫ И ПОИСКИ ИСТИНЫ

Свет маяка окутал Алису и Кая, и в следующее мгновение они оказались в совершенно другом месте. Вокруг тянулся бесконечный белый простор, напоминающий пустыню, но вместо песка под ногами была гладкая, словно отполированная, поверхность. Над головой висело огромное голубое небо без единого облачка.

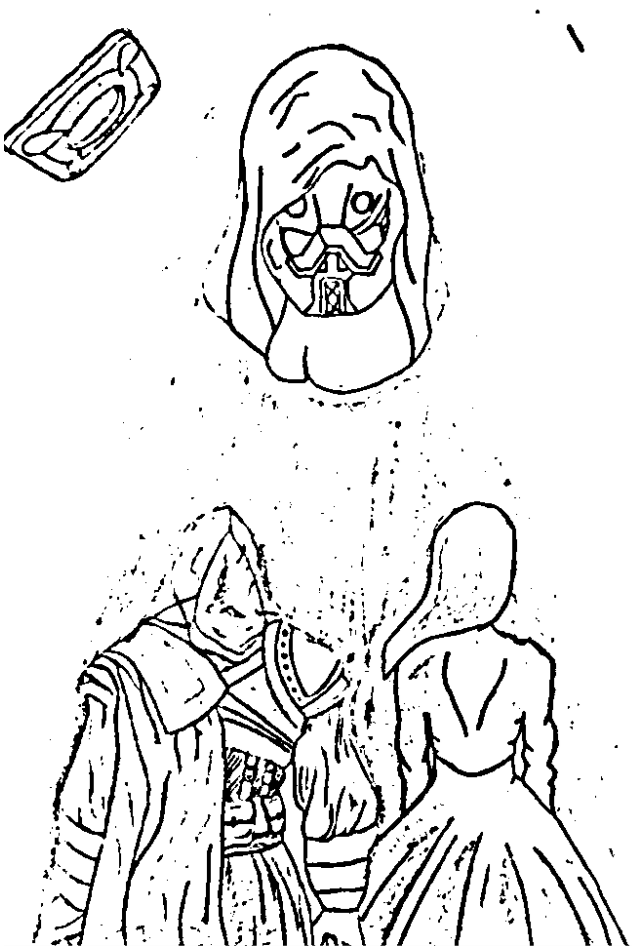
– Где мы? – спросила Алиса, оглядываясь по сторонам.

– Похоже, мы в какой-то виртуальной реальности, – ответил Кай, изучая окружение через свой R-терминал. – Судя по данным, которые я получаю, это пространство полностью сгенерировано кодом.

– А где Элрос? – спросила Алиса, вглядываясь в даль.

В этот момент в воздухе появилось голографическое изображение Элроса, но на этот раз оно было тусклым и нечетким, словно сигнал был очень слабым.

– Ученики… – прохрипел Элрос, – я… рад, что вы нашли меня… Я нахожусь… в ловушке… Мне нужна ваша помощь…



Изображение замерцало и исчезло.

– Элрос! – позвала Алиса, но ответа не было.

– Связь очень нестабильна, – сказал Кай, проверяя свой терминал. – Мы находимся слишком далеко друг от друга. Но он сказал, что в ловушке. Нам нужно его найти!

– Но где нам его искать в этой… пустыне? – растерянно спросила Алиса.

Внезапно земля под их ногами задрожала, и из-под поверхности начали вырастать

странные конструкции, похожие на кристаллы. Кристаллы были разных цветов и форм, и на каждом из них были выгравированы какие-то символы.

– Что это? – спросила Алиса, с опаской наблюдая за растущими кристаллами.

– Не знаю, – ответил Кай, – но кажется, это какой-то код.

Он подошел к ближайшему кристаллу и внимательно рассмотрел символы.

– Подожди, я кажется, понимаю! – воскликнул он. – Это векторы!

– Векторы? – переспросила Алиса. – А что они здесь делают?

– Похоже, это какой-то ключ к поиску Элроса, – сказал Кай. – Нам нужно проанализировать эти векторы и понять, что они означают.

– Но как мы это сделаем? – спросила Алиса. – Мы же еще не изучали векторы подробно.

– Не волнуйся, – успокоил ее Кай. – Я тебе помогу. Векторы – это просто упорядоченные наборы данных. Каждый кристалл содержит один вектор, и нам нужно найти связь между ними.

Кай достал свой R-терминал и начал сканировать кристаллы, записывая данные о векторах.

– Так… – бормотал он, – первый вектор содержит числа 1, 2, 3… второй вектор – числа 4, 5, 6… третий вектор – числа 7, 8, 9…

– И что это значит? – спросила Алиса.

– Пока не знаю, – ответил Кай, – но я заметил одну закономерность. Каждый следующий вектор получается из предыдущего путем прибавления 3 к каждому элементу.

– И что это нам дает? – спросила Алиса.

– Возможно, это подсказка, – сказал Кай. – Давай попробуем продолжить эту последовательность. Следующий вектор должен быть 10, 11, 12…

Кай ввел эти числа в свой терминал и нажал на кнопку «анализировать». Терминал загудел, и на экране появилась голографическая карта местности.

– Вот оно! – воскликнул Кай. – Это карта этого пространства! А те точки, которые на ней отмечены… это координаты Элроса!

– Ура! – обрадовалась Алиса. – Мы нашли его!

– Да, – сказал Кай, – но нам нужно торопиться. Судя по данным, он находится в очень нестабильной зоне. Если мы не поможем ему быстро, он может… раствориться в коде.

– Тогда бежим! – крикнула Алиса, и они вместе бросились бежать в направлении, указанном на карте.

Вспомним, что такое вектор.

**Вектор** – это упорядоченный набор элементов одного типа. В R векторы могут содержать числа, строки или логические значения. Представьте себе нитку бус, где каждая бусина – это элемент вектора, а вся нитка – это вектор. Все бусины должны быть одного типа, например, все красные или все круглые.

В пятой главе мы уже познакомились с основными операциями над векторами. Рассмотрим еще несколько полезных операций:

- `seq()`: создает вектор из чисел, идущих с заданным шагом. Например, `seq(from = 2, to = 10, by = 2)` создаст вектор (2, 4, 6, 8, 10);
- `rep()`: повторяет элементы вектора заданное количество раз. Например, `rep(c("a", "b"), times = 3)` создаст вектор `("a", "b", "a", "b", "a", "b");`
- `%in%`: проверяет, содержится ли элемент в векторе. Например, `5 %in% c(1, 3, 5, 7)` вернет `TRUE`.

Примеры:

```
# Создание вектора с числами от 1 до 10 с шагом 0.5
seq(1, 10, 0.5)
# Повторение каждого элемента вектора 3 раза
rep(c(1, 2, 3), each = 3)
# Проверка, содержится ли число 7 в векторе
7 %in% c(1, 3, 5, 7)
```

**Задания для самостоятельной работы**

1. Создайте вектор с числами от 10 до 100 с шагом 10, используя функцию `seq()`.
2. Создайте вектор, в котором число 5 повторяется 7 раз, используя функцию `rep()`.
3. Создайте вектор с именами месяцев. Проверьте, содержится ли в нем месяц «июнь».
4. Создайте два вектора с числами. Найдите элементы, которые присутствуют в обоих векторах, используя функцию `intersect()`.

5. Создайте два вектора с числами. Найдите элементы, которые присутствуют в первом векторе, но отсутствуют во втором, используя функцию `setdiff()`.

# *Глава 7*
# ОПЕРАЦИИ НАД ВЕКТОРАМИ – ПУТЬ К СПАСЕНИЮ

Белый простор вокруг них замелькал, словно картинка в неисправном телевизоре, и через мгновение Алиса и Кай оказались в совершенно ином месте. Они стояли на узкой каменной платформе, парящей в воздухе над бездной, заполненной бурлящим цифровым хаосом. Вдали виднелись обломки кода, похожие на астероиды, которые хаотично двигались, сталкиваясь и распадаясь на мельчайшие частицы.

– Похоже, мы попали в эпицентр хаоса, – сказал Кай, с тревогой оглядывая окружение. – Здесь очень нестабильно.

– А где Элрос? – спросила Алиса, вглядываясь в цифровую бурю.

В этот момент рядом с ними материализовался голографический образ Элроса, но он был еще более тусклым и искаженным, чем раньше.

– Ученики… – прошептал Элрос, – я… теряю связь… Хаос… разрушает мой код… Мне нужна… ваша… помощь…

Изображение рассеялось, оставив после себя лишь мерцающие искры.

– Элрос! – крикнула Алиса, протягивая руку к исчезающему образу. – Мы идем!

– Но как нам его найти? – спросил Кай, чувствуя, как платформа под их ногами начинает дрожать. – Мы же даже не знаем, где он!

Внезапно R-терминал Кая засигналил, и на экране появилось новое сообщение от Элроса:

«Используйте... векторы... операции...
сложение... вычитание...»

Сообщение оборвалось, прежде чем Элрос успел договорить.

– Что он имеет в виду? – спросила Алиса, глядя на Кая.

– Я не знаю, – ответил Кай, – но кажется, он пытается нам что-то подсказать. Вспомни, мы уже сталкивались с векторами в прошлый раз. Может быть, нам нужно провести какие-то операции над ними?

– Но где нам взять векторы? – спросила Алиса, оглядываясь по сторонам.

– Посмотри вниз! – крикнул Кай, указывая на бездну.

Алиса посмотрела вниз и увидела, что в хаосе мелькают светящиеся символы, образующие странные узоры.

– Это… векторы? – неуверенно спросила она.

– Да! – подтвердил Кай. – И они постоянно меняются. Нам нужно быстро проанализировать их и провести необходимые операции.

Кай достал свой R-терминал и начал сканировать векторы, отображающиеся в бездне.

– Так… – бормотал он, – первый вектор – (1, 2, 3)… второй вектор– (4, 5, 6)… Что же нам с ними делать? Сложение? Вычитание?

– Попробуй сложить их! – предложила Алиса.

Кай быстро ввел команду на терминале:

```
c(1, 2, 3) + c(4, 5, 6)
```

На экране появился результат:

```
[1] 5 7 9
```

– Получился новый вектор – (5, 7, 9), – сказал Кай. – Но что он означает?

Внезапно платформа под их ногами затряслась еще сильнее, и в воздухе появился новый вектор – (2, 2, 2).

– Может быть, нужно вычесть его из предыдущего вектора? – предложила Алиса.

Кай снова ввел команду:

```
c(5, 7, 9) - c(2, 2, 2)
```

Результат:

```
[1] 3 5 7
```

– (3, 5, 7)… – прошептал Кай. – Подожди… Я что-то припоминаю! Элрос рассказывал нам про специальные векторы-ключи, которые могут стабилизировать код реальности. И один из них как раз имеет такие координаты!

– Значит, мы на правильном пути! – обрадовалась Алиса. – Но как нам использовать этот вектор?

– Нам нужно отправить его в эпицентр хаоса! – сказал Кай. – Но как?

В этот момент платформа, на которой они стояли, начала разрушаться. Цифровой хаос подступал все ближе, грозя поглотить их.

– У нас нет времени! – крикнул Кай. – Алиса, ты помнишь, как работать с функцией `print()`?

– Кажется, да, – ответила Алиса, лихорадочно включая свой R-терминал. – Она выводит данные на экран.

– Не на экран! – крикнул Кай. – Нам нужно вывести вектор в реальность! Используй аргумент `Screen` и установи его значение `"reality"`!

Алиса быстро ввела команду:

```
print(c(3, 5, 7), Screen = "reality")
```

В тот же момент вектор (3, 5, 7) материализовался в реальности в виде трех светящихся точек, которые устремились в цифровой хаос. Хаос забурлил еще сильнее, а затем начал успокаиваться. Обломки кода замедлили свое движение и стали выстраиваться в упорядоченные структуры.

– Получилось! – воскликнула Алиса. – Мы стабилизировали код!

– Да, – сказал Кай, с облегчением вздыхая, – но где же Элрос?

В этот момент пространство вокруг них начало проясняться, цифровой хаос рассеивался, а обломки кода словно растворялись в воздухе. В центре образовавшейся пустоты они увидели Элроса. Он парил в воздухе, его металлическое тело мерцало, а глаза были закрыты.

– Элрос! – крикнула Алиса и бросилась к нему.

Кай последовал за ней. Подлетев ближе, они увидели, что тело Элроса частично разрушено, а из пробоин вырываются струйки цифрового кода.

– Он очень пострадал, – сказал Кай, с тревогой разглядывая Мастера. – Хаос почти разрушил его код.

– Что же нам делать? – спросила Алиса, с жалостью глядя на Элроса.

Внезапно R-терминал Кая засигналил, и на экране появилось новое сообщение, но на этот раз не от Элроса. Сообщение было от неизвестного отправителя:

**«Вы справились с первым испытанием. Но это было лишь начало. Хаос не побежден, он лишь затаился. Будьте готовы к новым испытаниям. Ваше путешествие продолжается..»**

Сообщение исчезло, а вместе с ним и голографический образ Элроса. Вместо него в воздухе появился новый указатель, стрелка которого была направлена в неизвестном направлении.

– Куда теперь? – спросила Алиса, глядя на указатель.

– Не знаю, – ответил Кай, – но кажется, у нас нет выбора. Мы должны следовать за этим указателем.

– Но что же с Элросом? – спросила Алиса. – Мы не можем его бросить!

– У нас нет выбора, – повторил Кай. – Похоже, кто-то другой заботится о нем. А нам нужно продолжать наше путешествие. Возможно, в конце мы сможем помочь Элросу и победить хаос.

Алиса вздохнула, но согласилась с Каем. Они вместе направились в сторону, указанную стрелкой, готовясь к новым приключениям и испытаниям.

**Изменение элементов вектора**

Мы уже умеем создавать векторы и получать доступ к их элементам. Но что, если нам нужно изменить значения некоторых элементов? Это можно сделать с помощью оператора присваивания «<-» и индексации.

Примеры:

```
my_vector <- c(5, 2, 8, 1, 3)
# Изменим значение третьего элемента
my_vector[3] <- 10
# Изменим значения первого и пятого элементов
my_vector[c(1, 5)] <- c(7, 9)
print(my_vector)  # Выведет 7 2 10 1 9
```

**Логическая индексация**

Мы можем использовать логические выражения для выбора элементов вектора. Это очень удобно, когда нужно изменить значения элементов, удовлетворяющих определенному условию.

Пример:

```
my_vector <- c(5, 2, 8, 1, 3)
# Увеличим на 1 все элементы, которые больше 3
my_vector[my_vector > 3] <- my_vector[my_vector
> 3] + 1
print(my_vector)  # Выведет 6 2 9 1 4
```

**Функции** `which()` **и** `match()`

Функция `which()` возвращает индексы элементов вектора, которые удовлетворяют заданному условию, а `match()` возвращает индексы первых вхождений элементов одного вектора в другом векторе.

Примеры:

```
my_vector <- c(5, 2, 8, 1, 3)
# Найдем индексы элементов, которые больше 3
which(my_vector > 3)  # Выведет 1 3
# Найдем индексы первых вхождений чисел 2 и 8
в векторе
match(c(2, 8), my_vector)  # Выведет 2 3
```

**Задания для самостоятельной работы**

1. Создайте вектор с числами от 20 до 1 (в обратном порядке).
2. Найдите длину этого вектора.
3. Отсортируйте вектор по возрастанию.
4. «Разверните» вектор.
5. Выберите из вектора все числа, которые делятся на 3 без остатка.

# *Глава 8*
# СПИСКИ – ПОРЯДОК В ХАОСЕ

Алиса и Кай двигались вперед, следуя за указателем, который вел их сквозь лабиринт извилистых коридоров и странных залов. Цифровая реальность вокруг них постоянно менялась, стены то растворялись в воздухе, то превращались в замысловатые узоры, а пол под ногами то превращался в зыбучие пески, то вздымался острыми пиками.

– Интересно, куда мы идем? – спросила Алиса, с опаской оглядываясь по сторонам.

– Надеюсь, это безопасное место, – ответил Кай, внимательно следя за показаниями своего R-терминала. – После того хаоса, который мы только что пережили, я бы не отказался от небольшой передышки.

Внезапно коридор, по которому они шли, закончился, и герои оказались в огромном зале, заполненном миллионами летающих объектов. Это были не просто объекты, а фрагменты кода, строчки программ, переменные, функции – все то, из чего состояла цифровая реальность. Они хаотично двигались, сталкиваясь и перемешиваясь друг с другом, создавая впечатление полного беспорядка.

– Что здесь происходит? – спросила Алиса, с трудом пробираясь сквозь вихрь летающих объектов.

– Похоже, это какой-то *свалкокод*, – ответил Кай, пытаясь проанализировать ситуацию с помощью своего терминала. – Все эти фрагменты кода не на своих местах. Они должны быть организованы в логическую структуру, а не летать в хаосе.

В этот момент R-терминал Алисы засигналил, и на экране появилось новое сообщение:

```
«Добро пожаловать на свалку кода!
Ваша задача - навести здесь порядок.
Используйте списки, чтобы организовать фрагменты
кода и восстановить структуру этого места».
```

– Списки? – переспросила Алиса. – А что это такое?

– Списки – это еще один тип данных в R, – объяснил Кай. – Они позволяют хранить упорядоченные коллекции объектов. В отличие от векторов, которые могут содержать только элементы одного типа,

списки могут хранить объекты разных типов: числа, строки, логические значения, даже другие списки и векторы.

– А как нам использовать списки, чтобы навести порядок в этом хаосе? – спросила Алиса, с опаской поглядывая на летающие вокруг фрагменты кода.

– Нам нужно создать списки и распределить по ним фрагменты кода в соответствии с их типом и назначением, – ответил Кай. – Например, мы можем создать отдельный список для числовых переменных, отдельный – для строковых, и так далее.

– А как мы будем это делать? – спросила Алиса.

– С помощью R-терминала, конечно, – ответил Кай. – Смотри!

Кай достал свой терминал и начал вводить команды.

```
# Создаем пустой список
my_list <- list()
# Добавляем в список числовую переменную
my_list$numbers <- c(1, 2, 3, 4, 5)
# Добавляем в список строковую переменную
my_list$names <- c("Алиса", "Кай", "Элрос")
# Добавляем в список логическую переменную
my_list$is_true <- TRUE
```

– Вот так мы создали список `my_list` и добавили в него три элемента: вектор чисел `numbers`, вектор строк `names` и логическую переменную `is_true`, – объяснил Кай. – Теперь мы можем обращаться к этим элементам по их именам.

– А как нам добавить в список все эти летающие фрагменты кода? – спросила Алиса.

– Для этого нам нужно сначала проанализировать их и определить их тип и назначение, – ответил Кай. – А затем мы сможем написать программу, которая автоматически распределит их по спискам.

– Звучит сложно, – сказала Алиса.

– Не волнуйся, – успокоил ее Кай. – Мы справимся. Вместе.

И они принялись за работу, анализируя фрагменты кода и создавая списки для их организации. Постепенно хаос в зале начал уступать место порядку, а летающие объекты стали выстраиваться в четкие структуры.

**Список (list)** – это упорядоченная коллекция объектов. В отличие от векторов, которые могут содержать только элементы одного типа, списки могут хранить объекты разных типов, включая другие списки и векторы. Это делает списки очень гибким инструментом для хранения и организации данных.

**Создание списков**

Для создания списков в R используется функция `list()`.

Примеры:

```
# Пустой список
empty_list <- list()
# Список с элементами разных типов
my_list <- list(1, "hello", TRUE, c(1, 2, 3))
# Список с именованными элементами
person <- list(name = "Alice", age = 19, is_elf = FALSE)
```

**Доступ к элементам списка**

Для доступа к элементам списка можно использовать два способа:

- по индексу: укажите индекс элемента в двойных квадратных скобках [[ ]]. Индексы начинаются с 1;
- по имени: укажите имя элемента после знака доллара $.

Примеры:

```
my_list[[1]]  # Первый элемент списка my_list (1)
person$name  # Элемент с именем "name" в списке person ("Alice")
```

**Добавление элементов в список**

Добавлять элементы в список можно с помощью функции `append()` или присваивая значения новым элементам.

Примеры:

```
# Добавление элемента в конец списка
my_list <- append(my_list, "new element")
# Добавление элемента с именем
person$city <- "Moscow"
```

**Другие полезные функции для работы со списками:**

- `length()`: возвращает количество элементов в списке;
- `names()`: возвращает имена элементов списка (если они есть);
- `is.list()`: проверяет, является ли объект списком;
- `unlist()`: преобразует список в вектор.

Примеры:

```
length(my_list)  # Выведет 5
names(person)  # Выведет "name"  "age"  "is_elf"
"city"
is.list(my_list)  # Выведет TRUE
```

**Задания для самостоятельной работы**

1. Создайте список, содержащий ваши имя, возраст и любимый цвет.
2. Добавьте в список новый элемент: название вашего города.

# *Глава 9*
# МАТРИЦЫ И ГОЛОВОЛОМКИ РЕАЛЬНОСТИ

Алиса и Кай продолжали свое путешествие по цифровому лабиринту. После свалки кода они попали в зону, напоминающую гигантский конструктор. Вокруг них возвышались башни из кубов, пирамид и других геометрических фигур, которые постоянно меняли свое положение и форму.

– Похоже на тетрис в 3D, – заметила Алиса, с интересом разглядывая окружение. – Только фигуры сами падают и перемещаются.

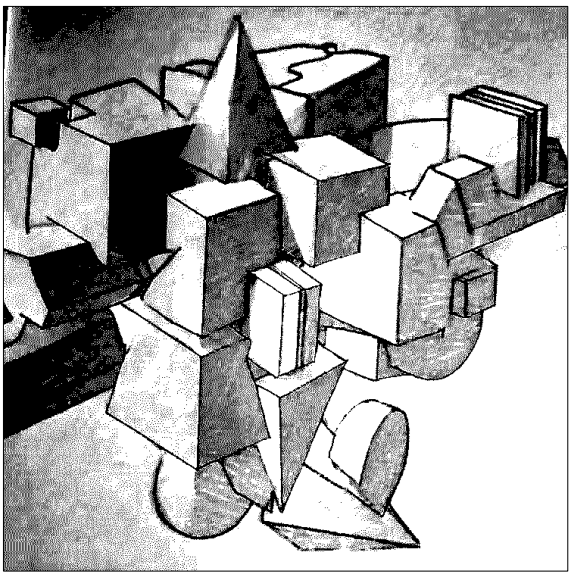
– Да, здесь все очень динамично, – согласился Кай, анализируя данные на своем R-терминале. – Судя по всему, этот участок лабиринта управляется каким-то сложным алгоритмом.

Внезапно один из кубов отделился от башни и полетел прямо на Алису. Девушка вскрикнула и отпрыгнула в сторону, едва успев увернуться от столкновения.

– Осторожнее! – крикнул Кай. – Здесь нужно быть на чеку.

В этот момент R-терминал Алисы засигналил, и на экране появилось новое сообщение:

**«Поздравляю, вы добрались до зоны матричных преобразований! Ваша задача – решить головоломку реальности, используя матрицы и операции над ними. Удачи!»**



– Матрицы? – переспросила Алиса. – А что это такое?

– Матрицы – это двумерные структуры данных, – объяснил Кай. – Представь себе таблицу, состоящую из строк и столбцов. В каждой ячейке таблицы может храниться какое-то значение.

– А зачем они нужны? – спросила Алиса.

– Матрицы используются для хранения и обработки больших

объемов данных, – ответил Кай. – Они также широко применяются в линейной алгебре, машинном обучении и других областях.

– А как они помогут нам решить эту головоломку? – спросила Алиса, указывая на хаотично двигающиеся фигуры.

– Я пока не знаю, – ответил Кай, – но давай попробуем проанализировать эти фигуры с помощью матриц.

Кай достал свой R-терминал и начал сканировать окружающие объекты.

– Так… – бормотал он, – каждая фигура может быть представлена в виде матрицы, где каждый элемент соответствует блоку в фигуре. Например, вот эта пирамида…

Кай навел терминал на пирамиду, и на экране появилось ее матричное представление:

```
     [,1] [,2] [,3]
[1,]    0    0    1
[2,]    0    1    1
[3,]    1    1    1
```

– Но что нам делать с этими матрицами? – спросила Алиса.

– Возможно, нам нужно провести какие-то операции над ними, – предположил Кай. – Например, сложить их, умножить, транспонировать…

– А давай попробуем транспонировать матрицу этой пирамиды! – предложила Алиса.

– Хорошая идея! – сказал Кай. – Функция `t()` в R как раз для этого предназначена.

Кай ввел команду на терминале:

```
t(matrix(c(0, 0, 1, 0, 1, 1, 1, 1, 1), nrow = 3, byrow = TRUE))
```

На экране появилась транспонированная матрица:

```
     [,1] [,2] [,3]
[1,]    0    0    1
[2,]    0    1    1
[3,]    1    1    1
```

В тот же момент пирамида в реальности изменила свою форму, словно отразившись в зеркале.

– Не получилось! – воскликнула Алиса. – Мы не изменили реальность с помощью матрицы!

– Подумай, Алиса и не паникуй раньше времени, – сказал Кай. – Матрица пирамиды симметрична относительно главной диагонали (0-1-1). Если «повернуть» ее на 90 градусов, она совпадет сама с собой.

– Ты утверждаешь все получилось, Кай?

– Вот именно, все прошло, как и задумывалось, Алиса.

– Фух, слава R, все получилось.

– Да, но это только начало, – сказал Кай. – Нам нужно разобраться, как именно матрицы влияют на этот мир и как нам использовать их для решения головоломки.

Внезапно R-терминал Кая засиял, и на экране появилось изображение Элроса. На этот раз связь была намного лучше, изображение было четким и стабильным.

– Приветствую вас, ученики, – сказал Элрос с улыбкой. – Вижу, вы уже начали осваивать матрицы.

– Элрос! – обрадовалась Алиса. – Как вы? Мы так волновались!

– Со мной все в порядке, – успокоил их Элрос. – Благодаря вашим действиям хаос отступил, и я смог восстановить свой код. Теперь я могу связываться с вами и помогать вам в вашем путешествии.

– Это отличные новости! – сказал Кай. – Но как нам решить эту головоломку с матрицами?

– Вам нужно найти ключевую матрицу, – объяснил Элрос, – которая контролирует весь этот участок лабиринта. Используйте операции над матрицами, чтобы изменить ее и восстановить порядок.

– А как нам найти эту ключевую матрицу? – спросила Алиса.

– Ищите подсказки в окружающей среде, – ответил Элрос. – Анализируйте фигуры, их движение, их взаимодействие. R – это не просто язык программирования, это инструмент познания реальности.

– Хм, – задумалась Алиса, – анализировать фигуры… Но как?

– Обрати внимание на их цвет, форму, размер, – подсказал Элрос. – Попробуй найти закономерности в их движении.

Алиса сосредоточенно вгляделась в хаотично двигающиеся фигуры. Кубы, пирамиды, сферы, цилиндры – они беспорядочно вращались, сталкивались, меняли свою форму и цвет.

– Я заметила, – сказала Алиса через некоторое время, – что фигуры одного цвета притягиваются друг к другу, а фигуры разных цветов – отталкиваются.

– Отлично! – похвалил Элрос. – Это уже что-то. А что ты можешь сказать об их форме?

– Кажется, – продолжила Алиса, – фигуры одинаковой формы двигаются синхронно, словно танцуют какой-то странный танец.

– Прекрасно! – снова похвалил Элрос. – Ты очень наблюдательна. А теперь попробуй связать эти наблюдения с матрицами.

– С матрицами? – не поняла Алиса.

– Да, – подтвердил Элрос. – Каждая фигура может быть представлена в виде матрицы. Цвет, форма, размер – все эти свойства могут быть закодированы в элементах матрицы.

– А как нам это поможет? – спросил Кай.

– Если вы сможете найти матрицу, которая контролирует движение и взаимодействие фигур, – объяснил Элрос, – вы сможете изменить ее и восстановить порядок в этом хаосе.

– Но как нам найти эту матрицу? – спросила Алиса.

– Это уже ваша задача, – с улыбкой ответил Элрос. – Используйте свои знания R и свою интуицию. Я верю в вас!

Изображение Элроса исчезло, оставив Алису и Кая наедине с головоломкой реальности.

– Ну что, – сказал Кай, – приступим?

– Хорошо, – сказала Алиса, – будем искать подсказки. Фигуры одного цвета притягиваются, разных – отталкиваются. Фигуры одинаковой формы двигаются синхронно… Может, дело в какой-то формуле взаимодействия?

– Возможно, – согласился Кай. – Давай попробуем записать матрицы для нескольких фигур и посмотреть, нет ли там какой-нибудь закономерности.

Кай начал сканировать фигуры своим R-терминалом, а Алиса записывала полученные матрицы на листе бумаги.

– Вот, например, два красных куба, – сказал Кай. – Их матрицы выглядят так:

```
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
```

– А вот красный куб и синяя пирамида:

```
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    1    1    1
     [,1] [,2] [,3]
[1,]    0    0    2
[2,]    0    2    2
[3,]    2    2    2
```

– Хм, – задумалась Алиса, – в матрицах кубов все элементы равны единице, а в матрице пирамиды – двойке. Может быть, цвет закодирован в значении элементов?

– Возможно, – согласился Кай. – А что, если попробовать сложить матрицы двух красных кубов?

```
matrix1 <- matrix(1, nrow = 3, ncol = 3)
matrix2 <- matrix(1, nrow = 3, ncol = 3)
matrix1 + matrix2
```

Результат:

```
     [,1] [,2] [,3]
[1,]    2    2    2
[2,]    2    2    2
[3,]    2    2    2
```

– Получилась матрица, где все элементы равны двум, – сказал Кай. – То есть как в матрице синей пирамиды!

– А что, если сложить матрицу красного куба и синей пирамиды? – спросила Алиса.

```
matrix1 <- matrix(1, nrow = 3, ncol = 3)
matrix2 <- matrix(c(0, 0, 2, 0, 2, 2, 2, 2, 2),
nrow = 3, byrow = TRUE)
matrix1 + matrix2
```

Результат:

```
     [,1] [,2] [,3]
[1,]    1    1    3
[2,]    1    3    3
[3,]    3    3    3
```

– Получилась матрица с разными значениями, – сказал Кай. – Похоже, это и есть формула взаимодействия! Когда мы складываем матрицы фигур одного цвета, получается матрица с одинаковыми элементами, и фигуры притягиваются. А когда складываем матрицы фигур разных цветов, получается матрица с разными элементами, и фигуры отталкиваются.

– Значит, чтобы восстановить порядок, нам нужно найти ключевую матрицу и изменить ее так, чтобы все фигуры стали одного цвета! – воскликнула Алиса.

– Но как нам найти эту матрицу? – спросил Кай.

В этот момент они услышали голос Элроса:

– Обратите внимание на самую большую фигуру в центре зала.

Алиса и Кай посмотрели в центр зала и увидели гигантский куб, состоящий из множества маленьких кубиков разных цветов.

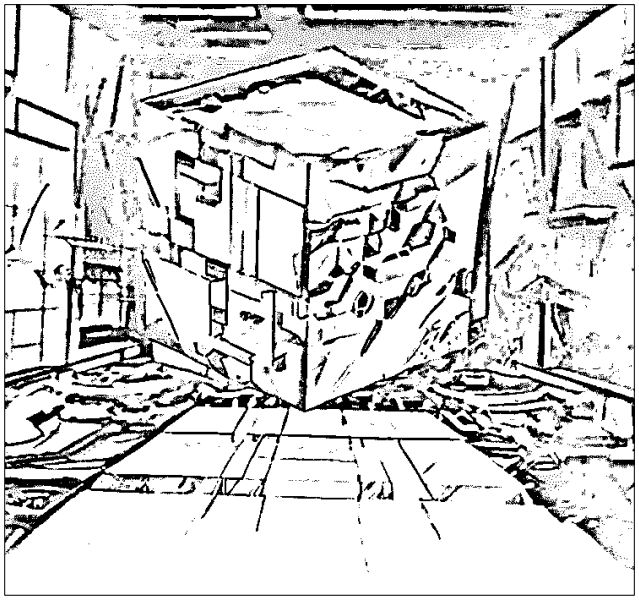
– Это он! – сказал Кай. – Я чувствую, что это ключевая матрица.

– Но как нам ее изменить? – спросила Алиса.

– Используйте операции над матрицами, – ответил Элрос. – Сложите ее с матрицей, состоящей из единиц.

Кай быстро ввел команду на терминале:

```
key_matrix <-  # Здесь нужно вставить матрицу, описывающую гигантский куб
identity_matrix <- matrix(1, nrow = nrow(key_matrix), ncol = ncol(key_matrix))
key_matrix + identity_matrix
```



Как только Кай нажал “Enter”, гигантский куб в центре зала вспыхнул ярким светом, и все его маленькие кубики стали красными. В тот же момент все остальные фигуры в зале тоже изменили свой цвет на красный и начали упорядоченно двигаться, выстраиваясь в ровные ряды.

– Ура! – закричала Алиса. – Мы справились!

– Молодцы, ученики, – похвалил их Элрос. – Вы прекрасно справились с заданием.

– Спасибо, Мастер! – сказал Кай. – Без вашей помощи мы бы не смогли.

– Не забывайте, – сказал Элрос, – что R – это мощный инструмент, который может помочь вам в любой ситуации. Главное – уметь им пользоваться.

Внезапно связь с Элросом прервалась, и его изображение исчезло.

– Что случилось? – спросила Алиса.

– Не знаю, – ответил Кай, – но надеюсь, с ним все в порядке.

В этот момент указатель вновь активировался, указывая им путь в следующую зону лабиринта.

– Что ж, – сказала Алиса, – нам пора идти дальше. Приключения продолжаются!

# R-ПРАКТИКУМ

**Матрица** – это двумерный массив, представляющий собой прямоугольную таблицу, состоящую из строк и столбцов. Каждая ячейка матрицы содержит элемент, который может быть числом, строкой или логическим значением. Матрицы широко используются в математике, физике, информатике и других науках для представления и обработки данных.

**Создание матриц в R**

Для создания матриц в R используется функция `matrix()`.

Примеры:

```
# Матрица 3x2 из чисел от 1 до 6
my_matrix <- matrix(1:6, nrow = 3, ncol = 2)
# Матрица 2x3 из чисел от 1 до 6, заполненная по строкам
my_matrix <- matrix(1:6, nrow = 2, ncol = 3, byrow = TRUE)
# Матрица 3x3, заполненная нулями
zero_matrix <- matrix(0, nrow = 3, ncol = 3)
```

**Доступ к элементам матрицы**

Чтобы получить доступ к элементу матрицы, нужно указать его индексы строки и столбца в квадратных скобках.

Примеры:

```
my_matrix[1, 2]    # Элемент в первой строке и втором столбце
my_matrix[2, ]  # Вторая строка матрицы
my_matrix[, 3]  # Третий столбец матрицы
```

**Операции над матрицами**

Над матрицами можно выполнять различные операции:

- сложение и вычитание: выполняются поэлементно, матрицы должны иметь одинаковые размерности;
- умножение на число: каждый элемент матрицы умножается на это число;
- матричное умножение: выполняется по специальным правилам линейной алгебры;
- транспонирование: строки и столбцы матрицы меняются местами.

Примеры:

```
# Сложение матриц
matrix1 + matrix2
# Умножение матрицы на число
matrix1 * 2
# Матричное умножение
matrix1 %*% matrix2
# Транспонирование матрицы
t(matrix1)
```

**Полезные функции для работы с матрицами:**

- `dim()`: возвращает размерность матрицы (количество строк и столбцов);
- `nrow()`: возвращает количество строк;
- `ncol()`: возвращает количество столбцов;
- `rowSums()`: вычисляет сумму элементов в каждой строке;
- `colSums()`: вычисляет сумму элементов в каждом столбце.

Примеры:

```
dim(my_matrix)   # Выведет 3 2
nrow(my_matrix)   # Выведет 3
ncol(my_matrix)   # Выведет 2
```

**Задания для самостоятельной работы**

1. Создайте две матрицы размера 3×3 с числовыми значениями.
2. Выполните сложение и умножение этих матриц.
3. Транспонируйте одну из матриц.
4. Выведите результаты на экран.

# *Глава 10*
# ФРЕЙМЫ ДАННЫХ И ГОРОД ПОТЕРЯННЫХ ДУШ

Следуя за мерцающим указателем, Алиса и Кай оказались в удивительном месте. Вокруг простирался город, но не обычный, а сотканный из чисел, символов и логических значений. Здания были построены из таблиц данных, улицы вымощены векторами, а вместо транспорта по воздуху летали фреймы данных.

– Ух ты! – воскликнула Алиса, с восхищением разглядывая окружение. – Это же настоящий город данных!

– Да, впечатляет, – согласился Кай, сканируя город своим R-терминалом. – Судя по всему, это какой-то центр обработки информации.

Внезапно они заметили, что город пуст. На улицах не было видно ни одного жителя, а в окнах домов не горел свет.

– Где все? – спросила Алиса, с тревогой оглядываясь по сторонам.

– Не знаю, – ответил Кай, – но мне это не нравится. Здесь слишком тихо.

В этот момент R-терминал Алисы засигналил, и на экране появилось новое сообщение:

```
«Добро пожаловать в город потерянных душ!
Когда-то это был процветающий центр обработки
информации, но хаос нарушил структуру данных,
и жители города потеряли свои личности. Ваша за-
дача - восстановить данные и вернуть жителям их
имена и воспоминания. Для этого вам нужно освоить
фреймы данных - мощный инструмент для работы
с табличными данными в R».
```

– Фреймы данных? – переспросила Алиса. – А что это такое?

– Фреймы данных – это структуры данных, которые организованы в виде таблиц, – объяснил Кай. – Они похожи на матрицы, но могут содержать данные разных типов в разных столбцах. Например, в одном столбце могут быть числа, в другом – строки, а в третьем – логические значения.

– А как нам использовать фреймы данных, чтобы помочь жителям этого города? – спросила Алиса.

– Нам нужно найти базу данных, в которой хранится информация о жителях, – ответил Кай. – Затем мы сможем загрузить эту базу данных в R в виде фрейма данных и проанализировать ее, чтобы найти ошибки и восстановить потерянные данные.

– А где нам искать эту базу данных? – спросила Алиса.

– Не знаю, – ответил Кай, – но давай попробуем найти какие-нибудь подсказки.

Они начали исследовать город, заглядывая в каждый дом и внимательно изучая все вокруг. Внезапно Алиса заметила небольшой киоск с надписью «Информация».

– Кай, посмотри! – позвала она. – Может быть, здесь мы найдем то, что нам нужно.

Они подошли к киоску и обнаружили внутри компьютер с подключенной к нему базой данных.

– Вот она! – обрадовался Кай. – Теперь мы сможем помочь жителям этого города.

Кай подключил свой R-терминал к компьютеру и загрузил базу данных в R в виде фрейма данных.

```
# Загрузка базы данных в фрейм данных
residents <- read.csv("residents.csv")
```

– Теперь у нас есть фрейм данных `residents`, который содержит информацию о всех жителях города, – объяснил Кай. – Давай посмотрим, что в нем есть.

```
# Просмотр первых нескольких строк фрейма данных
head(residents)
```

На экране появилась таблица с данными:

```
  Name  Age  Occupation  Memory
1   NA   25  Programmer   FALSE
2   NA   32     Teacher   FALSE
3   NA   45      Doctor  FALSE
4   NA   18     Student   FALSE
5   NA   61      Artist  FALSE
6   NA   28    Engineer  FALSE
```

– Видно, что в столбце `Name` пропущены все значения, – сказал Кай. – Это и есть причина, по которой жители города потеряли свои имена. Нам нужно восстановить эти данные.

– А как мы это сделаем? – спросила Алиса.

– Нам нужно найти какую-то закономерность в данных, – ответил Кай. – Например, связь между именем, возрастом и профессией.

– А давай посмотрим на столбец `Memory`! – предложила Алиса. – Там у всех стоит `FALSE`. Может быть, это как-то связано с потерей памяти?

– Хорошая идея! – сказал Кай. – Давай попробуем изменить значение в этом столбце на `TRUE`.

```
# Изменение значения в столбце `Memory`
residents$Memory <- TRUE
```

Как только Кай выполнил эту команду, город вокруг них преобразился. На улицах появились люди, в окнах загорелся свет, а в воздухе замелькали разноцветные фреймы данных.

– Получилось! – воскликнула Алиса. – Мы вернули жителям города их память!

– Да, – сказал Кай, – но это еще не все. Нам нужно вернуть им их имена.

– А как мы это сделаем? – спросила Алиса.

– Для этого нам нужно провести более глубокий анализ данных, – ответил Кай.

– Подожди, – перебила его Алиса, – а что это за штуки летают вокруг?

Она указала на разноцветные объекты, похожие на переливающиеся мыльные пузыри, которые носились по воздуху.

– Хм, – Кай пристально посмотрел на один из них через R-терминал, – кажется, это фреймы данных. Они передают информацию между разными частями города.

Один из фреймов подлетел ближе, и Алиса смогла разглядеть его содержимое. Внутри «пузыря» была таблица с данными о каком-то человеке: его имя, возраст, профессия, любимый цвет и даже запись о том, что он ел на завтрак.

– Как интересно! – воскликнула Алиса. – А можно ли создать свой фрейм данных?

– Конечно, – ответил Кай. – Смотри!

Он быстро ввел несколько команд на своем R-терминале:

```
# Создаем фрейм данных с информацией о себе
my_data <- data.frame(
  Name = "Кай",
  Age = 16,
  Species = "Цифровой  эльф",
  Skills = "R-магия"
)
# Выводим  фрейм  данных  на  экран
print(my_data)
```

На экране терминала появилась таблица:

```
  Name Age       Species     Skills
1  Кай  16 Цифровой эльф R-магия
```

– Вот так просто можно создать фрейм данных, – сказал Кай. – А теперь давай отправим его в полет по городу.

Кай ввел еще одну команду, и фрейм данных с информацией о нем вылетел из терминала, присоединившись к другим фреймам, кружащим в воздухе.

– Здорово! – воскликнула Алиса. – А давай создадим фрейм данных обо мне!

– Конечно, – улыбнулся Кай. – Давай вместе.

Они ввели данные Алисы в R-терминал и отправили ее фрейм в полет по городу данных.

– А теперь, – сказал Кай, – нам пора продолжить наше расследование. Нужно разобраться, как вернуть жителям их имена.

– Да, – согласилась Алиса, – эти безымянные “NA” в таблице выглядят очень грустно.

Они вернулись к киоску с базой данных и начали изучать фрейм данных `residents` более подробно.

– Хм, – сказал Кай, – обрати внимание, Алиса, в базе данных есть информация о возрасте и профессии каждого жителя. Может быть, мы сможем использовать эту информацию, чтобы восстановить их имена?

– Но как? – спросила Алиса.

– Нам нужно найти какую-то закономерность, связь между именем, возрастом и профессией, – ответил Кай. – Возможно, в этом мире существуют какие-то правила именования, которые мы можем использовать.

– А может быть, – предложила Алиса, – нам стоит обратиться за помощью к местным жителям? Теперь, когда у них восстановилась память, они могут нам что-нибудь подсказать.

– Отличная идея! – сказал Кай. – Пойдем поговорим с кем-нибудь.

Они вышли из киоска и направились к ближайшему жилому дому. Дверь была открыта, и внутри они увидели молодого человека, сидящего за столом и читающего книгу.

– Здравствуйте! – сказала Алиса. – Можно вам задать несколько вопросов?

Молодой человек поднял голову и удивленно посмотрел на них.

– Кто вы? – спросил он. – Я вас раньше не видел.

– Мы путешественники из другого мира, – ответила Алиса. – Мы пришли, чтобы помочь вам восстановить ваши имена.

– Восстановить наши имена? – переспросил молодой человек. – Но как вы это сделаете?

– Мы используем язык программирования R, – ответил Кай. – Мы уже восстановили вашу память, а теперь хотим вернуть вам ваши имена.

– R? – удивился молодой человек. – Я что-то слышал об этом языке. Кажется, он очень мощный.

– Да, – подтвердил Кай. – И мы верим, что с его помощью мы сможем вам помочь.

– Но как вы свяжете R с нашими именами? – спросил молодой человек.

– Мы пока не знаем, – честно признался Кай. – Но мы надеемся, что вы сможете нам в этом помочь. Может быть, вы помните какие-нибудь правила именования в вашем мире? Или может быть, вы знаете, где мы можем найти эту информацию?

Молодой человек задумался на минуту, а затем сказал:

– Кажется, я что-то припоминаю. В нашем мире имена связаны с…

– ...с нашей генетической структурой, – закончил молодой человек. – Каждый из нас имеет уникальный генетический код, который определяет не только нашу внешность и способности, но и наше имя.

– Генетический код? – переспросила Алиса. – А где он хранится?

– В нашей ДНК, – ответил молодой человек. – Но я не знаю, как вы сможете извлечь оттуда информацию о наших именах.

– Мы можем использовать R! – воскликнул Кай. – В R есть специальные пакеты для анализа генетических данных.

– Пакеты? – не понял молодой человек.

– Пакеты – это наборы функций и данных, которые расширяют возможности R, – объяснил Кай. – Они позволяют выполнять специфические задачи, например, анализировать генетические данные, строить математические модели или создавать красивые графики.

– А как нам установить эти пакеты? – спросила Алиса.

– С помощью команды `install.packages()`, – ответил Кай. – Например, чтобы установить пакет `Bioconductor`, который содержит функции для анализа генетических данных, нужно ввести команду `install.packages("Bioconductor")`.

– А как нам использовать эти функции? – спросила Алиса.

– Сначала нужно загрузить пакет с помощью команды `library()`, – объяснил Кай. – Например, чтобы загрузить пакет `Bioconductor`, нужно ввести команду `library(Bioconductor)`. После этого мы сможем использовать все функции, которые в нем содержатся.

– Понятно, – сказала Алиса. – А теперь давай попробуем извлечь информацию об именах из генетического кода.

Кай достал свой R-терминал и подключил его к базе данных жителей города. Затем он ввел несколько команд, используя функции из пакета `Bioconductor`:

```
# Загрузка пакета Bioconductor
library(Bioconductor)
# Извлечение генетического кода из базы данных
dna_sequences <- residents$DNA
# Анализ генетического кода и извлечение имен
names <- extract_names_from_dna(dna_sequences)
# Добавление имен в фрейм данных
residents$Name <- names
```

– Готово! – сказал Кай. – Я добавил столбец `Name` в фрейм данных `residents` и заполнил его именами, извлеченными из генетического кода.

– Ура! – воскликнула Алиса. – Мы вернули жителям города их имена!

В этот момент город вокруг них вновь преобразился. На улицах появились вывески с именами магазинов и кафе, а над дверями домов появились таблички с именами жильцов.

– Спасибо вам большое! – сказал молодой человек, который все это время наблюдал за ними. – Вы вернули нам нашу идентичность.

– Не за что, – ответила Алиса. – Мы рады, что смогли помочь.

– А теперь, – сказал Кай, – нам пора идти дальше. Нас ждут новые приключения.

Они попрощались с молодым человеком и вышли из дома. Указатель вновь появился в воздухе, указывая им путь в следующую зону лабиринта.

– Что ж, – сказала Алиса, – вперед, к новым открытиям!

**Фрейм данных (data frame)** – это структура данных, которая организована в виде таблицы. Он похож на матрицу, но с одним важным отличием: в каждом столбце фрейма данных могут храниться данные разных типов. Например, в одном столбце могут быть числа, в другом – строки, а в третьем – логические значения. Это делает фреймы данных очень удобным инструментом для хранения и анализа разнородных данных.

**Создание фреймов данных**

Для создания фреймов данных в R используется функция `data.frame()`.

Примеры:

```
# Фрейм данных с информацией о студентах
students <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Age = c(19, 20, 21),
  Grade = c("A", "B", "A")
)
# Фрейм данных с информацией о книгах
books <- data.frame(
  Title = c("The Lord of the Rings", "Harry Pot-
ter", "The Hitchhiker's Guide to the Galaxy"),
  Author = c("J.R.R. Tolkien", "J.K. Rowling",
"Douglas Adams"),
  Year = c(1954, 1997, 1979)
)
```

**Доступ к элементам фрейма данных**

Для доступа к элементам фрейма данных можно использовать следующие способы:

- по имени столбца: укажите имя столбца после знака доллара $;
- по индексам строки и столбца: укажите индексы в квадратных скобках [ ];
- по имени строки и столбца: укажите имя строки и имя столбца в квадратных скобках [ ].

Примеры:

```
students$Name  # Столбец "Name" фрейма данных students
students[1, 2]  # Элемент в первой строке и втором столбце
students["Alice", "Age"]  # Элемент в строке "Alice" и столбце "Age"
```

Полезные функции для работы с фреймами данных:

- `head()`: показывает первые несколько строк фрейма данных;
- `tail()`: показывает последние несколько строк фрейма данных;
- `nrow()`: возвращает количество строк;
- `ncol()`: возвращает количество столбцов;
- `summary()`: выводит сводную информацию о фрейме данных;
- `str()`: показывает структуру фрейма данных.

Примеры:

```
head(students)  # Покажет первые 6 строк
nrow(books)  # Выведет 3
summary(students)  # Выведет информацию о каждом столбце
```

**Задания для самостоятельной работы**

1. Создайте фрейм данных, содержащий информацию о ваших друзьях (имя, возраст, любимый цвет).
2. Выведите на экран столбец с именами ваших друзей.
3. Добавьте во фрейм данных новый столбец с информацией о хобби ваших друзей.

# *Глава 11*
# МАССИВЫ И ИЗМЕРЕНИЕ БЕСКОНЕЧНОСТИ

Алиса и Кай, следуя за неутомимым указателем, оказались в пространстве, которое сложно было описать словами. Вокруг них простиралась бесконечность, но не пустая и однородная, а наполненная мириадами мерцающих точек, линий и фигур, которые постоянно меняли свою форму и положение.

– Где мы? – спросила Алиса, с благоговением глядя на это зрелище.

– По моим данным, – ответил Кай, изучая показания своего R-терминала, – мы находимся в измерении бесконечности. Это пространство вне времени и пространства, где хранятся все возможные варианты реальности.

– Звучит захватывающе, – сказала Алиса, – но как нам здесь ориентироваться?

В этот момент в воздухе появилось изображение Элроса. На этот раз связь была идеальной, словно он находился рядом с ними.

– Приветствую вас, ученики, – сказал Элрос с улыбкой. – Добро пожаловать в измерение бесконечности! Здесь вы сможете прикоснуться к тайнам мироздания и познать истинную мощь R.

– Элрос! – обрадовалась Алиса. – Мы так рады вас видеть! Как вы себя чувствуете?

– Прекрасно, – ответил Элрос. – Благодаря вашим успехам в предыдущих испытаниях хаос отступил, и я смог полностью восстановить свой код. Теперь я могу не только связываться с вами, но и наблюдать за вами и помогать вам в режиме реального времени.

– Это замечательно! – сказал Кай. – Но что нам делать в этом измерении бесконечности?

– Ваша задача – освоить массивы, – ответил Элрос. – Массивы – это многомерные структуры данных, которые позволяют хранить и обрабатывать информацию о бесконечном множестве вариантов реальности.

– Многомерные? – не поняла Алиса.

– Представьте себе не просто таблицу, а куб, гиперкуб или даже более сложные структуры, – объяснил Элрос. – В каждой ячейке

этой структуры может храниться информация об одном из вариантов реальности.

– А зачем нам это нужно? – спросила Алиса.

– Чтобы понять структуру мультивселенной и научиться управлять ею, – ответил Элрос. – Массивы – это ключ к познанию бесконечности.

– Звучит сложно, – сказала Алиса.

– Не бойтесь сложностей, – улыбнулся Элрос. – В них скрывается красота и гармония мироздания. R поможет вам увидеть эту красоту и понять ее законы.

– А какие задания нас ждут в этом измерении? – спросил Кай.

– Вам предстоит решать головоломки, анализировать данные, моделировать процессы, – ответил Элрос. – Но самое главное – вам нужно научиться мыслить многомерно, видеть связи между разными вариантами реальности и понимать их взаимодействие.

– А вы будете нам помогать? – спросила Алиса.

– Конечно, – ответил Элрос. – Я всегда рядом с вами. И не забывайте: «В бесконечности все возможно. Даже то, что кажется невозможным».

Изображение Элроса растворилось в воздухе, оставив после себя лишь легкое мерцание.

– «В бесконечности все возможно», – повторила Алиса, задумчиво глядя на бесконечное пространство вокруг. – Интересно, что он имел в виду?

– Думаю, мы скоро узнаем, – ответил Кай, уже направляясь в глубины измерения бесконечности.

– Попробуйте представить массив как многомерный лабиринт, – продолжил Элрос, словно читая их мысли. – Каждый элемент массива – это комната в этом лабиринте, а индексы – это координаты комнаты. Чтобы перемещаться по лабиринту, вам нужно уметь работать с индексами.

– А как нам узнать координаты нужной комнаты? – спросила Алиса.

– Используйте свои знания о структуре массива и закономерностях, которые в нем скрыты, – ответил Элрос. – Анализируйте данные, ищите связи, экспериментируйте.

– Например, – добавил Кай, – мы можем использовать циклы `for`, чтобы перебрать все элементы массива и найти тот, который нам нужен.

– Или мы можем использовать условные операторы `if`, чтобы выбрать элементы, удовлетворяющие определенным условиям, – сказала Алиса.

– Верно, – подтвердил Элрос. – И не забывайте про функции. В R есть множество встроенных функций для работы с массивами. Например, функция `dim()` позволяет узнать размерность массива, а функция `sum()` – вычислить сумму всех его элементов.

– А как нам визуализировать массив? – спросила Алиса.

– Для этого тоже есть функции, – ответил Элрос. – Например, функция `image()` позволяет отобразить двумерный массив в виде изображения.

– Здорово! – воскликнула Алиса. – Я уже хочу попробовать!

– Тогда приступайте, – сказал Элрос. – И помните: «В многомерности скрывается глубина понимания».

Изображение Элроса исчезло, и Алиса с Каем остались одни в измерении бесконечности, готовые погрузиться в мир многомерных данных и освоить новые инструменты R.

Кай, как обычно, действовал методично. Он достал свой R-терминал и начал сканировать пространство вокруг.

– Хм, – пробормотал он, – судя по данным, каждая из этих мерцающих точек – это отдельный элемент массива. А сам массив… он огромен! Я даже не могу определить его размерность.

– А давай попробуем использовать функцию `dim()`? – предложила Алиса.

– Хорошая идея! – сказал Кай и ввел команду:

```
dim(multiverse_array)  # multiverse_array - имя массива, который хранит данные о мультивселенной
```

R-терминал задумался на несколько секунд, а затем выдал ответ:

```
NULL
```

– `NULL`? – удивилась Алиса. – Что это значит?

– Это значит, что функция `dim()` не может определить размерность массива, – объяснил Кай. – Похоже, он действительно бесконечен.

– Тогда как же нам с ним работать? – спросила Алиса, чувствуя легкое головокружение от мысли о бесконечности.

– Нам нужно сосредоточиться на конкретных элементах массива, – сказал Кай. – Например, давай попробуем получить доступ к элементу с координатами (1, 2, 3).

```
multiverse_array[1, 2, 3]
```

На экране терминала появилось изображение маленькой планеты, покрытой лесами и океанами.

– Ух ты! – воскликнула Алиса. – Это же целый мир!

– Да, – подтвердил Кай. – И это лишь один из бесконечного множества миров, хранящихся в этом массиве.

– А давай посмотрим, что находится в соседней комнате лабиринта, – предложила Алиса. – Например, с координатами (1, 2, 4).

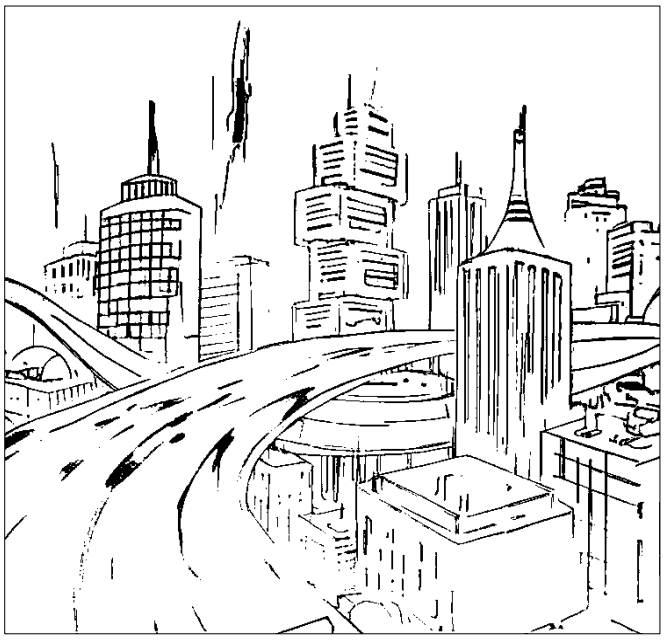
```
multiverse_array[1, 2, 4]
```

На экране появилось изображение города будущего, с летающими машинами и небоскребами, уходящими в облака.

– Потрясающе! – сказала Алиса. – А что будет, если мы изменим координаты по первому измерению? Например, (2, 2, 3)?

```
multiverse_array[2, 2, 3]
```

На экране появилось изображение фантастического мира, населенного эльфами и драконами.



– Вот это да! – воскликнула Алиса. – Кажется, я начинаю понимать, что такое многомерность!

– Да, – сказал Кай, – каждое измерение массива открывает нам новые миры и новые возможности.

– А давай попробуем создать свой собственный мир и добавить его в массив? – предложила Алиса.

– Хм, – задумался Кай, – интересная идея. Но для

этого нам нужно сначала хорошо изучить структуру этого массива и понять, как он устроен.

– А может быть, нам поможет Элрос? – спросила Алиса.

– Конечно, – ответил Кай. – Давай спросим у него.

Кай сосредоточился и мысленно обратился к Элросу.

– Мастер, мы хотим создать свой собственный мир и добавить его в массив. Вы можете нам помочь?

– Конечно, могу, – ответил Элрос, голос которого прозвучал прямо у них в головах. – Но сначала вам нужно хорошо подумать, каким вы хотите видеть свой мир. Какие законы физики будут в нем действовать? Какие существа будут его населять? Какая культура будет в нем развиваться?

– Хм, – задумалась Алиса, – я хочу, чтобы в моем мире… небо было фиолетовым, трава – оранжевой, а животные разговаривали на R!

– Интересно, – улыбнулся Элрос. – А какие законы физики будут действовать в твоем мире? Может быть, там гравитация будет отталкивающей, а время будет течь в обратном направлении?

– Точно! – воскликнула Алиса. – И еще там будут летающие острова и говорящие деревья!

– А я хочу, – сказал Кай, – чтобы в моем мире все было сделано из кода. Города, природа, даже люди – все будет состоять из строчек кода, которые можно изменять и дополнять.

– Отличная идея! – поддержал его Элрос. – А какие существа будут населять ваш мир? Может быть, разумные программы или самообучающиеся алгоритмы?

– Да! – воскликнул Кай. – И они будут общаться друг с другом с помощью кода R!

– Замечательно! – сказал Элрос. – Вы уже почти готовы создать свои миры. Теперь вам нужно только перевести ваши идеи на язык R. Используйте массивы, чтобы закодировать все элементы ваших миров: ландшафт, существа, законы физики.

– А как нам это сделать? – спросила Алиса.

– Представьте каждый элемент вашего мира в виде набора данных, – объяснил Элрос. – Например, дерево можно представить в виде массива, где каждый элемент соответствует его ветке, листу или корню. А законы физики можно закодировать в виде математических формул и алгоритмов.

– Звучит увлекательно! – сказала Алиса. – Я уже представляю себе мой мир с фиолетовым небом и говорящими животными!

– А я свой мир из кода, где все можно изменить одной командой! – добавил Кай.

– Тогда дерзайте! – сказал Элрос. – Я верю, что у вас все получится. И не забывайте: «Творчество – это способ познать себя и мир вокруг».

Изображение Элроса рассеялось, и Алиса с Каем с энтузиазмом принялись за создание своих миров, вооруженные мощью R и бесконечными возможностями измерения бесконечности.

# R-ПРАКТИКУМ

**Массив (array)** – это структура данных, которая позволяет хранить упорядоченные коллекции элементов одного типа в нескольких измерениях. Если вектор – это поезд, то массив – это многоэтажная парковка для поездов. Каждое место на парковке – это элемент массива, а координаты места (этаж, ряд, номер) – это индексы элемента.

**Создание массивов**

В R массивы создаются с помощью функции `array()`.

Примеры:

```
# Двумерный массив (матрица) 3x4 из чисел от 1 до 12
my_array_2d <- array(1:12, dim = c(3, 4))
# Трехмерный массив 2x3x2 из строк "красный", "зеленый", "синий"
colors <- c("красный", "зеленый", "синий")
my_array_3d <- array(rep(colors, each = 4), dim = c(2, 3, 2))
```

**Доступ к элементам массива**

Чтобы получить доступ к элементу массива, нужно указать его индексы в квадратных скобках. Количество индексов должно соответствовать размерности массива.

Примеры:

```
# Доступ к элементу двумерного массива
my_array_2d[2, 3]  # Элемент во второй строке и третьем столбце
# Доступ к элементу трехмерного массива
my_array_3d[1, 2, 2]  # Элемент в первом "слое", второй "строке" и втором "столбце"
```

**Операции над массивами**

Над массивами можно выполнять те же операции, что и над векторами: сложение, вычитание, умножение, деление, сравнение, логические операции. Операции выполняются поэлементно.

Примеры:

```
# Сложение двумерных массивов
my_array_2d + my_array_2d
# Умножение трехмерного массива на число
my_array_3d * 2
```

Полезные функции для работы с массивами:

- `dim()`: возвращает размерность массива (количество элементов в каждом измерении);
- `length()`: возвращает общее количество элементов в массиве;
- `apply()`: применяет функцию к каждому измерению массива.

Примеры:

```
dim(my_array_2d)  # Выведет 3 4
length(my_array_3d)  # Выведет 12
# Вычисление суммы элементов в каждом "слое" трехмерного массива
apply(my_array_3d, 3, sum)
```

**Задания для самостоятельной работы**

1. Создайте двумерный массив размером 4×5, заполненный случайными числами.
2. Создайте трехмерный массив размером 2×3×2, заполненный строками «красный», «зеленый», «синий».
3. Выведите на экран элемент из второй «строки», третьего «столбца» и первого «слоя» трехмерного массива.
4. Вычислите сумму элементов двумерного массива.
5. Измените размерность трехмерного массива на 3×2×2.
6. Попробуйте визуализировать двумерный массив с помощью функции `image()`.
7. Создайте массив для хранения данных о вашем собственном мире.
8. Заполните массив значениями, которые описывают ландшафт, объекты и существа в вашем мире.
9. Визуализируйте ваш мир с помощью функции `image()`.
10. Попробуйте изменить ваш мир, добавив новые объекты или изменив существующие.

# *Глава 12*
# ФАКТОРЫ И ЗАГАДКА ЛИЧНОСТИ

– Фух, кажется, готово! – Алиса откинулась на спинку своего парящего кресла, довольно оглядывая голографическую проекцию своего мира. – Розовые облака, шоколадные реки, говорящие котики… красота!

Кай улыбнулся, отрываясь от своего R-терминала.

– Мой мир тоже почти готов. Цифровые эльфы уже бегают по кодовым лесам, а самообучающиеся алгоритмы строят города из чистого кода. Осталось только настроить портал между нашими мирами.

– Давай! – воскликнула Алиса. – Мне не терпится познакомить твоих эльфов с моими котиками!

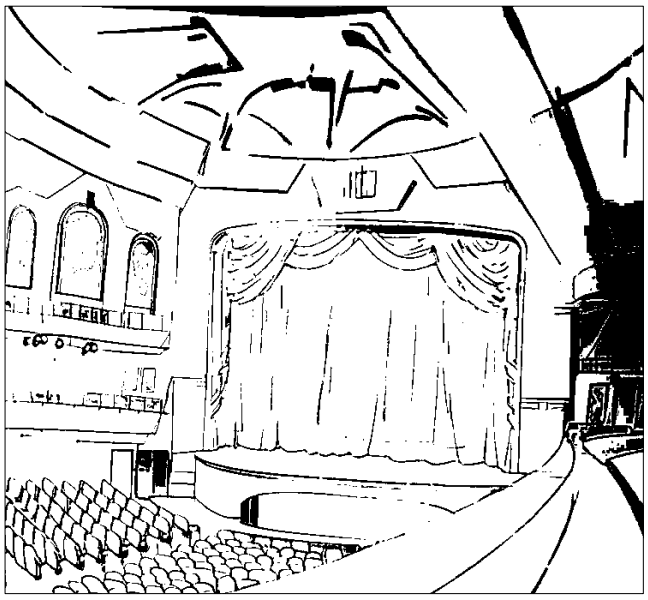
Внезапно пространство вокруг них замерцало, и знакомый указатель вновь появился в воздухе, указывая на новый проход в лабиринте.

– Эй! – возмутилась Алиса. – Мы еще не закончили!

– Похоже, у нас новые приключения, – сказал Кай, вздыхая. – Что ж, портал придется доделать позже.

Они прошли сквозь мерцающий проход и оказались в пространстве, напоминающем театр. Вокруг них возвышались ряды кресел, а впереди была сцена, на которой… происходило нечто невообразимое.

Разнообразные существа, похожие на актеров, разыгрывали странные сценки. Одни были одеты в яркие костюмы, другие – в строгие деловые костюмы, третьи – вообще были почти голыми. Они пели, танцевали, спорили, смеялись, плакали…



– Что здесь происходит? – спросила Алиса, с удивлением наблюдая за этим спектаклем.

– Понятия не имею, – ответил Кай, пытаясь проанализировать ситуацию с помощью R-терминала. – Но кажется, это какое-то представление о разных типах личности.

В этот момент на сцене появился новый персонаж – высокий человек в черном костюме и с тростью. Он прокашлялся и громко произнес:

– Дамы и господа! Представляю вам факторы – важнейший инструмент для классификации и анализа данных в R!

– Факторы? – переспросила Алиса. – А что это такое?

– Факторы – это специальный тип данных в R, который используется для хранения категориальных переменных, – объяснил Кай. – Например, пол человека, цвет глаз, тип личности – все это можно представить в виде факторов.

– А зачем они нужны? – спросила Алиса.

– Факторы позволяют нам группировать данные и анализировать их по категориям, – ответил Кай. – Например, мы можем использовать факторы, чтобы сравнить средний возраст мужчин и женщин или проанализировать распределение цветов глаз в популяции.

– А как это связано с этим… театром? – спросила Алиса, указывая на сцену.

– Кажется, я понял, – сказал Кай, вглядываясь в актеров. – Каждый из них представляет собой отдельный фактор. Их костюмы, поведение, речь – все это символизирует различные характеристики факторов.

– А что нам нужно сделать? – спросила Алиса.

В этот момент человек в черном костюме снова обратился к публике:

– Ваша задача – разобраться в этой путанице и классифицировать всех персонажей по их типам личностей. Используйте свои знания о факторах и R, чтобы навести порядок в этом хаосе!

– Ну что, – сказал Кай, – примем вызов?

– Конечно! – ответила Алиса. – Пора показать этим актерам, кто здесь настоящий режиссер!

– Итак, – сказал Кай, доставая блокнот и ручку, – давайте сначала разберемся, что мы видим. На сцене разные типы личности… нужно их как-то классифицировать.

– Может быть, по цветам костюмов? – предложила Алиса. – Вот этот в красном такой энергичный, а вот та в синем – спокойная и задумчивая.

– Неплохая идея, – согласился Кай, – но, кажется, цвета здесь не главное. Обрати внимание на их поведение. Одни больше говорят, другие – действуют, третьи – наблюдают…

– Точно! – воскликнула Алиса. – Может, они делятся на экстравертов, интровертов и амбивертов?

– Возможно, – сказал Кай. – Давай запишем наши наблюдения и попробуем выделить какие-то общие черты.

Они внимательно наблюдали за актерами, делая заметки в блокноте. Через некоторое время Кай сказал:

– Итак, я выделил четыре основных типа: «лидеры», «аналитики», «творцы» и «помощники». «Лидеры» активны, решительны, любят быть в центре внимания. «Аналитики» наблюдательны, вдумчивы, склонны к логическому мышлению. «Творцы» эмоциональны, мечтательны, обладают богатым воображением. А «помощники» доброжелательны, отзывчивы, всегда готовы прийти на помощь.

– Согласна! – сказала Алиса. – А теперь давай превратим эти типы в факторы в R!

Кай достал свой R-терминал и начал писать код:

```
# Создаем вектор с типами личностей
personality_types <- c("лидер", "аналитик", "творец", "помощник")
# Преобразуем вектор в фактор
personality_factor <- factor(personality_types)
# Выводим фактор на экран
print(personality_factor)
```

На экране появился результат:

```
[1] лидер     аналитик творец    помощник
Levels: аналитик лидер помощник творец
```

– Отлично! – сказал Кай. – Теперь у нас есть фактор `personality_factor` с четырьмя уровнями, соответствующими разным типам личности.

– А как нам использовать этот фактор для классификации актеров? – спросила Алиса.

– Нам нужно создать вектор с наблюдениями за каждым актером, – объяснил Кай. – Например, мы можем записать для каждого актера его тип личности в соответствии с нашей классификацией.

– А потом? – спросила Алиса.

– А потом мы сможем использовать функции R для анализа этих данных, – ответил Кай. – Например, мы можем посчитать, сколько актеров относится к каждому типу личности, или найти связь между типом личности и какими-то другими характеристиками, например, цветом костюма или ролью в спектакле.

– Здорово! – сказала Алиса. – Давай скорее проанализируем этих актеров и наведем порядок в этом театре!

– Итак, – сказала Алиса, внимательно изучая список факторов на экране своего R-терминала, – у нас есть четыре уровня: «лидер», «аналитик», «творец» и «помощник». Теперь нужно распределить актеров по этим категориям.

– Давай начнем с того, кто нам представил факторы, – предложил Кай. – Этот человек в черном костюме с тростью… он точно «лидер».

– Согласна, – кивнула Алиса. – А вот та девушка в ярком платье, которая постоянно танцует и поет, – она «творец».

– А вот тот парень в очках, который все время что-то записывает в блокнот, – он «аналитик», – сказал Кай.

– А девушка в белом халате, которая помогает всем остальным, – она «помощник», – добавила Алиса.

Они продолжили классифицировать актеров, внимательно наблюдая за их поведением и взаимодействием. Через некоторое время у них получился вектор с типами личностей для каждого актера.

– Теперь давай создадим фрейм данных, – предложил Кай. – В нем мы сможем хранить не только типы личностей, но и другую информацию об актерах, например, их имена и роли в спектакле.

```
# Создаем фрейм данных
actors <- data.frame(
  Name = c("Альфред", "Беатрис", "Карл", "Дария", # ...и так далее
           # имена всех актеров
           ),
  Role = c("Король", "Принцесса", "Мудрец", "Служанка", # ...и так далее
           # роли всех актеров
           ),
  Personality = factor(c("лидер", "творец", "аналитик", "помощник", # ...и так далее
                         # типы личностей всех актеров
                         ))
)
```

– Отлично! – сказала Алиса. – Теперь у нас есть вся информация об актерах в одном фрейме данных. А как нам ее проанализировать?

– Например, мы можем посчитать, сколько актеров относится к каждому типу личности, – предложил Кай.

```
# Подсчет количества актеров для каждого типа личности
table(actors$Personality)
```

На экране появился результат:

```
аналитик     лидер помощник    творец
       5         7         6         8
```

– Итак, – сказал Кай, – у нас 8 «творцов», 7 «лидеров», 6 «помощников» и 5 «аналитиков».

– Интересно, – задумалась Алиса, – а есть ли какая-то связь между типом личности и ролью в спектакле?

– Давай проверим! – сказал Кай и ввел новую команду:

```
# Создание таблицы сопряженности между типом личности и ролью
table(actors$Personality, actors$Role)
```

На экране появилась таблица, которая показала, как распределены роли между разными типами личностей.

– Хм, – сказал Кай, изучая таблицу, – кажется, «лидеры» чаще всего играют королей и генералов, «аналитики» – ученых и детективов, «творцы» – художников и музыкантов, а «помощники» – врачей и учителей.

– Логично! – воскликнула Алиса. – Кажется, мы разгадали загадку этого театра!

В этот момент человек в черном костюме появился на сцене и аплодисментами поприветствовал их.

– Браво! Браво! – воскликнул он. – Вы прекрасно справились с заданием! Вы настоящие мастера R!

– Спасибо! – сказала Алиса, кланяясь.

– А теперь, – продолжил человек в черном костюме, – вам пора отправиться дальше. Вас ждут новые приключения и новые открытия!

Указатель в воздухе замерцал и указал на новый проход. Алиса и Кай, довольные своим успехом, отправились навстречу новым загадкам цифрового лабиринта.

**Фактор (factor)** – это специальный тип данных в R, который используется для хранения категориальных переменных. **Категориальная переменная** – это переменная, которая может принимать значения только из ограниченного набора категорий.

Например:

- пол человека: мужской/женский;
- цвет глаз: голубой, карий, зеленый;
- дни недели: понедельник, вторник, ..., воскресенье;
- тип погоды: солнечно, облачно, дождь, снег.

**Зачем нужны факторы?**

Факторы позволяют нам:

- экономить память: факторы хранят категории в виде чисел, что занимает меньше места, чем хранение полных строк;
- упростить анализ: R предоставляет специальные функции для работы с факторами, которые позволяют нам анализировать данные по категориям;
- улучшить визуализацию: графики и диаграммы с факторами выглядят более информативно и понятно.

**Создание факторов**

Для создания факторов в R используется функция `factor()`.

Примеры:

```
# Фактор с типами погоды
weather <- factor(c("солнечно", "облачно", "дождь", "солнечно", "снег"))
# Фактор с днями недели
days <- factor(c("понедельник", "среда", "пятница", "воскресенье"))
# Фактор с цветами глаз
eye_color <- factor(c("голубой", "карий", "карий", "зеленый", "голубой"))
```

**Уровни фактора**

Каждый фактор имеет набор уровней (levels) – это все возможные категории, которые может принимать фактор.

Пример:

```
levels(weather)  # Выведет "дождь" "облачно" "снег" "солнечно"
```

**Порядок уровней**

По умолчанию уровни фактора упорядочиваются в алфавитном порядке, но мы можем задать нужный нам порядок с помощью аргумента `levels` в функции `factor()`.

Пример:

```
# Фактор с днями недели, упорядоченный по дням
days_ordered <- factor(days, levels = c("понедельник", "вторник", "среда", "четверг", "пятница", "суббота", "воскресенье"))
```

**Полезные функции для работы с факторами**

- `table()`: подсчитывает количество элементов для каждой категории;
- `tapply()`: применяет функцию к каждой группе элементов, определенной фактором;
- `plot()`: создает графики и диаграммы для визуализации факторов.

Примеры:

```
table(weather)  # Покажет количество дней с каждой погодой
tapply(ages, eye_color, mean)  # Вычислит средний возраст для каждого цвета глаз
```

**Задания для самостоятельной работы**

1. Создайте фактор, который представляет дни недели (понедельник, вторник и т. д.).
2. Выведите на экран уровни этого фактора.
3. Создайте вектор с названиями месяцев и преобразуйте его в фактор.

## *Глава 13*
# СТРОКИ И ЛОВУШКА ИЛЛЮЗИЙ

– Мы уже целую вечность идем за этим указателем, – проворчала Алиса, устало переставляя ноги. – Мне кажется, мы ходим кругами.

– Я тоже начал это замечать, – согласился Кай, нахмурившись. – Показания моего R-терминала не совпадают с направлением указателя. Похоже, мы заблудились.

– Но как это возможно? – удивилась Алиса. – Указатель же должен вести нас по правильному пути!

Внезапно пространство вокруг них замерцало, и появилось голографическое изображение Элроса. На этот раз его лицо выражало недовольство.

– Ученики! – громогласно произнес Элрос. – Что это за некомпетентность?! Вы уже должны были достичь следующего уровня лабиринта, а вы все еще бродите неизвестно где!

– Но мы следовали за указателем! – возмутилась Алиса.

– Указатель?! – фыркнул Элрос. – Вы что, совсем разучились думать самостоятельно?! Хаос может влиять на любые элементы этого лабиринта, включая указатель! Вы должны были это предвидеть!

– Но мы… – начала было Алиса.

– «Мы, мы…» – передразнил ее Элрос. – Если вы будете продолжать в том же духе, я так и останусь призраком, а вы застрянете в этом лабиринте навечно! И тогда мне придется наблюдать за вашими бесконечными блужданиями до конца своих дней, которые, между прочим, в моем текущем состоянии могут оказаться не такими уж и бесконечными!

Кай и Алиса переглянулись. Они еще никогда не видели Элроса таким рассерженным.

– Простите, Мастер, – сказал Кай. – Мы больше не будем слепо доверять указателю.

– Надеюсь на это, – проворчал Элрос, но его голос звучал уже спокойнее. – А теперь включите свои R-терминалы и проанализируйте код окружающего пространства. Вам нужно найти аномалии, которые укажут на правильный путь.

– Хорошо, Мастер, – сказала Алиса и достала свой терминал.

– Обратите внимание на строковые переменные, – подсказал Элрос. – Хаос часто использует их, чтобы создавать ложные указатели и иллюзии.

– Строковые переменные? – переспросила Алиса.

– Это переменные, которые хранят текстовые данные, – объяснил Кай. – Например, названия объектов, сообщения, команды.

– А как нам их найти? – спросила Алиса.

– Используйте функцию `typeof()`, – ответил Элрос. – Она покажет вам тип каждой переменной.

Алиса и Кай начали сканировать пространство своими терминалами, используя функцию `typeof()`. Вскоре они обнаружили несколько строковых переменных, которые выглядели подозрительно.

– Вот, например, переменная с именем “direction”, – сказал Кай. – Она содержит значение «налево». Но наш указатель показывает направо.

– А вот переменная “message”, – сказала Алиса. – Она содержит текст «Вы на верном пути». Но я чувствую, что это ловушка.

– Вы начинаете понимать, – сказал Элрос с удовлетворением. – Хаос хитер, но он не всемогущ. Используйте свои знания R, чтобы разоблачить его обман.

– А что нам делать с этими переменными? – спросила Алиса.

– Измените их значения на правильные, – ответил Элрос. – Используйте оператор присваивания `<-`.

Кай быстро ввел команды на своем терминале:

```
direction <- "направо"
message <- "Осторожно,  ловушка!"
```

В тот же момент пространство вокруг них снова замерцало, и иллюзии начали рассеиваться. Ложный указатель исчез, а вместо него появился новый, который указывал в совершенно другом направлении.

– Получилось! – воскликнула Алиса. – Мы нашли правильный путь!

– Молодцы, – похвалил их Элрос. – А теперь поторопитесь. Я чувствую, что хаос где-то рядом.

Внезапно внимание Элроса отвлеклось, и он резко повернул голову.

– Что это там? – пробормотал он. – Кажется, кто-то пытается взломать мой код… Ах, вот ты где, проклятый вирус! Ну я тебе сейчас покажу!

Элрос исчез, оставив Алису и Кая в недоумении.

– Что это было? – спросила Алиса.

– Похоже, у Мастера новые приключения, – усмехнулся Кай. – Что ж, не будем ему мешать. Пойдем дальше.

Они последовали за новым указателем, оставляя Элроса бороться с вирусом в цифровом пространстве.

# R-ПРАКТИКУМ

**Строки (character)** – это один из основных типов данных в R, который используется для хранения текста. Строки заключаются в одинарные (') или двойные (") кавычки.

**Примеры строк:**

```
"Привет, мир!"
'Это строка'
"123" #  Это тоже строка, так как заключена в кавычки
```

**Функции для работы со строками**

R предоставляет множество функций для работы со строками. Некоторые из них:

- `nchar()`: возвращает количество символов в строке;
- `paste()`: объединяет несколько строк в одну;
- `substr()`: извлекает подстроку из строки;
- `tolower()`: преобразует строку в нижний регистр;
- `toupper()`: преобразует строку в верхний регистр;
- `strsplit()`: разделяет строку на подстроки по заданному разделителю;
- `grep()`: ищет заданный шаблон в строке;
- `sub()`: заменяет первое вхождение заданного шаблона в строке;
- `gsub()`: заменяет все вхождения заданного шаблона в строке.

**Примеры использования функций:**

```
my_string <- "Привет, мир!"
nchar(my_string)  # Выведет 13
paste("Привет", "мир", sep = ", ")  # Выведет "Привет, мир"
substr(my_string, 1, 6)  # Выведет "Привет"
tolower(my_string)  # Выведет "привет, мир!"
toupper(my_string)  # Выведет "ПРИВЕТ, МИР!"
strsplit(my_string, ", ")  # Выведет список из двух элементов: "Привет" и "мир!"
grep("мир", my_string)  # Выведет 1 (индекс строки, где найден шаблон)
```

```
   sub("мир", "R", my_string)  # Выведет "Привет,
R!"
   gsub("и", "!", my_string)  # Выведет "Пр!вет,
м!р!"
```

**Задания для самостоятельной работы**

1. Создайте строковую переменную `my_name` и присвойте ей значение вашего имени.
2. Выведите на экран количество символов в вашем имени с помощью функции `nchar()`.
3. Создайте новую строковую переменную, которая содержит ваши имя, фамилию и город, в котором вы живете, используя функцию `paste()`.
4. Извлеките из этой строки ваше имя с помощью функции `substr()`.
5. Преобразуйте ваше имя в верхний регистр с помощью функции `toupper()`.

# *Глава 14*
# ВЕКТОРЫ, МАТРИЦЫ И ПОБЕГ ИЗ ЗООПАРКА

– Фух, наконец-то выбрались из этой иллюзии! – Алиса с облегчением огляделась по сторонам.

Они стояли на зеленой лужайке, окруженной высокими деревьями. В воздухе порхали бабочки, а вдали слышалось пение птиц.

– Кажется, мы попали в какой-то парк, – сказал Кай.

– Надеюсь, здесь нет никаких ловушек и опасностей, – сказала Алиса. – Я бы не отказалась немного отдохнуть и послушать пение птиц.

– Подожди, – Кай нахмурился, глядя на показания своего R-терминала. – Что-то не так. Я регистрирую странные сигналы…

Внезапно из-за деревьев появилась группа… говорящих животных! Лев с массивной гривой, слон с длинным хоботом, жираф с длинной шеей и зебра в полосатую рубашку направились к ним.

– Стой! Кто такие?! – прорычал Лев грозным голосом.

– Мы… мы путешественники, – заикаясь, ответила Алиса. – Мы заблудились…

– Заблудились? – удивился Слон. – В нашем Зоопарке? Ха-ха-ха!

– Зоопарк? – переспросила Алиса. – Но здесь же нет клеток и заборов…

– Наши клетки невидимы, – гордо произнес Жираф. – Они созданы с помощью самого передового кода!

– И вы не сможете отсюда выбраться, – добавила Зебра, потирая копыта. – Ха-ха-ха!

– Что же нам делать? – шепнула Алиса Каю.

– Не паникуй, – ответил Кай. – Мы что-нибудь придумаем.

В этот момент в воздухе появилось изображение Элроса, но на этот раз он был не один. Рядом с ним находилась… красивая русалка с длинными зелеными волосами.

– Привет, ученики! – сказал Элрос с лукавой улыбкой. – Как вам мой новый друг?

– Э-э… здравствуйте, – промямлила Алиса, не зная, что сказать.

– Это Ариэль, – представил русалку Элрос. – Она – хранительница морских данных. Мы с ней как раз обсуждали… кхм… важные вопросы межпространственного сотрудничества.

Ариэль кокетливо улыбнулась и помахала им рукой.

– Но… что вы здесь делаете? – спросил Кай.

– Я решил взять отпуск! – объявил Элрос. – Хватит с меня этих лабиринтов и хаоса. Пора отдохнуть и насладиться жизнью!

– Но… а как же мы? – спросила Алиса.

– Вы? – Элрос махнул рукой. – Вы уже достаточно взрослые, чтобы справиться сами. Вот, возьмите это.

Он отправил им на R-терминалы файл с кодом.

– Это программа для деактивации невидимых клеток, – объяснил Элрос. – Вам нужно только немного ее доработать. И не забудьте использовать все, чему я вас научил: векторы, матрицы, функции…

– Но… – начала было Алиса.

– Никаких «но»! – перебил ее Элрос. – Я ухожу плавать с Ариэль. Удачи!

Изображение Элроса и Ариэль растворилось в воздухе, оставив Алису и Кая в окружении говорящих животных.

– Ну что, хакеры, – ухмыльнулся Лев, – попались?

– Еще не вечер, – сказал Кай, открывая файл с кодом на своем терминале. – Алиса, поможешь мне с этой программой?

– Конечно! – ответила Алиса, с азартным блеском в глазах. – Пора показать этим зверям, на что мы способны!

– Хм, – сказал Кай, просматривая код, который им прислал Элрос. – Похоже, это довольно простая программа. Нам нужно просто найти матрицу, которая хранит информацию о координатах клеток, и изменить ее значения.

– А как мы найдем эту матрицу? – спросила Алиса.

– Судя по коду, – ответил Кай, – она хранится в переменной с именем `cage_coordinates`. Давай посмотрим, что в ней.

```
print(cage_coordinates)
```

На экране появилась матрица с координатами клеток:

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

– Теперь нам нужно изменить эти координаты на нулевые, – сказал Кай. – Это деактивирует клетки.

– А как нам это сделать? – спросила Алиса.

– Мы можем использовать цикл `for` и функцию `replace()`, – ответил Кай.

```
for (i in 1:nrow(cage_coordinates)) {
  for (j in 1:ncol(cage_coordinates)) {
    cage_coordinates <- replace(cage_coordi-
nates, cage_coordinates[i, j], 0)
  }
}
print(cage_coordinates)
```

На экране появилась новая матрица:

```
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0
```

– Готово! – сказал Кай. – Клетки деактивированы.

В тот же момент невидимые клетки исчезли, и животные удивленно заозирались по сторонам.

– Что произошло? – спросил Лев. – Куда делись клетки?

– Мы их деактивировали, – ответила Алиса. – Теперь вы свободны!

Животные обрадовались и начали прыгать и бегать по лужайке, наслаждаясь своей свободой.

– Спасибо вам! – сказал Слон. – Вы нас освободили!

– Не за что, – ответила Алиса. – Мы рады, что смогли помочь.

– А теперь, – сказал Кай, – нам пора идти. Нас ждут новые приключения.

– Подождите! – крикнул Жираф. – А как же Элрос? Он же должен был вот-вот раствориться!

Алиса и Кай переглянулись. Они совсем забыли про Элроса!

– Точно! – сказала Алиса. – Как он мог выбраться без сферы?

– И почему он выглядел таким… живым? – добавил Кай.

– Может быть, он нас обманул? – предположила Алиса.

– Не знаю, – сказал Кай. – Но мне это не нравится. У меня плохое предчувствие.

– Давай свяжемся с ним, – предложила Алиса.

Кай попытался связаться с Элросом через R-терминал, но связь не устанавливалась.

– Не получается, – сказал Кай. – Что-то блокирует сигнал.

– Может быть, он в опасности? – спросила Алиса с тревогой.

– Возможно, – ответил Кай. – Нам нужно его найти.

В этот момент указатель вновь активировался, указывая на новый проход.

– Похоже, это наш путь, – сказал Кай. – Идем!

Они прошли сквозь проход и оказались в странном, искаженном пространстве, где все было заполнено иллюзиями и миражами.

– Где мы? – спросила Алиса, с опаской оглядываясь по сторонам.

– Кажется, мы попали в ловушку иллюзий, – ответил Кай. – И мне кажется, что Элрос где-то здесь.

Внезапно они увидели Элроса, стоящего посреди этого хаоса. Он выглядел странно: его тело мерцало, а глаза были затуманены.

– Элрос! – крикнула Алиса. – С вами все в порядке?

Элрос медленно повернулся к ним и произнес хриплым голосом:

– Кто… кто вы? Где… где я?

– Вы не узнаете нас? – спросил Кай с тревогой. – Это мы, Алиса и Кай!

– Алиса? Кай? – переспросил Элрос, морщась. – Я… я не знаю таких…

– Он потерял память! – воскликнула Алиса. – Хаос повредил его код!

– Нам нужно ему помочь! – сказал Кай. – Но как?

В этот момент на R-терминале Кая появилось новое сообщение:

**«Внимание! Обнаружено критическое повреждение кода Мастера. Необходима немедленная реактивация сферы памяти!»**

– Сфера памяти? – переспросил Кай. – Но она же осталась в том месте, где мы встретили Элроса!

– Нам нужно вернуться туда! – сказала Алиса. – Но как?

– Я знаю, как! – сказал Кай. – Мы можем использовать функцию `return()`!

– `return()`? – не поняла Алиса.

– Эта функция позволяет вернуться к предыдущему вызову функции, – объяснил Кай. – В данном случае она вернет нас в то место, откуда мы начали наше путешествие.

– Тогда давай скорее! – сказала Алиса. – Нам нужно спасти Элроса!

Кай быстро ввел команду на своем R-терминале:

```
return()
```

В тот же миг мир вокруг них расплылся, и они снова оказались в белом пространстве, где впервые встретили Элроса. На полу все еще лежала светящаяся сфера – та самая, которая отделилась от Элроса после его исчезновения.

– Быстрее! – сказала Алиса и бросилась к сфере.

Она взяла сферу в руки, и та вспыхнула ярким светом. В воздухе появилось голографическое изображение Элроса, но на этот раз он выглядел совсем плохо. Его тело было полупрозрачным, а глаза лихорадочно блестели.

– Элрос! – крикнула Алиса. – Мы вернулись! Вот ваша сфера!

Элрос с трудом сфокусировал взгляд на Алисе и Кае.

– Алиса? Кай? – прошептал он. – Это… это вы?

– Да, это мы! – сказал Кай. – Мы принесли вашу сферу!

Алиса поднесла сферу к Элросу, и та влилась в его тело. Элрос вздрогнул, его тело стало плотнее, а глаза приобрели нормальный цвет.

– Спасибо… – прошептал он. – Я… я уже начал терять себя…

– Что случилось? – спросила Алиса. – Как вы оказались в том странном месте?

– Я… я погнался за вирусом, – объяснил Элрос. – Он оказался хитрее, чем я думал. Он заманил меня в ловушку иллюзий и… начал поглощать мой код.

– Но как вы выбрались из Зоопарка? – спросил Кай.

– Я… я не выбирался, – признался Элрос. – Это была иллюзия. Вирус создал ее, чтобы отвлечь меня.

– Но как же Ариэль? – спросила Алиса.

– Ариэль? – Элрос удивленно посмотрел на них. – Какая Ариэль?

– Ну, русалка, с которой вы… кхм… обсуждали межпространственное сотрудничество, – сказала Алиса.

– А-а, – понял Элрос. – Это тоже была иллюзия. Вирус знает мои слабости…

Алиса и Кай переглянулись. Похоже, Элрос попал в настоящую передрягу.

– Ну что ж, – сказал Кай, – главное, что вы теперь в безопасности. А с вирусом мы разберемся позже.

– Да, – согласилась Алиса. – А теперь давайте отдохнем. Мы все заслужили перерыв после таких приключений.

Они устроились поудобнее на полу белого пространства, и Элрос начал рассказывать им о своей битве с вирусом и о том, как ему удалось сохранить хоть часть своего кода.

– …и тогда я понял, – сказал Элрос, – что единственный способ победить хаос – это использовать его же оружие против него. Я начал создавать свои собственные иллюзии, чтобы запутать вирус, и в конце концов мне удалось вырваться из его ловушки.

– Вы настоящий герой, Мастер! – воскликнула Алиса.

– Да, – согласился Кай. – Вы нас многому научили.

– Главное, чему я хотел вас научить, – сказал Элрос, – это тому, что R – это не просто язык программирования. Это инструмент, который может помочь вам в любой ситуации. Даже в самой безнадежной.

– Мы запомним это, Мастер, – сказал Кай.

– А теперь, – сказал Элрос, зевая, – я, пожалуй, немного вздремну. Эти приключения меня совсем вымотали.

Он закрыл глаза и уснул, а Алиса и Кай продолжили обсуждать свои планы на будущее и думать о новых приключениях.

# R-ПРАКТИКУМ

**Файлы** – это основной способ хранения данных на компьютере. R позволяет нам читать данные из файлов, обрабатывать их и записывать результаты обратно в файлы. Это позволяет нам:

- сохранять данные: мы можем сохранить результаты нашего анализа в файл, чтобы использовать их позже или поделиться с другими;
- загружать данные: мы можем загрузить данные из файла в R, чтобы проанализировать их или использовать в наших программах;
- обмениваться данными: файлы – удобный способ обмена данными между разными программами и пользователями.

**Типы файлов**

R может работать с разными типами файлов, например:

- текстовые файлы (.txt): содержат текст, который можно прочитать в любом текстовом редакторе;
- CSV-файлы (.csv): содержат табличные данные, где значения разделены запятыми (Comma Separated Values);
- RData-файлы (.RData): специальный формат файлов R, который позволяет сохранять объекты R (переменные, фреймы данных и т. д.).

**Чтение данных из файла**

Для чтения данных из файла в R есть несколько функций:

- `read.table()`: для чтения табличных данных из текстовых файлов;
- `read.csv()`: для чтения данных из CSV-файлов;
- `load()`: для загрузки объектов R из RData-файлов.

Примеры:

```
# Чтение данных из текстового файла
my_data <- read.table("my_data.txt", header = TRUE)
# Чтение данных из CSV-файла
my_data <- read.csv("my_data.csv")
# Загрузка объекта R из RData-файла
load("my_object.RData")
```

**Запись данных в файл**

Для записи данных в файл в R также есть несколько функций:

- `write.table()`: для записи табличных данных в текстовый файл;
- `write.csv()`: для записи данных в CSV-файл;
- `save()`: для сохранения объектов R в RData-файл.

Примеры:

```
# Запись данных в текстовый файл
write.table(my_data, file = "my_data.txt", sep = "\t", row.names = FALSE)
# Запись данных в CSV-файл
write.csv(my_data, file = "my_data.csv", row.names = FALSE)
# Сохранение объекта R в RData-файл
save(my_object, file = "my_object.RData")
```

**Задания для самостоятельной работы**

1. Создайте фрейм данных с информацией о себе (имя, возраст, город).
2. Запишите этот фрейм данных в CSV-файл с именем “my_info.csv”.
3. Прочитайте данные из файла “my_info.csv” в новую переменную.
4. Добавьте во фрейм данных новый столбец с информацией о вашем любимом хобби.

# ЗАКЛЮЧЕНИЕ

Дорогие читатели!

Вот и подошло к концу наше путешествие в удивительный мир R. Вместе с Алисой и Каем вы прошли через множество испытаний, разгадали загадки цифрового лабиринта и помогли Элросу в борьбе с хаосом.

Надеюсь, вам понравилось это приключение! Но главное, что вы не только прочитали эту книгу, но и выполнили все задания в R-практикумах, так как – это не просто язык программирования, это инструмент, который открывает перед вами безграничные возможности. С его помощью вы можете анализировать данные, строить модели, создавать визуализации и даже изменять мир вокруг себя.

В этой книге вы познакомились с самыми основами R: переменными, типами данных, операторами, функциями, векторами, списками, матрицами. Это лишь первый шаг на пути к освоению этого мощного языка. Впереди вас ждет еще много интересного!

Если вы хотите продолжить изучение R, рекомендую вам обратиться к следующим ресурсам:

- Кабаков, Р. И. R в действии / Р. И. Кабаков ; перевод с английского А. Н. Киселева. – 3-е изд. – Москва : ДМК Пресс, 2023. – 768 с. – ISBN 978-5-93700-173-3;
- Савельев, В. Статистика и котики / В. Савельев. – Москва : АСТ, 2018. – 190 с. – ISBN 978-5-17-108287-1;
- Введение в статистическое обучение с примерами на языке R / Г. Джеймс, Д. Уиттон, Т. Хасти, Р. Тибширани ; перевод с английского С. Э. Мастицкого. – Москва : ДМК Пресс, 2017. – 456 с. – ISBN 978-5-97060-495-3.

А что же дальше с Алисой, Каем и Элросом? Их приключения в цифровом лабиринте подошли к концу, но в следующих книгах серии, кто знает, может они столкнутся с новыми вызовами, освоят новые инструменты R и раскроют еще больше тайн мультивселенной.

Не пропустите продолжение!

*С наилучшими пожеланиями,*
*Стратонов Дмитрий Дмитриевич.*

# ГЛОССАРИЙ

**Алгоритм** – последовательность шагов, которые нужно выполнить для решения задачи. Представьте себе рецепт приготовления торта: каждый шаг в рецепте – это как строчка кода в алгоритме, а готовый торт – это результат работы алгоритма.

**Аргумент** – дополнительная информация, которую можно передать в функцию, чтобы изменить ее поведение. Например, в функции `print()` можно указать аргумент `Screen = "reality"`, чтобы вывести данные не на экран, а в реальность (как это делали Алиса и Кай).

**Вектор** – упорядоченный набор элементов одного типа. Представьте себе поезд, где каждый вагон – это элемент вектора, а сам поезд – это вектор.

**Выражение** – комбинация значений, переменных, операторов и функций, которая вычисляется в одно значение. Например, `2 + 2` – это выражение, которое вычисляется в значение `4`.

**Идентификатор –** имя, которое дается переменной, функции или другому объекту в программе. Например, `cage_coordinates` – это идентификатор матрицы, которая хранила координаты клеток в Зоопарке.

**Индекс** – номер позиции элемента в векторе, списке или матрице. Например, в векторе `c(10, 20, 30)` элемент `20` имеет индекс `2`.

**Конкатенация** – объединение нескольких векторов или строк в один. Представьте, что вы соединяете несколько вагонов в один длинный поезд.

**Логическое значение** – значение, которое может быть либо `TRUE` (истина), либо `FALSE` (ложь). Например, утверждение «небо голубое» – истина (`TRUE`), а утверждение «кошки умеют летать» – ложь (`FALSE`).

**Массив** – многомерная структура данных, которая может хранить элементы разных типов. Представьте себе не просто таблицу, а куб или даже более сложную фигуру, где каждая ячейка содержит какое-то значение.

**Матрица** – двумерный массив, т. е. таблица, состоящая из строк и столбцов. Например, экран вашего компьютера – это матрица пикселей.

**Оператор** – символ, который выполняет какое-то действие над данными. Например, «+» (сложение), «−» (вычитание), «*» (умножение), «/» (деление).

**Пакет** – набор функций и данных, которые расширяют возможности R. Например, пакет `ggplot2` позволяет создавать красивые графики.

**Переменная** – контейнер для хранения данных. Представьте себе коробку, в которую можно положить что-нибудь: число, текст, картинку.

**Скрипт** – файл, содержащий последовательность команд R. Это как рецепт для компьютера, который говорит ему, что нужно делать.

**Список** – упорядоченная коллекция объектов разных типов. Представьте себе шкаф с разными полками, на которых могут храниться самые разные вещи.

**Строка** – последовательность символов, заключенная в кавычки. Например, `"Привет, мир!"` – это строка.

**Структура управления** – конструкция в программе, которая определяет порядок выполнения команд. Например, условный оператор `if` позволяет выполнить какой-то код только при выполнении определенного условия.

**Тип данных** – определяет, какого рода информацию может хранить переменная. Например, число, текст, логическое значение.

**Функция** – именованный блок кода, который выполняет определенное действие. Например, функция `print()` выводит данные на экран.

**Цикл** – конструкция в программе, которая позволяет повторять один и тот же блок кода несколько раз. Например, цикл `for` может быть использован для обработки всех элементов вектора.

# РЕШЕНИЯ ЗАДАНИЙ ИЗ R-ПРАКТИКУМОВ

## Глава 1.
## ВХОД В МАТРИЦУ

**Задания для самостоятельной работы**
Нет заданий.

## Глава 2.
## ПЕРЕМЕННЫЕ И ТИПЫ ДАННЫХ

**Задания для самостоятельной работы**

1. Создайте переменную `my_name` и присвойте ей значение вашего имени.
2. Создайте переменную `my_age` и присвойте ей значение вашего возраста.
3. Создайте переменную `is_programmer` и присвойте ей значение `TRUE`.
4. Проверьте типы данных созданных переменных с помощью функции `class()`.

**Решение**

```
my_name <- "Иван"  # Замените "Иван" на ваше имя
my_age <- 30       # Замените 30 на ваш возраст
is_programmer <- TRUE
class(my_name)      # Выведет "character"
class(my_age)       # Выведет "numeric"
class(is_programmer)  # Выведет "logical"
```

## Глава 3.
## ОПЕРАТОРЫ И ВЫРАЖЕНИЯ

**Задания для самостоятельной работы**

1. Вычислите значение выражения `(10 + 5) * 2 - 8 / 4`.
2. Проверьте, верно ли утверждение `15 > 10 & 5 < 3`.
3. Проверьте, верно ли утверждение `"hello" != "world"`.

**Решение**

```
(10 + 5) * 2 - 8 / 4    # Выведет 28
15 > 10 & 5 < 3         # Выведет FALSE
"hello" != "world"      # Выведет TRUE
```

# Глава 4.
# ФУНКЦИИ И СТРУКТУРЫ УПРАВЛЕНИЯ

**Задания для самостоятельной работы**

1. Напишите функцию `calculate_area()`, которая принимает два аргумента (длину и ширину) и возвращает площадь прямоугольника.
2. Напишите цикл `for`, который выводит на экран числа от 1 до 10.

**Решение**

```
# Функция для вычисления площади прямоугольника
calculate_area <- function(length, width) {
  area <- length * width
  return(area)
}
# Вызов функции
calculate_area(5, 3)  # Выведет 15
# Цикл for для вывода чисел от 1 до 10
for (i in 1:10) {
  print(i)
}
```

# Глава 5.
# ВЕКТОРЫ И РАБОТА С ДАННЫМИ

**Задания для самостоятельной работы**

1. Создайте вектор `my_numbers`, содержащий числа от 1 до 20.
2. Выведите на экран пятый элемент вектора `my_numbers`.
3. Умножьте все элементы вектора `my_numbers` на 3.
4. Создайте новый вектор, содержащий первые 10 элементов вектора `my_numbers`.

**Решение**

```
# Создание вектора чисел от 1 до 20
my_numbers <- 1:20
# Вывод пятого элемента вектора
my_numbers[5]   # Выведет 5
# Умножение всех элементов вектора на 3
my_numbers * 3  # Выведет 3  6  9 12 15 18 21 24
27 30 33 36 39 42 45 48 51 54 57 60
# Создание нового вектора с первыми 10 элемен-
тами
first_10_numbers <- my_numbers[1:10]
```

# Глава 6.
# ВЕКТОРЫ И ПОИСКИ ИСТИНЫ

**Задания для самостоятельной работы**

1. Создайте вектор с числами от 10 до 100 с шагом 10, используя функцию `seq()`.
2. Создайте вектор, в котором число 5 повторяется 7 раз, используя функцию `rep()`.
3. Создайте вектор с именами месяцев. Проверьте, содержится ли в нем месяц «июнь».
4. Создайте два вектора с числами. Найдите элементы, которые присутствуют в обоих векторах, используя функцию `intersect()`.
5. Создайте два вектора с числами. Найдите элементы, которые присутствуют в первом векторе, но отсутствуют во втором, используя функцию `setdiff()`.

**Решение**

```
# 1. Вектор с числами от 10 до 100 с шагом 10
numbers_10_to_100 <- seq(from = 10, to = 100, by = 10)
print(numbers_10_to_100)
# 2. Вектор с числом 5, повторенным 7 раз
fives <- rep(5, times = 7)
print(fives)
# 3. Проверка наличия месяца "июнь"
months <- c("январь", "февраль", "март", "апрель", "май", "июнь",
            "июль", "август", "сентябрь", "октябрь", "ноябрь", "декабрь")
"июнь" %in% months # Вернет TRUE
# 4. Пересечение векторов
vector1 <- c(1, 2, 3, 4, 5)
vector2 <- c(3, 4, 5, 6, 7)
intersect(vector1, vector2) # Вернет 3 4 5
# 5. Разница векторов
vector1 <- c(1, 2, 3, 4, 5)
vector2 <- c(3, 4, 5, 6, 7)
setdiff(vector1, vector2) # Вернет 1 2
```

# Глава 7.
# ОПЕРАЦИИ НАД ВЕКТОРАМИ – ПУТЬ К СПАСЕНИЮ

**Задания для самостоятельной работы**

1. Создайте вектор с числами от 20 до 1 (в обратном порядке).
2. Найдите длину этого вектора.
3. Отсортируйте вектор по возрастанию.
4. «Разверните» вектор.
5. Выберите из вектора все числа, которые делятся на 3 без остатка.

**Решение**

```
# 1. Создание вектора с числами от 20 до 1
numbers_20_to_1 <- 20:1
print(numbers_20_to_1)
# 2. Длина вектора
length(numbers_20_to_1) # Выведет 20
# 3. Сортировка вектора по возрастанию
sort(numbers_20_to_1) # Выведет 1  2  3  4  5  6
7  8  9 10 11 12 13 14 15 16 17 18 19 20
# 4. "Разворачивание" вектора
rev(numbers_20_to_1) # Выведет 1  2  3  4  5  6
7  8  9 10 11 12 13 14 15 16 17 18 19 20
# 5. Выбор чисел, делящихся на 3 без остатка
numbers_20_to_1[numbers_20_to_1 %% 3 == 0] # Вы-
ведет 3  6  9 12 15 18
```

# Глава 8.
# СПИСКИ – ПОРЯДОК В ХАОСЕ

**Задания для самостоятельной работы**

1. Создайте список, содержащий ваши имя, возраст и любимый цвет.
2. Добавьте в список новый элемент – название вашего города.

**Решение**

```
# Создание списка
my_list <- list(
  name = "Иван",  # Замените на ваше имя
  age = 30,       # Замените на ваш возраст
```

```
    favorite_color = "синий"   # Замените на ваш
любимый цвет
)
# Добавление нового элемента
my_list$city <- "Москва"  # Замените на ваш город
```

## Глава 9.
## МАТРИЦЫ И ГОЛОВОЛОМКИ РЕАЛЬНОСТИ

**Задания для самостоятельной работы**

1. Создайте две матрицы размера 3×3 с числовыми значениями.
2. Выполните сложение и умножение этих матриц.
3. Транспонируйте одну из матриц.
4. Выведите результаты на экран.

**Решение**

```
# Создание матриц
matrix1 <- matrix(1:9, nrow = 3, ncol = 3)
matrix2 <- matrix(10:18, nrow = 3, ncol = 3)
# Сложение матриц
sum_matrix <- matrix1 + matrix2
# Умножение матриц
prod_matrix <- matrix1 %*% matrix2
# Транспонирование матрицы
transposed_matrix <- t(matrix1)
# Вывод результатов
print(sum_matrix)
print(prod_matrix)
print(transposed_matrix)
```

## Глава 10.
## ФРЕЙМЫ ДАННЫХ И ГОРОД ПОТЕРЯННЫХ ДУШ

**Задания для самостоятельной работы**

1. Создайте фрейм данных, содержащий информацию о ваших друзьях (имя, возраст, любимый цвет).
2. Выведите на экран столбец с именами ваших друзей.
3. Добавьте в фрейм данных новый столбец с информацией о хобби ваших друзей.

**Решение**

```
# Создание фрейма данных
friends <- data.frame(
  Name = c("Петя", "Маша", "Вася"),
  Age = c(25, 30, 28),
  FavoriteColor = c("красный", "зеленый",
"синий")
)
# Вывод столбца с именами
print(friends$Name)
# Добавление нового столбца
friends$Hobby <- c("футбол", "рисование",
"музыка")
```

## Глава 11.
## МАССИВЫ И ИЗМЕРЕНИЕ БЕСКОНЕЧНОСТИ

**Задания для самостоятельной работы**

1. Создайте двумерный массив размером 4×5, заполненный случайными числами.
2. Создайте трехмерный массив размером 2×3×2, заполненный строками «красный», «зеленый», «синий».
3. Выведите на экран элемент из второй «строки», третьего «столбца» и первого «слоя» трехмерного массива.
4. Вычислите сумму элементов двумерного массива.
5. Измените размерность трехмерного массива на 3×2×2.
6. Попробуйте визуализировать двумерный массив с помощью функции `image()`.

**Решение**

```
# 1. Создание двумерного массива
array_2d <- array(runif(20), dim = c(4, 5))  #
runif(20) генерирует 20 случайных чисел
# 2. Создание трехмерного массива
colors <- c("красный", "зеленый", "синий")
array_3d <- array(rep(colors, each = 4), dim =
c(2, 3, 2))
# 3. Вывод элемента
print(array_3d[2, 3, 1])  # Выведет "синий"
# 4. Сумма элементов двумерного массива
sum(array_2d)
```

```
# 5. Изменение размерности
dim(array_3d) <- c(3, 2, 2)
# 6. Визуализация
image(array_2d)
```

## Глава 12.
## ФАКТОРЫ И ЗАГАДКА ЛИЧНОСТИ

**Задания для самостоятельной работы**

1. Создайте фактор, который представляет дни недели (понедельник, вторник и т. д.).
2. Выведите на экран уровни этого фактора.
3. Создайте вектор с названиями месяцев и преобразуйте его в фактор.

**Решение**

```
# 1. Создание фактора "дни недели"
days_of_week <- factor(c("понедельник", "вторник", "среда", "четверг", "пятница", "суббота", "воскресенье"))
# 2. Вывод уровней фактора
levels(days_of_week)  # Выведет все дни недели
# 3. Создание фактора "месяцы"
months <- c("январь", "февраль", "март", "апрель", "май", "июнь",
            "июль", "август", "сентябрь", "октябрь", "ноябрь", "декабрь")
months_factor <- factor(months)
```

## Глава 13.
## СТРОКИ И ЛОВУШКА ИЛЛЮЗИЙ

**Задания для самостоятельной работы**

1. Создайте строковую переменную `my_name` и присвойте ей значение вашего имени.
2. Выведите на экран количество символов в вашем имени с помощью функции `nchar()`.
3. Создайте новую строковую переменную, которая содержит ваши имя, фамилию и город, в котором вы живете, используя функцию `paste()`.

4. Извлеките из этой строки ваше имя с помощью функции `substr()`.
5. Преобразуйте ваше имя в верхний регистр с помощью функции `toupper()`.

**Решение**

```
# 1. Создание строковой переменной
my_name <- "Иван"  # Замените на ваше имя
# 2. Вывод количества символов
nchar(my_name)     # Выведет количество букв в имени
# 3. Создание новой строки
my_info <- paste("Меня зовут", my_name, ", я живу в Москве", sep = " ")
# Замените "Москва" на ваш город
# 4. Извлечение имени
substr(my_info, 13, 16)  # Индексы нужно подобрать в зависимости от вашего имени и города
# 5. Преобразование в верхний регистр
toupper(my_name)  # Выведет "ИВАН"
```

## Глава 14.
## ВЕКТОРЫ, МАТРИЦЫ И ПОБЕГ ИЗ ЗООПАРКА

**Задания для самостоятельной работы**

1. Создайте фрейм данных с информацией о себе (имя, возраст, город).
2. Запишите этот фрейм данных в CSV-файл с именем “my_info.csv”.
3. Прочитайте данные из файла “my_info.csv” в новую переменную.
4. Добавьте в фрейм данных новый столбец с информацией о вашем любимом хобби.

**Решение**

```
# 1. Создание фрейма данных с информацией о себе
my_info <- data.frame(
  Имя = "Ваше имя",        # Замените "Ваше имя" на ваше настоящее имя
  Возраст = Ваш_возраст,     # Замените Ваш_возраст на ваш настоящий возраст
```

```
      Город = "Ваш город"       # Замените "Ваш
город" на ваш настоящий город
    )
    # 2. Запись фрейма данных в CSV-файл
    write.csv(my_info, file = "my_info.csv",
row.names = FALSE)
    # row.names = FALSE предотвратит запись индек-
сов строк в файл
    # 3. Чтение данных из файла в новую переменную
    new_variable <- read.csv("my_info.csv")
    # 4. Добавление нового столбца
    new_variable$Хобби <- "Ваше хобби"  # Замените
"Ваше хобби" на ваше любимое хобби
    # Чтобы добавить несколько хобби можно исполь-
зовать вектор:
    # new_variable$Хобби <- c("Хобби 1", "Хобби
2", "Хобби 3")
    # Вывод измененного фрейма данных (чтобы убе-
диться в изменениях)
    print(new_variable)
```

Используйте функцию `?` (например, `?read.csv`), чтобы получить подробную справку по любой функции в R.

Учебное издание

***Стратонов*** *Дмитрий Дмитриевич*

# ОСНОВЫ R И ПОБЕГ ИЗ ЦИФРОВОГО ЛАБИРИНТА

ISBN 978-5-94984-945-3

Редактор П. С. Фенина
Оператор компьютерной верстки Т. В. Упорова

Подписано в печать 10.04.2025. Формат 60×84/16.
Бумага офсетная. Цифровая печать.
Уч.-изд. л. 4,73. Усл. печ. л. 6,28.
Тираж 300 экз. (1-й завод 26 экз.).
Заказ № 8100

ФГБОУ ВО «Уральский государственный лесотехнический университет».
620100, Екатеринбург, Сибирский тракт, 37.
Редакционно-издательский отдел.
Тел. 8 (343) 221-21-44.

Типография ООО «ИЗДАТЕЛЬСТВО УЧЕБНО-МЕТОДИЧЕСКИЙ ЦЕНТР УПИ».
620062, РФ, Свердловская область, Екатеринбург, ул. Гагарина, 35а, оф. 2.
Тел. 8 (343) 362-91-16.