

Игра «Камень, ножницы, бумага» на языке Rust

Кокухина Софья Александровна

Приамурский государственный университет им.Шолом-Алейхема

Студент

Аннотация

Целью исследования является разработка консольной игры «Камень, ножницы, бумага» на языке Rust. В работе показаны основы организации пользовательского ввода, генерации случайных чисел, использования условий и функций. Программа предоставляет пользователю возможность сыграть против компьютера, который делает случайный выбор. Работа демонстрирует практическое применение Rust для создания простых игровых приложений.

Ключевые слова Rust, игра, консольное приложение, генератор случайных чисел, программирование.

Rock, Paper, Scissors Game in Rust

Kokukhina Sofya Alexandrovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The article describes the development of a console game “Rock, Paper, Scissors” using the Rust programming language. The project demonstrates basic input/output operations, random number generation, conditional statements, and function implementation. The program allows the user to play against the computer, which makes a random choice. This work illustrates practical applications of Rust for simple game development.

Keywords Rust, game, console application, random number generator, programming.

1 Введение

1.1 Актуальность

Игры — один из самых доступных способов освоения основ программирования. Проект «Камень, ножницы, бумага» позволяет наглядно продемонстрировать такие элементы языка Rust, как ввод-вывод, использование условных операторов, циклов и генерации случайных значений. Данный пример полезен для школьников и студентов, которые только начинают изучать Rust.

1.2 Обзор исследований

В статье Е.А. Ульянова рассматривается и описывается разработка игры камень, ножницы, бумага на языке программирования C#. Практическим результатом является разработанная игра [1]. В публикации К. П Трубаева описаны современные языки программирования на примере языка rust [2]. В статье Д.М. Берлизова представлена реализация программной транзакционной памяти (software transactional memory) на основе алгоритма TLRW для языка программирования Rust как альтернатива традиционному подходу к синхронизации конкурентных потоков в многопоточных приложениях на основе механизмов взаимного исключения. Ключевая идея представленной методики состоит в использовании блокировок чтения-записи (read-write lock), что позволяет достичь более высокого уровня эффективности и производительности по сравнению с другими алгоритмами транзакционной памяти [3]. В.С. Широченко рассматривает стремительно набирающий популярность язык программирования Rust, использующийся в различных областях разработки программного обеспечения, который применяется в высокопроизводительных вычислениях, а также в предметных областях, в которых критична безопасность данных [4].

1.3 Цель исследования

Цель исследования заключается не только в разработке консольной игры «Камень, ножницы, бумага» на языке Rust, но и в демонстрации базовых принципов программирования на современном системном языке. В рамках работы ставятся задачи: изучить основы ввода и вывода данных в Rust, освоить использование внешней библиотеки rand для генерации случайных чисел, реализовать обработку пользовательского ввода с помощью строк и условных операторов, а также закрепить навыки структурирования программы через функции.

2 Материалы и методы

Для реализации программы используется язык программирования Rust версии 1.79.0, установленный через менеджер пакетов rustup. В комплект поставки входят компилятор rustc и система сборки проектов Cargo, которая позволяет быстро создавать и управлять проектами.

В качестве среды разработки применяется Visual Studio Code, дополненный расширением rust-analyzer, обеспечивающим подсветку синтаксиса, автодополнение кода и подсказки по типам. Такой набор инструментов является оптимальным для начинающих разработчиков, так как сочетает простоту использования и функциональность.

На первом этапе подготовки создается новый проект командой cargo new rps_game --bin (Рис.1).

```
C:\Users\Nktt\Documents\Учеба\RustApp>cargo new rps_game --bin
  Creating binary (application) `rps_game` package
note: see more `Cargo.toml` keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html
```

Рисунок 1 – Создание проекта в консоли

Следующим шагом идет подключение внешней библиотеки `rand`, которая необходима для генерации случайных чисел и выбора варианта хода компьютера (Рис.2).



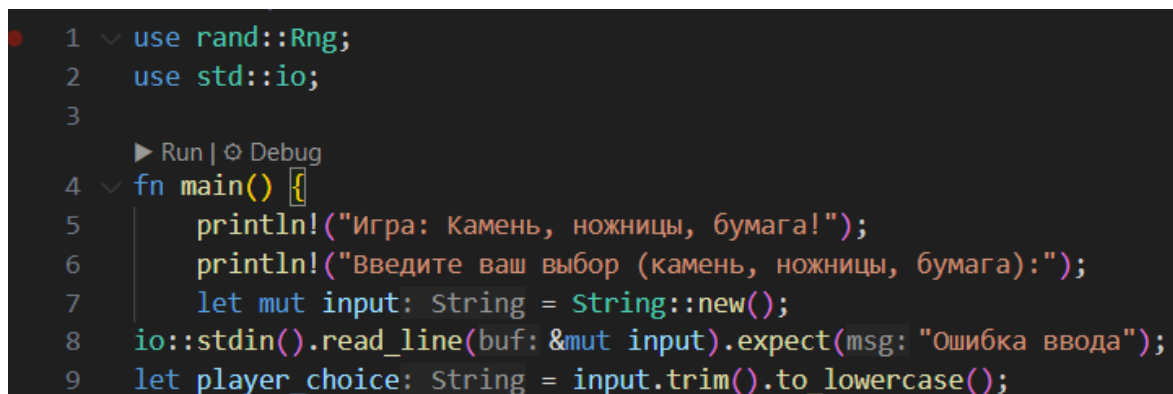
```
Cargo.toml
Cargo.toml > ...
1  [package]
2  name = "rps_game"
3  version = "0.1.0"
4  edition = "2024"
5
6  [dependencies]
7  rand = "0.9.2" ✓
8
```

Рисунок 2 – Добавление зависимости `rand` в `Cargo.toml`

3 Результаты

После подготовки проекта можно переходить к реализации логики игры. Программа состоит из нескольких ключевых этапов: ввод данных от пользователя, генерация случайного выбора компьютера, сравнение результатов и вывод ответа.

Сначала в основной функции `main` организуется ввод данных с клавиатуры. Для этого используется модуль `std::io`. Игроку предлагается ввести один из вариантов: «камень», «ножницы» или «бумага». Введённое значение обрабатывается методом `trim()`, который удаляет лишние пробелы и перевод строки, а затем переводится в нижний регистр методом `to_lowercase()`, чтобы избежать ошибок при сравнении (Рис. 3).



```
1  use rand::Rng;
2  use std::io;
3
4  fn main() {
5      println!("Игра: Камень, ножницы, бумага!");
6      println!("Введите ваш выбор (камень, ножницы, бумага):");
7      let mut input: String = String::new();
8      io::stdin().read_line(buf: &mut input).expect(msg: "Ошибка ввода");
9      let player_choice: String = input.trim().to_lowercase();
```

Рисунок 3 - Организация ввода данных игрока

В случае, когда пользователь вводит слово, отличающееся от предложенных, программа выдаст ошибку и предложит повторить ввод (Рис.4).

```
16         if options.contains(&choice.as_str()) {
17             break choice;
18         } else {
19             println!("Ошибка: неверный ввод! Пожалуйста, введите 'камень', 'ножницы' или 'бумага'.");
20         }
21     };
22 }
```

Рисунок 4 – Ошибка ввода для пользователя

Для выбора хода компьютера используется библиотека `rand`. Сначала создаётся массив с доступными вариантами ходов: «камень», «ножницы» и «бумага». Затем при помощи генератора случайных чисел выбирается одно из трёх значений. Это обеспечивает непредсказуемость и честность игры (Рисунок 5).

```
10 let options: [&'static str; 3] = ["камень", "ножницы", "бумага"];
11 let computer_choice: &'static str = options[rand::thread_rng().gen_range(0..3)];
12
```

Рисунок 5 – Генерация случайного хода компьютера

После получения данных от игрока и компьютера программа выводит их на экран. Это помогает пользователю видеть, какие именно ходы были сделаны обеими сторонами (Рисунок 6).

```
println!("Вы выбрали: {}", player_choice);
println!("Компьютер выбрал: {}", computer_choice);
```

Рисунок 6 – Вывод выбранных ходов

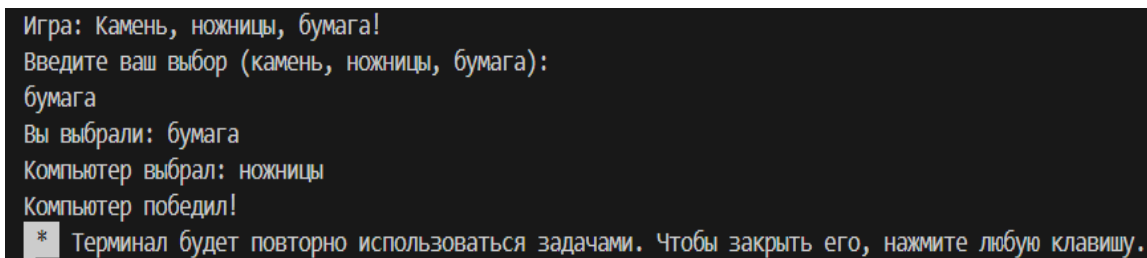
Основная логика игры реализуется с помощью условных операторов `if-else`. Сначала проверяется ситуация ничьей, когда выбор игрока совпадает с выбором компьютера. Если ничья не наступила, проверяется три возможных комбинации победы игрока: камень побеждает ножницы, ножницы побеждают бумагу, бумага побеждает камень.

Во всех остальных случаях победителем признаётся компьютер (Рисунок 7).

```
15 if player_choice == computer_choice {
16     println!("Ничья!");
17 } else if (player_choice == "камень" && computer_choice == "ножницы")
18     || (player_choice == "ножницы" && computer_choice == "бумага")
19     || (player_choice == "бумага" && computer_choice == "камень") {
20     println!("Вы победили!");
21 } else {
22     println!("Компьютер победил!");
23 }
24
```

Рисунок 7 – Логика определения победителя

После завершения кода программа запускается командой `cargo run` в консоли от имени администратора. Пример работы программы представлен на рисунке 8 (Рис.8).



```
Игра: Камень, ножницы, бумага!  
Введите ваш выбор (камень, ножницы, бумага):  
бумага  
Вы выбрали: бумага  
Компьютер выбрал: ножницы  
Компьютер победил!  
* Терминал будет повторно использоваться задачами. Чтобы закрыть его, нажмите любую клавишу.
```

Рисунок 8 – Пример работы программы

4

Выводы. В ходе работы была создана игра «Камень, ножницы, бумага» на языке Rust. Проект позволил освоить основы ввода-вывода, генерации случайных чисел, работы с условиями и строками. Реализация подобной игры демонстрирует доступность Rust для обучения школьников и студентов, а также может служить базой для расширения (например, добавления счёта или повторных раундов).

Разработанная программа успешно реализует консольную игру «Камень, ножницы, бумага». Она корректно обрабатывает ввод пользователя, использует генерацию случайного выбора для компьютера и выводит результат игры.

Библиографический список

1. Ульянов Е. А. Разработка игры камень, ножницы, бумага на C# // Постулат. 2021. № 1(63).
2. Трубаев К. П. Современные языки программирования на примере языка Rust // Материалы XIV Международного молодежного форума, 13–14 октября 2022 года. Белгородский государственный технологический университет им. В.Г. Шухова. 2022. С. 197-200.
3. Берлизов Д. М. Оптимизация выполнения критических секций на языке программирования Rust // Труды 12-й Международной Азиатской школы-семинара / Под редакцией С.И. Кабанихина А.В. Кельманова. Институт вычислительной математики и математической геофизики Сибирского отделения РАН. 2016. С. 65-72.
4. Широченко В. С. Использование языка программирования Rust для высокопроизводительных и безопасных вычислений // Сборник научных статей по материалам XIII Международной научно-практической конференции. Общество с ограниченной ответственностью "Научно-издательский центр "Вестник науки". 2023. С. 238-241.