

Peter the Great  
Saint-Petersburg Polytechnic University

# **Язык программирования Rust. Введение.**

Исполнитель: Команда №2  
Казакевич Анна, Лапина Ольга, Марченко  
Елизавета

# План презентации

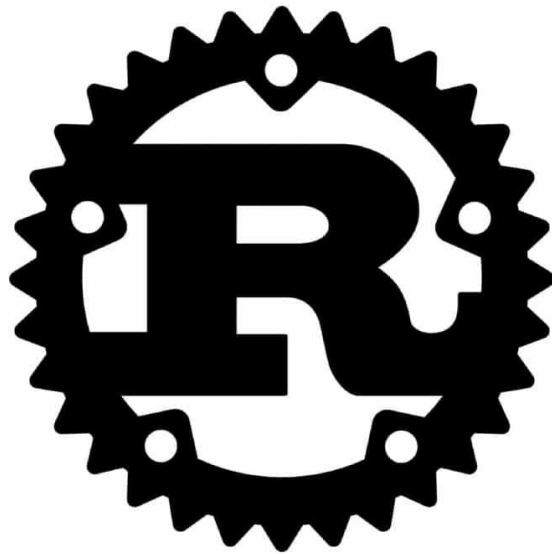
---

- ▶ История языка программирования
- ▶ Основные области применения и примеры проектов
- ▶ Рекомендуемые редакторы и отладка
- ▶ Первая программа на Rust
- ▶ Dockerfile для запуска программы
- ▶ Список литературы
- ▶ План для будущих презентаций

# История языка программирования (1/2)

---

- ▶ Rust был разработан в компании Mozilla Research, и его создание началось в 2006 году.
- ▶ Rust - это системный язык программирования с открытым исходным кодом, который фокусируется на скорости, безопасности памяти и параллелизме.



**The Rust  
Programming  
Language**

# История языка программирования (2/2)

---

1. **2006 год:** Начало разработки языка.
2. **2009 год:** Публикация первой версии Rust.
3. **2010 год:** Mozilla официально приняла Rust в свой проект.
4. **2012 год:** Релиз первой версии Rust, известной как Rust 0.1.
5. **2015 год:** Вышла версия 1.0, что ознаменовало стабильность языка и его готовность для использования в реальных проектах.
6. **2016–2020 годы:** Rust продолжал активно развиваться и улучшаться.
7. **2021–2024 годы:** Rust продолжает развиваться, и его сообщество активно работает над расширением возможностей языка и улучшением экосистемы.

# Основные области применения и примеры проектов (1/2)

---

## 1. Системное программирование

**Пример: Redox OS** — операционная система, написанная целиком на Rust. Этот проект демонстрирует возможности языка в разработке системного программного обеспечения.

## 2. Разработка веб-серверов и веб-приложений

**Пример: Actix и Warp** — популярные веб-фреймворки на Rust, которые обеспечивают высокую производительность и безопасность для веб-разработки.

## 3. Встраиваемые системы и IoT

**Пример: Tock OS** — операционная система для встраиваемых устройств, написанная на Rust. Этот проект иллюстрирует использование языка в области IoT.

## 4. Криптография и безопасность

**Пример: Serenity** — библиотека для работы с криптографией на Rust, обеспечивающая безопасные и эффективные операции.

# Основные области применения и примеры проектов

## (2/2)

---

### 5. Научные и аналитические вычисления

**Пример: Polars** — библиотека для обработки и анализа данных, написанная на Rust, которая предлагает высокую производительность для работы с большими наборами данных.

### 6. Игровая разработка

**Пример: Bevy** — игровая движок, написанный на Rust, который предлагает современные возможности для разработки игр с акцентом на производительность и безопасность.

### 7. Библиотеки и инструменты для разработчиков

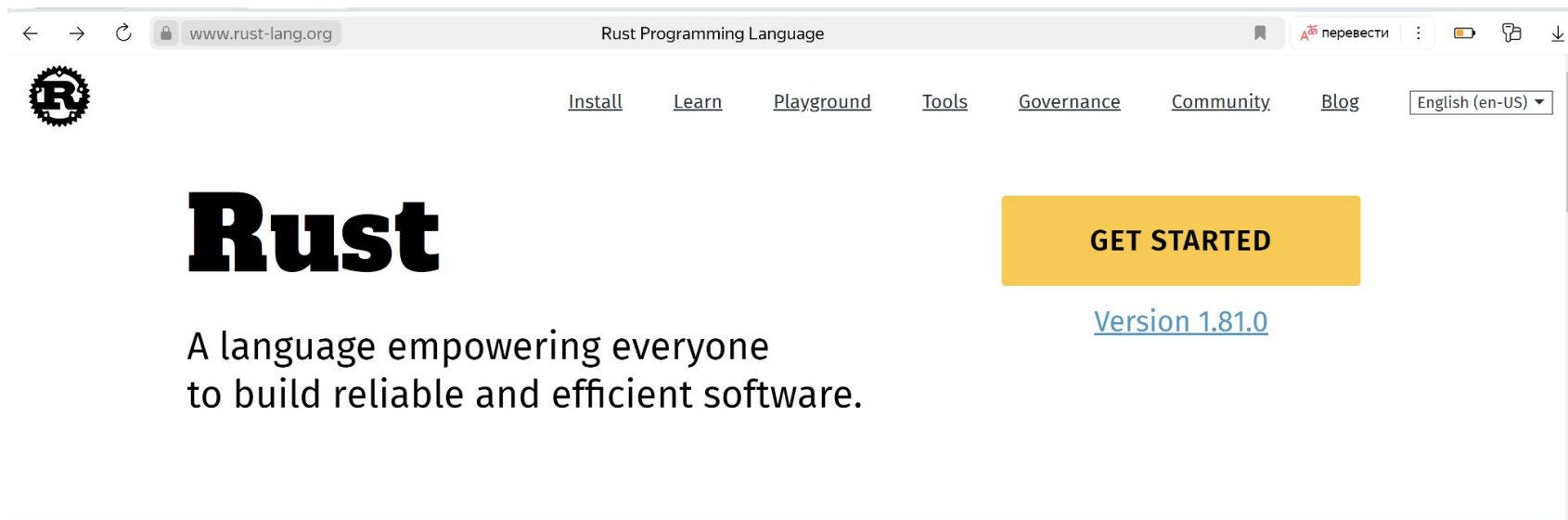
**Пример: Cargo** — система управления зависимостями и сборки на Rust, которая значительно упрощает процесс разработки и управления проектами.

# Установка Rust и рекомендуемые редакторы и отладка (1/4)

## 1. Онлайн-компилятор.

<https://play.rust-lang.org/?version=stable&mode=debug&edition=2021>

## 2. Установите Rust. Следуйте официальному руководству по установке на сайте [rust-lang.org](https://rust-lang.org). Это позволит установить компилятор Rust, cargo (менеджер пакетов и инструмент сборки), rustup (менеджер обновления) и другие полезные инструменты.



# Установка Rust и рекомендуемые редакторы и отладка (2/4)

## 2. Установите Rust (продолжение). После нажатия на Get Started:

### Rustup: the Rust installer and version management tool

The primary way that folks install Rust is through a tool called Rustup, which is a Rust installer and version management tool.

It looks like you're running Windows. To start using Rust, download the installer, then run the program and follow the onscreen instructions. You may need to install the [Visual Studio C++ Build tools](#) when prompted to do so. If you are not on Windows see "[Other Installation Methods](#)".

DOWNLOAD RUSTUP-INIT.EXE (32-BIT)

DOWNLOAD RUSTUP-INIT.EXE (64-BIT)

### Windows Subsystem for Linux

If you're a Windows Subsystem for Linux user run the following in your terminal, then follow the on-screen instructions to install Rust.

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```



# Установка Rust и рекомендуемые редакторы и отладка (3/4)

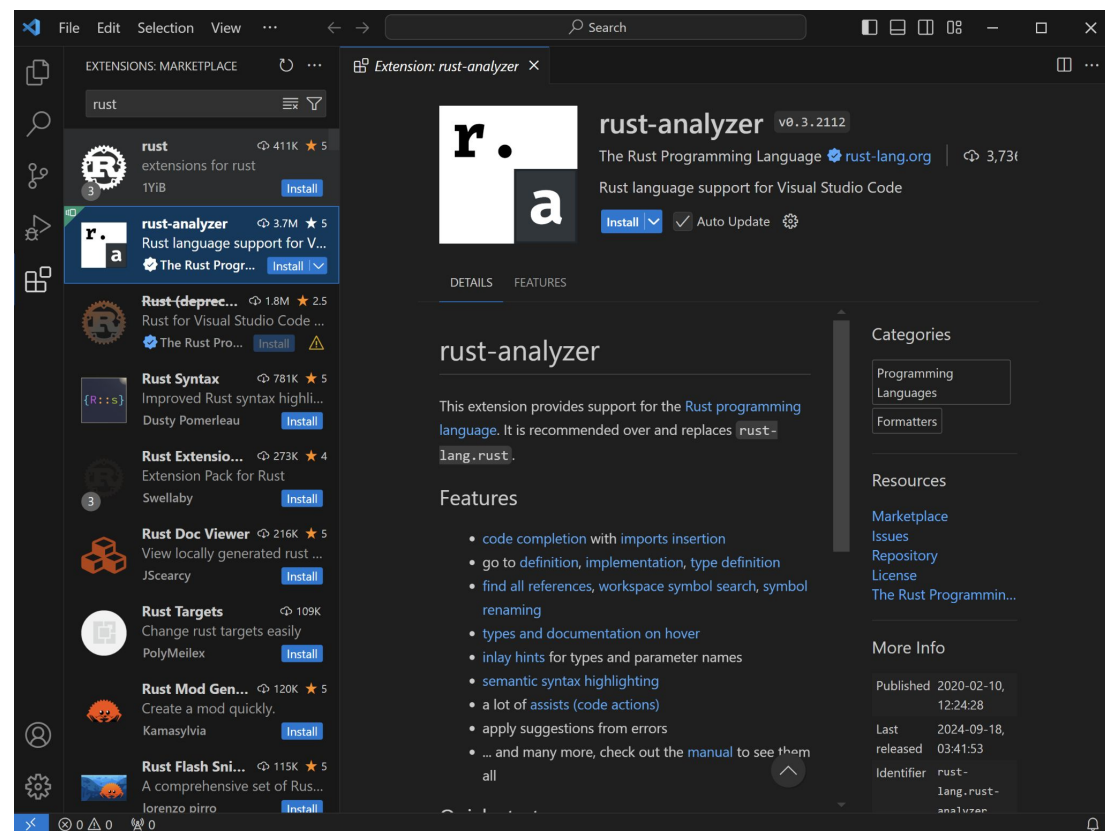
## 2. Варианты редакторов и установка расширений для них.

- ❑ **Visual Studio Code** с расширением Rust (rust-analyzer). Это один из самых популярных редакторов для Rust благодаря своей простоте и множеству функций.

Вы можете найти и установить расширение **rust-analyzer** из VS Code через **представление расширений**

**(Ctrl + Shift + X)**

и выполнить поиск "rust-analyzer". Вы должны установить релизную версию.

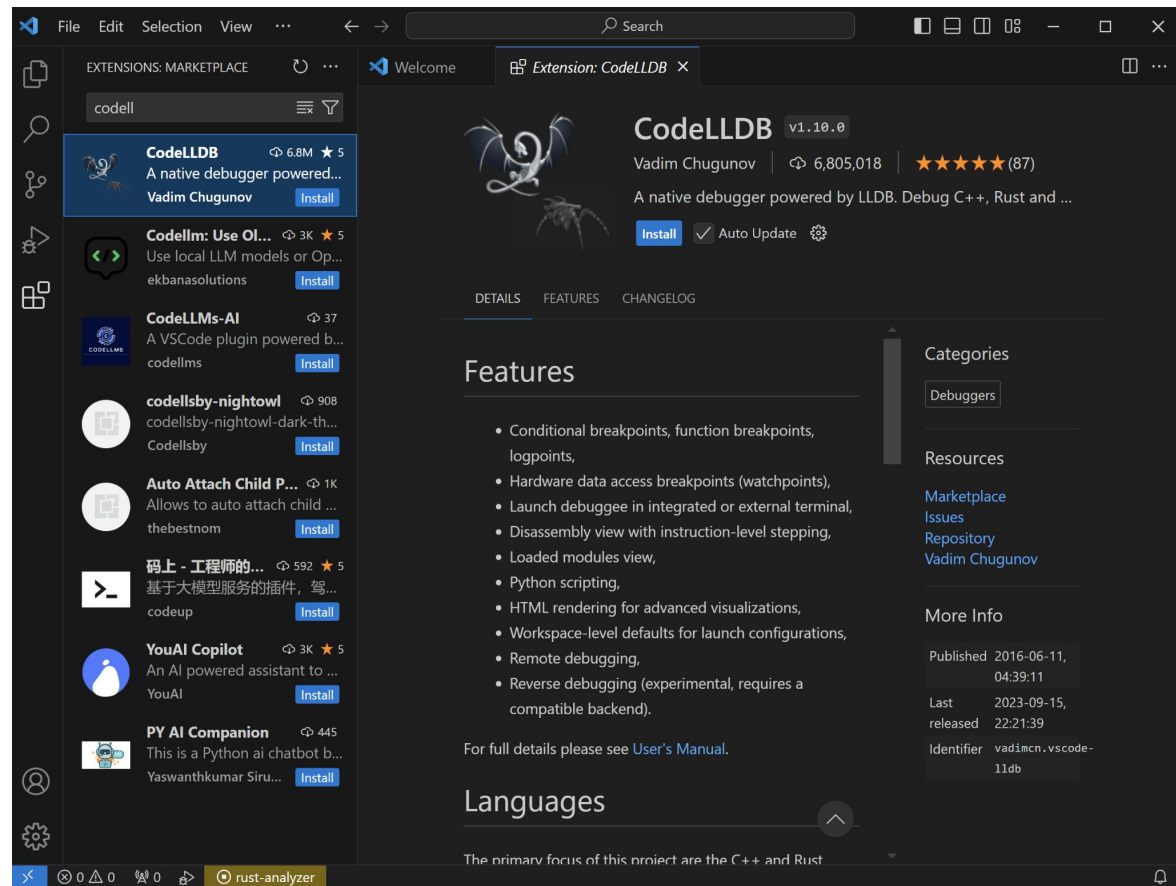


# Установка Rust и рекомендуемые редакторы и отладка (3/4)

## 2. Варианты редакторов и установка расширений для них.

- ❑ **Visual Studio Code** с расширением Rust (rust-analyzer).

Для поддержки отладки нужно установить расширение CodeLLdb с помощью меню расширений.



# Установка Rust и рекомендуемые редакторы и отладка (4/4)

2. Варианты редакторов и установка расширений для них.

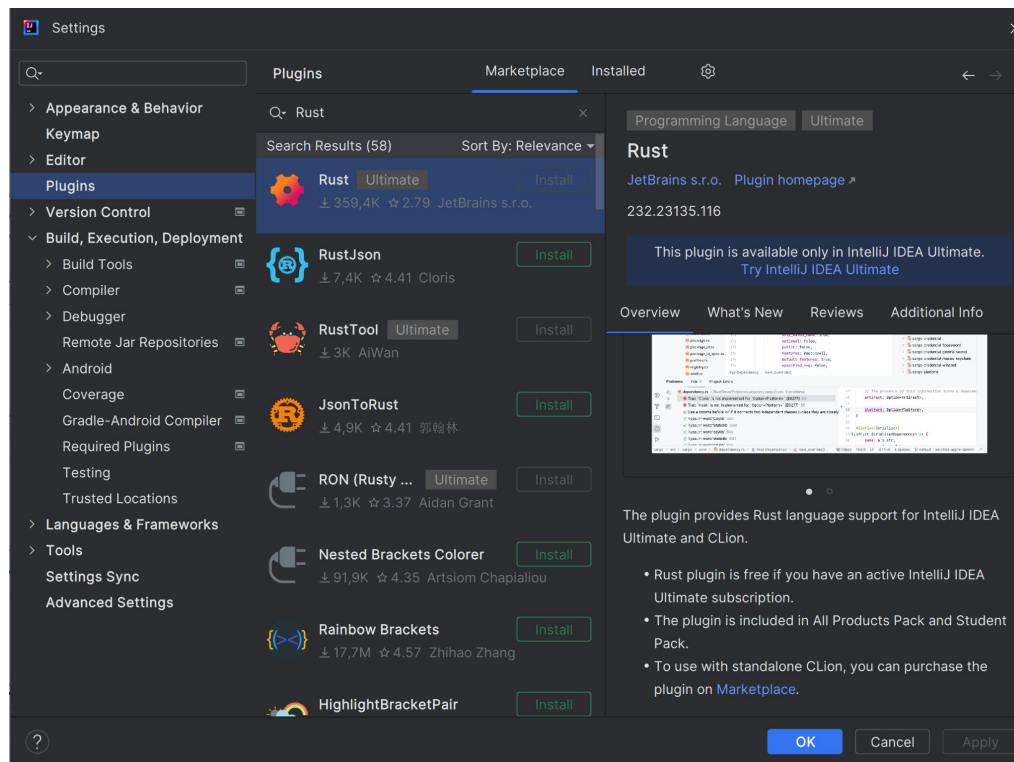
- **IntelliJ IDEA** с плагином Rust. Поддерживает богатую функциональность и интеграцию с инструментами отладки.

## Установка плагина:

1. Запустите IntelliJ IDEA. На экране приветствия нажмите Плагины.

Если у вас уже открыт проект, перейдите в Настройки (Ctrl+Alt+S) | Плагины.

2. Выберите вкладку Marketplace и найдите плагин для Rust.








## Первая программа на Rust: “Hello world” (1/5 )

---





- Для создания новой программы с помощью менеджера пакетов Cargo в терминале нужно ввести следующую последовательность команд:
  1. Создайте новый проект: **cargo new имя\_проекта**
  2. Перейдите в директорию проекта: **cd имя\_проекта**
  3. В файле **src/main.rs** введите необходимый код.
  4. Постройте и запустите проект: **cargo run**
- По умолчанию в файле **src/main.rs** создастся файл с кодом программы Hello world.
- Заметим, что файлы с кодом на Rust имеют расширение **.rs**

## Первая программа на Rust: “Hello world” (2/5 )

- После **cargo new имя\_проекта** будет создано новые файлы (Cargo.toml и Cargo.lock) и папка src (в которой хранятся все файлы для программы):

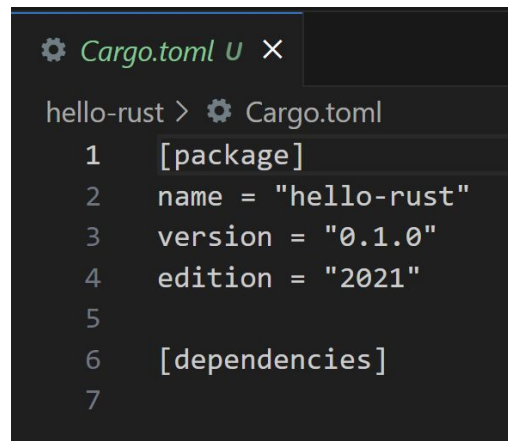
 src	18.09.2024 21:37	Папка с файлами	
 target	18.09.2024 22:15	Папка с файлами	
 .gitignore	18.09.2024 21:37	Текстовый докум...	1 КБ
 Cargo	18.09.2024 21:39	Файл "LOCK"	1 КБ
 Cargo	18.09.2024 21:37	Файл "TOML"	1 КБ

- После **cargo run** формируется папка target, в которой лежат папки с файлом расширения .exe

Имя	Дата изменения	Тип	Размер
 debug	18.09.2024 21:39	Папка с файлами	
 release	18.09.2024 22:15	Папка с файлами	
 .rustc_info	18.09.2024 23:42	JSON File	2 КБ
 CACHEDIR.TAG	18.09.2024 21:39	Файл "TAG"	1 КБ

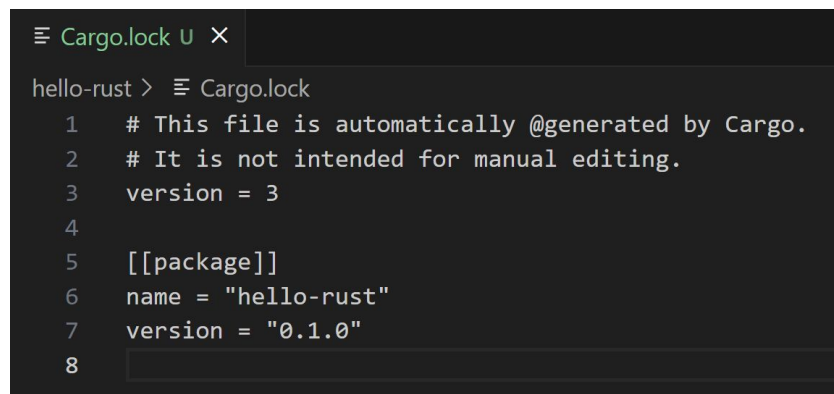
# Первая программа на Rust: “Hello world” (3/ 5)

- **Cargo.toml** — это файл конфигурации для менеджера пакетов Rust, Cargo. Он используется для указания зависимостей и настроек сборки для проекта Rust. Редактируется вами.



```
hello-rust > Cargo.toml
1  [package]
2  name = "hello-rust"
3  version = "0.1.0"
4  edition = "2021"
5
6  [dependencies]
7
```

- **Cargo.lock** - содержит точную информацию о ваших зависимостях. Она поддерживается Cargo и **не должна редактироваться вручную**.



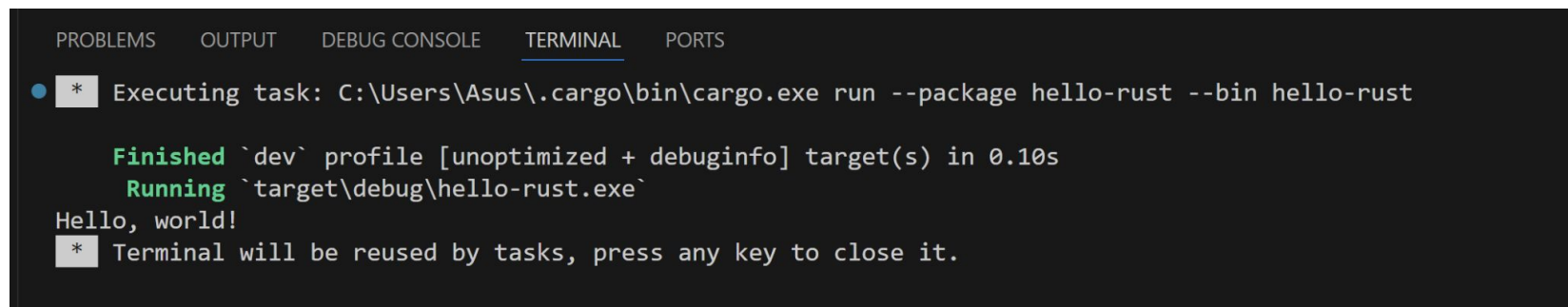
```
hello-rust > Cargo.lock
1  # This file is automatically @generated by Cargo.
2  # It is not intended for manual editing.
3  version = 3
4
5  [[package]]
6  name = "hello-rust"
7  version = "0.1.0"
8
```

## Первая программа на Rust: “Hello world” (4/5 )



The screenshot shows an IDE window titled 'main.rs' with a Rust icon, a search icon, and a close icon. The breadcrumb path is 'hello-rust > src > main.rs > ...'. Below the path is a 'Run | Debug' button. The code editor displays the following Rust code:

```
1 fn main() {  
2     println!("Hello, world!");  
3 }  
4
```

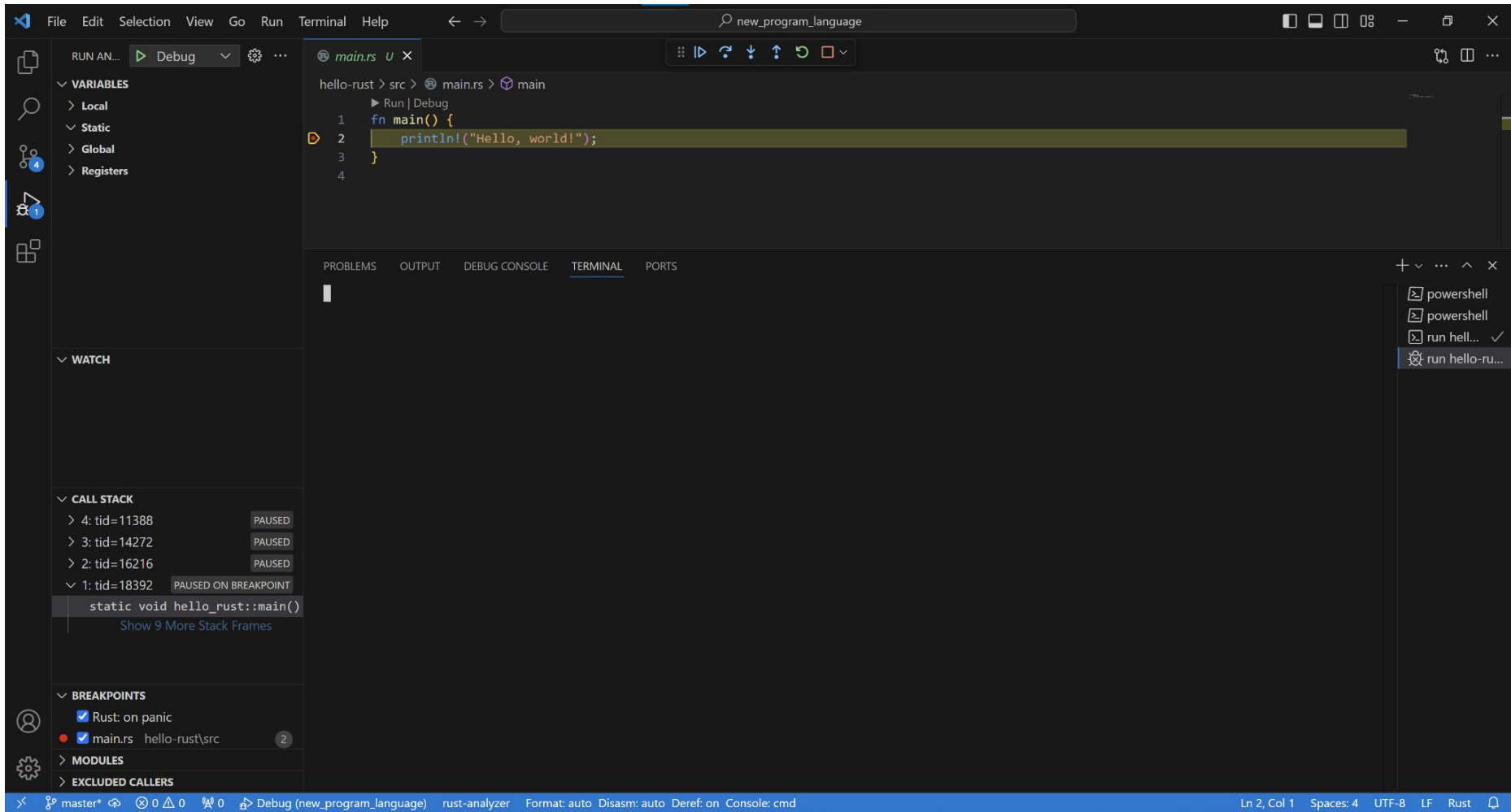


The screenshot shows a terminal window with tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is active. The terminal output shows the execution of the Rust program:

```
* Executing task: C:\Users\Asus\.cargo\bin\cargo.exe run --package hello-rust --bin hello-rust  
  
Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.10s  
Running `target\debug\hello-rust.exe`  
Hello, world!  
* Terminal will be reused by tasks, press any key to close it.
```



# Первая программа на Rust: “Hello world”. Окно отладки в VSCode (5/5 )





# Dockerfile

```
Dockerfile X
test-rust > Dockerfile
1  # Используем официальный образ Rust
2  FROM rust:latest
3
4  # Устанавливаем рабочую директорию
5  WORKDIR /app
6
7  # Создаём новый проект на Rust
8  RUN cargo new app-rust
9
10 # Копируем файл Cargo.toml
11 # (если он отличается от исходного)
12 # COPY Cargo.toml ./app-rust
13
14 # Копируем исходный код
15 COPY main.rs ./app-rust/src
16
17 # Переходим в папку проекта и собираем его
18 RUN cd app-rust && cargo build --release
19
20 # Определяем команду для запуска
21 CMD ["/app-rust/target/release/app-rust"]
22
```

Для запуска Docker - контейнера нужно выполнить следующие команды:

1. Постройте Docker образ:  
**docker build -t my-rust-app .**
2. Запустите контейнер:  
**docker run --rm my-rust-app**

my-rust-app можно заменить на любое другое название

# Dockerfile. Пример запуска.

labpython > new\_program\_language > test-rust

Имя	Дата изменения	Тип	Размер
Dockerfile	19.09.2024 19:49	Файл	1 КБ
main	19.09.2024 19:52	Файл "RS"	1 КБ

```
Командная строка
Microsoft Windows [Version 10.0.19045.4780]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Asus>cd C:\labpython\new_program_language\test-rust

C:\labpython\new_program_language\test-rust>docker build -t hello-world .
[+] Building 2.4s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 724B                                0.0s
=> [internal] load metadata for docker.io/library/rust:latest      1.4s
=> [internal] load .dockerignore                                   0.0s
=> => transferring context: 2B                                       0.0s
=> [1/5] FROM docker.io/library/rust:latest@sha256:fcd390e0a3a6bfcf26969861efbe7b864df052aa71a361cf3cd7c5c585b1b 0.0s
=> [internal] load build context                                   0.0s
=> => transferring context: 79B                                       0.0s
=> CACHED [2/5] WORKDIR /app                                       0.0s
=> CACHED [3/5] RUN cargo new app-rust                             0.0s
=> [4/5] COPY main.rs ./app-rust/src                             0.0s
=> [5/5] RUN cd app-rust && cargo build --release                  0.7s
=> exporting to image                                              0.1s
=> => exporting layers                                              0.1s
=> => writing image sha256:c5b25df3262f0c95512b4101e6f54d151db01eaafaaa0b6ba21cffcca55248d3 0.0s
=> => naming to docker.io/library/hello-world                     0.0s

View build details: docker-desktop://dashboard/build/default/default/oedy2tzg4u6ztty8b7ljpoxbr

What's Next?
  1. Sign in to your Docker account → docker login
  2. View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\labpython\new_program_language\test-rust>docker run --rm hello-world
Hello, world!

C:\labpython\new_program_language\test-rust>
```

# Список литературы

---

- **The Rust Programming Language** - Steve Klabnik, Carol Nichols. [Читать онлайн](<https://doc.rust-lang.org/book/>)
- **Rust by Example** - [Читать онлайн](<https://doc.rust-lang.org/rust-by-example/>)
- **Programming Rust** - Jim Blandy, Jason Orendorff.
- **Управление зависимостями в Rust с Cargo** - <https://habr.com/ru/companies/otus/articles/788642/>

# План для будущих презентаций

---

2. Углубленное изучение типов данных и структур - подробное рассмотрение типов данных, структур и перечислений.
3. Функции, конструкторы потока управления (циклы, условия и т.д.) и стандартная библиотека ввода/вывода
4. Коллекции и обобщенные типы
5. Обработка ошибок и управление памятью
6. Параллелизм и ООП