بنام خدا

تمرین‌های درس: برنامه نویسی سیستمی

نام و نام خانوادگی: محمد صادق محبی

دانشجوی ترم 4 کارشناسی ناپیوسته

شماره دانشجویی :01221153705003

تاریخ: **جمعه ۱۴ دی**

1 زبان برنامه نویسی Rust چیست؟

Rust یک زبان برنامه نویسی سیستمی است که برای عملکرد بالا، امنیت حافظه و همزمانی طراحی شده است. این زبان توسط موزیلا توسعه داده شد و اکنون به صورت مستقل توسط بنیاد Rust مدیریت می‌شود.

2 ویژگی های زبان Rust را به اختصار توضیح دهید؟

* مدیریت حافظه ایمن (بدون نیاز به garbage collector)
* همزمانی ایمن (عدم وجود data race)
* سرعت بالا و کارایی بهینه
* قابلیت برنامه نویسی سیستمی و سطح پایین
* پشتیبانی از برنامه نویسی تابعی و شیءگرا
* مدیریت خطا به صورت کارآمد

3 زبان Rust درچه حوزه هایی کاربرد بیشتری دارد؟

* توسعه سیستم‌های عامل
* موتورهای بازی
* برنامه‌های تحت وب با استفاده از WebAssembly
* نرم‌افزارهای توزیع شده
* ابزارهای خط فرمان
* بلاکچین و قراردادهای هوشمند

4 تفاوت زبان Rust با زبان های C و ++C چیست؟

* ایمنی حافظه بدون نیاز به garbage collector
* جلوگیری از باگ‌های مربوط به اشاره‌گرها (null pointer, dangling pointer)
* مدیریت مالکیت داده‌ها (Ownership)
* جلوگیری از data race
* کدنویسی مدرن‌تر و راحت‌تر

5 محدودیت های زبان Rust را توضیح دهید؟

* یادگیری دشوارتر نسبت به زبان‌های سطح بالا
* زمان کامپایل طولانی‌تر
* کتابخانه‌های کمتر نسبت به زبان‌هایی مانند C++

6 درمورد ابزار Cargo توضیح دهید؟

**Cargo**  ابزار مدیریت بسته (package manager) و ساخت (build system) برای Rust است که کارهایی مانند کامپایل، اجرای تست‌ها و مدیریت وابستگی‌ها را انجام می‌دهد.

7 فایل lock.Cargo چه عملکردی دارد؟

فایل Cargo.lock نسخه دقیق وابستگی‌ها را قفل می‌کند تا اطمینان حاصل شود که همه وابستگی‌ها در طول توسعه ثابت می‌مانند.

8 در مورد IDE مطرح RustRover توضیح دهید؟

**RustRover** یک محیط توسعه یکپارچه (IDE) برای Rust است که توسط JetBrains توسعه داده شده و امکاناتی مانند تکمیل کد، دیباگ و مدیریت پروژه را ارائه می‌دهد.

9 چند مورد از برنامه ها یی که به زبان Rust نوشته شده اند را نام ببرید؟

* سیستم‌عامل Redox
* موتور بازی Amethyst
* مرورگر Servo
* ابزار خط فرمان ripgrep
* بلاکچین Solana

10 قابلیت های زبان Rust در Concurrency و Parallelism را به اختصار و با ذکر مثال بیان کنید؟

**ایمنی حافظه (Data Race-Free):**

* **برنامه‌های چند نخی (Multithreaded): نوشتن برنامه‌هایی که همزمان روی داده‌ها کار می‌کنند بدون نگرانی از خطاهای ناشی از دسترسی همزمان.**
* **سیستم‌های حساس به پایداری: در سیستم‌های Embedded و نرم‌افزارهای سیستمی که خطاهای حافظه خطرناک هستند.**

**Arc اشتراک امن داده‌ها بین threadها:**

* **وب‌سرورها: اشتراک‌گذاری درخواست‌ها یا داده‌های مشترک بین چندین thread برای پاسخ‌دهی سریع‌تر.**
* **پایگاه داده‌ها: دسترسی همزمان به رکوردهای ثابت و بدون تغییر (Read-Only).**

**Mutex هم‌رسانی داده‌های mutable:**

* **پردازش‌های موازی: به‌روزرسانی داده‌های مشترک مانند cache یا وضعیت کلی سیستم.**
* **سیستم‌های گیمینگ و گرافیکی: همگام‌سازی داده‌های mutable بین پردازش‌های گرافیکی و فیزیکی.**

**Rayon پردازش موازی داده‌ها:**

* **پردازش داده‌های حجیم: پردازش مجموعه‌های بزرگ مانند تصاویر، ویدیو یا داده‌های علمی.**
* **Machine Learning و AI: پردازش موازی برای بهینه‌سازی الگوریتم‌های یادگیری.**
* **تحلیل داده (Data Analysis): پردازش سریع‌تر روی حجم زیادی از داده‌ها با توزیع بار بین هسته‌های CPU**

11 درمورد امکان برنامه نویسی زبان Rust برای Microcontroller ها و Robotها توضیح دهید؟

**Robot** برای سیستم‌های تعبیه شده (Embedded Systems) از طریق پروژه‌هایی مانند embedded-hal پشتیبانی می‌کند که برای برنامه‌نویسی میکروکنترلرها و ربات‌ها استفاده می‌شود.

12 کاربرد و مفهوم Crate در زبان Rust را بیان کنید؟

**Crate**  یک بسته یا کتابخانه در Rust است که می‌تواند شامل کد منبع یا باینری باشد.

13 تفاوت مفهوم Iterator و Generator در زبان Rust را با ذکر مثال توضیح دهید؟

* Iterator یک API برای پیمایش روی عناصر است.
* Generator مشابه iterator است اما lazy evaluation دارد.

مثال Iterator:



14 Monomorphization در زبان Rust چیست؟

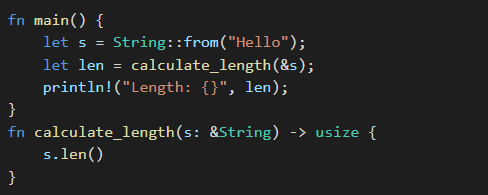
**Monomorphization**  فرآیندی که طی آن کد جنریک (generic) به کد مشخص (concrete) برای هر نوع تبدیل می‌شود.

15 درمورد مفهوم Ownership در زبان Rust توضیح دهید؟

هر مقدار در Rust دارای مالک (owner) است و در هر لحظه فقط یک مالک وجود دارد.

16 قابلیت Borrowing در زبان Rust را با ذکر مثال توضیح دهید؟

امکان قرض گرفتن مقادیر بدون انتقال مالکیت.



17 انواع Inference Type در زبان Rust را با ذکر مثال توضیح دهید؟

**استنباط نوع متغیر (Variable Inference)**

**استنباط نوع بازگشتی توابع (Function Return Type Inference)**

**استنباط نوع کلوزر (Closure Type Inference)**

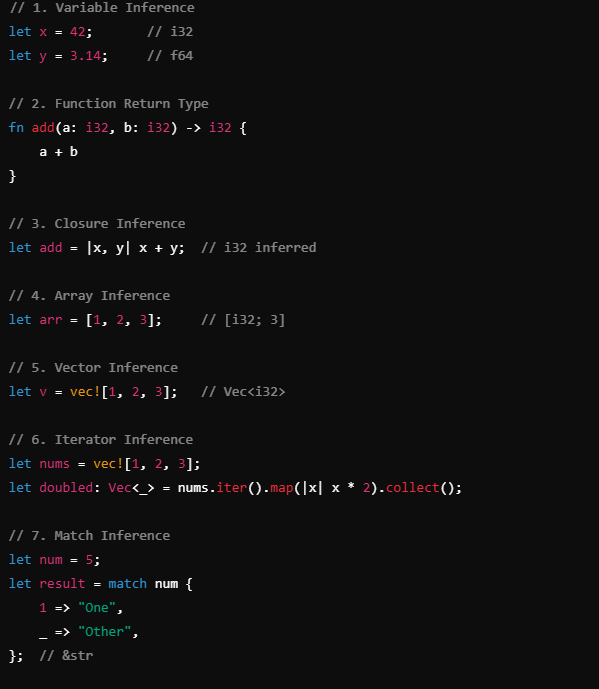
**استنباط نوع آرایه (Array Inference)**

**استنباط نوع وکتور (Vector Inference)**

**استنباط نوع Iterator**

**استنباط نوع در Match**

و غیره...

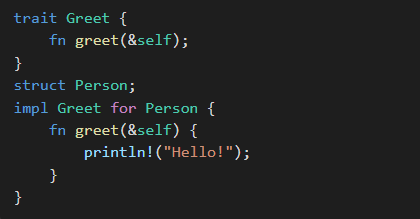


18 مفهوم Pointer Smart در زبان Rust را توضیح دهید؟

اشاره‌گرهای هوشمند مانند Box, Rc, و Arc برای مدیریت حافظه استفاده می‌شوند.

19 ساختار Trait چیست؟ با ذکر مثال توضیح دهید ؟

در زبان **Rust**، Trait مشابه **interface** در زبان‌های دیگر است و برای تعریف رفتار (behavior) که توسط انواع مختلف پیاده‌سازی می‌شود، استفاده می‌شود



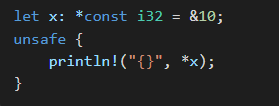
20 Closure چیست را ذکر مثال توضیح دهید؟

تابع بدون نام که می‌تواند متغیرهای محیط را بگیرد



21 توضیح دهید زبان Rust با کدهای نا امن) Code unsafe )چطور رفتار می کند؟ با ذکر مثال توضیح دهید؟

اجازه می‌دهد تا از کد ناامن استفاده شود



22 درمورد کاربرد زبان برنامه نویسی Rust در حوزه Contracts Smart توضیح دهید؟

Rust به دلیل امنیت بالا برای نوشتن قراردادهای هوشمند در بلاکچین استفاده می‌شود.

23 درمورد کاربرد زبان برنامه نویسی Rust در حوزه Blockchain توضیح دهید؟

Rust به دلیل کارایی بالا برای توسعه بلاکچین‌هایی مانند Solana استفاده شده است.

24 درمورد کاربرد زبان برنامه نویسی Rust در حوزه Intelligence Artificial توضیح دهید؟

Rust در توسعه کتابخانه‌های AI و ML مانند tch-rs برای TensorFlow و PyTorch کاربرد دارد.

25 ابزارهای Test نویسی در زبان Rust را نام ببرید و تفاوت آنها را بیان کنید؟

* cargo test
* assert!
* assert\_eq!
* proptest برای تست‌های مبتنی بر property

**26 ابزارهای برنامه‌نویسی Web در زبان Rust**

در زبان Rust برای توسعه وب چندین ابزار وجود دارد که معمولاً از آنها در ساخت وب‌سایت‌ها و API‌ها استفاده می‌شود:

1. **Rocket**: فریم‌ورک توسعه وب است که بر سادگی و سرعت تمرکز دارد. Rocket به توسعه‌دهندگان این امکان را می‌دهد که API‌های پیچیده را به راحتی بسازند و از ویژگی‌هایی مانند درخواست‌های ایمن و فرمت‌های داده ساده پشتیبانی می‌کند.
2. **Actix-web**: یک فریم‌ورک وب سریع و مقاوم است که از مدل‌های تطبیق‌کننده و ناهمگام استفاده می‌کند. Actix به دلیل سرعت بالا شناخته شده است و برای ساخت سیستم‌های مقیاس‌پذیر مناسب است.
3. **Warp**: یک فریم‌ورک وب دیگر است که بر سادگی، ایمنی و عملکرد تمرکز دارد. Warp از ویژگی‌هایی مانند فیلترهای ترکیبی برای مدیریت مسیرها و درخواست‌ها پشتیبانی می‌کند.
4. **Tide**: فریم‌ورک ساده و ناهمگام است که برای ایجاد API‌های وب ساخته شده است. Tide بر روی Snafu و async/await تمرکز دارد.

تفاوت‌ها:

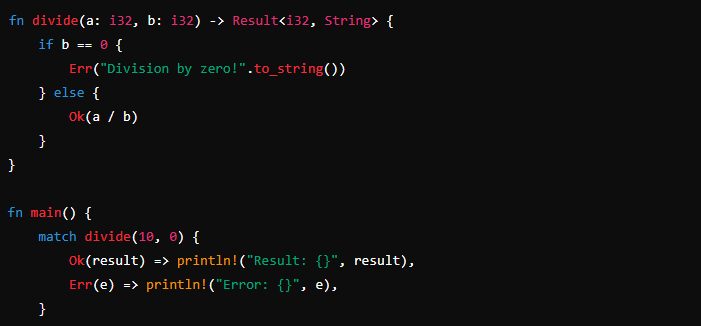
* **Rocket** بیشتر بر سادگی و استفاده از ویژگی‌های Rust تمرکز دارد.
* **Actix-web** بر سرعت و عملکرد بالا تاکید دارد و برای پروژه‌های مقیاس‌پذیر توصیه می‌شود.
* **Warp** سادگی را برای توسعه‌دهندگان و ساختار ترکیب‌پذیر فیلترها ارائه می‌دهد.
* **Tide** برای توسعه‌دهندگانی که به سادگی و بهره‌برداری از async/await علاقه‌مند هستند مناسب است.

**27 مکانیزم کنترل خطا (Error Handling) در زبان Rust**

Rust برای مدیریت خطاها از دو نوع اصلی استفاده می‌کند:

1. **Result<T, E>**: برای مدیریت خطاهای قابل پیش‌بینی یا خطاهایی که می‌توان آنها را کنترل کرد، استفاده می‌شود.
2. **Option<T>**: برای زمانی که مقدار ممکن است موجود نباشد (مثل null در زبان‌های دیگر).

**مثال:**



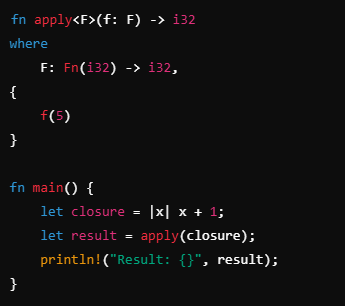
در اینجا، اگر مقسوم‌علیه صفر باشد، یک خطا باز می‌گردد که به شکل Err نمایش داده می‌شود.

**28 قابلیت‌های زبان Rust در Programming Functional**

Rust از مفاهیم برنامه‌نویسی تابعی (Functional Programming) پشتیبانی می‌کند:

1. **توابع مرتبه اول (First-Class Functions)**: در Rust می‌توان توابع را به عنوان مقادیر دریافت یا بازگرداند.
2. **توابع ناشناس (Anonymous Functions) یا Closure**: این توابع می‌توانند متغیرهایی از محیط خود را به طور خودکار ذخیره کنند.

**مثال:**

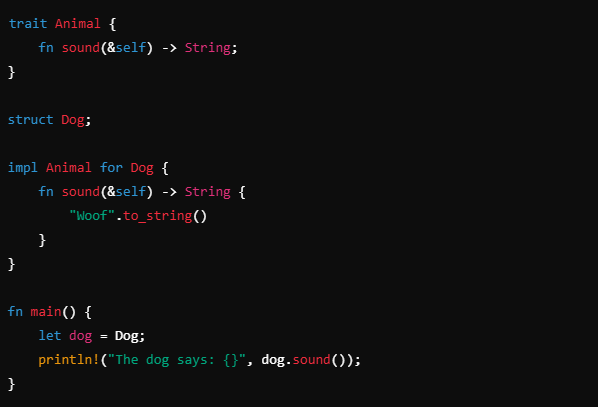


در اینجا یک تابع ناشناس (closure) استفاده می‌شود که بر روی عدد ۵ اعمال می‌شود.

**29 برنامه‌نویسی شی‌گرا (Object-Oriented Programming) در زبان Rust**

Rust به طور مستقیم از مفاهیم کلاس‌ها و ارث‌بری مانند سایر زبان‌های شی‌گرا پشتیبانی نمی‌کند، اما می‌تواند مفاهیم شی‌گرای مشابه را با استفاده از ویژگی‌هایی مانند **ساختارها (Structs)** و **متحولات (Traits)** پیاده‌سازی کند.

**مثال:**

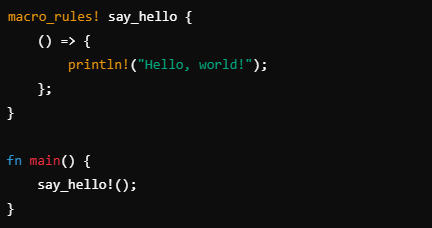


در اینجا از **Traits** برای تعریف رفتارهایی مانند صدای یک حیوان استفاده شده است.

**30 مفهوم Meta-programming (Profiling، Reflection) در زبان Rust**

Rust به طور کامل از **Reflection** پشتیبانی نمی‌کند (یعنی نمی‌توانید به طور داینامیک اطلاعات نوع را در زمان اجرا بدست آورید). اما قابلیت‌هایی مانند **Profiling** و **Macros** برای تولید کد و انجام برخی کارهای مشابه به متا-برنامه‌نویسی وجود دارند.

**مثال (Macro برای تولید کد):**



در اینجا یک **Macro** برای چاپ پیام «سلام، دنیا!» تعریف شده است که به نوعی متا-برنامه‌نویسی را شبیه‌سازی می‌کند.

31 راهکار زبان Rust در مواجهه با مشکالت Safety Memory چیست؟ با ذکر یک مثال ساده توضیح دهید؟

Rust با استفاده از **سیستم مالکیت (Ownership)**، **وام‌گیری (Borrowing)** و **قوانین طول عمر (Lifetimes)**، از مشکلات رایج ایمنی حافظه جلوگیری می‌کند. این مکانیزم‌ها در زمان کامپایل بررسی می‌شوند و باعث می‌شوند خطاهای حافظه به حداقل برسند.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **مشکل حافظه** |  |  | | --- | |  | | | **راهکار Rust** | | --- |  |  | | --- | |  | |
| |  | | --- | | **Use after free** |  |  | | --- | |  | | |  | | --- | | سیستم مالکیت اجازه دسترسی به داده‌های آزادشده را نمی‌دهد. |  |  | | --- | |  | |
| |  | | --- | | **Double free** |  |  | | --- | |  | | |  | | --- | | فقط یک مالک برای داده وجود دارد، از آزادسازی مجدد جلوگیری می‌شود. |  |  | | --- | |  | |
| |  | | --- | | **Memory leaks** |  |  | | --- | |  |  |  | | --- | |  | | |  | | --- | | استفاده از Drop برای آزادسازی حافظه به صورت خودکار. |  |  | | --- | |  | |
| |  | | --- | | **Null pointers** |  |  | | --- | |  | | |  | | --- | | استفاده از نوع Option به جای اشاره‌گر Null. |  |  | | --- | |  | |
| |  | | --- | | **Buffer overreads/overwrites** |  |  | | --- | |  | | |  | | --- | | ایمنی آرایه‌ها و جلوگیری از دسترسی خارج از محدوده. |  |  | | --- | |  | |
| |  | | --- | | **Data races** |  |  | | --- | |  | | استفاده از Mutex و Arc برای اشتراک ایمن داده بین threadها. |

32 عملکرد کتابخانه های زیر را با ذکر یک مثال ساده بیان کنید؟

1 **Serde** – سریال‌سازی و دی‌سریال‌سازی



2 **Lazy\_static** – تعریف متغیرهای استاتیک که به صورت تنبل مقداردهی می‌شوند



3 **Thiserror** – مدیریت خطاها



4 **Itertools** – افزودن قابلیت‌های بیشتر به Iteratorها



5 **Time** – مدیریت و محاسبات زمان



6 **Url** – تجزیه و ترکیب URLها



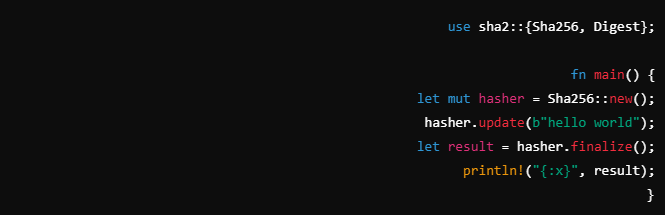
7 **Reqwest** – ارسال درخواست‌های HTTP



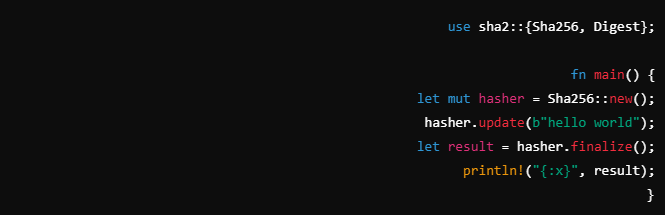
8 **Anyhow** – مدیریت خطاها به صورت ساده



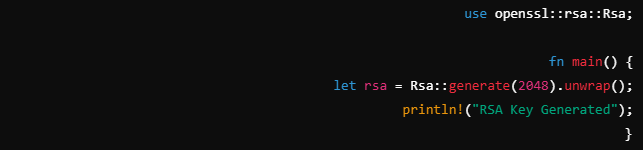
9 **Digest** – محاسبه هش



10 **Signature** – مدیریت امضاهای دیجیتال



11 **Openssl** – رمزنگاری و SSL



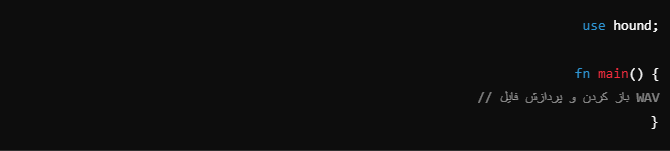
12 **Chrono** – مدیریت تاریخ و زمان



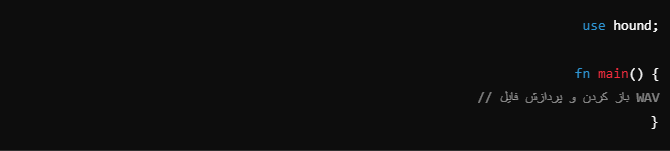
13 **Geo** – کار با موقعیت‌های جغرافیایی



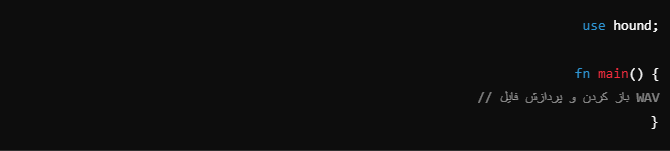
14 **Hound** – کار با فایل‌های صوتی



15 **Rustls** – پیاده‌سازی TLS امن



16 **Ring** – رمزنگاری و عملیات رمزنگاری



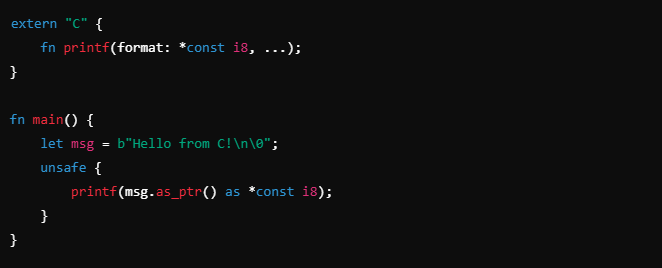
33 درمورد امکان فراخوانی کدهای دیگرزبان ها در زبان Rust توضیح دهید؟

Rust امکان فراخوانی کدهای نوشته‌شده به زبان‌هایی مانند C و C++ را از طریق **FFI (رابط تابع خارجی)** فراهم می‌کند. این قابلیت به Rust اجازه می‌دهد تا از کتابخانه‌های موجود و کدهای سیستمی استفاده کند.

**روش:**

* استفاده از extern "C" برای تعریف توابع خارجی.
* پیاده‌سازی توابع با استفاده از unsafe برای اطمینان از ایمنی.

**مثال (فراخوانی کد C:**



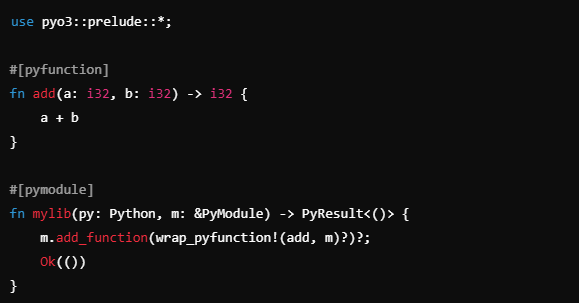
34 درمورد امکان فراخوانی کدهای زبان Rust در زبان هایی مانند Java ، #C، C وPython توضیح دهید؟

Rust امکان استفاده از کتابخانه‌های Rust در زبان‌هایی مانند C، Python، Java و #C را با تولید **کتابخانه‌های مشترک (Shared Library)** یا **Rust DLL** فراهم می‌کند.

**روش‌ها:**

* ایجاد کتابخانه‌های cdylib یا staticlib برای تولید فایل .so یا .dll.
* استفاده از #[no\_mangle] برای جلوگیری از تغییر نام توابع توسط کامپایلر.
* ابزارهایی مانند PyO3 برای Python و jni برای Java.

**مثال (ایجاد کتابخانه برای Python):**



35 در قالب یک مثال Programming Asynchronous در زبان Rust را توضیح دهید؟

Rust از برنامه‌نویسی ناهمگام (Asynchronous) با استفاده از async و await پشتیبانی می‌کند. این قابلیت اجازه می‌دهد تا بدون مسدود کردن thread، عملیات ورودی/خروجی انجام شود.

**مثال (برنامه ناهمگام):**

