Домашнее задание

Всего баллов: 12

Бонусные баллы: 2

1. Максимум «на максималках»

Баллы: 2

> Зло, добро, вещественные числа, – серьезно сказал Ведь'*Макс* — меньшее, большее, среднее – все едино, пропорции условны, а границы размыты. Главное — чтобы удовлетворяло единому интерфейсу, аминь! Я не pure functional отшельник, не только хороший код писал в жизни. И если приходится выбирать, для какого типа реализовать функцию, я предпочитаю не выбирать вообще.

> Ведьмакам платят меньше, чем программистам, и не называют мутантами на улицах (ведь программисты не выходят на улицу), поэтому Макс решил переквалифицивроваться.

Напишите функцию `max(a, b) → ?`, работающую на любых типах, которые корректно сравнивать.

Задание заключается в том, что нагуглить и придумать корректную сигнатуру нужно самим.

2. Ord-Обертка вокруг `float`

Баллы: 4

> В Первом отделу собираются реализовать Stalin sort для системы материального поощрения сотрудников. Они пишут сортировку, работающую с любым типом, поддерживающим сравнение, а ваша задача — заставить его работать для `float`-ов. Они обладают индивидуальностью, и не каждый сравним с каждым, но Первому отделу наплевать, они хотят подогнать всех под одну гребёнку.
>
> Вы работаете в другом отделе и не хотите часто связываться с Первым. К счастью, вам и не придётся, ведь в стандартной библиотеке есть `trait`, который поможет вам понять друг друга: `Ord`.

Создайте структуру `OrdFloat` с единственным полем внутри — `f32`, которая реализует `Ord`. `NaN` должны считаться больше всех остальных чисел, даже `inf`.

*На самом деле, конечно, для этого есть готовая библиотека (на сленге **Rust** — crate). Мы ею попробуем вопользоваться через два занятия, когда обсудим сборку в **Rust**.*

3. Обобщение языков разметки

Баллы: 6

> Вы решили написать свою обобщённую систему разметки, из которой можно будет конвертировать в любую другую. Уже готовы и трейты, и сигнатуры, так что полдела сделано. На этом вы *потеряли интерес к задаче*, впали в депрессию от того, что *тесты не проходят (ведь код ещё не написан)*, погрузились в беспросветную прокрастинацию и уже полгода присылаете друзьям текст в примерно таким формате:
>
> ```
> Article {
>       title: "About something (todo!())".into(),
>       sections: vec!["Section A".into(), "Section B".into()],
>       conclusion: "Severe case of procrastination".into(),
>       references: vec![Link {
>           content: "Google".into(),
>           url: "www.google.com".into(),
>       }],
>   }
> ```
>
> Пора положить этому конец, ведь вы знаете, какому другу какой язык разметки больше нравится. У вас есть *неделя*, чтобы заменить все `todo!()` в проекте на что-то осмысленное.

Посмотрим на языки разметки. Несмотря на то, что у них отличается синтаксис, можно найти общий функционал, который есть в любом языке разметки. Ограничемся html и Markdown. В обоих языках есть загаловки, разные выделения текста и гиперссылки. Дано несколько структур, для которых нужно написать преобразовать в html и Markdown. Чтобы не писать кучу функций вида `fn convert_<struct name>_to_<lang name>` разделим интерфейс на конвертацию структуры (`Convert`) и конвентора в отдельный язык разметки (`Conventer`).

```
pub trait Convert {
    fn convert<C>(&self, conventer: &mut C) -> Result<(), C::Error>
    where
        C: Conventer;
}
```

```
pub trait Conventer {
    type Error;

    fn write_header(&mut self, text: &str) -> Result<(), Self::Error>;
    fn write_text(&mut self, text: &str) -> Result<(), Self::Error>;
    fn write_bold_text(&mut self, text: &str) -> Result<(), Self::Error>;
    fn write_link(&mut self, content: &str, url: &str) -> Result<(),
Self::Error>;
}
```

Требуется реализовать `Convert` для нескольких структур с данными и реализовать конвенторы в html и Markdown. Для обобщенности, конвентор может писать результат в что-угодно, что удволетворяет `std::io::Write`, например, в локальный буффер, файл или сокет. Рекомендуется ознакомиться с этим трейтом в *документации*. Для форматированной записи в `Write` рекомендуется использовать *макрос* `writeln`