Crate out_of_the_box

- Алгоритм обирає із *доступних значень* так, щоб вони *задовільняли усі перерахованні умови* та повертає їх у вигляді вектора.
- Якщо алгоритм не може повернути *бажане значення*, тоді він намагається повернути *найближче значення, яке є не менше бажаного*, якщо і це не можливо тоді *найближче менше*.
- Алгоритм повертає лише ті значення, які є *дозволеними і доступними*.
- Алгоритм повертає *порожній вектор, якщо жодне із значень не підходить*.
- Результат виконання алгоритму *не має повторів*.
- Всі вхідні дані *відсортовані у порядку зростання*.
- Також масиви *'allowed'* та *'preferred'* можуть мати один елемент *"any"*. "any" в масиві дозволених значень означає, що дозволеними є всі значення. "any" в масиві бажаних значень означає, що бажаним є будь-яке значення.

Structs

Value

`available` - вектор доступних значень. `allowed` - вектор дозволених значень. `preferred` - вектор бажаних значень.

Functions

check_key_any

Функція **check_key_any** шукає ключове слово 'any'. Якщо воно відсутнє вектор залишається без змін. У разі співпадіння, вектор очищається і переписується новими даними.

i32_parse

Функція **i32_parse** приймає Vec<&str> та повертає Vec<&i32>.

Function out_of_the_box::check_key_any

```
pub fn check_key_any(item: &mut Vec<&str>)
```

Функція **check_key_any** шукає ключове слово 'any'. Якщо воно відсутнє вектор залишається без змін. У разі співпадіння, вектор очищається і переписується новими даними.

Приклад

```
let mut allowed = vec!["360", "any"];
let mut preferred = vec!["360", "720"];

check_key_any(&mut allowed);
check_key_any(&mut preferred);

assert_eq!(vec!["240", "360", "480", "720", "1080"], allowed);
assert_eq!(vec!["360", "720"], preferred);
```

Function out_of_the_box::i32_parse

```
pub fn i32_parse(vec: Vec<&str>) -> Vec<i32>
```

Функція **i32_parse** приймає Vec<&str> та повертає Vec<&i32>.

Приклад

```
let allowed = vec!["240", "360", "480", "720", "1080"];

let allowed = i32_parse(allowed);

assert_eq!(vec![240, 360, 480, 720, 1080], allowed);
```

Struct out_of_the_box::Value

```
pub struct Value {
    pub available: Vec<i32>,
    pub allowed: Vec<i32>,
    pub preferred: Vec<i32>,
}
```

- `available` - вектор доступних значень.
- `allowed` - вектор дозволених значень.
- `preferred` - вектор бажаних значень.

Fields

`available: Vec<i32>`
`allowed: Vec<i32>`
`preferred: Vec<i32>`

Implementations

```
impl Value
```

```
pub fn attempt(&self) -> Vec<i32>
```

Функція **attempt** приймає посилання self, реалізує алгоритм та повертає вектор значень, які задовільняють умови запиту.

Приклад

```
let query = Value {
  available: vec![240, 720],
  allowed: vec![240, 360, 720, 1080],
  preferred: vec![240, 360],
};

let result = query.attempt();

assert_eq!(vec![240, 720], result);
```

```
pub fn ready_to_use(&mut self)
```

Функція **ready_to_use** приймає змінне посилання на self. Сортує значення від найменшого до найбільшого та видаляє дублікати.

Приклад

```
let mut query = Value {
  available: vec![360, 240, 360, 720],
  allowed: vec![1080, 720, 240, 720],
  preferred: vec![240, 480],
};

query.ready_to_use();

assert_eq!(vec![240, 360, 720], query.available);
```

```
pub fn print_results(&self, result: Vec<i32>)
```

Функція **print_results** приймає посилання self і результат функції **attempt** та друкує результати.

Приклад

```
let query = Value {
   available: vec![240, 360, 720, 1080],
   allowed: vec![240, 360, 720],
   preferred: vec![360, 720],
};

let result = query.attempt();

query.print_results();
```

Trait Implementations

```
impl Debug for Value
```

```
fn fmt(&self, f: &mut Formatter<'_>) -> Result
```

Formats the value using the given formatter. Read more

Auto Trait Implementations

```
impl RefUnwindSafe for Value
```

```
impl Send for Value
```

```
impl Sync for Value
```

```
impl Unpin for Value
```

```
impl UnwindSafe for Value
```

Blanket Implementations

```
impl<T> Any for T
where
    T: 'static + ?Sized,
```

```
impl<T> Borrow<T> for T
where
    T: ?Sized,
```

```
impl<T> BorrowMut<T> for T
where
    T: ?Sized,
```

```
impl<T> From<T> for T
```

```
impl<T, U> Into<U> for T
where
    U: From<T>,
```

```
impl<T, U> TryFrom<U> for T
where
    U: Into<T>,
```

```
impl<T, U> TryInto<U> for T
where
    U: TryFrom<T>,
```