Vec の作り方

```
let v = vec![型; サイズ];
```

inputの仕方

proconio

```
input!{
    a: 型,
    b: [[型; サイズ]; サイズ],
}
```

';' じゃなくて ';' ！！！

example

```
use proconio::{input, marker::*};
input! {
    n: usize,
    m: usize,
    a: [[i32; m]; n],
    s: Chars,
}
```

次のような入力に対応する

```
4 2
3 1
4 1
5 9
2 6
makabe_mizuki_is_very_cute
```

text_io

```
let x: 型 = read!();
```

一行ずつ入力を行う。Java の Scanner のイメージ。

配列の注意

```
let v = vec![3, 1, 4, 1, 5];
let x_0 = v[0_i32];   // error
let x_0 = v[0_usize]; // ok
```

配列の中身の参照はusize型でしか行うことができない。ばか

いろいろなソート

'''rust vec.sort_by(|a, b| a.0.cmp(&b.0));//キーでソートする場合 vec.sort_by(|a, b| a.1.cmp(&b.1));//バリューでソートする場合 fn sortby(mut v:Vec){ v.sort_by(|a, b| a.cmp(&b)); } '''

```
    let mut x = (1..n+1).collect::<Vec<usize>>();

    for x in (0..n).permutations(n){
     //n！回の順列を作成するもの
    }
```