

# listings-rust demo

Tobias Denking  
tobias.denking@tu-dresden.de

January 25, 2019

## Some identifiers

```
1 // keywords
2 as break const continue crate dyn else enum extern fn for if impl in let loop
3 match mod move mut pub ref return Self self static struct super trait type
4 union unsafe use where while
5
6 // reserved keywords (for future use)
7 abstract alignof become box do final macro offsetof override priv proc pure
8 sizeof typeof usize virtual yield
9
10 // strings
11 "this_is_a_test"
12
13 // characters
14 'a' 'b' 'c'
15
16 // primitive types
17 bool char f32 f64 i8 i16 i32 i64 isize str u8 u16 u32 u64 unit usize i128 u128
18
19 // some common type and value constructors
20 Box Option Result String Vec Some None Ok Err true false
21
22 // some common traits
23 Copy Send Sized Sync Drop Fn FnMut FnOnce ToOwned Clone PartialEq PartialOrd
24 Eq Ord AsRef AsMut Into From Default Iterator Extend IntoIterator
25 DoubleEndedIterator ExactSizeIterator SliceConcatExt ToString
26
27 // some common macros
28 assert! assert_eq! assert_ne! cfg! column! compile_error! concat!
29 concat_ids! debug_assert! debug_assert_eq! debug_assert_ne! env! eprint!
30 eprintln! file! format! format_args! include! include_bytes! include_str!
31 line! module_path! option_env! panic! print! println! select! stringify!
32 thread_local! try! unimplemented! unreachable! vec! write! writeln!
```

## Example file

```
1 use std::rc::Rc;
2
3 /// upside-down tree with a designated position (the *stack pointer*)
4 /// and *nodes* of type 'A'.
5 #[derive(Clone, Debug)]
6 pub struct TreeStack<A> {
7     parent: Option<(usize, Rc<TreeStack<A>>>>)>,
8     value: A,
9     children: Vec<Option<Rc<TreeStack<A>>>>>>,
10 }
11
12 impl<A> TreeStack<A> {
13     /// Creates a new 'TreeStack<A>' with root label 'a'.
14     pub fn new(a: A) -> Self {
15         TreeStack { value: a, children: Vec::new(), parent: None }
16     }
17
18     /// Applies a function 'FnMut(&A) -> B' to every node in a 'TreeStack<A>'.
19     pub fn map<F, B>(&self, f: &mut F) -> TreeStack<B>
20     where F: FnMut(&A) -> B,
21     {
22         let new_value = f(&self.value);
23         let new_parent = match self.parent {
24             Some((i, ref p)) => Some((i, Rc::new(p.map(f)))),
25             None => None,
26         };
27         let new_children = self.children
28             .iter()
29             .map(|o| o.clone().map(|v| Rc::new(v.map(f))))
30             .collect();
31
32         TreeStack {
33             parent: new_parent,
34             value: new_value,
35             children: new_children
36         }
37     }
38 }
```