Системне Програмування

З використанням мови програмування **Rust.**
Fundamentals. String. Char. Encodings

© Олексій Рихальський, 2024

Строки

Рядки (англ. **String**) — це базова структура даних у програмуванні, яка використовується для зберігання та обробки текстової інформації. У будь-якій програмі, від найпростішого калькулятора до складних систем управління базами даних, тексти або символи займають центральне місце. Вони можуть представляти імена, повідомлення, URL-адреси, файли конфігурації, текстові поля в програмах і багато іншого.

© Олексій Рихальський, 2024

Що таке String?

У загальному випадку рядок — це послідовність символів, які можуть складатися з букв, цифр, спеціальних символів або навіть пробілів. У різних мовах програмування реалізація рядків може відрізнятися, але основна суть залишається однаковою — це послідовність символів, яку можна маніпулювати, зберігати, копіювати або змінювати.

© Олексій Рихальський, 2024

Два основних типа String

String — це змінний (heap-allocated) тип даних, який дозволяє зберігати та модифікувати текстові дані.

&str або стрічка (slice) — це посилання на текстові дані, що зазвичай використовуються для зберігання незмінних рядків.

Обидва типи мають свої переваги й недоліки залежно від контексту, але String є основним вибором, якщо потрібно працювати з динамічними, змінними текстовими даними.

© Олексій Рихальський, 2024

String. Basics. Створення

```rust
/// creation
let s1 : &str = "Hello!";
let s2 : String = String::from( s: "Hello!");
let mut s3 : String = String::new();

/// conversion
let s4 : String = s1.to_string();
let s5 : &str = s2.as_str();
```


© Олексій Рихальський, 2024

String. Basics. Локація

```rust
/// stack
let data : &str = "initial contents";
/// move to heap
let s1 : String = data.to_string();
let s2 : String = String::from(data);
/// initially in heap
let s3 : String = String::from( s: "initial contents");
```


© Олексій Рихальський, 2024

String. Basics. Конкатенація, довжина

```rust
let mut s3 : String = String::from( s: "Hello!");
let l1 : usize = s3.len();
println!("{}", l1);

s3.push( ch: ',');
s3.push_str( string: " world!");
let l1 : usize = s3.len();

println!("{}", s3); // Hello!, world!
println!("{}", l1);
```


© Олексій Рихальський, 2024

String. Basics. Пошук

```rust
let x: bool = s.contains(pat: "ll");
println!("{}", x); // true
let x: bool = s.contains(pat: "ee");
println!("{}", x); // false

let x: bool = s.is_empty();
println!("{}", x); // false

let x: bool = s.starts_with(pat: "He");
println!("{}", x); // true
let x: bool = s.ends_with(pat: "!?");
println!("{}", x); // false
```


© Олексій Рихальський, 2024

String. Basics. Регістр

```rust
let s : &str = "Привіт";
let upper : String = s.to_uppercase();
let lower : String = s.to_lowercase();
println!("{}", upper); // ПРИВІТ
println!("{}", lower); // привіт
```


© Олексій Рихальський, 2024

String. Basics. Редагування

```rust
/// trim
let raw : &str  =  "     Rust     ";
let trimmed : &str  = raw.trim();
println!("{}", trimmed); // "Rust"

/// replace
let mut edited : String  = trimmed.replace( from: "us",  to: "uuuussss");
println!("{}", edited); // "Ruuuussssst"

/// insert
edited.insert( idx: 5,  ch: '-');
println!("{}", edited); // "Ruuuu-sssst"
```


© Олексій Рихальський, 2024

String. Basics. Slices

```rust
let s : String = String::from( s: "hello world");
//                              012345678910
/// slicing doesn't take a memory
let hello : &str = &s[0..5]; // "hello"

let hello2 : &str = &s[..5]; // "hello"

let hello3 : &str = &s[..=6]; // "hello w"

let world : &str = &s[6..11]; // "world"

let world2 : &str = &s[6..]; // "world"

let whole : &str = &s[..]; // "hello world"
```


© Олексій Рихальський, 2024

Ітерація по символам

```rust
let s5 : String = "Hello".to_string();
for c : char in s5.chars() {
    print!("{} ", c);
}
// H e l l o
```

Ітерація по байтам

```rust
let s5 : String = "Hello".to_string();
for c : u8 in s5.bytes() {
    print!("{} ", c);
}
// 72 101 108 108 111
```


© Олексій Рихальський, 2024

ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

© Олексій Рихальський, 2024

Розбивання на частини

```rust
/// split by index
let word : &str = "programming";
let (l : &str, r : &str) = word.split_at( mid: 7);
assert_eq!("program", l);
assert_eq!("ming", r);
```


© Олексій Рихальський, 2024

Розділення на частини

```
/// split by whitespaces
let sentence : &str = "Раст це цікава мова програмування.";
sentence.split_whitespace() : impl Iterator<Item=&str>
    .for_each(|w : &str | println!("{}", w));
// Раст
// це
// цікава
// мова
// програмування
```


© Олексій Рихальський, 2024

Colors...

```rust
println!("{}", "This is red".red());
println!("{}", "And this is blue".blue());
println!("{}", "This is yellow and striked".yellow().strikethrough());
println!("{}", "This is green with underline".green().underline());
```

```
This is red
And this is blue
This is yellow and striked
This is green with underline
```


© Олексій Рихальський, 2024

Length...

```rust
let s1 : &str = "hello";
println!("{}", s1.len());

let s1 : &str = "Привіт";
println!("{}", s1.len());
```


© Олексій Рихальський, 2024

Length...

```rust
let s1 : &str = "hello";
println!("{}", s1.len()); // 5

let s1 : &str = "Привіт";
println!("{}", s1.len()); // 12
```


© Олексій Рихальський, 2024

Slicing

```rust
let hello : &str = "hello";
dbg!(&hello[0..2]); // "he"
dbg!(&hello[0..4]); // "hell"
```

```rust
let hello : &str = "Привіт";
dbg!(&hello[0..4]); // Пр
```

```rust
let hello : &str = "Привіт";
dbg!(&hello[0..3]); // panic
```


© Олексій Рихальський, 2024

The dark side. Char

```rust
let c1 : char = 'z';
let c2 : char = 'я';
let c3 : char = '你';
let c4 : char = '😀';
```

```rust
let s5 : String = "Hello".to_string();
for c : u8 in s5.bytes() {
    print!("{} ", c);
}
// 72 101 108 108 111
```

```rust
println!("{}", size_of_val(&c1));
println!("{}", size_of_val(&c2));
println!("{}", size_of_val(&c3));
println!("{}", size_of_val(&c4));
```


© Олексій Рихальський, 2024

The dark side. Char

```rust
let c1 : char = 'z';
let c2 : char = 'я';
let c3 : char = '你';
let c4 : char = '😀';
```

```rust
let s5 : String = "Hello".to_string();
for c : u8 in s5.bytes() {
    print!("{} ", c);
}
// 72 101 108 108 111
```

```rust
println!("{}", size_of_val(&c1)); // 4
println!("{}", size_of_val(&c2)); // 4
println!("{}", size_of_val(&c3)); // 4
println!("{}", size_of_val(&c4)); // 4
```


© Олексій Рихальський, 2024

UTF-8

- Char може бути 1 байт
- Char може бути 2 байти
- Char може бути 3 байти
- Char може бути 4 байти

але... в Rust size_of(char) == 4 байти

© Олексій Рихальський, 2024

Ітерація по символам

```
let s5 : String = "Привіт".to_string();
for c : char in s5.chars() {
    print!("{} ", c);
}
// П р и в і т
```

Ітерація по байтам

```
let s5 : String = "Привіт".to_string();
for c : u8 in s5.bytes() {
    print!("{} ", c);
}
// 208 159 209 128 208 184 208 178 209 150 209 130
```


© Олексій Рихальський, 2024

```rust
let hello : String = String::from( s: "Hello");
println!("{}", hello);

hello.chars().for_each(|c : char | print!("{}, ", c));
println!();

hello.bytes().for_each(|b : u8 | print!("{}, ", b));
println!();

hello.bytes().for_each(|b : u8 | print!("{:#02x}, ", b));
println!();
// Hello
// H, e, l, l, o,
// 72, 101, 108, 108, 111,
// 0x48, 0x65, 0x6c, 0x6c, 0x6f,
```


© Олексій Рихальський, 2024

```rust
let hello : String = String::from( s: "Привіт");
let chars: Chars = hello.chars();
let bytes: Bytes = hello.bytes();

dbg!(&hello); // "Hello"
dbg!(&chars); // Chars(['П',      'р',      'и',      'в',      'і',      'т')
dbg!(&bytes); // Bytes([208,159, 209,128, 208,184, 208,178, 209,150, 209,130)

dbg!(hello.len()); // 12
```

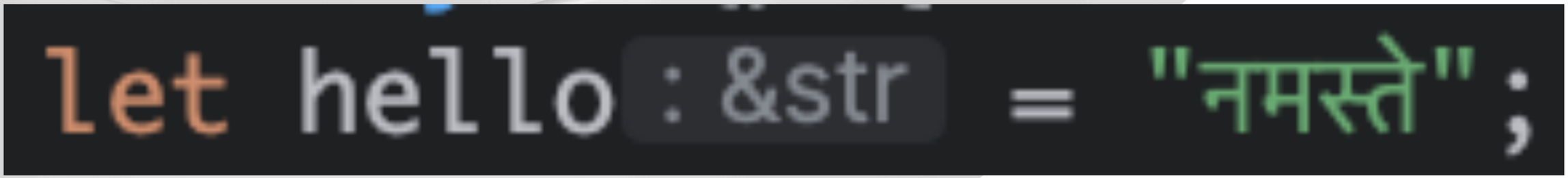

© Олексій Рихальський, 2024

```
let hello : String = String::from( s: "pa");
```


© Олексій Рихальський, 2024

```rust
let hello : String = String::from( s: "pa");
let chars: Chars = hello.chars();
let bytes: Bytes = hello.bytes();
dbg!(&hello); // "pa"
dbg!(chars.count()); // 2
dbg!(bytes.count()); // 3
dbg!(hello.len()); // 3
```


© Олексій Рихальський, 2024



© Олексій Рихальський, 2024

```rust
let hello: &str = "नमस्ते";
// ['न', 'म', 'स', '्', 'त', 'े']
// ["न", "म", "स्",      "ते"     ]

let chars: Chars = hello.chars();
let bytes: Bytes = hello.bytes();

dbg!(hello); // "नमस\u{94d}त\u{947}"
dbg!(&chars); // ['न', 'म', 'स', '\u{94d}', 'त', '\u{947}']
              // ['न', 'म', 'स', '्',       'त', 'े'       ]
```

```
dbg!(&bytes); // [224, 164, 168,  224, 164, 174,  224, 164, 184,
              //   ------------    ------------    ------------
```

```
224, 165, 141,  224, 164, 164,  224, 165, 135]
------------    ------------    ------------
```


© Олексій Рихальський, 2024

```rust
let hello: &str = "😀🤪😐🙄";
println!("{}", hello); // 😀🤪😐🙄

hello.chars().for_each(|c: char| print!("{}, ", c));
println!(); // 😀, 🤪, 😐, 🙄,

hello.bytes().for_each(|b: u8| print!("{}, ", b));
println!();
// 240, 159, 152, 128,  240, 159, 164, 170,
// -------------------  -------------------
// 240, 159, 152, 144,  240, 159, 153, 132,
// -------------------  -------------------
println!("{}", hello.len()) // 16
```


© Олексій Рихальський, 2024

```rust
let s:char = 'a';
println!("{:?}", bytes(s)); // [97]
let s:char = 'Щ';
println!("{:?}", bytes(s)); // [208, 169]
let s:char = '你';
println!("{:?}", bytes(s)); // [228, 189, 160]
let s:char = '💡';
println!("{:?}", bytes(s)); // [240, 159, 146, 161]
```


© Олексій Рихальський, 2024

**Код з лекцій,
презентації Keynote,
PDF-файли
знаходяться на GitHub:**

https://github.com/djnzx/rust-course

© Олексій Рихальський, 2024