

05. string

string1.rs

```
1 fn main()
2 {
3     let s1 = "ABCD";           // &str 타입
4     let s2 = String::from("ABCD"); // String 타입
5
6
7     println!("{}", {}, s2.len(), s2.capacity());
8         // 4, 4
9
10    println!("{}", std::mem::size_of_val(&s2));
11        // 24
12
13
14    println!("{:p}", &s2);
15    println!("{:p}", s2.as_ptr());
16 }
17
```

String

핵심 정리

- Rust 문자열 타입의 종류

String	문자열 자원을 소유. C++ std::string 과 유사
&str	문자열 자원을 소유하지 않음. C++ std::string_view 와 유사

"Reference", "&str" 강좌 참고

- String 객체의 메모리 구조

"ABCD"

Heap

*
len 4
capacity 4
s2

s2 가
"문자열 자원을 소유"

Stack

string2.rs

```
2 {
3     // ❶ string 객체를 생성하는 방법
4     let s1 = String::new();
5     let s2 = String::from("ABCD");
6     let s3 = "ABCD".to_string();
7
8     // ❷ method
9     println!("{}", {}, s1.len(), s2.len()); // 0, 4
10    println!("{}", {}, s1.is_empty(), s2.is_empty()); // true, false
11    println!("{}", {}, s2.starts_with("AB"), s2.ends_with("AB")); // true, false
12
13    // ❸ 문자(열) 추가, 변경
14    let s4 = "ABCD".to_string();
15    let mut s5 = "ABCD".to_string();
16
17    // s4.push('X'); // error
18    s5.push('X');
19    s5.push_str("OPQ");
20
```

```
L3 // ③ 문자(열) 추가, 변경
L4 let s4 = "ABCD".to_string();
L5 let mut s5 = "ABCD".to_string();
L6
L7 // s4.push('X'); // error
L8 s5.push('X');
L9 s5.push_str("OPQ");
L10
L11 println!("{}", s5); // ABCDXOPQ
L12
L13 let s6 = s5.replace("BCD", "-----");
L14 println!("{}", s6); // A-----XOPQ
L15
L16 // ④ 검색
L17 let ret = s5.find("CD"); // Option<T>
L18
L19 println!("{:?}", ret); // Some(2)
L20 }
```

