Study Rust Day1

Output

- Hello world

```
fn main() {
        println!("Hello World!");
}
```

- `println!` is not a funtion. it is a macro

Cargo

- Rust build system
- Package Manager
  using when build code ,download dependency and make library

intructions

- `cargo new`로 새 프로젝트를 생성할 수 있습니다.
- `cargo build` 명령으로 프로젝트를 빌드할 수 있습니다.
- `cargo run` 명령어는 한 번에 프로젝트를 빌드하고 실행할 수 있습니다.
- `cargo check` 명령으로 바이너리를 생성하지 않고 프로젝트의 에러를 체크할 수 있습니다.
- 빌드로 만들어진 파일은 작성한 소스 코드와 뒤섞이지 않도록 *target/debug* 디렉터리에 저장됩니다.

Condition

- 조건식에는 bool 타입의 값만올 수 있음 다음과 같은 식은 잘못됨.

```
fn main(){
        let x = 5;
        if x {
                println!("number is not zero");
        }
}
```

- 이러면 옳은 식이 됨.

```
fn main(){
        let x = 5;
        if x != 0 {
                println!("number is not zero");
        }
}
```

- 아래처럼 상항 연산자처럼 사용이 가능함.

```
fn main(){
        let condition = true;
        let x = if condition { 1 } else { 0 };
}
```

loop statement

`loop` 는 기본적으로 무한 반복함.
`break`을 통해 나갈 수 있음.

value return

```
fn main(){
        let mut cnt = 0;
        let s = loop{
                cnt+=1;
                if cnt == 10{
                        break cnt;
                }
        }
}
```

- `;`는 없어도 됨.

labeling loop

```
fn main(){
        let mut cnt = 0;
        'out_loop: loop{
                let mut rem = 0;
                loop{
                        if rem == 2{
                                break;
                        }
                        if cnt == 2{
                                break 'out_loop;
                        }
                        rem+=1;
                }
                cnt+=1;
        }
        println!("cnt : {cnt}");
}
```

- value return 과 응용

```
fn main(){
    let mut cnt = 0;
    let k = 'out_loop: loop{
        let mut rem = 0;
        loop{
                println!("rem : {rem}");
            if rem == 2{
                break;
            }
            if cnt == 2{
                break 'out_loop cnt*2
            }
            rem+=1;
        }
        cnt+=1;
    };
    println!("cnt : {cnt}\nk : {k}");
}
```

while statement

- 우리가 아는 while 문임.

for statement

- python for문
- count down 예제

```
fn main(){
        for n in (1..4).rev(){ // for n in range(3,0,-1)
                println!("{number}!");
        }
}
```

debugging

`std::fmt`

- `fmt::Debug`의 `{:?}`는 디버깅 목적으로 사용됨.

dead code

- 사용되지 않지만 선언만 된 변수/ 함수들

Debug모드

- `{:?}` 타입 그대로 출력함.
-