

Вау! ИИ готовит к ЕГЭ по информатике Попробовать

Новый конкурс! Нарисуйте <u>комикс</u> или <u>инфографику</u> на любую тему из области информатики. Приём работ до 17 марта.

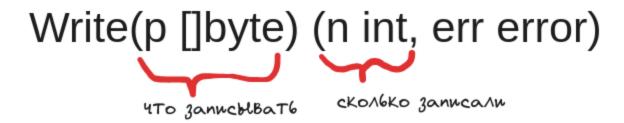
Урок Спринт 2.2

io.Writer

Интерфейс Writer в Go определён следующим образом:

```
type Writer interface {
    Write(p []byte) (n int, err error)
}
```

Write записывает len(p) байт из слайса p в место назначения, возвращая количество записанных байт и ошибку (если возникла). Как и в случае с интерфейсом Reader, важно проверить возвращаемое значение n, чтобы убедиться, что все данные записаны.



Интерфейс Writer также используется различными типами пакета io, такими как File, Buffer и Net.Conn, для записи данных в разные места назначения.

Использование Writer позволяет не зависеть от реализации места назначения записи, будь то локальный файл, удалённое хранилище, либо что-то ещё. Например, если есть функция, которой нужно передать io.Writer для записи:

```
func WriteData(writer io.Writer, data []byte) error{
   bytesWritten, err := writer.Write(data) // Записываем данные
   if err != nil {
        // TODO: handle error.
   }
```

1 of 3

```
fmt.Printf("Записано %d байт", bytesWritten)
}
```

Можно использовать её для записи данных в файл:

```
file, err := os.Create("output.txt") // Создаём файл
if err != nil {
  // TODO: handle error.
}
defer file.Close() // закроем файл после записи
data := []byte("Hello, World!")
if err = WriteData(file, data); err != nil { // запишем данные в файл
   // TODO: handle error.
}
```

Либо реализовать этот интерфейс в своей структуре:

```
type myWriter struct {
   content []byte // сюда будем записывать данные
}
// реализация интерфейса io.Writer
func (w *myWriter) Write(buf []byte) (int, error) {
   w.content = append(w.content, buf...) // запишем данные в слайс
    return len(buf), nil // вернём сколько данные записали
}
// для удобного представления реализуем интерфейс Stringer
func (w *myWriter) String() string {
 return string(w.content)
}
```

И использовать с функцией WriteData из примера выше:

```
w := &myWriter{}
WriteData(w, []byte("привет\n")) // запишем данные в слайс
fmt.Println(w.String()) // "привет\n"
```

Таким образом, io.Reader и io.Writer — это интерфейсы, которые предоставляют общий способ чтения и записи данных из/в различные источники и места назначения. Они обеспечивают гибкие и несвязанные операции ввода-вывода в вашем коде.

Справка

2 of 3 3/16/24, 16:24 Задача «io.Writer» — io.Reader/io.Writer — Программир...

Исключительное право на учебную программу и все сопутствующие ей учебные материалы, доступные в рамках сервиса, принадлежат АНО ДПО «Образовательные технологии Яндекса». Воспроизведение, копирование, распространение и иное использование программы и материалов допустимо только с предварительного письменного согласия АНО ДПО «Образовательные технологии Яндекса». Пользовательское соглашение.

© 2018 - 2024 ООО «Яндекс»

3 of 3