**Бинарный поиск**

package main

import (

"fmt"

)

func binarySearch(arr []int, target int) int {

low := 0

high := len(arr) - 1

for low <= high {

mid := (low + high) / 2

if arr[mid] == target {

return mid

} else if arr[mid] < target {

low = mid + 1

} else {

high = mid - 1

}

}

return -1

}

func main() {

arr := []int{1, 3, 5, 7, 9, 11, 13, 15, 17, 19}

target := 9

index := binarySearch(arr, target)

if index != -1 {

fmt.Printf("Элемент %d найден по индексу %d\n", target, index)

} else {

fmt.Printf("Элемент %d не найден\n", target)

}

}

**Сортировка слиянием**

package main

import (

"fmt"

)

func merge(left []int, right []int) []int {

result := make([]int, 0)

for len(left) > 0 || len(right) > 0 {

if len(left) > 0 && len(right) > 0 {

if left[0] <= right[0] {

result = append(result, left[0])

left = left[1:]

} else {

result = append(result, right[0])

right = right[1:]

}

} else if len(left) > 0 {

result = append(result, left[0])

left = left[1:]

} else if len(right) > 0 {

result = append(result, right[0])

right = right[1:]

}

}

return result

}

func mergeSort(arr []int) []int {

if len(arr) <= 1 {

return arr

}

mid := len(arr) / 2

left := mergeSort(arr[:mid])

right := mergeSort(arr[mid:])

return merge(left, right)

}

func main() {

arr := []int{12, 11, 13, 5, 6, 7}

fmt.Println("Исходный массив:", arr)

arr = mergeSort(arr)

fmt.Println("Отсортированный массив:", arr)

}

**Работа со стеком**

package main

import (

"fmt"

)

type Stack struct {

items []int

}

func (s \*Stack) Push(item int) {

s.items = append(s.items, item)

}

func (s \*Stack) Pop() int {

if len(s.items) == 0 {

panic("Стек пуст")

}

index := len(s.items) - 1

item := s.items[index]

s.items = s.items[:index]

return item

}

func main() {

stack := Stack{}

stack.Push(1)

stack.Push(2)

stack.Push(3)

fmt.Println(stack.Pop())

fmt.Println(stack.Pop())

fmt.Println(stack.Pop())

}