**Липецкий государственный технический университет**  
Факультет автоматизации и информатики  
  
Кафедра Автоматизированных систем управления

Лабораторная работа №1

по курсу «Организация графических систем»

Система наложения фильтров на загруженное изображение

|  |  |
| --- | --- |
| Студенты | Попов А.Д. |
| Группа М-АС-21 |  |
| Руководитель  доцент | Кургасов В.В. |

Липецк 2022 г.

**Цель работы**

Изучить основы синтаксиса языка Swift, работу с базовыми типами данных, управляющими конструкциями, функциями и замыканиями.

**Задание**

Написать консольное приложение, предназначенное для сортировки массива кортежей (Int, Float, String) и поиска данных по массиву. Метод сортировки и поиска приведен в таблице №1. Массив может быть отсортирован по полю Int или по полю Float; поле, по которому осуществляется сортировка и поиск, выбирается пользователем.

Программа должна выполнять следующее:

1. Запрос размера массива у пользователя и заполнение массива случайными данными

2. Выбор поля, по которому выполняется сортировка

3. Сортировка массива по заданному полю заданным по варианту методом

4. Поиск в упорядоченном массиве значений заданным по варианту методом

5. Сортировка и поиск должны выполняться в отдельных функциях

6. Передача информации о выбранном поле для поиска и сортировки должно выполняться с помощью замыкания

**Вариант 13 (3)**

|  |  |  |
| --- | --- | --- |
| № варианта | Метод сортировки | Метод поиска |
| 3 | Метод вставок | Двоичный поиск |

**Листинг программы**

//

//  main.swift

//  Lab1

//

//  Created by Admin on 30.03.2022.

//

import Foundation

// кастомный тип кортежа

typealias Product = (article: Int, price: Float, name: String)

/\*

 \* ввод количества элементов массива

 \*/

func getCount() -> Int {

    print("Введите количетсво элементов массива")

    let count = Int(readLine() ?? "0") ?? 0

    if (count <= 0) {

        print("!Значение должно быть больше 0!")

        return getCount()

    }

    return count

}

/\*

 \* выбор поля для сортировки

 \*/

func getSortField() -> Int {

    print("Введите индекс поля по которому будет осуществляться сортировка и поиск 1: article 2: name 3: price")

    return Int(readLine() ?? "1") ?? 1

}

/\*

 \* генерация массива

 \*/

func generateArr(count: Int) -> [Product] {

    var arr = [Product]()

    for \_ in 1...count {

        let article = Int.random(in: 1...100)

        let name = "Product \(article)"

        let price = Float.random(in: 10.0...1000.0)

        arr.append((article, price, name))

    }

    return arr

}

/\*\*

 \* сортировка массива методов вставок

 \*/

func sortArray(arr: [Product], comparison: (Product, Product) -> Bool) -> [Product]{

    var list = arr

    for (index, element) in list.enumerated() {

        if (index != 0) {

            var location: Int = index - 1

            while (location >= 0 && comparison(list[location], element)) {

                list[location + 1] = list[location]

                list[location] = element

                location = location - 1

            }

        }

    }

    return list

}

/\*

 \* поиск

 \*/

func search(arr: [Product], field: (index: Int, name: String), firstComparison: (Product, Any) -> Bool, secondComparison: (Product, Any) -> Bool) {

    print("Введите \(field.name): ")

    let query = readLine() ?? "0"

    var mid: Int = 0

    var left: Int = 0

    var right: Int = arr.count - 1

    var flag: Bool = false

    while((left <= right) && (!flag)) {

        mid = (left + right) / 2

        if (firstComparison(arr[mid], query)) {

            flag = true

        } else if (secondComparison(arr[mid], query)) {

            right = mid - 1

        } else {

            left = mid + 1

        }

    }

    if (flag) {

        print(arr[mid])

    } else {

        print("not found \(query)")

        preSearch(arr: arr, field: field)

    }

}

func preSearch(arr: [Product], field: (index: Int, name: String)) {

    search(arr: arr, field: field, firstComparison: {

        (element: Product, query: Any) -> Bool in

        switch field.index {

        case 1: return element.article == Int(query as! String) ?? 0

        case 2: return element.name == query as! String

        case 3: return element.price == Float(query as! String) ?? 0.0

        default: return element.article == Int(query as! String) ?? 0

        }

    }, secondComparison: {

        (element: Product, query: Any) -> Bool in

        switch field.index {

        case 1: return element.article > Int(query as! String) ?? 0

        case 2: return element.name > query as! String

        case 3: return element.price > Float(query as! String) ?? 0.0

        default: return element.article > Int(query as! String) ?? 0

        }

    })

}

func main() {

    let count = getCount()

    var arr = generateArr(count: count)

    // 1 - article, 2 - name, 3 - price

    let sortField = getSortField()

    arr = sortArray(arr: arr) {

        switch sortField {

        case 1: return $0.article > $1.article

        case 2: return $0.name > $1.name

        case 3: return $0.price > $1.price

        default: return $0.article > $1.article

        }

    }

    for element in arr {

        print(element)

    }

    var field: String = ""

    switch sortField {

    case 1: field = "article"

    case 2: field = "name"

    case 3: field = "price"

    default: field = "article"

    }

    preSearch(arr: arr, field: (index: sortField, name: field))

}

main()

**Вывод**

В результате выполнения данной лабораторной работы было разработано консольное приложение на языке Swift, которое реализует сортировку вставками и двоичный поиск по массиву.