**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

по дисциплине: «**Разработка приложений для iPhone и iPad»**

на тему: «Коллекции, обобщения»

Выполнил: студент гр. ИТП-22

Расшивалов Н.И.  
 Принял: ст. преподаватель

Семенченя Т.С.

Гомель 2021

**Цель работы:** изучить основы программирования на языке *Swift*

**Вариант 9**

9. Предметная область: Интернет-оператор. Провайдер имеет различные тарифы доступа в Интернет за 1 Мбайт в зависимости от величины абонентской платы. Информационная система провайдера хранит данные о клиентах.

Система должна позволять выполнять следующие задачи:

* ввод тарифов;
* регистрация пользователя;
* ввод данных о потребленном трафике для конкретного пользователя;
* подсчет общей стоимости реализованного трафика;
* поиск клиента, заплатившего наибольшую стоимость за услуги.

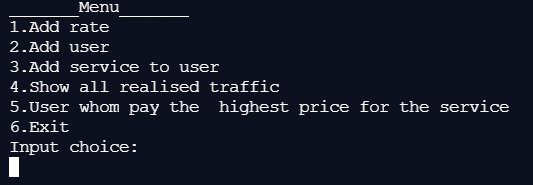


Рисунок 1 – Меню программы

Добавление тарифа представлено на рисунке 2.

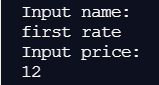


Рисунок 2 – Добавление тарифа

Добавление жильца представлено на рисунке 3.

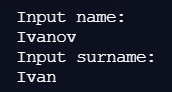


Рисунок 3 – Добавление пользователя

Добавление услуги пользователю представлено на рисунке 4

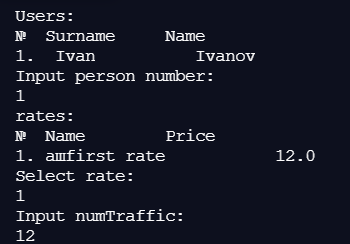


Рисунок 4 – В Добавление услуги пользователю

Вывод реализованного траффика представлен на рисунке 5.



Рисунок 5 – Вывод реализованного траффика

Вывод пользователя заплатившего наибольшую стоимость за услуги представлен рисунке 6.



Рисунок 6 – Вывод пользователя заплатившего наибольшую стоимость за услуги

Листинг программы представлен в Приложении А

**Вывод:** в ходе выполнения лабораторной работы были изучены основы работы с классами в языке *Swift*.

**ПРИЛОЖЕНИЕ А**

**Листинг приложения**

**Main**

import Glibc

var choice:Int = getChoice()

let user = UserService("Service")

while choice != 6

{

switch choice

{

case 1:

user.Rates.append(addRate())

case 2:

user.Users.append(addPerson())

case 3:

addService(user)

case 4:

showAllTraff(user)

case 5:

showAllSum(user)

default:

print("Ошибка выбора")

}

choice = getChoice()

Glibc.system("clear")

}

**menu**

//Возвращает выбор

func getChoice()-> Int

{

print("\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\n1.Add rate\n2.Add user\n3.Add service to user\n4.Show all realised traffic\n5.User whom pay the highest price for the service\n6.Exit\nInput choice: ")

let choice = (readLine())!

var num = Int(choice)

if(num == nil)

{

num = 0

}

return num!

}

//Добавляет тариф

func addRate()-> Rate

{

print("Input name:")

let name = (readLine())!

let price = getDoubleNumber("Input price:")

return Rate(name,price)

}

//Добавляет жителя

func addPerson()-> User

{

print("Input name:")

let name = (readLine())!

print("Input surname:")

let surname = (readLine())!

return User(surname,name)

}

//Добавляет услугу

func addService(\_ user:UserService)

{

var i = 1

print("Users:\n№ Surname Name ")

for user in user.Users

{

print("\(i). \(user.Surname) \(user.Name)")

i += 1

}

let number = getIntNumber("Input person number:")

if number > 0 && number <= user.Users.count

{

var i = 1

print("rates:\n№ Name Price")

for rate in user.Rates

{

print("\(i). \(rate.Name) \(rate.Price)")

i += 1

}

let rateNumber = getIntNumber("Select rate:")

if rateNumber > 0 && rateNumber <= user.Rates.count

{

let numTraffic = getDoubleNumber("Input numTraffic:")

user.Users[number-1].Services.append(Service(user.Rates[rateNumber-1],numTraffic))

}

else

{

print("Incorrect")

}

}

else

{

print("Incorrect")

}

}

//Выводит полную сумму услуг

func showAllSum(\_ user:UserService)

{

print("\_\_\_User\_\_\_\_\_Sum\_\_\_")

var max = 0.0

for user in user.Users

{

if max < user.GetSum()

{

max = user.GetSum()

}

}

for user in user.Users

{

if max == user.GetSum()

{

print("\(user.Surname) \(user.Name) \(user.GetSum()) ---max")

}

else

{

print("\(user.Surname) \(user.Name) \(user.GetSum())")

}

}

if user.Users.count == 0

{

print("No persons")

}

else

{

print("All sum:\(user.GetSum())")

}

}

//Выводит весь трафик

func showAllTraff(\_ user:UserService)

{

print("All traffic: \(user.GetAllTraffic())")

}

//Проверяет является ли числом

func isNumber(\_ string:String) -> Bool

{

let variable = Double(string)

if(variable != nil)

{

return true

}else{

return false

}

}

//Возращает введенное int число

func getIntNumber(\_ message:String) -> Int{

var stringNumber:String = ""

var number: Int = 0

var error:Bool = true

while(error)

{

print(message)

stringNumber = (readLine())!

if(isNumber(stringNumber))

{

number = Int(stringNumber)!

if(number>=0)

{

error = false

}

else

{

print("Incorrect")

}

}

else

{

print("Incorrect")

}

}

return number

}

//Возращает введенное double число

func getDoubleNumber(\_ message:String) -> Double{

var stringNumber:String = ""

var number: Double = 0.0

var error:Bool = true

while(error)

{

print(message)

stringNumber = (readLine())!

if(isNumber(stringNumber))

{

number = Double(stringNumber)!

if(number>0)

{

error = false

}

else

{

print("Incorrect")

}

}

else

{

print("Incorrect")

}

}

return number

}

**UserService class**

public class UserService

{

init(\_ name: String)

{

self.Name = name

}

public var Name:String

public var Users:[User] = []

public var Rates:[Rate] = []

public func GetSum() -> Double

{

var sum = 0.0

for user in Users

{

for service in user.Services

{

sum += service.GetPrice()

}

}

return sum

}

public func GetAllTraffic() -> Double

{

var sum = 0.0

for user in Users

{

for service in user.Services

{

sum += service.GetTraff()

}

}

return sum

}

}

**User class**

public class User

{

init(\_ surname: String, \_ name: String)

{

self.Surname = surname

self.Name = name

}

public var Surname: String

public var Name: String

public var Services: [Service] = []

public func GetSum() -> Double

{

var sum = 0.0

for service in Services

{

sum += service.GetPrice()

}

return sum

}

}

**Rate class**

public class Rate

{

init(\_ name: String,\_ price: Double)

{

self.Name = name

self.Price = price

}

public var Name: String = ""

public var Price: Double = 0.0

}

**Service class**

public class Service

{

init(\_ rate: Rate,\_ numTraffic:Double){

self.Rate = rate

self.NumTraffic = numTraffic

}

public var NumTraffic:Double

public var Rate:Rate

public var Name:String

{

get{

return Rate.Name

}

}

public func GetTraff() -> Double

{

return NumTraffic\*1

}

public func GetPrice() -> Double

{

return NumTraffic\*Rate.Price

}

}