Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Отчет по лабораторной работе №1

По дисциплине «СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ»

Выполнили:

Замотаев Егор

Карагужина Гульсария

Мингазов Артур

студенты группы R3336

Преподаватель:

Капитонов Александр Александрович

Санкт-Петербург

2018

**Цель работы:**

Научится программировать и решать более сложные задачи.

**Ход работы:**

Задача №1.

Реализовать алгоритм фасетного поиска для определенной группы объектов (Жанры). Реализовать алгоритм не менее чем для 16 объектов. Пользователь может отвечать на вопросы только «да» и «нет».

genre = {"nyy":"Шутер", "nyny":"Файтинг", "nynny":"Избей их всех", "nynn":"Слешер", "ynn":"Стратегии в реальном времени", "yyy":"Карточные стратегии", "yyn":"Пошаговые стратегии", "ynny":"Глобальные стратегии","ynnn":"Симуляторы бога", "nnyyn":"Текстовые приключения", "nnyyy":"Графические квесты", "nnyny":"Головоломки", "nnynnny":"Приключенческие боевики", "nnynny":"Симуляторы свиданий", "nnynnnn":"Визуальные ноыеллы","nnn":"Аркада"}

k=[]

k.append(input("В игре во главу угла ставится стратегическое мышление? y/n"))

if k[0]=='y':

k.append(input("Игрок производит действия за ход? y/n"))

if k[1]=='y':

k.append(input("Игра основанна на настольных карточных играх? y/n"))#карточные либо пошаговые стратегии

else:

k.append(input("К основным элементам игры относится: сбор ресурсов, создание боевых едениц, штурм базы противника? y/n"))

if k[2]=='n':

k.append(input("Игрок управляет крупной системой, такой как город, государство и.т.д.? y/n"))#получаем глобальную стратению или симулятор бога

else:

k.append(input("Целью игры является сражение? y/n"))

if k[1]=='y':

k.append(input("Использует ли персонаж огнестрельное оружие во время боя? y/n"))

if k[2]=='n': #получаем шутер или идем дальше

k.append(input("Действие игры происходит на арене? y/n"))

if k[3]=='n':

k.append(input("Игрок сражается в рукопашную? y/n")) #получаем слешер или избей их всех

else:

k.append(input("Игрок продигается по сюжету общаясь с игровыми персонажами и решая логические задачи? y/n"))#в случае отрицательного ответа получаем аркаду

if k[1]=='y':

k.append(input("Игрок использует командную строку для взаимодействия с игрой? y/n"))

if k[2]=='y':

k.append(input("В игре присутсвует графика? y/n"))#получаем в случае положительного ответа графический квест, в случае отрицательного текстовые приключения

else:

k.append(input("В игре основной упор делается на интеллектуальные задачи? y/n"))#при положительном ответе получаем головоломки

if k[3]=='n':

k.append(input("Имеется ли в игре романтический сюжет? y/n"))#при положительном ответе получаем симулятор свиданий

if k[4]=='n':

k.append(input("Игровой процесс основан на рефлексах и реакции игрока, но включает в себя элементы классических квестов? y/n"))#при положительном ответе получаем приключенческий боевик, при отрицательном визуальную новеллу

key=''.join(k)

print("Ответ: ",genre[key])

Задача №2

Пользователь задает любое количество чисел с экрана, разделяя их запятыми. Реализовать алгоритм, который распределяет числа на натуральные, целые, рациональные, вещественные, комплексные, четные, нечетные и простые. Обратите внимание, что цифры могут попадать в несколько категорий.

numbers=input().split(',')

rational=[]

whole=[]

natural=[]

even=[]

odd=[]

prime=[]

a=len(numbers)

for i in range(a):

numbers[i]=complex(numbers[i])

x=numbers[i].real

if numbers[i].imag==0:

rational.append(x)

if x-int(x)==0:

whole.append(x)

if x>0:

natural.append(x)

if x%2==0:

even.append(x)

else:

odd.append(x)

for j in range (2,int(x)+1):

if x%j==0:

break

else:

prime.append(x)

print("комплексные: ", numbers)

print("вещественные: ", rational)

print("рациональные: ", rational)

print("целые: ", whole)

print("натуральные: ", natural)

print("четные: ", even)

print("нечетные: ", odd)

print("простые: ", prime)

Задача №3

Пользователь задает массив натуральных чисел. Реализовать для них алгоритмы сортировки следующими методами: пузырьковый, гномий, блочный, пирамидальный. Проанализировать достоинства и недостатки данных методов.

import math

#bubble sort

def BubbleSort(x):

n = len(x)

i = 0; j = 0; k = 0;

for i in range(0, n-1):

j = i

for k in range(i, n):

if x[k]<x[j]:

j = k

x[i], x[j] = x[j], x[i]

#gnome sort

def GnomeSort(list):

n = len(a)

i = 0

while i < n-1:

if a[i] <= a[i+1]:

i +=1

else:

a[i], a[i+1] = a[i+1], a[i]

if i != 0:

i -= 1

else:

i +=1

#bucket sort

def listmerge(lstlst):

all=[]

for lst in lstlst:

all.extend(lst)

return all

def BucketSort(x):

l=min(x)

m=int((max(x)-min(x))/10)+1

z=[]

for i in range(m):

f=[]

for j in range(len(x)):

if int(x[j])<l+(i+1)\*10 and int(x[j])>=l+i\*10:

f.append(x[j])

z.append(f)

for bucket in range(len(z)):

BubbleSort(z[bucket])

z=listmerge(z)

for num in range(len(x)):

x[num]=z[num]

def hashing(a):

m = a[0]

for i in range(1, len(a)):

if (m < a[i]):

m = a[i]

result = [m, int(math.sqrt(len(a)))]

return result

def re\_hashing(i, code):

return int(i / code[0] \* (code[1] - 1))

#heatsort

# To heapify subtree rooted at index i.

def heapify(arr, n, i):

largest = i # обозначить наибольшее значение как корень

l = 2 \* i + 1

r = 2 \* i + 2

# Проверить, есть ли левое поддерево и больше ли оно, чем корень

if l < n and arr[i] < arr[l]:

largest = l

# Проверить, есть ли правое поддерево и больше ли оно, чем корень

if r < n and arr[largest] < arr[r]:

largest = r

# Поменять корень, если значение largest изменились

if largest != i:

arr[i],arr[largest] = arr[largest],arr[i]

heapify(arr, n, largest)

def HeapSort(arr):

n = len(arr)

for i in range(n, -1, -1):

heapify(arr, n, i)

# Уменьшаем количество элементов по одному

for i in range(n-1, 0, -1):

arr[i], arr[0] = arr[0], arr[i]

heapify(arr, i, 0)

a = []

for i in range(int(input())):

a.append(int(input()))

func=input("Выберите вид сортировки: BubbleSort, GnomeSort, BucketSort, HeapSort ")

if func=="BubbleSort":

BubbleSort(a)

if func=="GnomeSort":

GnomeSort(a)

if func=="BucketSort"

BucketSort(a)

if func=="HeapSort"

HeapSort(a)

print(a)

Задача №4

С **A** третьекурсниками часто происходит событие **B**. Зная вероятность, что данное событие происходит с ними **N** раз в **C** дней (**N** вводится пользователем при запуске программы), определите вероятность того, что за **D** дней данная ситуация произойдет только с третьекурсником **E**. Реализовать алгоритм для решения данной задачи (см. Таблица 2 и Теорему Байеса).

N=int(input("Введите N "))

C=12

D=100

P=(N/C\*1/3)\*\*D

print('Вероятность равна: ',P)