**ALGORITHM AND DATA STRUCTURE PRACTICUM**

**MODULE 4**

**SEARCH**



**CREATED BY :**

**KURNIAWAN BAGASKARA**

**L200214253**

**INFORMATICS STUDY PROGRAM**

**FACULTY OF COMMUNICATION AND INFORMATION SCIENCE**

**MUHAMMADIYAH SURAKARTA UNIVERSITY**

**Task.**

class Mahasiswa(object):

"""Mahasiswa yang dibangun dari class Manusia."""

def \_\_init\_\_(self, nama, NIM, kota, us):

"""Metode inisiasi ini menutupi metode inisiasi di class Manusia"""

self.nama = nama

self.NIM = NIM

self.kotaTinggal = kota

self.uangSaku = us

c0 = Mahasiswa('Ika',10,'Sukoharjo',240000)

c1 = Mahasiswa('Budi',51,'Sragen',230000)

c2 = Mahasiswa('Ahmad',2,'Surakarta',250000)

c3 = Mahasiswa('Chandra',18,'Surakarta',235000)

c4 = Mahasiswa('Eka',4,'Boyolali',240000)

c5 = Mahasiswa('Fandi',31,'Salatiga',250000)

c6 = Mahasiswa('Deni',13,'Klaten',245000)

c7 = Mahasiswa('Galuh',5,'Wonogiri',245000)

c8 = Mahasiswa('Janto',23,'Klaten',245000)

c9 = Mahasiswa('Hasan',64,'Karanganyar',270000)

c10 = Mahasiswa('Khalid',29,'Purwodadi',265000)

Daftar = [c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10]

target = 'Klaten'

def cari(target, Daftar):

o = []

d = 0

for i in range (len(Daftar)):

if Daftar[i].kotaTinggal == target:

o.append(d)

d += 1

else:

d += 1

return o

print ("\n-----------------------------NO. 1-----------------------------")

print(cari(target, Daftar))

##-----------------------------NO. 2-----------------------------

def cariUangSakuTerkecil(kumpulan):

terkecil = kumpulan[0].uangSaku

for i in kumpulan:

if i.uangSaku < terkecil:

terkecil = i.uangSaku

return terkecil #kembalikan yang terkecil

print ("\n-----------------------------NO. 2-----------------------------")

print(cariUangSakuTerkecil(Daftar))

##-----------------------------NO. 3-----------------------------

def cariyangTerkecil(kumpulan):

n = []

terkecil = kumpulan[0].uangSaku

for i in kumpulan:

if i.uangSaku < terkecil:

terkecil = i.uangSaku

n.append(kumpulan.index(i))

return n

print ("\n-----------------------------NO. 3-----------------------------")

print(cariyangTerkecil(Daftar))

##-----------------------------NO. 4-----------------------------

def cariKurangDari(kumpulan):

b = []

for i in kumpulan:

if i.uangSaku < 250000:

terkecil = i.uangSaku

b.append(kumpulan.index(i))

return b

print ("\n-----------------------------NO. 4-----------------------------")

print(cariKurangDari(Daftar))

##-----------------------------NO. 5-----------------------------

class node (object):

def \_\_init\_\_(self, data, next = None):

self.data = data

self.next = next

def cari (self, cari):

curNode = self

while curNode is not None :

if curNode.next != None :

if curNode.data != cari :

curNode = curNode.next

else:

print ("Item", cari, "ada dalam Linked List")

break

elif curNode.next == None :

print ("Item", cari, "tidak ada Linked list")

break

a = node (12)

menu = a

a.next = node (34)

a = a.next

a.next = node (10)

a = a.next

a.next = node (45)

print ("\n-----------------------------NO. 5-----------------------------")

menu.cari(10)

menu.cari(110)

##-----------------------------NO. 6-----------------------------

def binSe(kumpulan, target):

low = 0

high = len(kumpulan) -1

data = []

#Secara berulang belah runtutan itu menjadi separuhnya

#sampai targetnya ditemukan

while low <= high:

#Temukan pertengahan runtut itu

mid = (high + low) //2

#Apakah pertengahannya memuat target?

if kumpulan[mid] == target:

data.append(kumpulan.index(target))

return True

#ataukah targetnya di sebelah kirinya?

elif target < kumpulan[mid]:

high = mid -1

#ataukah targetnya di sebelah kanannya?

else :

low = mid +1

#Jika runtutnya tidak bisa dibelah lagi, berarti targetnya tidak ada

return False

a = [2,3,5,6,8,9,10,11,12,13,14]

index\_a = 12

index\_b = 17

print ("\n-----------------------------NO. 6-----------------------------")

print ("Index :", index\_a)

print (binSe(a, index\_a))

print ("\nIndex :", index\_b)

print (binSe(a, index\_b))

##-----------------------------NO. 7-----------------------------

def binSearch(kumpulan, target):

#Mulai dari seluruh runtutan elemen

low = 0

high = len(kumpulan) -1

data = []

#Secara berulang belah runtutan itu menjadi separuhnya

#sampai targetnya ditemukan

while low != high:

#Temukan pertengahan runtut itu

mid = (high + low) //2

#Apakah pertengahannya memuat target?

if kumpulan[mid] == target:

break

#ataukah targetnya di sebelah kirinya?

elif target < kumpulan[mid]:

high = mid -1

#ataukah targetnya di sebelah kanannya?

else :

low = mid +1

for i in range (low, high):

if target == kumpulan[i]:

data.append(i)

return data

List = [2, 3, 5, 6, 6, 6, 8, 9, 9, 10, 11, 12, 13, 13, 14]

elemen = 6

print ("\n-----------------------------NO. 7-----------------------------")

print ("Indeks lokasi elemen ", elemen, "yang terdapat pada list ", List, "yaitu ")

print (binSearch(List, elemen))

##-----------------------------NO. 8-----------------------------

print("""\n-----------------------------NO. 8-----------------------------

\nUntuk membuat permainan tebak angka, kalau angka yang ditebak di antara 1 dan 100 maksimal jumlah tebakan adalah 7.

\nKalau antara 1 dan 1000 maksimal jumlah tebakan adalah 10.

\nHal ini terjadi karena jumlah tebakan nya bila dipangkatkan 2 tidak boleh lebih dari 100 atau 1000.

\nPola yang digunakan pada tebakan adalah 2^n.

""")

print ('Kurniawan Bagaskara')

print('L200214253')

Output :



