



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»
КАФЕДРА ИУК5 «Системы обработки информации»

Лабораторная работа №1

«АЛГОРИТМЫ. СЛОЖНОСТЬ АЛГОРИТМОВ»

ДИСЦИПЛИНА: «Вычислительные алгоритмы»

Выполнил: студент гр. ИУК5-41Б

(Подпись)

(Иванов Н.В.)
(Ф.И.О.)

Проверил:

(Подпись)

(Верховский Е.В.)
(Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

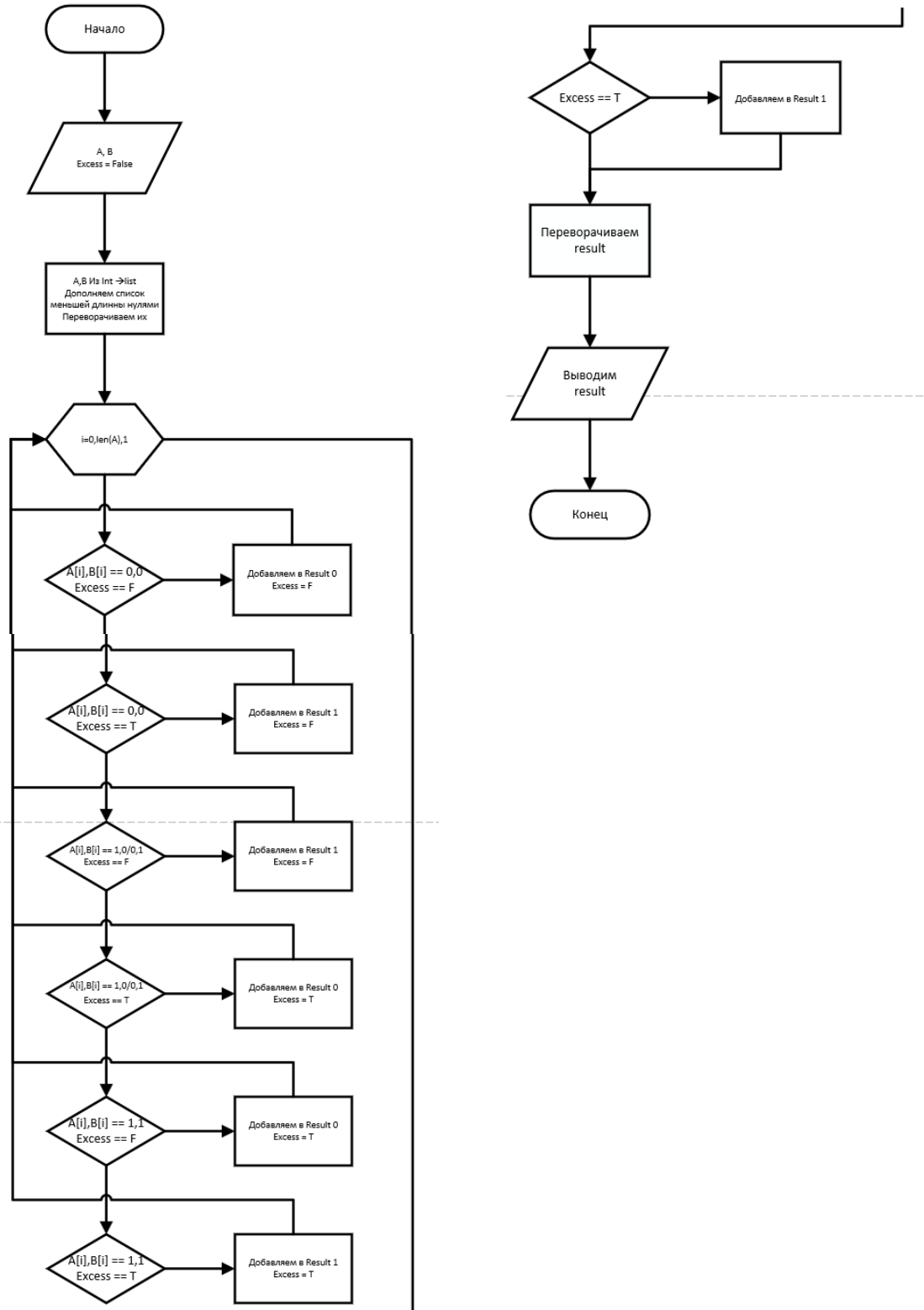
Калуга, 2024

Цель: выработать навыки оценки вычислительной сложности алгоритма.

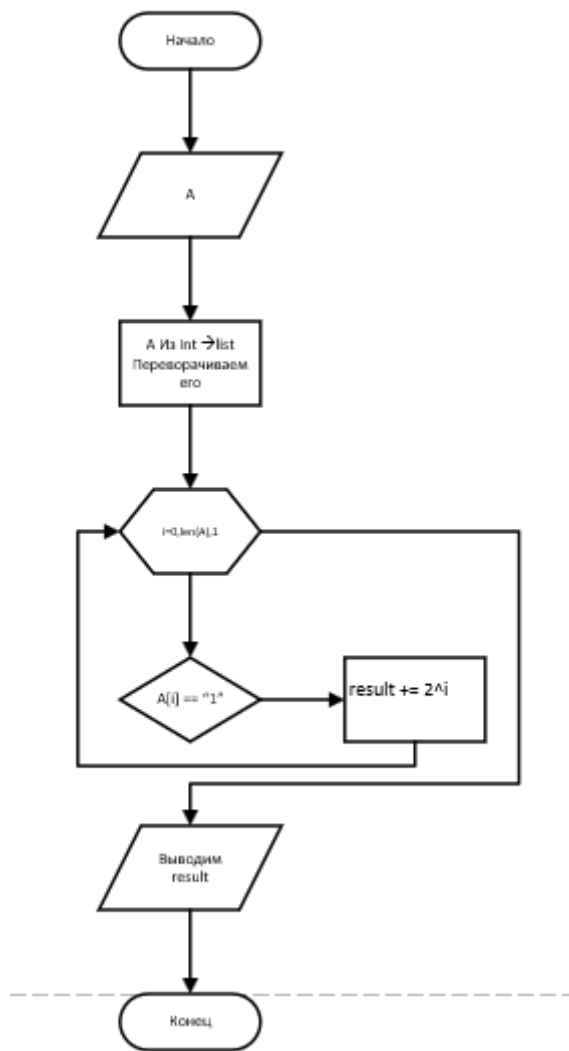
Вариант 9

Дано число в двоичной системе. Определите это число в десятичной системе. Составьте программу, которая получает два целых числа, записанных в двоичной системе, складывает их и результат показывает также в двоичной системе.

Блок-схема ко второй части задачи



Блок-схема ко второй части задачи



Код

```
import time

def to_dec(a):
    a = list(str(a))[::-1]
    result = 0
    for i in range(len(a)):
        if a[i] == "1":
            result += 2 ** (i)
    print("Результат перевода первого числа в десятичную систему счисления:",
          result)

def sum_bin(a, b):
    a = list(str(a))
    b = list(str(b))
    excess = False
    result = []
    if len(a) > len(b):
        b = (["0" for i in range(len(a) - len(b))]) + b[::-1]
        a = a[::-1]
    else:
        a = (["0" for i in range(len(b) - len(a))]) + a[::-1]
```

```

        b = b[::-1]

    for i in range(len(a)):
        if a[i] == '0' and b[i] == '0' and not (excess):
            result.append("0")
            excess = False
        elif a[i] == '0' and b[i] == '0' and excess:
            result.append("1")
            excess = False
        elif ((a[i] == '1' and b[i] == '0') or (a[i] == '0' and b[i] == '1'))
and not (excess):
            result.append("1")
            excess = False
        elif ((a[i] == '1' and b[i] == '0') or (a[i] == '0' and b[i] == '1'))
and excess:
            result.append("0")
            excess = True
        elif a[i] == '1' and b[i] == '1' and not (excess):
            result.append("0")
            excess = True
        elif a[i] == '1' and b[i] == '1' and excess:
            result.append("1")
            excess = True
    if excess:
        result.append("1")

    print("Ответ:", "".join(result[::-1]))

a = int(input())
b = int(input())
start_time = time.perf_counter_ns()
sum_bin(a, b)
elapsed_time = time.perf_counter_ns() - start_time
print("Время работы программы:", elapsed_time / (10 ** 9))

```

Входные данные	Время работы, с.
10 1	0.0001066
111 11111	8.85e-05
1010101010 11111	8.32e-05
1000 111	3.86e-05
101010101 1111001	7.98e-05
100110011 11	0.0001117
1111 1111	6.84e-05
1000 1000	4.88e-05
11 1	0.0001071
1111110001 101010	8.53e-05

Сложность алгоритма $O(n)$.

Вывод: в процессе работы я выработал навыки оценки вычислительной сложности алгоритма.



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК-КФ «Информатика и управление»
КАФЕДРА ИУК5-КФ «Системы обработки информации»

Лабораторная работа №2

«Одномерные и двумерные массивы»

ДИСЦИПЛИНА: «Вычислительные алгоритмы»

Выполнил: студент гр. ИУК5-41Б

Иванов Н.В.
(Подпись)

(Иванов Н.В.)
(Ф.И.О.)

Проверил:

Варшавский Е.В.
(Подпись)

(Варшавский Е.В.)
(Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

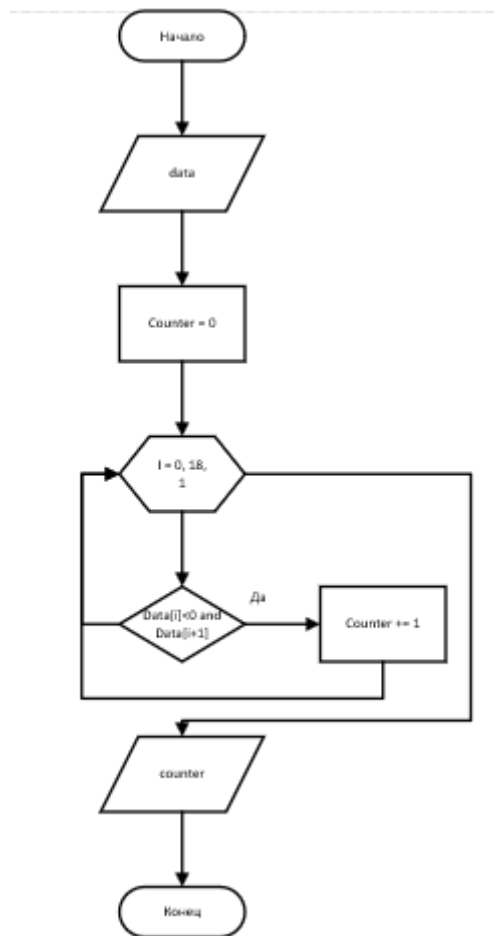
- Балльная оценка:
- Оценка:

Калуга, 2024

Цель: выработать навыки реализации типовых алгоритмов обработки одномерных и двумерных массивов.

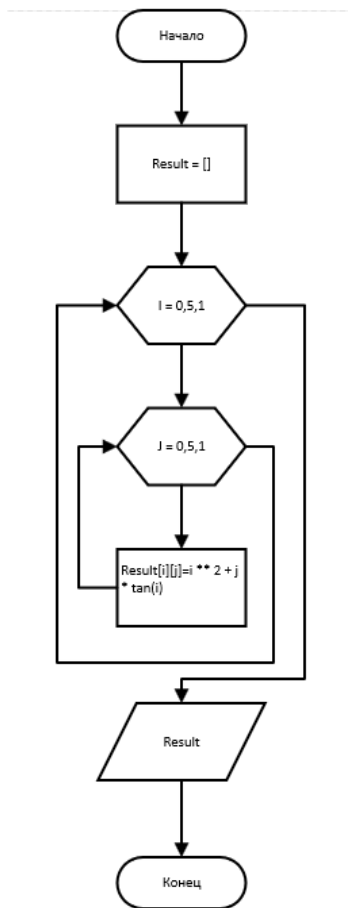
№1

```
def func1():  
    data = [1, -2, -8, -6, -3, -7, -4, 10, 9, -3, 8, 7, -1, 7, 4, -5, 7, -4,  
            -7, -2]  
    counter = 0  
    for i in range(19):  
        if data[i] < 0 and data[i + 1] < 0:  
            counter += 1  
  
    print(counter)
```



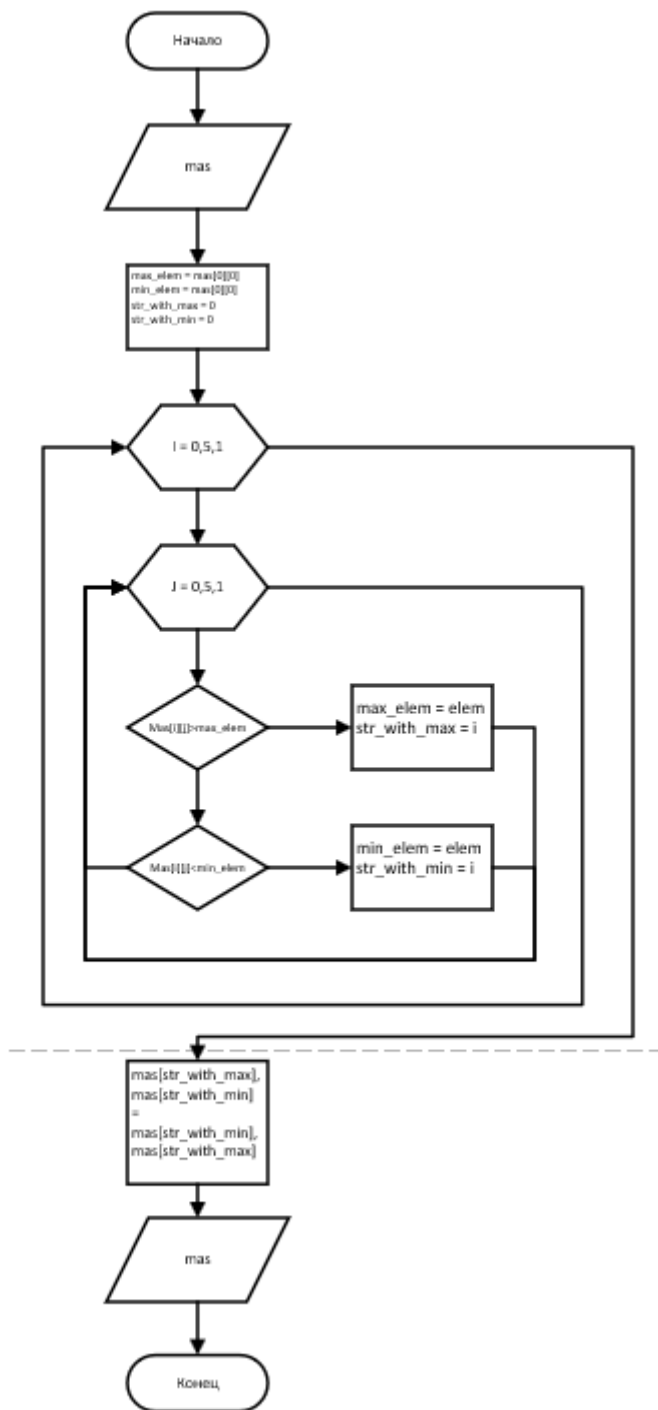
№2

```
def show(mas):  
    for i in range(6):  
        print(mas[i])  
    print()  
  
def funcA():  
    result = []  
    for i in range(6):  
        result.append([])  
        for j in range(6):  
            elem = round(i ** 2 + j * math.tan(i), 1)  
            result[i].append(elem)  
    show(result)  
    return result
```



```

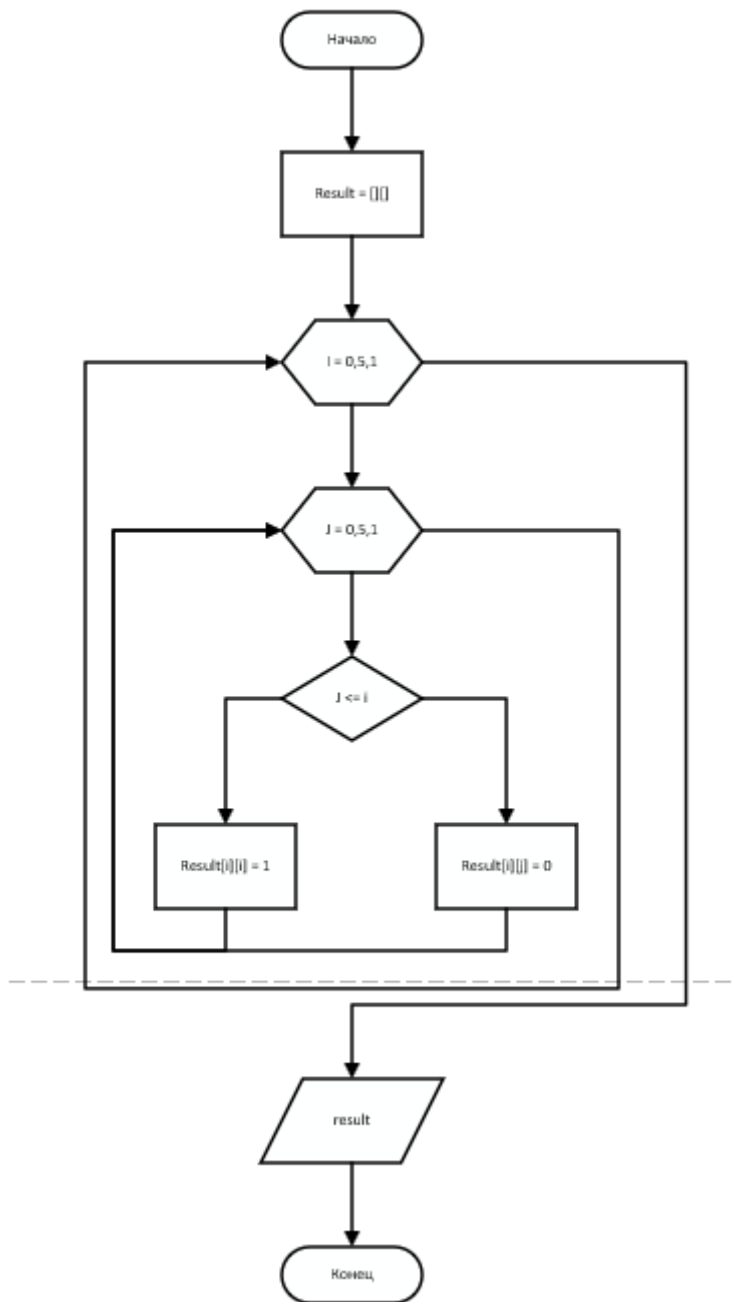
def funcB(mas):
    max_elem = mas[0][0]
    min_elem = mas[0][0]
    str_with_max = 0
    str_with_min = 0
    for i in range(6):
        for j in range(6):
            elem = mas[i][j]
            if elem > max_elem:
                max_elem = elem
                str_with_max = i
            if elem < min_elem:
                min_elem = elem
                str_with_min = i
    print(str_with_max, str_with_min)
    mas[str_with_max], mas[str_with_min] = mas[str_with_min],
mas[str_with_max]
    show(mas)
    return mas
  
```

```

def funcC():
    result = []
    for i in range(6):
        result.append([])
        for j in range(6):
            if j <= i:
                result[i].append(0)
            else:
                result[i].append(1)
    show(result)
    return result

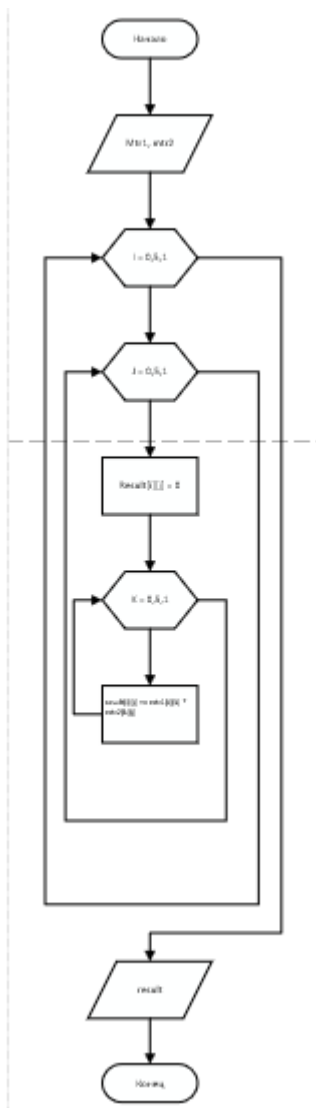
```



```

def multiply(mtr1, mtr2):
    result = []
    for i in range(6):
        result.append([])
        for j in range(6):
            result[i].append(0)
            for k in range(6):
                result[i][j] += mtr1[i][k] * mtr2[k][j]
            result[i][j] = round(result[i][j], 1)
    show(result)
    return result

```



```

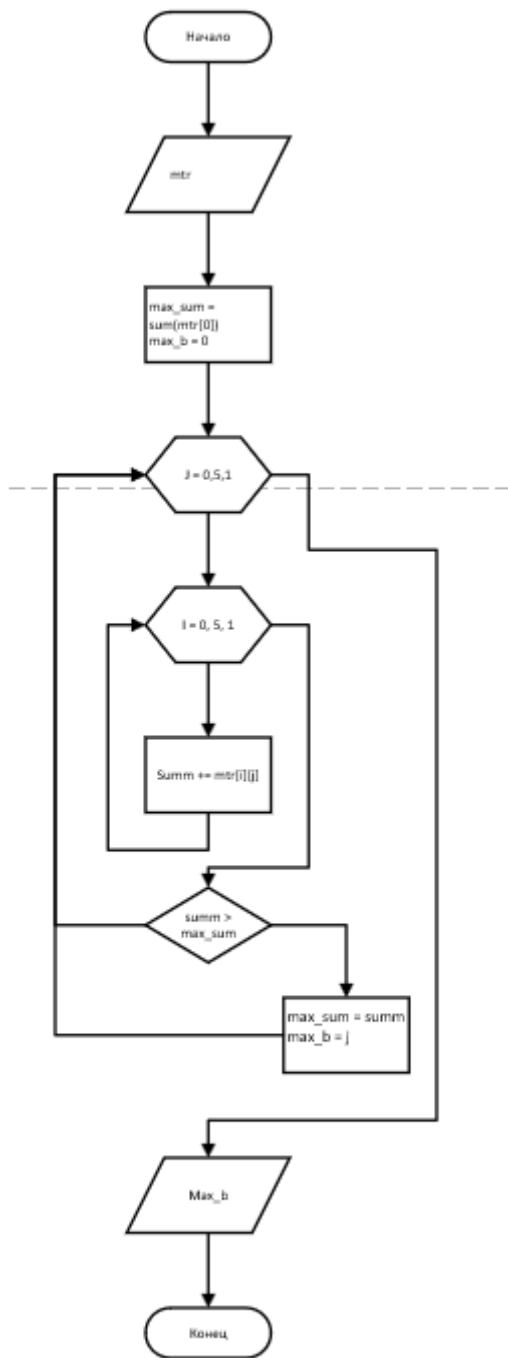
def swap_b(mtr):
    mtr[0][2], mtr[0][3], mtr[1][2], mtr[1][3], mtr[2][0], mtr[2][1],
mtr[3][0], mtr[3][1] \
    = mtr[2][0], mtr[2][1], mtr[3][0], mtr[3][1], mtr[0][2], mtr[0][3],
mtr[1][2], mtr[1][3]
    show(mtr)
    return mtr

```

```

def max_sum(mtr):
    max_sum = sum(mtr[0])
    max_b = 0
    for j in range(6):
        summ = sum([mtr[i][j] for i in range(6)])
        print(summ)
        if summ > max_sum:
            max_sum = summ
            max_b = j
    print(max_b + 1)
    return max_b + 1

```



Вывод: в процессе работы выработаны навыки реализации типовых алгоритмов обработки одномерных и двумерных массивов.



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК-КФ «Информатика и управление»
КАФЕДРА ИУК5-КФ «Системы обработки информации»

Лабораторная работа №3

«Связный список, стек, очередь»

ДИСЦИПЛИНА: «Вычислительные алгоритмы»

Выполнил: студент гр. ИУК5-41Б

(Подпись)

(Иванов Н.В.)
(Ф.И.О.)

Проверил:

(Подпись)

(Возниukhina E.B.)
(Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2024

Цель: получение практических навыков при работе со связными списками, стеками, очередями.

№1 9. Создайте двусвязный список групп факультета. Каждая группа представляет собой односвязный список студентов.

```
from pprint import pprint
import time

class El:
    def __init__(self, value, next=None):
        self.value = value
        self.next = next

    def __str__(self):
        return str(self.value)

class Fir_list:
    def __init__(self):
        self.head = None
        self.tail = None

    def append(self, value):
        elem = El(value)
        if None == self.head:
            self.head = elem
            self.tail = elem
        else:
            self.tail.next = elem
            self.tail = elem

    def __str__(self):
        res = ""
        elem = self.head
        res += str(elem)
        res += " "
        while None != elem.next:
            elem = elem.next
            res += str(elem)
            res += " "

        return res

class Elem:
    def __init__(self, value, next=None, previous=None):
        self.value = value
        self.next = next
        self.previous = previous

    def __str__(self):
        return Fir_list.__str__(self.value)

class Sec_list:
    def __init__(self):
        self.head = None
        self.tail = None

    def append(self, value):
        elem = Elem(value)
```

```

        if None == self.head:
            self.head = elem
            self.tail = elem
        else:
            self.tail.next = elem
            elem.previous = self.tail
            self.tail = elem

    def show(self):
        elem = self.head
        print(elem)
        while None != elem.next:
            elem = elem.next
            print(elem)

start1 = time.perf_counter()
l2 = Fir_list()
l2.append('Сидоров')
l2.append('Иванов')
l2.append('Иванов')
l2.append('Иванов')

l3 = Fir_list()
l3.append('Иванов')
l3.append('Сидоров')
l3.append('Иванов')
l3.append('Иванов')

l4 = Fir_list()
l4.append('Иванов')
l4.append('Иванов')
l4.append('Сидоров')
l4.append('Иванов')

l = Sec_list()
l.append(l2)
l.append(l3)
l.append(l4)
l.show()
finish1 = time.perf_counter()
print(round((finish1 - start1) * 1000, 6))

start1 = time.perf_counter()
l22 = []
l22.append('Сидоров')
l22.append('Иванов')
l22.append('Иванов')
l22.append('Иванов')

l33 = []
l33.append('Иванов')
l33.append('Сидоров')
l33.append('Иванов')
l33.append('Иванов')

l44 = []
l44.append('Иванов')
l44.append('Иванов')
l44.append('Сидоров')

```

```

l44.append('Иванов')

l1 = []
l1.append(l22)
l1.append(l33)
l1.append(l44)

pprint(l1)
finish1 = time.perf_counter()
print(round((finish1 - start1) * 1000, 6))

```

№2 9. Карту, определяющую прямоугольную область моря, представили матрицей с логическими элементами (false – море, true - суша). Островом будем называть совокупность соприкасающихся (вертикальной или горизонтальной стороной) клеток матрицы со значениями true. Рассчитайте число островов на матрице-карте.

```

class Stack_elem():
    def __init__(self, value, next=None):
        self.value = value
        self.next = next

    def __str__(self):
        return str(self.value)

class Stack():
    def __init__(self):
        self.head = None

    def push(self, value):
        elem = Stack_elem(value)
        if None == self.head:
            self.head = elem
        else:
            elem.next = self.head
            self.head = elem

    def pop(self):
        elem = self.head
        self.head = self.head.next
        return elem

    def __str__(self):
        res = " "
        elem = self.head
        res += str(elem)
        res += " "
        while None != elem.next:
            elem = elem.next
            res += str(elem)
            res += " "
        res += '\n'
        return res

    def len(self):
        l = 0
        if self.head is None:
            return 0
        elem = self.head
        l += 1
        while None is not elem.next:
            elem = elem.next
            l += 1

```



```

        return l

    def __getitem__(self, item):
        l = self.len()
        elem = self.head
        if item == l:
            return elem
        else:
            i = 0
            j = l - item - 1
            while i != j:
                i += 1
                elem = elem.next
            return elem

    def extend(self, s):
        # print(s.len())
        if s.len() != 0:
            elem = s.head

            self.push(elem.value)
            while None != elem.next:
                elem = elem.next
                self.push(elem.value)

    def remove(self, el):
        elem = self.head
        # print(type(elem))
        if elem is None:
            return l
        if elem.value[0].value == el.value[0].value and elem.value[1].value
        == el.value[1].value:
            self.head = self.head.next
        else:
            elem1 = elem
            while None != elem.next:
                elem = elem.next
                if elem.value[0].value == el.value[0].value and
                elem.value[1].value == el.value[1].value:
                    elem1.next = elem.next
            return l
            elem1 = elem

    def __contains__(self, item):
        elem = self.head
        if elem.value == item:
            return True
        else:
            while None != elem.next:
                elem = elem.next
                if elem.value == item:
                    return True
            return False

    def ncontains(self, el):
        for i in range(self.len()):
            if self[i].value[0].value == el[0].value and
            self[i].value[1].value == el[1].value:
                return False
        return True

s1 = Stack()
s1.push(1)

```

```
s1.push(0)
s1.push(0)
s1.push(1)
s1.push(0)

s2 = Stack()
s2.push(0)
s2.push(0)
s2.push(1)
s2.push(1)
s2.push(0)

s3 = Stack()
s3.push(1)
s3.push(1)
s3.push(0)
s3.push(0)
s3.push(1)

s4 = Stack()
s4.push(0)
s4.push(0)
s4.push(0)
s4.push(1)
s4.push(0)

s5 = Stack()
s5.push(0)
s5.push(1)
s5.push(0)
s5.push(1)
s5.push(0)

# s1 = Stack()
# s1.push(0)
# s1.push(0)
# s1.push(0)
# s1.push(0)
# s1.push(0)
#
# s2 = Stack()
# s2.push(0)
# s2.push(0)
# s2.push(0)
# s2.push(0)
# s2.push(0)
#
# s3 = Stack()
# s3.push(0)
# s3.push(0)
# s3.push(0)
# s3.push(0)
# s3.push(0)
#
# s4 = Stack()
# s4.push(0)
# s4.push(0)
# s4.push(0)
# s4.push(0)
# s4.push(0)
#
# s5 = Stack()
# s5.push(0)
# s5.push(0)
```

```

# s5.push(0)
# s5.push(0)
# s5.push(0)

s = Stack()
s.push(s1)
s.push(s2)
s.push(s3)
s.push(s4)
s.push(s5)

print(s)

# print(s4[3])
#
#
# s3.extend(s3)
# print(s3)
#
# s4.remove(1)
# print(s4)
#
# s5 = Stack()
#
# s6 = Stack()
# s6.push(1)
# s6.push(0)
#
# s8 = Stack()
# s8.push(1)
# s8.push(1)
#
# s5.push(s6)
# s5.push(s8)
# print(s5)
#
# s7 = Stack()
# s7.push(1)
# s7.push(0)
#
# s5.remove(s7)
# print(s5)

def crest2(mas, n, m, mas2):
    # print(type(mas.value[1].value))
    i, j = mas.value[1].value, mas.value[0].value

    res = Stack()
    s1 = Stack()
    s1.push(j)
    s1.push(i - 1)

    s2 = Stack()
    s2.push(j)
    s2.push(i + 1)

    s3 = Stack()
    s3.push(j - 1)
    s3.push(i)

    s4 = Stack()
    s4.push(j + 1)
    s4.push(i)

```

```

        if i != 0 and mas2.ncontains(s1):
            res.push(s1)
        if i != (n - 1) and mas2.ncontains(s2):
            res.push(s2)
        if j != 0 and mas2.ncontains(s3):
            res.push(s3)
        if j != (m - 1) and mas2.ncontains(s4):
            res.push(s4)

    return res

def islands2(map):
    counter = 0
    n = map.len()
    m = map[0].value.len()
    # print(n, m)
    elems = Stack()
    map_ = Stack()
    for i in range(n):
        for j in range(m):
            e = Stack()
            e.push(j)
            e.push(i)
            elems.push(e)
    # print(elems)
    while elems.len() != 0:
        # maps = []
        el = elems.pop()
        i, j = el.value[1].value, el.value[0].value
        if map[i].value[j].value == 1:
            map_.push(el.value)
            # print(type(map_))
            maps = crest2(el, n, m, map_)
            # print(map_)
            while maps.len() != 0:
                # print(maps)
                el = maps.pop()
                # print(type(el.value))
                # print(el.value)
                # print(type(el.value[1]))
                elems.remove(el)
                i, j = el.value[1].value, el.value[0].value
                if map[i].value[j].value == 1:
                    map_.push(el.value)
                    maps.extend(crest2(el, n, m, map_))
                else:
                    continue
            counter += 1
            # print(counter)
        else:
            continue
    print(" Количество островов = ", counter)

islands2(s)

```

№3 9. Дан текстовый файл. За один просмотр файла напечатать элементы файла в следующем порядке: сначала все слова, начинающиеся с прописной буквы, потом все слова, начинающиеся со строчной буквы, сохраняя исходный порядок в каждой группе слов.

```

import string

punc = string.punctuation + " "

class Elem:
    def __init__(self, value, next=None, previous=None):
        self.value = value
        self.next = next

    def __str__(self):
        return self.value

class Queue:
    def __init__(self):
        self.head = None
        self.tail = None

    def append(self, value):
        elem = Elem(value)
        if None == self.head:
            self.head = elem
            self.tail = elem
        else:
            self.tail.next = elem
            self.tail = elem

    def __str__(self):
        res = ""
        elem = self.head
        res += str(elem.value)
        res += " "
        while None != elem.next:
            elem = elem.next
            res += str(elem.value)
            res += " "
        return res

with open('t.txt', 'r') as f:
    res1 = Queue()
    res2 = Queue()
    s = f.readline()
    while len(s) != 0:
        word = ""
        for w in s:
            if w not in punc or w == '\n':
                word += w
            else:
                if word != "":
                    if word[0].isupper():
                        res1.append(word)
                    else:
                        res2.append(word)
                word = ""
        s = f.readline()
    if word != "":
        if word[0].isupper():
            res1.append(word)
        else:
            res2.append(word)
    print(res1)
    print(res2)

```

Вывод: в процессе работы были получены практические навыки при работе со связными списками, стеками, очередями



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК-КФ «Информатика и управление»
КАФЕДРА ИУК5-КФ «Системы обработки информации»

Лабораторная работа №4

«Алгоритмы сортировки»

ДИСЦИПЛИНА: «Вычислительные алгоритмы»

Выполнил: студент гр. ИУК5-41Б

(Подпись)

(Иванов Н.В.)
(Ф.И.О.)

Проверил:

(Подпись)

(Верbitsкий Е.В.)
(Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2024

Цель: формирование практических навыков разработки алгоритма сортировки

9. Написать программу, сортирующую методом слияния главную диагональ двумерного массива целых чисел. Массив считать из текстового файла, содержащего в первой строке кол-во строк и кол-во столбцов, а далее в каждой строке целые числа, соответствующие элементам строки массива.

```
from pprint import pprint

with open("t4.1.txt") as f:
    nm = list(map(int, list(f.readline().split())))
    n, m = nm[0], nm[1]

    matrix = []
    for i in range(n):
        matrix.append(list(map(int, list(f.readline().split()))))

    for elem in matrix:
        print(' '.join(map(str, elem)))

    m1 = [matrix[i][i] for i in range(min(n, m))]
    m2 = []
    m3 = []

    # print(m1)
    while True:
        while len(m1) != 0:
            elem = m1.pop(0)
            flag = True
            if flag:
                if len(m2) == 0:
                    m2.append(elem)
                elif m2[-1] > elem:
                    m3.append(elem)
                    flag = False
                else:
                    m2.append(elem)
            else:
                if m2[-1] > elem:
                    m2.append(elem)
                    flag = True
                else:
                    m3.append(elem)
        if len(m3) == 0:
            m1 = m2
            break
    # print(m2)
    # print(m3)

    while True:
        el2 = m2[0]
        el3 = m3[0]
        if el2 < el3:
            m1.append(el2)
            m2.pop(0)
            if len(m2) == 0:
                m1.extend(m3)
                break
        else:
            m1.append(el3)
            m3.pop(0)
```



```

        if len(m3) == 0:
            m1.extend(m2)
            break

    m3, m2 = [], []
    # print(m1)
    print()

    for i, el in enumerate(m1):
        matrix[i][i] = el

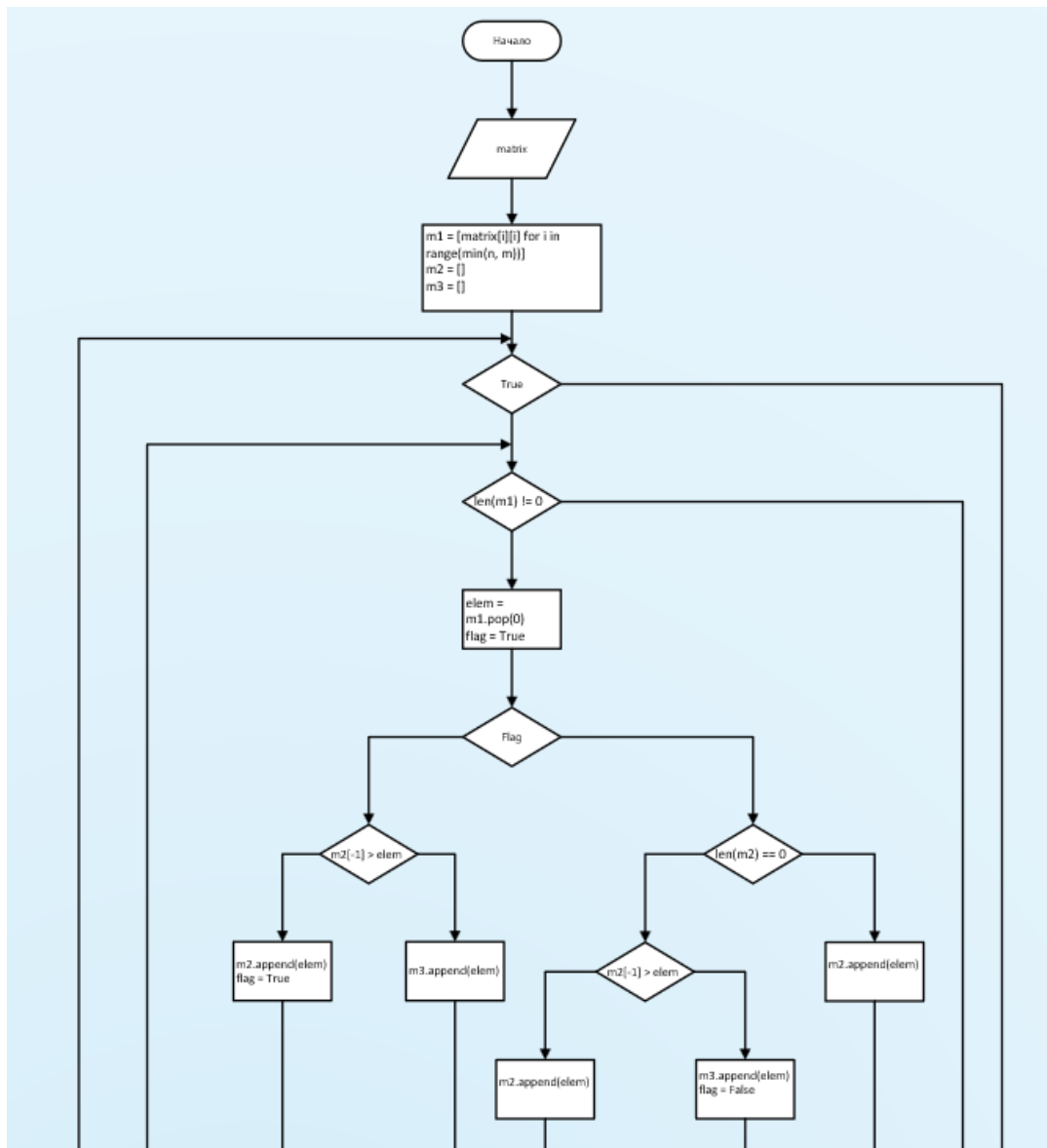
    for elem in matrix:
        print(' '.join(map(str, elem)))

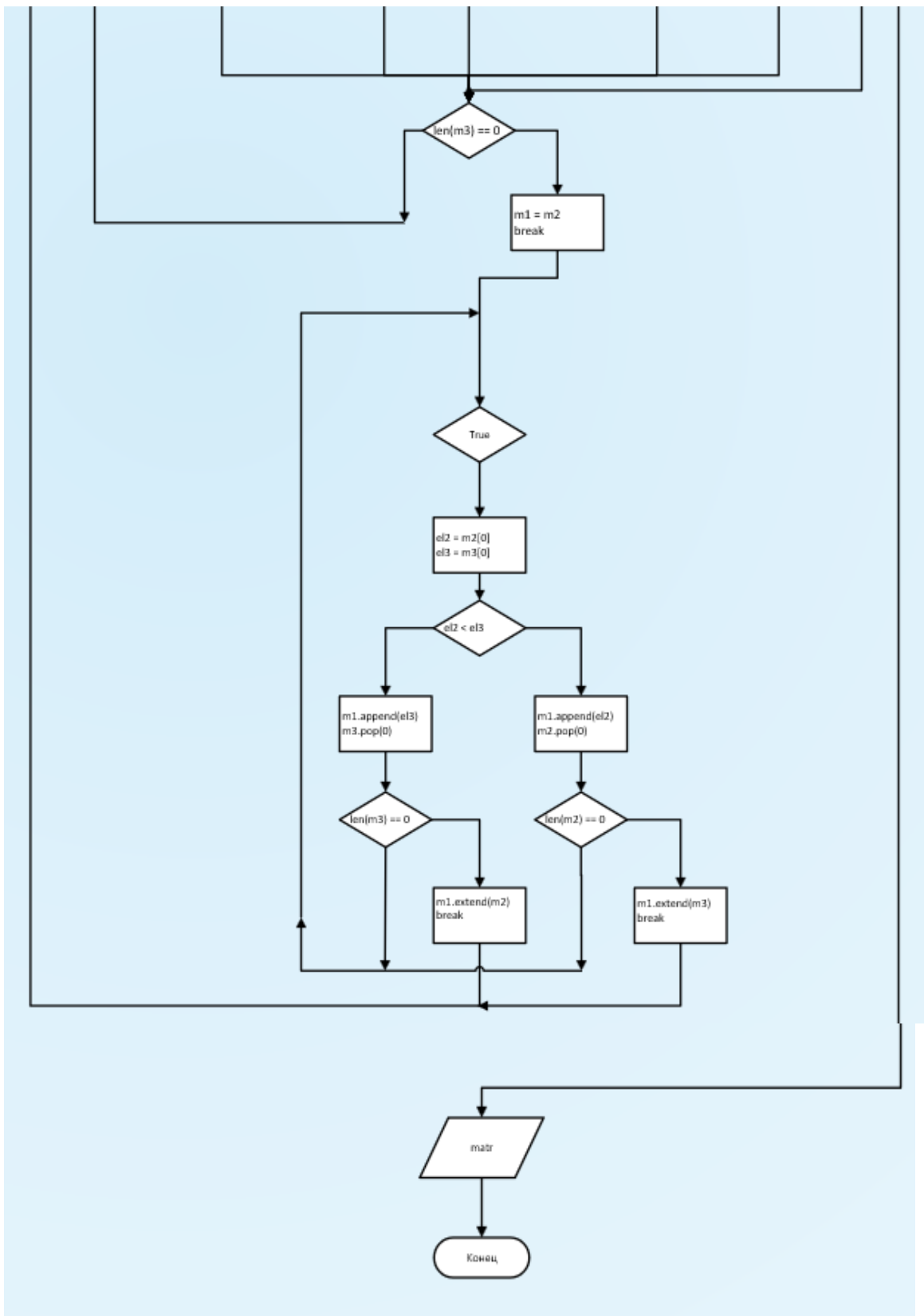
```

```

5 5
4 2 3 4 5
1 2 3 4 6
1 2 5 4 7
1 2 3 1 8
1 2 3 1 8

```





Вывод: в процессе работы я сформировал практические навыки разработки алгоритма сортировки.



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК-КФ «Информатика и управление»
КАФЕДРА ИУК5-КФ «Системы обработки информации»

Лабораторная работа №5

«Рекурсия. Поиск подстроки в строке»

ДИСЦИПЛИНА: «Вычислительные алгоритмы»

Выполнил: студент гр. ИУК5-41Б

Иванов
(Подпись)

(Иванов Н.В.)
(Ф.И.О.)

Проверил:

В.В. Виноградов
(Подпись)

(Виноградов В.В.)
(Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2024

Цель: получение практических навыков при реализации рекурсивных функций, типовых алгоритмов поиска подстроки в строке

9. Дано натуральное число $n > 1$. Выведите все простые множители этого числа в порядке неубывания с учетом кратности. Алгоритм должен иметь сложность $O(\log n)$.

```
def prime_factors(n, divi=2):
    if n <= 1:
        return
    if n % divi == 0:
        count = 0
        while n % divi == 0:
            n //= divi
            count += 1
        if count > 1:
            print(f"{divi}^{count}", end=" ")
        else:
            print(divi, end=" ")

    prime_factors(n, divi + 1)

n = int(input("Число: "))
print(f"Простые множители числа :")
prime_factors(n)
```

$O(\log)$

Число: 345

Простые множители числа :

3 5 23

9. Алгоритм Райта

```
from pprint import pprint

def pre(word):
    n = len(word)
    res = {}
    for i, elem in enumerate(word[::-1]):
        if i == 0:
            continue
        else:
            if elem not in res.keys():
                res[elem] = i

    print(word)
    pprint(res)
    return res

def rait(word, proposal):
    n = len(word)
    m = len(proposal)

    offset = 0

    off_word = 0
    off_word2 = n - 1
    off_word3 = (n - 1) // 2
```

```

shift = pre(word)

res = []
while off_word2 + offset <= m - 1:
    print(off_word2 + offset)
    key = proposal[off_word2 + offset]
    if (word[off_word2] == key and word[off_word] == proposal[offset]
        and word[off_word3] == proposal[off_word3 + offset]):
        end = True
        for i in range(1, n - 2):
            if word[i] != proposal[i + offset]:
                end = False
                break

        if end:
            res.append([offset, off_word2 + offset])
            offset += n
            # print(res)
    else:
        if key in shift.keys():
            offset += shift[key]
        else:
            offset += n

print(res)

rait("a ", "mmasdmmasda sdas")

```

$O(M*N)$

“a “

{'a': 1}

[[10, 11]]

Вывод: в процессе работы я получил практические навыки при реализации рекурсивных функций, типовых алгоритмов поиска подстроки в строке.



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК-КФ «Информатика и управление»
КАФЕДРА ИУК5-КФ «Системы обработки информации»

Лабораторная работа №6

«Хеш-таблицы»

ДИСЦИПЛИНА: «Вычислительные алгоритмы»

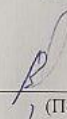
Выполнил: студент гр. ИУК5-41Б



(Подпись)

(Иванов Н.В.)
(Ф.И.О.)

Проверил:



(Подпись)

(Верховский Е.В.)
(Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2024

Цель: изучить построение функции хеширования и алгоритмов хеширования данных и научиться разрабатывать алгоритмы открытого и закрытого хеширования при решении задач

9. Построить хеш-таблицу для зарезервированных слов языка C++ (не менее 20 слов), содержащую HELP для каждого слова. Выдать на экран подсказку по введенному слову. Выполнить программу для различных размерностей таблицы и сравнить время поиска и количество сравнений. Сравнить эффективность добавления ключа в таблицу или ее реструктуризацию для различной степени заполненности таблицы.

```
import time

class HashTable:
    def __init__(self, size):
        self.size = size
        self.table = [None] * size

    def hash(self, word):
        return sum([ord(c) for c in word]) % self.size

    def insert(self, word, help_message):
        start = time.perf_counter()

        index = self.hash(word)
        if self.table[index] is None:
            # print(index)
            self.table[index] = [(word, help_message)]
        else:
            # print(index)
            self.table[index].append((word, help_message))

        finish = time.perf_counter()

        return finish - start

    def search(self, word):
        start = time.perf_counter()

        index = self.hash(word)
        if self.table[index] is not None:
            for w, help_message in self.table[index]:
                if w == word:
                    finish = time.perf_counter()
                    print(
                        f"Время поиска элемента в таблице с размером {self.size} = {round((finish - start) * 1000, 4)}")
                    return help_message
            finish = time.perf_counter()

            print(f"Время поиска элемента в таблице с размером {self.size} = {round((finish - start) * 1000, 4)}")

            return "Word not found"

reserved_words = {
    "auto": "The auto keyword declares an automatic variable.",
    "break": "The break keyword is used to terminate the loop or switch statement.",
    "case": "The case keyword is used in switch statements to specify
```

```

different code blocks to be executed.",
    "char": "The char keyword is used to declare a character type variable.",
    "const": "The const keyword is used to define constants.",
    "continue": "The continue keyword is used to skip the current iteration
in a loop and move to the next one.",
    "default": "The default keyword is used in switch statements to "
               "specify the code block that is executed if no case matches.",
    "do": "The do keyword is used to start a do-while loop.",
    "double": "The double keyword is used to declare a double precision
floating point variable.",
    "else": "The else keyword is used in conditional statements to specify "
            "the code block that is executed if the condition is false.",
    "enum": "The enum keyword is used to declare an enumeration.",
    "extern": "The extern keyword is used to declare a global variable or
function that is defined in another file.",
    "float": "The float keyword is used to declare a floating point
variable.",
    "for": "The for keyword is used to create a for loop.",
    "goto": "The goto keyword is used to transfer control to a labeled
statement in a program.",
    "if": "The if keyword is used to create conditional statements.",
    "int": "The int keyword is used to declare an integer variable.",
    "long": "The long keyword is used to declare a long integer variable.",
    "register": "The register keyword is used to declare a register
variable.",
    "return": "The return keyword is used to exit a function and return a
value."
}

hash_table_size = 20
hash_table = HashTable(hash_table_size)

res = []
for word, help_message in reserved_words.items():
    t = hash_table.insert(word, help_message)
    res.append(round(t * 1000, 4))

print(f"Время вставки элемента в таблицу с размерностью
{hash_table_size}")
print(res)

word_to_search = input("Enter a reserved word to get help message: ")
help_message = hash_table.search(word_to_search)
print(help_message)

reserved_words = {
    "alignas": "Specifies the alignment requirement for a variable or
structure",
    "alignof": "Obtains the alignment required of the type of an expression",
    "and": "Logical AND operator",
    "and_eq": "Bitwise AND assignment operator",
    "asm": "Introduces an inline assembly language statement",
    "auto": "Declares automatic variables",
    "bitand": "Bitwise AND operator",
    "bitor": "Bitwise OR operator",
    "bool": "Defines the bool type",
    "break": "Exits a loop",
    "case": "Labels a statement as the target of a switch statement",
    "catch": "Handles an exception",
    "char": "Defines the char type",
    "char8_t": "Defines the char8_t type",
    "char16_t": "Defines the char16_t type",
    "char32_t": "Defines the char32_t type",

```



```

        "class": "Defines a class",
        "compl": "Bitwise NOT operator",
        "concept": "Specifies a constrained template",
        "const": "Specifies that an object is constant",
        "consteval": "Specifies a compile-time evaluation",
        "constexpr": "Specifies that a function is a compile-time constant
expression",
        "constinit": "Specifies that a variable is initialized once",
        "const_cast": "Converts a variable to a different type",
        "continue": "Jumps to the next iteration of a loop",
        "co_await": "Suspends the coroutine",
        "co_return": "Returns from a coroutine",
        "co_yield": "Yields a value from a coroutine",
        "decltype": "Obtains the type of an expression",
        "default": "Specifies a default case in a switch statement",
        "delete": "Deletes an object",
        "do": "Starts a do-while loop",
        "double": "Defines the double type",
        "dynamic_cast": "Performs a dynamic cast",
        "else": "Indicates the alternative statement for an if statement",
        "enum": "Describes a set of named integer constants",
        "explicit": "Specifies an explicit constructor",
        "export": "Declares a template to be exported",
        "extern": "Specifies that a variable or function exists externally",
        "false": "Boolean value false",
        "float": "Defines the float type"
    }

hash_table_size = 40
hash_table = HashTable(hash_table_size)

res = []
for word, help_message in reserved_words.items():
    t = hash_table.insert(word, help_message)
    res.append(round(t * 1000, 4))

print(f"Время вставки элемента лементах в таблице с размерностью
{hash_table_size}")
print(res)

word_to_search = input("Enter a reserved word to get help message: ")
help_message = hash_table.search(word_to_search)
print(help_message)

```

Время вставки элемента лементах в таблице с размерностью 20

[0.0066, 0.0022, 0.001, 0.0006, 0.0008, 0.0008, 0.0011, 0.0009, 0.0007, 0.0006, 0.0006, 0.0009, 0.0007, 0.0005, 0.0006, 0.0006, 0.0004, 0.0005, 0.0008, 0.0006]

Enter a reserved word to get help message: float

Время поиска элемента в таблице с размером 20 = 0.011

The float keyword is used to declare a floating point variable.

Время вставки элемента лементах в таблице с размерностью 40

[0.0021, 0.0008, 0.0007, 0.0009, 0.0005, 0.0007, 0.0006, 0.0006, 0.0005, 0.0008, 0.0006, 0.0005, 0.0005, 0.0008, 0.0009, 0.0007, 0.0007, 0.0005, 0.0007, 0.0006, 0.0009, 0.0007, 0.0007, 0.0007, 0.0008, 0.0007, 0.0007, 0.001, 0.0007, 0.0006, 0.0007, 0.0009, 0.0005, 0.0006, 0.0006, 0.0007, 0.0006, 0.0005, 0.0005]

Enter a reserved word to get help message: enum

Время поиска элемента в таблице с размером 40 = 0.0096

Describes a set of named integer constants

Вывод: в процессе работы я изучил построение функции хеширования и алгоритмов хеширования данных и научился разрабатывать алгоритмы открытого и закрытого хеширования при решении задач.



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК-КФ «Информатика и управление»
КАФЕДРА ИУК5-КФ «Системы обработки информации»

Лабораторная работа №7

«Бинарные деревья»

ДИСЦИПЛИНА: «Вычислительные алгоритмы»

Выполнил: студент гр. ИУК5-41Б

(Подпись)

(Иванов Н.В.)
(Ф.И.О.)

Проверил:

(Подпись)

(Верховский Е.В.)
(Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2024

Цель: получение практических навыков реализации бинарных деревьев

9. Построить дерево поиска с элементами — вещественными числами. Определить количество элементов дерева на каждом уровне. Удалить элементы с заданными значениями.

```
class Tree_node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

class Tree:
    def __init__(self):
        self.root = None

    def addr(self, node, parent, value):
        if node is None:
            return None, parent, False

        if value == node.data:
            return node, parent, True

        if value < node.data:
            if node.left:
                return self.addr(node.left, node, value)

        if value > node.data:
            if node.right:
                return self.addr(node.right, node, value)

        return node, parent, False

    def append(self, obj):
        if self.root is None:
            self.root = obj
            return obj

        s, p, fl_find = self.addr(self.root, None, obj.data)

        if not fl_find and s:
            if obj.data < s.data:
                s.left = obj
            else:
                s.right = obj

        return obj

    def show(self, node):
        if node is None:
            return

        self.show_tree(node.left)
        print(node.data)
        self.show_tree(node.right)

    def show_tree(self, node):
        if node is None:
            return

        v = [node]
        while v:
```

```

        vn = []
        for x in v:
            print(x.data, end=" ")
            if x.left:
                vn += [x.left]
            if x.right:
                vn += [x.right]
        print()
        v = vn

def count_elem(self):
    node = self.root
    if node is None:
        return

    print(1)
    v = [node]
    while v:
        vn = []
        for x in v:
            if x.left:
                vn += [x.left]
            if x.right:
                vn += [x.right]
        if len(vn) != 0:
            print(len(vn))
        v = vn

def del_leaf(self, s, p):
    if p.left == s:
        p.left = None
    elif p.right == s:
        p.right = None

def del_one_child(self, s, p):
    if p.left == s:
        if s.left is None:
            p.left = s.right
        elif s.right is None:
            p.left = s.left
    elif p.right == s:
        if s.left is None:
            p.right = s.right
        elif s.right is None:
            p.right = s.left

def find_min(self, node, parent):
    if node.left:
        return self.find_min(node.left, node)

    return node, parent

def del_node(self, key):
    s, p, fl_find = self.addr(self.root, None, key)

    if not fl_find:
        return None

    if s.left is None and s.right is None:
        self.del_leaf(s, p)
    elif s.left is None or s.right is None:
        self.del_one_child(s, p)
    else:
        sr, pr = self.find_min(s.right, s)

```

```

        s.data = sr.data
        self.del_one_child(sr, pr)

v = [10, 5, 7, 16, 13, 2, 20, 1]
# v = [20, 5, 24, 2, 16, 11, 18]

t = Tree()
for x in v:
    t.append(Tree_node(x))

t.show_tree(t.root)

t.del_node(5)
print()

t.show_tree(t.root)
print()

t.count_elem()

```

```

10
5 16
2 7 13 20
1

10
7 16
2 13 20
1

1
2
3
1

```

Вывод: в процессе работы я получил практические навыки реализации бинарных деревьев