|  |
| --- |
| МИНОБРНАУКИ РОССИИ  Федеральное государственное  бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт информационных технологий

Кафедра корпоративных информационных систем

|  |  |  |
| --- | --- | --- |
| **ОТЧЕТ**  **по лабораторной работе №1** | | |
| **по дисциплине** | | |
| **«Структуры и алгоритмы обработки данных»**  **Тема лабораторной работы: «**Структуры данных: список, очередь, стек**»** | | |
| Студент группы | ИКБО-09-18 | Лебедев О.А. |
| Принял | ассистент кафедры КИС | Габриелян Г.А. |
|  |  |  |
| Выполнено | «\_\_\_» \_\_\_\_\_\_ 201\_\_ г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
|  |  | *(подпись студента)* |
| Зачтено | «\_\_\_» \_\_\_\_\_\_ 201\_\_ г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
|  |  | *(подпись преподавателя)* |

Москва 2019

1. **Задача №1**
   1. **Постановка задачи**

Составить программу, которая объединяет два упорядоченных по возрастанию линейных однонаправленных списка в один упорядоченный по возрастанию ЛОС

* 1. **Описание используемых структур данных**

Линейный однонаправленный список — это структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей. Каждый элемент списка имеет указатель на следующий элемент. Последний элемент списка указывает на NULL. Элемент, на который нет указателя, является первым (головным) элементом списка. Здесь ссылка в каждом узле указывает на следующий узел в списке. В односвязном списке можно передвигаться только в сторону конца списка.



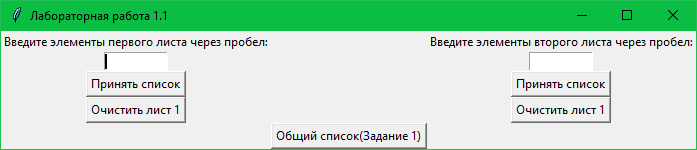
* 1. **Пользовательский интерфейс**

Рисунок . Пользовательский интерфейс задачи 1

Кнопки:

* «Принять список» - добавляет элементы введенные в текстовое поле выше в Связанный список.
* «Очистеть лист» - очищает связанный список.
* «Общий список(Задание 1)» - объединяет 2 связанных списка.
  1. **Описание алгоритма**
* Вводим элементы в первый список, список сортируется каждый раз при вводе элемента или нескольких элементов
* Вводим элементы в второй список, список сортируется каждый раз при вводе элемента или нескольких элементов
* Если нужно очистить списки, очищаем специальной кнопкой
* В задание мы объединяем два введенных списка в один и сортируем

* 1. **Тестирование**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| list 1 | list 2 | result | result |
| 1. | 1 2 3 4 5 6 | 6 5 4 3 2 1 | 1 1 2 2 3 3 4 4 5 5 6 6 | 1 1 2 2 3 3 4 4 5 5 6 6 | пройден |
| 2. | 32 44 0 4 | 64 88 0 8 | 0 0 4 8 32 44 64 99 | 0 0 4 8 32 44 64 99 | пройден |
| 3. | 11 342 543 645 99 1 | 4 89 1 7 | 1 1 4 7 11 89 99 342 543 645 | 1 1 4 7 11 89 99 342 543 645 | пройден |
| 4. | 1 3 5 7 9 0 | 3 3 3 | 0 1 3 3 3 3 5 7 9 | 0 1 3 3 3 3 5 7 9 | пройден |
| 5. | 323 654 743 678 | 74 12 8 5 | 5 8 12 74 323 654 678 743 | 5 8 12 74 323 654 678 743 | пройден |
| 6. | 1 1 1 1 | 0 0 0 0 | 0 0 0 0 1 1 1 1 | 0 0 0 0 1 1 1 1 | пройден |

* 1. **Листинг программы**

1. Класс List

**import** random  
**class** Node:  
 **def** \_\_init\_\_(self,value = **None**, next = **None**):  
 self.value= value  
 self.next = next  
  
**class** LinkList:  
 **def** \_\_init\_\_(self):  
 self.first = **None** self.last = **None** self.lenght = 0  
  
 **def** \_\_str\_\_(self):  
 **if** self.first != **None**:  
 current = self.first  
 out = **'List ['** + str(current.value) + **' '  
 while** current.next != **None**:  
 current = current.next  
 out += str(current.value) + **' '  
 return** out + **']'  
 return 'List []'  
  
 def** clear(self):  
 self.\_\_init\_\_()  
  
 **def** add(self,x):  
 self.lenght+=1  
 **if** self.first ==**None**:  
 self.first= self.last=Node(x,**None**)  
 **else**:  
 self.last.next =self.last =Node(x,**None**)  
  
 **def** sort(self):  
 a = Node(0, **None**)  
 b = Node(0, **None**)  
 c = Node(0, **None**)  
 e = Node(0, **None**)  
 tmp = Node(0, **None**)  
  
 **while** (e != self.first.next):  
 c = a = self.first  
 b = a.next  
  
 **while** a != e:  
 **if** a **and** b:  
 **if** a.value > b.value:  
 **if** a == self.first:  
 tmp = b.next  
 b.next = a  
 a.next = tmp  
 self.first = b  
 c = b  
 **else**:  
 tmp = b.next  
 b.next = a  
 a.next = tmp  
 c.next = b  
 c = b  
 **else**:  
 c = a  
 a = a.next  
 b = a.next  
 **if** b == e:  
 e = a  
 **else**:  
 e = a  
  
 **def** lenght(self):  
 **return** (lenght)  
  
 **def** createList(self):  
 temp = self.first  
 list1 = []  
 **while** (temp != **None**):  
 list1.append(temp.value)  
 temp = temp.next  
 **return** list1

1. Основная программа

**from** tkinter **import**\*  
**from** tkinter **import** messagebox  
**import** linklist  
*#1methods***def** click1():  
 tempint = txt1.get().split(**" "**)  
 tempint = [int(i) **for** i **in** tempint]  
 **for** i **in** tempint:  
 l1.add(i)  
 l1.sort()  
 messagebox.showinfo(**"First list"**, l1)  
**def** click2():  
 tempint = txt2.get().split(**" "**)  
 tempint = [int(i) **for** i **in** tempint]  
 **for** i **in** tempint:  
 l2.add(i)  
 l2.sort()  
 messagebox.showinfo(**"Second list"**, l2)  
**def** delite1():  
 l1.clear()  
 messagebox.showinfo(**"First list"**, **"list 1 clear"**)  
**def** delite2():  
 l2.clear()  
 messagebox.showinfo(**"Second list"**, **"list 2 clear"**)  
**def** click3():  
 l3.clear()  
 templist3 = []  
 templist1 = l1.createList()  
 templist2 = l2.createList()  
 templist3 = templist1 + templist2  
 **for** i **in** templist3:  
 l3.add(i)  
 l3.sort()  
 lbl3.configure(text = l3)  
  
  
*#1methods*l1 = linklist.LinkList()  
l2 = linklist.LinkList()  
l3 = linklist.LinkList()  
  
window= Tk()  
window.title(**"Лабораторная работа 1.1"**)  
*#label*lbl1 = Label(window,text = **"Введите элементы первого листа через пробел:"**)  
lbl2 = Label(window,text = **"Введите элементы второго листа через пробел:"**)  
lbl3 = Label(window)  
lbl2.grid(column = 2,row = 0)  
lbl1.grid(column = 0,row = 0)  
lbl3.grid(column = 1,row = 3)  
*#label  
  
#region Button*btn1 =Button(window,text = **"Принять список"**,command = click1)  
btn2 =Button(window,text = **"Принять список"**,command = click2)  
btn3 = Button(window,text = **"Общий список(Задание 1)"**, command = click3 )  
btn4 = Button(window, text = **"Очистить лист 1"**, command = delite1)  
btn5 = Button(window, text = **"Очистить лист 1"**, command = delite2)  
btn1.grid(column = 0,row = 2)  
btn2.grid(column = 2,row = 2)  
btn3.grid(column = 1,row = 4)  
btn4.grid(column = 0,row = 3)  
btn5.grid(column = 2,row = 3)  
*#endregion Button  
  
  
#TextBox*txt1 = Entry(window,width =10)  
txt2 = Entry(window,width =10)  
txt1.grid(column = 0, row = 1)  
txt2.grid(column = 2, row = 1)  
  
window.mainloop()

1. **Задача №2**
   1. **Постановка задачи**

Составить программу построения стека, элементами которого являются словами. Сформировать новый стек, содержащий в качестве элементов палиндромы слов из исходного текста

* 1. **Описание используемых структур данных**

Стек — абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»).



* 1. **Пользовательский интерфейс**

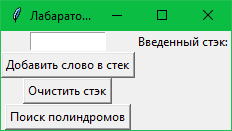


Рисунок . Пользовательский интерфейс задачи 2

Кнопки:

* «Добавить слово в стэк» - добавляет введенное в текстовое поле слово в стэк.
* «Очистить стэк» - очищает стэк.
* «Поиск полиндромов» - ищет в стэке слова полиндромы.
  1. **Описание алгоритма**
* Создаём стек
* Проходим по всем элементам исходного стека
* На каждой итерации, если первая половина текста элемента равна второй половине справа налево, то добавляем этот элемент в созданный массив
  1. **Тестирование**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| stack | result | result |
| 1. | qwer rest oo | oo | oo | пройден |
| 2. | wow task slip | wow | wow | пройден |
| 3. | 321 123 wriirw | wriirw | wriirw | пройден |
| 4. | 110011 1 zxcv | 110011 1 | 110011 1 | пройден |
| 5. | huh what lool | huh lool | huh lool | пройден |

* 1. **Листинг программы**

1. Класс Stack

**class** Stack:  
 **def** \_\_init\_\_(self):  
 self.items = []  
 **def** push(self,item):  
 self.items.append(item)  
 **def** pop(self):  
 **return** self.items.pop()  
 **def** \_\_str\_\_(self):  
 out = **""  
 for** i **in** self.items:  
 out += i + **' '  
 return** out  
 **def** is\_empty(self):  
 **return** (self.items == [])  
 **def** clear(self):  
 self.items = []  
 **def** task1(self):  
 list = []  
 **for** i **in** self.items:  
 x = len(i)- 1  
 y = 0  
 k = 0  
 **while** x- y >= y:  
 **if** i[x- y] == i[y]:  
 y+=1  
 **else**:  
 k = 1  
 **break  
 if** k == 0:  
 list.append(i)  
 **return** list

1. Основная программа

**from** tkinter **import**\*  
**from** tkinter **import** scrolledtext  
**import** Stack  
  
**def** click1():  
 temp = txt1.get().split()  
 **for** i **in** temp:  
 st1.push(i)  
 lbl1.configure(text = st1)  
**def** click2():  
 st1.clear()  
 lbl1.configure(text=st1)  
**def** click3():  
 st2.clear()  
 temp = []  
 temp = st1.task1()  
 **if** len(temp) == 0:  
 lbl3.configure(text = **"Нет полиндромов"**)  
 **else**:  
 **for** i **in** temp:  
 st2.push(i)  
 lbl3.configure(text=st2)  
  
  
st1 =Stack.Stack()  
st2 = Stack.Stack()  
  
win = Tk()  
win.title(**"Лабараторная 1.2"**)  
  
txt1 = Entry(win, width = 12)  
txt1.grid(column= 0, row = 0)  
  
btn1 = Button(win,text = **"Добавить слово в стек"**,command = click1)  
btn2 = Button(win, text = **"Очистить стэк"**,command = click2)  
btn3 = Button(win,text = **"Поиск полиндромов"**, command = click3)  
btn1.grid(column = 0,row = 1)  
btn2.grid(column = 0,row = 3)  
btn3.grid(column = 0,row = 4)  
  
lbl1 = Label(win)  
lbl2 = Label(win,text = **"Введенный стэк:"**)  
lbl3 = Label(win)  
lbl1.grid(column = 1, row = 1)  
lbl2.grid(column = 1, row = 0)  
lbl3.grid(column = 1, row = 4)  
  
win.mainloop()

1. **Задача №3**
   1. **Постановка задачи**

Составить программу построения очереди, содержащей целые числа. Вычислить количество совершенных чисел, содержащихся в очереди. Натуральное число называют совершенным, если оно равно сумме всех своих делителей, не считая его самого.

* 1. **Описание используемых структур данных**

Очередью называется структура данных, из которой удаляется первым тот элемент, который был первым в очередь добавлен. То есть очередь в программировании соответствует «бытовому» понятию очереди. Очередь также называют структурой типа FIFO



* 1. **Пользовательский интерфейс**

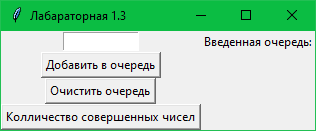


Рисунок . Пользовательский интерфейс задачи 3

Кнопки:

* «Добавить в очередь» - добавляет введенный элемент в список.
* «Очистить очередь» - очищает очередь.
* «Количество совершенных чисел» - находит количество совершенных чисел в очереди.
  1. **Описание алгоритма**
* Добавление в очередь элемента
* Проходим по каждому элементу очереди
* Делим каждый элемент на числа от 1 до самого числа и если делятся суммируем с суммой
* Сравниваем сумму и число.
  1. **Тестирование**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| Queue | result | result |  |
| 1. | 1 2 3 4 5 | 1 | 1 | пройден |
| 2. | 6 98 42 18 | 1 | 1 | пройден |
| 3. | 6 1 128 89 | 2 | 2 | пройден |
| 4. | 12 54 1024 | 0 | 0 | пройден |
| 5. | 58 21 0 4 | 1 | 1 | пройден |

* 1. **Листинг программы**

1. Класс Queue

**class** Queue:  
 **def** \_\_init\_\_(self):  
 self.queue = []  
 self.max = 50  
 **def** \_\_str\_\_(self):  
 out = **""  
 for** i **in** self.queue:  
 out +=str(i) + **" "  
 return** out  
 **def** clear(self):  
 self.queue[:] = []  
  
 **def** add(self, x):  
 **if** len(self.queue) == self.max:  
 **return None** self.queue.append(x)  
 **def** dequeue(self):  
 **if not** self.queue:  
 **return** (**None**)  
 **return** self.queue.pop(0)  
 **def** task(self):  
 task = 0  
 **for** i **in** self.queue:  
 s =0  
 **for** j **in** range(1,i):  
 **if** i%j==0:  
 s+=j  
 **if** s == i **or** i == 1:  
 task +=1  
 print (i)  
 **return** task

1. Основная программа

**from** tkinter **import**\*  
**import** Queue  
  
**def** click1():  
 tempint = txt1.get().split(**" "**)  
 tempint = [int(i) **for** i **in** tempint]  
 **for** i **in** tempint:  
 q1.add(i)  
 lbl1.configure(text = q1)  
 txt1.configure(text = **""**)  
**def** click2():  
 q1.clear()  
 lbl1.configure(text=q1)  
**def** click3():  
 lbl3.configure(text=q1.task())  
  
q1 = Queue.Queue()  
win = Tk()  
win.title(**"Лабараторная 1.3"**)  
  
txt1 = Entry(win, width = 12)  
txt2 = Entry(win, width = 3)  
txt1.grid(column= 0, row = 0)  
txt2.grid(column = 0,row = 1)  
  
btn1 = Button(win,text = **"Добавить в очередь"**,command = click1)  
btn2 = Button(win, text = **"Очистить очередь"**,command = click2)  
btn3 = Button(win, text = **"Колличество совершенных чисел"**,command = click3)  
btn1.grid(column = 0,row = 1)  
btn2.grid(column = 0,row = 2)  
btn3.grid(column = 0, row = 3)  
  
lbl1 = Label(win)  
lbl2 = Label(win,text = **"Введенная очередь:"**)  
lbl3 = Label(win)  
lbl1.grid(column = 1, row = 1)  
lbl2.grid(column = 1, row = 0)  
lbl3.grid(column = 1, row = 3)  
  
win.mainloop()