Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

РАСЧЕТНО-ГРАФИЧЕСКОЕ ЗАДАНИЕ

по дисциплине «Функциональное и логическое программирование»

Вариант 8

Выполнил:

студент группы ИП-715

Комашко Тимур Максимович

Работу проверила: ассистент кафедры ПмиК

Василина Валерьевна Долгополова

Новосибирск 2019 г.

**Содержание**

[1. Задание № 1. Списки. 3](#_Toc24299860)

[1.1. Условие задачи. 3](#_Toc24299861)

[1.2. Алгоритм решения поставленной задачи. 3](#_Toc24299862)

[1.3. Листинг программы. 3](#_Toc24299863)

[1.3 Скриншоты работы программы. 4](#_Toc24299864)

[2. Задание № 2. Строки, файлы. 5](#_Toc24299865)

[2.1 Условие задачи. 5](#_Toc24299866)

[2.2. Алгоритм решения поставленной задачи. 5](#_Toc24299867)

[2.3. Листинг программы. 5](#_Toc24299868)

[2.4. Скриншоты программы. 6](#_Toc24299869)

# **1. Задание № 1. Списки.**

## **1.1. Условие задачи.**

Подсчитайте число вхождений каждого элемента списка и сформируйте новый список, каждый элемент которого - это список из двух элементов: самого элемента и числа его вхождений в список. Например, [2,4,2,3,2,3]-> [[2,3], [4,1], [3,2]].

## **1.2. Алгоритм решения поставленной задачи.**

Вводится список, с которым будут производиться дальнейшие действия. Было решено реализовать задачу в три этапа:

1. Создается список из элементов без повторений;
2. Создается список с количеством повторений (частота встречаемости того или иного элемента);
3. Объединение этих двух списков.

1 и 2 пункты могут меняться местами.

Полученный список необходимо отсортировать, сохранив повторения. Это было реализовано через *in\_sort* (сортировка вставками). Далее в функции *kol* список частот:

* Голова сравнивается с головой хвоста, и если они равны, то счетчик количества одинаковых элементов увеличивается. В противном случае, значение счетчика заносится в список, а сам счетчик обнуляется (в нашем случае, становится равен 1).
* Если хвост основного списка не пустой, заново запускаем нашу функцию *kol*.
* Когда основной хвост оказался пустым, проверяем совпадает ли наша голова с головой хвоста (остался 1 элемент, который не проверили). Если да, то увеличиваем счетчик на 1, если нет, то заносим 1 (новое число).

Далее, через вспомогательную функцию попадаем в функцию *del*, где формируется список без повторяющихся элементов.

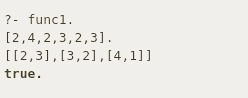
* Голова добавляется в новый список.
* Все одинаковые с головой элементы удаляются.
* Пока не конец списка (хвост не равен пустому списку) возвращаемся в функцию *del.*
* Когда хвост равен пустому списку переходим во вторую вспомогательную функцию, откуда попадаем в функцию формирования списка.

*spis* – функция, где список частот и список без повторений объединяются, пока хвост (не важно от какого из двух списков) не станет равен пустому списку. Когда списки заканчиваются, выводим итоговый список.

## **1.3. Листинг программы.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | in\_sort([ ],[ ]). | | in\_sort([X|Tail],Sort\_list):- | | in\_sort(Tail,Sort\_tail), | | insert(X,Sort\_tail,Sort\_list). | | insert(X,[Y|Sort\_list],[Y|Sort\_list1]):- | | X@>Y,!, | | insert(X,Sort\_list,Sort\_list1). | | insert(X,Sort\_list,[X|Sort\_list]). | | kol(L,[Head|Tail],S,Res):- | | Tail = [H|\_], | | (Head =:= H, S1 is S + 1, Res1 = Res; append([S],Res,Res1), S1 is 1), | | (Tail \== [], kol(L,Tail,S1,Res1),!; | | (Head =:= H, S2 is S1 + 1; S2 is S1), | | append([S2],Res1,Res2),f(L,Res2)). | |  | | del(List,Del,Kol):- | | List = [Head|Tail], | | append([Head],Del,Del1), | | delete(List,Head,List2), | | (Tail \== [], del(List2,Del1,Kol),!; f2(Del1,Kol)). | |  | | spis([Head|Tail],[H|T],Res):- | | append([[Head, H]],Res,Res1), | | (Tail \== [], spis(Tail,T,Res1),!;write(Res1)). | |  | | f(List,Kol):- | | Del = [], | | del(List,Del,Kol). | | f2(Del,Kol):- | | Res = [], | | spis(Del,Kol,Res). | |  | | func1:- | | write('Список: '), | | read(List), | | in\_sort(List,L1), | | S is 1, | | Kol = [], | | kol(L1,L1,S,Kol). | |
|  |  |
|  |  |

## **1.3 Скриншоты работы программы.**



# **2. Задание № 2. Строки, файлы.**

## **2.1 Условие задачи.**

В текстовом файле, состоящем из нескольких строк, определите количество слов в каждой строке и информацию об этом занесите в отдельную строку нового файла в виде: cтрока № <номер строки> содержит <количество> слов.

## **2.2. Алгоритм решения поставленной задачи.**

Через предикат *open* открываются два файла: первый на чтение, второй на запись. Потоки вывода и ввода привязываются к соответствующим файлам (для чтения и записи). Счетчик строк обнуляется и вместе с файловой переменной 1 файла передается в функцию для чтения строк. В функции *r* обнуляется количество слов (ставится в единицу), увеличивается счетчик строк, считывается строка и записывается в виде кодов символов в список. Затем, полученный список, счетчики слов и строк передаются в функцию для подсчета. А функция *r* запускается, пока не будет найден конец файла.

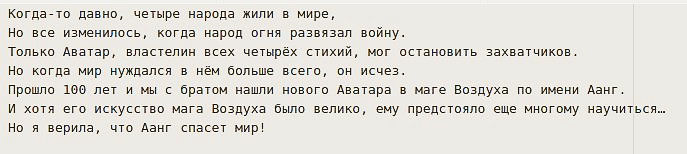
В функции подсчета *chet* считается количество пробелов (слов на 1 больше, чем пробелов, поэтому счетчик с 1). Если же пробелы не были найдены, то количество слов будет равно 1. Данная функция запускается, пока хвост списка не будет равен пустому списку. В противном случае, уходим в функцию *w*.

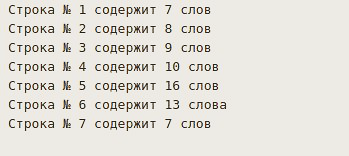
В функции *w* происходит запись в файл необходимой информации.

## **2.3. Листинг программы.**

|  |
| --- |
| w(S,Res):- |
|  | R is Res mod 10, |
|  | write('Строка № '), |
|  | write(S), |
|  | write(' содержит '), |
|  | write(Res), |
|  | (Res =:= 1,write(' слово ');write('')), |
|  | (R =:= 2,write(' слова ');write('')), |
|  | (R =:= 3,write(' слова ');write('')), |
|  | (R =:= 4,write(' слова ');write('')), |
|  | (R >=5 ,write(' слов ');write('')), |
|  | (R =:= 0, write(' слов '); write('')), |
|  | nl. |
|  |  |
|  | chet([Head|Tail],Res,S):- |
|  | (Head =:= 32, Res1 is Res+1; Res1 is Res), |
|  | (Tail \== [], chet(Tail,Res1,S),!;w(S,Res1)). |
|  |  |
|  | r(F1,S):- |
|  | read\_line\_to\_codes(F1,L), |
|  | Res is 1, |
|  | S1 is S + 1, |
|  | chet(L,Res,S1), |
|  | (at\_end\_of\_stream, write('');r(F1,S1),!). |
|  |  |
|  | func2:- |
|  | open('1.txt',read,F1), |
|  | open('2.txt',write,F2), |
|  | set\_input(F1), |
|  | set\_output(F2), |
|  | S is 0, |
|  | r(F1,S), |
|  | close(F1), |
|  | close(F2). |

## **2.4. Скриншоты программы.**

****

****