

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 11

Выполнил:

студент группы ИП-013

Копытина Т.А.

Работу проверил:  
Доцент Мачикина Е.П.

Новосибирск 2022 г.

## Задание

1. Список целых чисел разделите на два списка: из чётных элементов и нечётных элементов;

Например, [1,-2,3,5,-4]-> [1,3,5], [-2,-4].

2. Найдите в файле все слова максимальной длины. Сформируйте новый файл из найденных слов.

3. Создайте базу данных об игрушках: название, стоимость, возрастные границы. Получите названия всех самых дешевых игрушек, подходящих ребенку 3 лет.

## Выполнение

### Задание 1:

Предикат *odd\_even\_sort* распределяет числа по спискам в зависимости от чётности чисел и выводит списки чётных и нечётных чисел на экран. Если число чётное (при делении числа на 2 остаток равен нулю), то оно заносится в список *Even*, иначе – в список *Odd*.

```
odd_even([], [], []).
odd_even([Head | Tail], [Head | Even], Odd):-
Head mod 2 == 0, !, odd_even(Tail, Even, Odd).
odd_even([Head | Tail], Even, [Head | Odd]):-
Head mod 2 == 1, !, odd_even(Tail, Even, Odd).

odd_even_sort(List):-
odd_even(List, Even, Odd),
write('Четные: '), write(Even), nl,
write('Нечетные: '), write(Odd), nl.
```

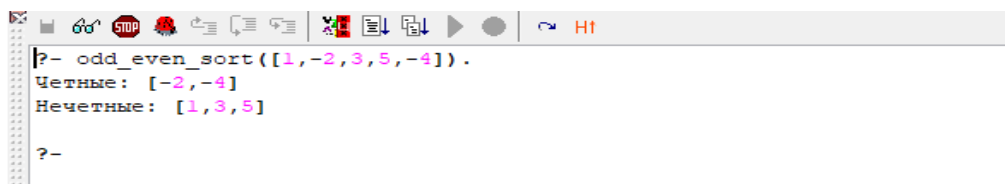


Рисунок 1 – Пример работы 1-ого задания

## Задание 2:

- 1) На вход подаётся строка, состоящая из слов, разделённых пробелом, и заканчивающаяся точкой.
- 2) В предикат `str_to_list_words(S, Lw)` на вход подаётся строка(S) и возвращается список(Lw), атомы которого – слова из строки S.
- 3) В предикат `arr_len_word (Lw, Ll)` подаётся на вход список из слов (Lw), и возвращается список длин этих слов (Ll).
- 4) Предикат `maximum (Ll, MaxLen)` в списке длин слов (Ll) находит максимальную длину (MaxLen) слова (ов), которое (ые) есть в списке (Ll).
- 5) Предикат `list_of_max_words (Lw, MaxLen, Ln)` из списка слов (Lw) формирует новый список слов (Ln), длина которых равна атому MaxLen.
- 6) Предикаты `read_from_file(Filename, S)` и `write_to_file(Filename, S)` читают и записывают данные в файл
- 7) предикат `t2` открывает файл и создает новый файл куда необходимо записать новые данные.

```
str_to_list_words(S, [Hw | Tn]):-
```

```
    front_token(S, W, S2), not(S2 = " "), !,
```

```
    Hw = W, str_to_list_words(S2, Tn).
```

```
str_to_list_words(_, []).
```

%Предикат, который выделяет слова из строки.

```
front_token(S, W, L1):-
```

```
    atom_chars(S, L), %строка > список символов
```

```
    append(ListW, [' ' | List1], L), !,
```

```
    atom_chars(W, ListW),
```

```
    atom_chars(L1, List1).
```

```
front_token(S, W, L1):-
```

```
    atom_chars(S, L), %строка > список символов
```

```
    append(ListW, ['. ' | List1], L), !,
```

```
    atom_chars(W, ListW),
```

```
    atom_chars(L1, List1).
```

%Предикат формирования списка длин слов по списку этих слов

```
arr_len_word([], []):- !.
```

```
arr_len_word([H | T], [Hn | Tn]):-
```

```
    atom_length(H, LenW),
```

```
    Hn = LenW,
```

```
    arr_len_word(T, Tn).
```

%Предикат нахождения максимума в списке (длин заданных слов)

```
maximum([X], X).
```

```
maximum([H | T], H):-maximum(T, M), H > M, !.
```

```
maximum([_ | T], M):-maximum(T, M).
```

%Предикат формирования нового списка путем удаления из заданного списка слов длины, отличной от заданной величины

```
list_of_max_words([], _, []):- !.
```

```
list_of_max_words([H | T], X, [Hn | Tn]):-
```

```
    atom_length(H, LenW), %длина строки
```

```
    LenW = X, !,
```

```
    Hn = H,
```

```
    list_of_max_words(T, X, Tn).
```

```
list_of_max_words([_ | T], X, Ln):- list_of_max_words(T, X, Ln).
```

%Предикат для чтения строки в файл

```
read_from_file(Filename, S):-
```

```
    open(Filename, read, Input),
```

```
    readln(Input, S),
```

```
    close(Input).
```

%Предикат для записи строки в файл

```
write_to_file(Filename, S):-
```

```
    open(Filename, write, Out),
```

```
    write(Out, S),
```

```
    close(Out).
```

```
max_len_words(S, Sn):-
```

```
    write("Список слов:"), nl,
```

```

str_to_list_words(S, Lw), %формирование списка Lw из слов строки S
write(Lw), nl,
arr_len_word(Lw, L1), %нахождение длин слов и занесение их в список
maximum(L1, MaxLen), %нахождение максимальной из длин слов
write("Длина наиболее длинного слова: "), write(MaxLen), nl,
list_of_max_words(Lw, MaxLen, Ln), %формирования списка слов макс.длины
atomics_to_string(Ln, " ", Sn), %список > строка
write(Sn).

```

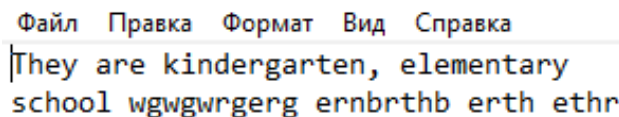
%предикат считывания файла и записи в файл

t2:-

```

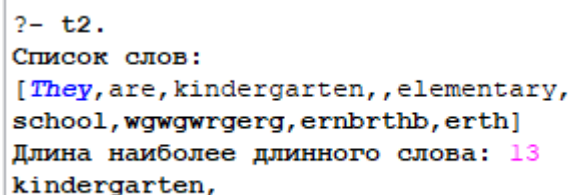
see('input.txt'),
seeing(Stream),
read_stream_to_codes(Stream, List_of_codes),
string_to_list(String, List_of_codes),
max_len_words(String, FinalListMaxWords),
seen,
tell('output.txt'),
write(FinalListMaxWords),
told.

```



Файл Правка Формат Вид Справка  
 They are kindergarten, elementary  
 school wgwgrgerg ernbrthb erth ethr

Рисунок 2. Пример предложения в файле input

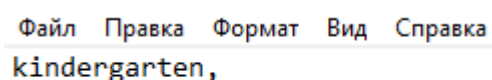


```

?- t2.
Список слов:
[They,are,kindergarten,,elementary,
school,wgwgrgerg,ernbrthb,erth]
Длина наиболее длинного слова: 13
kindergarten,

```

Рисунок 3 – Пример работы 2-ого задания



Файл Правка Формат Вид Справка  
 kindergarten,

Рисунок 4. Пример записи в файл наибольшего слова.

### Задание 3:

- 1) На вход подается пустая база данных DataBase.txt
- 2) Далее вызывается меню, где мы можем добавить записи в базу данных, просматривать базу данных, удалять оттуда записи, просматривать поиск по заданию или закрытие и сохранение базу данных с новыми данными.
- 3) Предикат func(2) добавляет новые записи в базу данных.
- 4) Предикат func(3) удаляет записи из базы данных.
- 5) Предикат func(4) вызывает функции которые ищут записи по заданным параметрам.
- 6) Предикат find\_toys(L\_Res) производит поиск записей удовлетворяющих заданным параметрам.
- 7) Предикаты min\_toys([H],Min,L\_Res), min\_toys([H|T],Min,L\_Res),  
check\_min(H,Min1,\_,Min2,L\_Res,Min),  
check\_min(H,Min1,L\_Res2,Min2,L\_Res,Min) и  
check\_min(\_,\_,L\_Res2,Min2,L\_Res,Min) выполняют поиск записей по минимальной цене, уже найденных записей по предыдущему параметру.
- 8) Предикат write\_list([H|T]) выполняет вывод информации в консоль.
- 9) Предикат func(5) выполняет функцию сохранения и закрытия базы данных.

```
:- discontiguous func/1.
```

```
:- dynamic toy/3.
```

```
task3:-
```

```
    retractall(toy(_,_,_)),  
    exists_file('DataBase.txt'),!,  
    consult('DataBase.txt'),  
    menu.
```

```
task3:-
```

```
    tell('DataBase.txt'),  
    told,
```

```
menu.
```

```
menu:-
```

```
repeat,  
n1,  
writeln('1) - просмотр содержимого базы данных;'),  
writeln('2) - добавления записи'),  
writeln('3) - удаления записи'),  
writeln('4) - самые дешёвые игрушки до 3 лет'),  
writeln('5) - выход и сохранение базы данных'),  
read(X),  
X>0,  
X<6,  
func(X),  
X:=5,!.
```

```
func(1):-
```

```
listing(toy/3).
```

```
func(2):-
```

```
repeat,  
writeln("Введите название игрушки:"),  
read(N),  
writeln("Введите стоимости:"),  
read(Pr),  
writeln("Введите возрастные ограничения:"),  
read(A),  
assertz(toy(N,Pr,A)),  
writeln("Вы хотите продолжить добавление [y/n]?"),  
read(Sw),  
Sw=n,!.
```

```
func(3):-
```

```
repeat,  
writeln("Введите название игрушки:"),  
read(N),  
retract(toy(N,_,_)),
```



```

writeln("Вы хотите продолжить удаление [y/n]?"),
read(Sw),
Sw=n,!

func(4):-
    find_toys(L_Res),
    write_list(L_Res).

find_toys(L_Res):-
    findall(toy(N,Pr,A),(toy(N,Pr,A),A=<3),L),
    min_toys(L,_,L_Res).

min_toys([],_,[]):-
    writeln("Игрушки не найдены").

min_toys([H],Min,L_Res):-
    H=toy(_,Min,_),!,
    L_Res=[H].

min_toys([H|T],Min,L_Res):-
    H=toy(_,Min1,_),
    min_toys(T,Min2,L_Res2),
    check_min(H,Min1,L_Res2,Min2,L_Res,Min).

check_min(H,Min1,_,Min2,L_Res,Min):-
    Min1<Min2,!,
    Min=Min1,
    L_Res=[H].

check_min(H,Min1,L_Res2,Min2,L_Res,Min):-
    Min1==Min2,!,
    Min=Min1,
    L_Res=[H|L_Res2].

check_min(_,_,L_Res2,Min2,L_Res,Min):-
    Min=Min2,
    L_Res=L_Res2.

```

```
write_list([]):-!.
```

```
write_list([H|T]):-  
    writeln(H),  
    write_list(T).
```

```
func(5):-  
    tell('DataBase.txt'),  
    listing(toy/3),  
    told.
```

```
Файл  Правка  Формат  Вид  Справка  
:- dynamic toy/3.
```

Рисунок 5. Пример пустой базы данных.

```
Введите название игрушки:  
|: "Lego".  
Введите стоимости:  
|: 300.  
Введите возрастные ограничения:  
|: 3.  
Вы хотите продолжить добавление [y/n]?  
|: y.  
Введите название игрушки:  
|: "Poni".  
Введите стоимости:  
|: 100.  
Введите возрастные ограничения:  
|: 2.  
Вы хотите продолжить добавление [y/n]?  
|: y.  
Введите название игрушки:  
|: "Pokemon".  
Введите стоимости:  
|: 500.  
Введите возрастные ограничения:  
|: 1.  
Вы хотите продолжить добавление [y/n]?  
|: n.
```

Рисунок 6. Пример добавления записей в базу данных.

```
%%  
%% |: 1.  
%% :- dynamic toy/3.  
%%  
%% toy("Lego", 300, 3).  
%% toy("Poni", 100, 2).  
%% toy("Pokemon", 500, 1).  
%%
```

Рисунок 7. Просмотр заполненной базы данных.

```

1) - просмотр содержимого базы данных;
2) - добавления записи
3) - удаления записи
4) - самые дешёвые игрушки до 3 лет
5) - выход и сохранение базы данных
|: 4.
toy(Poni, 100, 2)

```

Рисунок 8. Поиск самой дешевой игрушки.

```

|: 3.
Введите название игрушки:
|: "Lego".
Вы хотите продолжить удаление [y/n]?
|: n.

1) - просмотр содержимого базы данных;
2) - добавления записи
3) - удаления записи
4) - самые дешёвые игрушки до 3 лет
5) - выход и сохранение базы данных
|: 1.
:- dynamic toy/3.

toy("Poni", 100, 2).
toy("Pokemon", 500, 1).

```

Рисунок 9. Удаление записи.

```

Файл  Правка  Формат  Вид  Справка
|:- dynamic toy/3.

toy("Poni", 100, 2).
toy("Pokemon", 500, 1).

```

Рисунок 10. Сохранение базы данных.