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 1

Выполнил:

студент группы ИП-814

Якунин А.В.

ФИО студента

Работу проверил: ассистент кафедры ПМиК

Пащенко А.А.

ФИО преподавателя

Новосибирск 2020 г.

**Содержание:**

1. Задание……………………………….…………………………..………3
2. Листинг программы……………………..………………………………4
3. Демонстрация работы программы…………..………………………....9

# Задание

**1. Списки (списки вводятся с клавиатуры во время выполнения программы)**

1.1. Исходный список преобразуйте в новый "двойной" список таким образом, чтобы каждый элемент исходного списка удвоился. Например, [1,2,3]-> [1,1,2,2,3,3].

**2. Строки, файлы**

2.1. Текстовый файл состоит из нескольких строк, длина каждой из которых меньше 40. Выровняйте строки текста по ширине (равномерным внесением дополнительных пробелов между словами). Длины новых строк должны быть равны 40. Сформируйте новый файл.

**3. Динамические базы данных Напишите программу для работы с базой данных по заданию.**

Начальная база данных должна храниться в файле базы данных. В программе должно присутствовать меню из 5 пунктов, реализующих следующие возможности: − просмотр содержимого базы данных;

− добавления записи (за один вход в этот пункт должна быть возможность добавления нескольких записей);

− удаления записи (за один вход в этот пункт должна быть возможность удаления нескольких записей);

− выполнения запроса к базе данных по заданию;

− выход из программы с сохранением содержимого базы данных в файл базы данных.

3.1. Создайте базу данных об игрушках: название, стоимость. Получите названия всех наиболее дорогих игрушек (цены которых отличаются от самой дорогой не более чем на 100 рублей).

**2. Листинг программы**

**rgr.pl**

doubleList([H|T], List, N):-

New is N - 1,

New > 0, !,

append([H], [H], NewList),

append(List, NewList, ListX2),

doubleList(T, ListX2, New);

append([H], [H], NewList),

append(List, NewList, ListX2),

write(ListX2).

resultWrite(ListX2):-

write(ListX2).

main1(List, ListX2):-

length(List, N),

doubleList(List, ListX2, N).

goal:-

writeln("File"),

read(Filename),

check\_exist(Filename),

open(Filename,read,F), set\_input(F),

read\_file(F,1),

write\_screen(F), close(F).

check\_exist(Filename):-

exists\_file(Filename),!.

check\_exist(\_):-

writeln("error"),fail.

read\_file(\_,\_):-

at\_end\_of\_stream,!, fail.

read\_file(\_,1):- !.

read\_file(F,N):-

read\_line\_to\_codes(F,\_),

N1 is N-1,

read\_file(F,N1).

write\_screen(\_):-

at\_end\_of\_stream,!.

write\_screen(F):-

read\_line\_to\_string(F,L),

string\_to\_list(S,L),

writeln(S),

writekek(L),

write\_screen(F).

check([H|T], L, Add, Spaces):-

string\_length(H,H1),

Al is (H1 + Add + 1 + Spaces),

length(L,N),

N > 1, !,

check(T, T, Al, Spaces);

string\_length(H,H1),

Al is (H1 + Add),

checkSpace(Al, Spaces).

checkSpace(All, Spaces):-

All > 40, !,

Need is All - 40,

write(Need),

write(" "),

write(Spaces),

write(" "),

write(All),

write\_in\_file(Spaces - Need);

All == 40,!,

write\_in\_file(Spaces);

Need is 41 - All,

write(Need),

write\_in\_file(Spaces + Need).

kek([H|T],L, Spaces, LL):-

open("write1.txt", append, Out),

write(Out, H),

close(Out),

length(L,N),

N > 2,!,

write\_in\_file(Spaces),

kek(T,T, Spaces, LL);

length(L,N),

N == 2,

check(LL, LL, 0, Spaces),

kek(T,T, Spaces, LL);

writeln("").

write\_in\_file(Spaces):-

Spaces > 0,!,

open("write1.txt", append, Out),

write(Out, " "),

close(Out),

write\_in\_file(Spaces - 1);

writeln("").

writekek(S):-

string\_length(S,N),

N < 40,!,

split\_string(S, " ", " ", L),

length(L, Space),

NeedSpace is (40 - N) / (Space-1),

kek(L, L, floor(NeedSpace), L),

open("write1.txt", append, Out),

write(Out, "\n"),

close(Out);

open("write1.txt", append, Out),

write(Out,S),

write(Out, "\n"),

close(Out).

:-dynamic maxPrice/1.

domains:-

s = string,

i = integer.

database:-

toys(s,i).

menu:-

consult("toys.ddb"),

repeat,

nl,

writeln("---------------------------------------"),

writeln("1 - Display information about all toys"),

writeln("2 - Input"),

writeln("3 - Delete"),

writeln("4 - The most expensive toys"),

writeln("5 - Exit"),nl,

write("Input: (1-5) "),

read(X), nl, X<6, X > 0,

writeln("---------------------------------------"), nl, process(X), nl, X = 5,!.

process(5).

process(1):-

toys(E,R),

format("[~a ~a]~n",[E, R]),fail.

process(2):-

repeat,

write("Input name toy: "),nl,read(Name),

write("Input price: "),nl,read(Price),

asserta(toys(Name, Price)),

write("Input next? 1/0"),nl,read(Answer),

answerInput(Answer),!,

fail.

process(3):-

write("Input name toy: "),nl,read(Name),

retractall(toys(Name, \_)),

write("Input next? 1/0"),nl,read(Answer),

answerOutput(Answer),!,

fail.

process(4):-

findall(X, toys(\_, X), PriceList),

max\_list(PriceList, MaxList),

retractall(maxPrice(\_)),

asserta(maxPrice(MaxList)),

toys(E,R),

maxPrice(N),

R > N -100,

format("[~a ~a]~n",[E, R]),fail.

answerInput(0).

answerInput(1):- process(2), fail.

answerOutput(0).

answerOutput(1):- process(3), fail.

**40.txt**

content management system

differs from

markup language

engines are

proprietary Some

organize content

Web browser without

process of creation

website landscape.

collaboratively. All that

by hyperlinks, is

evolving, complex,

and networked text,

**toys.ddb**

:-dynamic toys/2.

toys("Herringbone", 200).

toys("Car", 322).

toys("Puzzles", 190).

toys("Ball", 10).

toys("Racket", 50).

toys("Constructor", 490).

toys("Yula", 320).

toys("Locomotive", 600).

toys("Pistol", 650).

toys("Kick scooter", 545).

toys("A bike", 430).

toys("Tank", 210).

toys("Aircraft", 150).

toys("Helicopter", 160).

toys("Soft toy", 70).

toys("Table billiards", 333).

toys("Boomerang", 45).

toys("Table soccer", 350).

toys("Coloring Pages", 100).

toys("Sledge", 110).

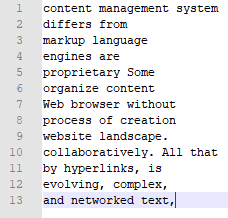
**3. Демонстрация работы программы**

****

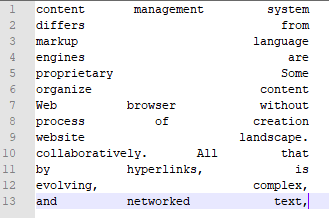
Задание 1.1



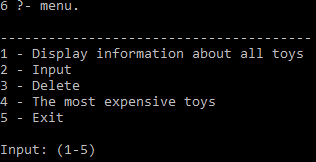
Задание 2.1(Запуск)



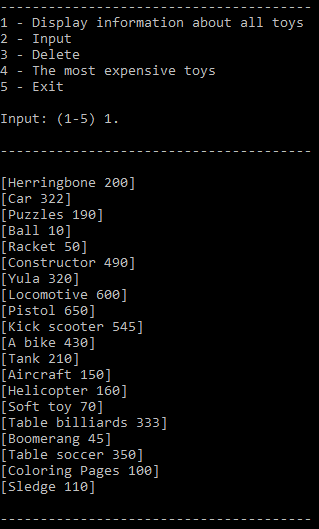
Задание 2.1(Исходный файл “40.txt”)



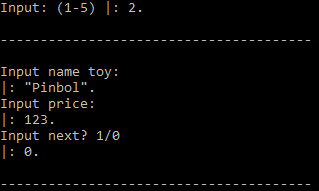
Задание 2.1(Конечный файл “write1.txt”)



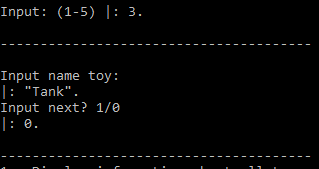
Задание 3.1(Запуск)



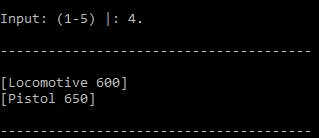
Задание 3.1(Вывод всех элементов базы данных)



Задание 3.1(Добавление нового элемента)



Задание 3.1(Удаление существующего элемента)



Задание 3.1(Вывод всех элементов, цены которых отличаются от самой дорогой не более чем на 100 рублей)