Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ»

Институт системной и программной инженерии

и информационных технологий (СПИНТех)

**Отчёт**

по дисциплине «Функциональное и логическое программирование»

**Лабораторная работа №4**

**Вариант-6**

Руководитель

\_\_\_\_\_\_\_\_\_\_ Петрова A. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

Студент группы ПИН-33

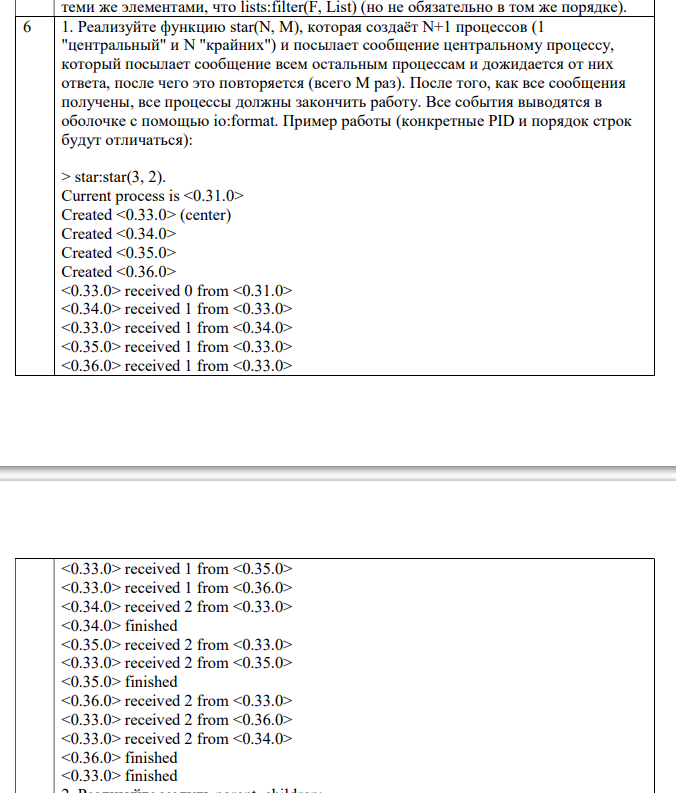
\_\_\_\_\_\_\_\_\_\_ Исламов Р. Р.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

*Москва*

*2024*

# Задание



# 1.1 Листинг

-module(task1).

-export([star/2]).

star(N, M) ->

Self = self(),

io:format("Current process is ~p~n", [Self]),

Center = spawn\_link(fun() -> center(Self, N, M) end),

io:format("Created ~p (center)~n", [Center]),

create\_processes(Center, N).

create\_processes(\_, 0) ->

ok;

create\_processes(Center, N) ->

Pid = spawn\_link(fun() -> process(Center, N) end),

io:format("Created ~p~n", [Pid]),

create\_processes(Center, N - 1).

center(Parent, N, M) ->

receive

{message, From, 0} ->

io:format("~p received 0 from ~p~n", [self(), From]),

Parent ! {reply, self(), 1},

center(Parent, N, M);

{message, From, Num} when Num < M ->

io:format("~p received ~p from ~p~n", [self(), Num, From]),

Parent ! {reply, self(), Num + 1},

center(Parent, N, M);

{message, From, M} ->

io:format("~p received ~p from ~p~n", [self(), M, From]),

Parent ! {reply, self(), M},

ok

end.

process(Center, N) ->

Center ! {message, self(), 0},

receive

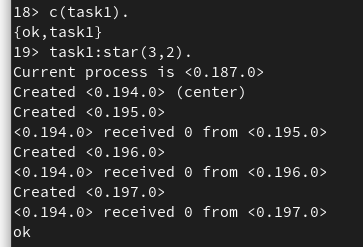
{reply, From, Num} ->

io:format("~p received ~p from ~p~n", [self(), Num, From]),

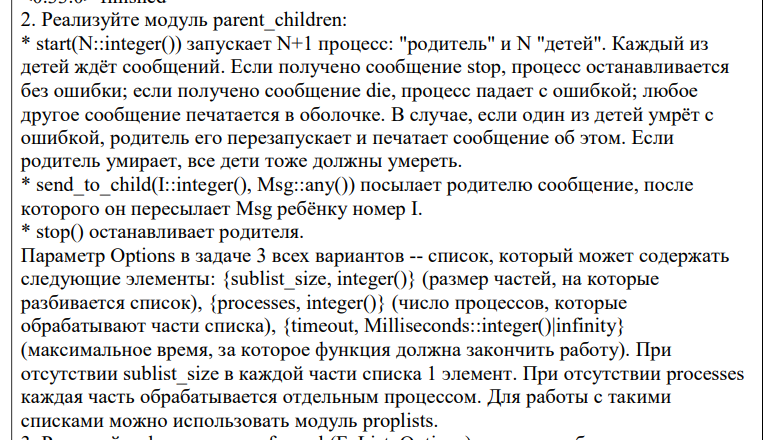
process(Center, N)

end.

# 1.2 Результат работы



# 2.1 Листинг



-module(task2).

-export([start/1, send\_to\_child/2, stop/0]).

start(N) ->

ParentPid = self(),

spawn\_link(fun() -> parent(N, ParentPid) end),

spawn\_children(N).

send\_to\_child(I, Msg) ->

ParentPid = whereis(parent),

ParentPid ! {send\_to\_child, I, Msg}.

stop() ->

ParentPid = whereis(parent),

ParentPid ! stop.

spawn\_children(0) -> ok;

spawn\_children(N) ->

spawn\_link(fun() -> child(N) end),

spawn\_children(N - 1).

parent(N, ParentPid) ->

register(parent, self()),

loop(N, ParentPid, []).

loop(N, ParentPid, Children) ->

receive

{send\_to\_child, I, Msg} when I > 0, I =< length(Children) ->

lists:nth(I, Children) ! Msg,

loop(N, ParentPid, Children);

stop ->

lists:foreach(

fun(ChildPid) -> ChildPid ! stop end,

Children

),

ok;

Msg ->

io:format("Parent received message: ~p~n", [Msg]),

loop(N, ParentPid, [spawn\_link(fun() -> child(N) end) | Children])

end.

child(N) ->

loop(N).

loop(N) ->

receive

stop ->

ok;

die ->

exit(die);

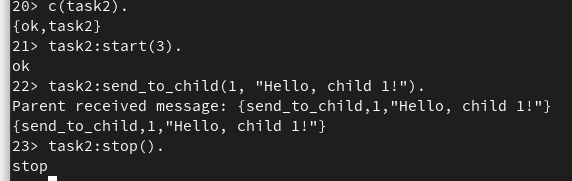
Msg ->

io:format("Child ~p received message: ~p~n", [self(), Msg]),

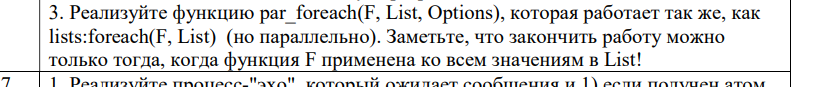
loop(N)

end.

# 2.2 Результат работы



# 3.1 Листинг



-module(task3).

-export([par\_foreach/3]).

-module(task3).

-export([par\_foreach/3]).

par\_foreach(F, List, Options) ->

Pids = [spawn\_link(fun() -> F(Item) end) || Item <- List],

[erlang:monitor(process, Pid) || Pid <- Pids],

%% Ожидаем завершения всех процессов

wait\_for\_pids(Pids).

wait\_for\_pids([]) ->

ok;

wait\_for\_pids(Pids) ->

receive

{'DOWN', \_MonitorRef, process, Pid, Reason} ->

io:format("Process ~p exited with reason: ~p~n", [Pid, Reason]),

wait\_for\_pids(lists:delete(Pid, Pids))

end.

# 3.2 Результат работы

