Университет ИТМО

Кафедра ВТ

Лабораторная работа №4

по ЯСП

Выполнил: Татаринов Д. Ю.

Группа: P3211

Преподаватель: Дергачёв А. М.

Санкт-Петербург

2017

# Задание

Linked List

1. Save them all in a **linked list** *in reverse order*.

2. Write a function to compute the sum of elements in a linked list.

3. Use this function to compute the sum of elements in the saved list.

4. Write a function to output the *n*-th element of the list. If the list is too

short, signal about it.

5. Free the memory allocated for the linked list.

Higher-Order Functions and Lists

• foreach accepts a pointer to the list start and a function (which returns void and

accepts an int). It launches the function on each element of the list.

• map accepts a function *f* and a list. It returns a new list containing the results of the

*f* applied to all elements of the source list. The source list is not affected.

For example, *f* (*x*) = *x* + 1 will map the list (1*,* 2*,* 3) into (2*,* 3*,* 4).

• map\_mut does the same but changes the source list.

• foldl is a bit more complicated. It accepts:

–– The accumulator starting value.

–– A function *f* (*x, a*).

–– A list of elements.

• iterate accepts the initial value *s*, list length *n*, and function *f*.

# Код main.c

#include "stdafx.h"

#include "limits.h"

#include <stdbool.h>

#include "list.h"

#include "func.h"

void foreach\_spaces(const int value) {

printf("%d ", value);

}

void foreach\_new\_line(const int value) {

printf("%d\n", value);

}

int map\_square(const int value){

return value\*value;

}

int map\_cube(const int value) {

return value\*value\*value;

}

int foldel\_sum(int acc, int const value){

acc = acc + value;

return acc;

}

int foldel\_min(int min, int const value){

if (value < min) { min = value;}

return min;

}

int foldel\_max(int max, int const value) {

if (value > max) { max = value; }

return max;

}

int map\_mut\_abs(const int value){

if (value < 0) { return -value;}

return value;

}

int iterate\_double(const int value){

return value \* 2;

}

int main() {

struct linked\_list\* the\_list = list\_create();

int input=0;

printf("Enter the array\n");

char c = ' ';

while (c != '\n') {

scanf("%d%c", &input, &c);

list\_add\_front(input, &the\_list);

}

printf("The summ is: %d\n", list\_sum(\*the\_list));

printf("\nCheck \"foreach\_spaces\"\n");

foreach(the\_list, foreach\_spaces);

printf("\nCheck \"foreach\_new\_line\"\n");

foreach(the\_list, foreach\_new\_line);

printf("\nCheck \"map\_square\"\n");

struct linked\_list cheking\_map\_list = map(the\_list,map\_square);

foreach(&cheking\_map\_list, foreach\_spaces);

printf("\nCheck \"map\_cube\"\n");

cheking\_map\_list = map(the\_list, map\_cube);

foreach(&cheking\_map\_list, foreach\_spaces);

printf("\n\nCheck \"foldel\_sum\"\n");

int acc = foldl(0, the\_list, foldel\_sum);

printf("The summ is: %d\n", acc);

printf("\nCheck \"foldel\_min\"\n");

int min = foldl(INT\_MAX, the\_list, foldel\_min);

printf("The min is: %d\n", min);

printf("\nCheck \"foldel\_max\"\n");

int max = foldl(INT\_MIN, the\_list, foldel\_max);

printf("The max is: %d\n", max);

printf("\nCheck \"map\_mut\_abs\"\n");

map\_mut(the\_list, map\_mut\_abs);

foreach(the\_list, foreach\_spaces);

printf("\n\nCheck \"iterate\_double\"\n");

struct linked\_list\* cheking\_iterate\_list = iterate(1,10,iterate\_double);

foreach(cheking\_iterate\_list, foreach\_spaces);

printf("\n\nCheck \"save\" and \"load\"\n");

save(the\_list,"t.txt");

bool isOk = load(&the\_list, "t.txt");

foreach(the\_list, foreach\_spaces);

printf("\n\nCheck \"serialize\" and \"desrialize\"\n");

struct linked\_list\* new\_list = list\_create();

serialize(the\_list, "ser.txt");

deserialize(&new\_list, "ser.txt");

foreach(new\_list, foreach\_spaces);

list\_free(\*the\_list);

}

# Вывод

Написал простой Linked List.

Научился писать функции высших порядков.

Научился использовать Include Guard.

Научился пользоваться хедерами.

Научился пользоваться функциями fprintf, fscanf, fwrite, fread.