Университет ИТМО, факультет ПИиКТ

Лабораторная работа №6
Дисциплина: Низкоуровневое программирование

Выполнил: Чангалиди Антон

Группа: P33113

Преподаватель: Логинов Иван Павлович

г. Санкт-Петербург

2020 г.

ЛАБОРАТОРНАЯ РАБОТА № 6

MAIN.C

```
#include "list.h"
#include <math.h>
#include <limits.h>

void foreach(void(*func)(int), list *list) {
     while (list != NULL) {
          func(list->value);
          list = list->next;
     }
}

void map_mut(int(*func)(int), list* list) {
     while (list != NULL) {
          list->value = func(list->value);
          list = list->next;
     }
}

list* map(int(*func)(int), list* list) {
     if (list->next != NULL)
          return list_add_front(func(list->value), map(func,
list->next));
     else
          return list_create(func(list->value));
}

list* iterate(int f_val, int length, int(*func)(int)) {
     int i;
     list *new_list = list_create(f_val);
     for (i = 1; i < length; i++) {
          new_list = list_add_front(func(new_list->value), new_list);
     }
     return new_list;
}

int foldl(int accum, int(*func)(int, int), list* list) {
     while (list != NULL) {
          accum = func(accum, list->value);
          list = list->next;
     }
     return accum;
}


int square(int n) {
     if (n < sqrt(INT_MAX))
          return n * n;
     else
          return 0;
```

```
}
int mul_2(int n) {
     if (n < INT_MAX / 2)
          return n * 2;
     else
          return 0;
}
int mul_3(int n) {
     if (n < INT_MAX / 3)
          return n * 3;
     else
          return 0;
}
int sum(int a, int b) {
     if (a < INT_MAX - b)
          return a + b;
     else
          return 0;
}


void printnewline(int n) {
     printf("%d\n", n);
}
void printspace(int n) {
     printf("%d ", n);
}

list* read_list(FILE* fp) {
     int n;
     list *link_list = NULL;
     while (fscanf(fp, "%d", &n) != EOF) {
          if (link_list == NULL)
               link_list = list_create(n);
          else
               link_list = list_add_front(n, link_list);
     }
     return link_list;
}
void printlist(list *list, void(*print_format)(int)) {
     foreach(print_format, list);
}

int main(int argc, char** argv) {
          printf("Input list with spaces:\n");
          list *modif_list, *list = read_list(stdin);
          printf("\n");

          printf("List: ");
          foreach(&printnewline, list);

          printf("Squared:\n");
```

```
        printlist(modif_list = map(&square, list), &printspace);
        free_list(modif_list);
        printf("\n");

        printf("Foldl: %d\n", foldl(0, &sum, list));

        printf("Mul 3: \n");
        map_mut(&mul_3, list);
        printlist(list, &printspace);
        printf("\n");

        printf("Iterations:\n");
        printlist(iterate(2, 10, &mul_2), &printspace);
        printf("\n");

        free_list(modif_list);
        free_list(list);
    return 0;

}
```

LIST.C

```
#include "list.h"

list* list_create(int value) {
     return list_add_front(value, NULL);
}

list* list_add_front(int value, list* link) {
     list* new_node;
     new_node = malloc(sizeof(list));
     new_node->value = value;
     new_node->next = link;
     return new_node;
}

void list_add_back(int value, list* link) {
     list* new_node;


     while (link->next != NULL) {
          link = link->next;
     }

     new_node = malloc(sizeof(link));
     new_node->value = value;
     new_node->next = NULL;
     link->next = new_node;
}

int list_get(list *link, int index) {
     link = list_node_at(link, index);
     if (NULL == link)
          return 0;
     else
          return link->value;
}

void free_list(list*link) {
     list* forFree;
     while (link != NULL) {
          forFree = link;
          link = link->next;
          free(forFree);
     }
}

int list_length(list* link) {
     long length = 0;
     while (link->next != NULL) {
          length++;
          link = link->next;
     }
```

```
    return length;
}

list* list_node_at(list *link, int index) {
    int i = 0;
    while (i < index && link->next != NULL) {
        link = link->next;
        i++;
    }
    if (i == index)
        return link;
    else
        return NULL;
}

long list_sum(list *link) {
    long sum = 0;
    while (link->next != NULL) {
        sum = sum + link->value;
        link = link->next;
    }
    return sum;
}
```

LIST.H

```
#pragma once

#include <stdio.h>
#include <stdlib.h>

typedef struct list {
    int value;
    struct list* next;
} list;

list* list_create(int);
list* list_add_front(int, list*);
void list_add_back(int, list*);
int list_get(list*, int);
void free_list(list*);
int list_length(list*);
list* list_node_at(list*, int);
long list_sum(list*);
```