

Факультет программной инженерии и компьютерной техники

Низкоуровневое программирование

Лабораторная работа №4

Преподаватель:

Логинов Иван Павлович

Выполнил:

Рождественский Никита Сергеевич

Р33113

## Задание:

Реализовать все функции, необходимые для работы со связным списком, из [задания на Stepik](https://stepik.org/lesson/408350/step/8).

 Функции для работы со списком должны быть оформлены в отдельный модуль (файл .c), к нему необходим заголовочный файл с Include guard.

 В основном файле нужно:

* считывать элементы пока возможно (пока не дошли до конца потока)
* добавлять их в начало списка
* вывести сумму списка
* вывести элемент списка по индексу, соответствующему вашему номеру в группе. Если список слишком короткий, то вывести в stderr сообщение об ошибке.
* для всей этой функциональности можно завести сколь угодно много вспомогательных функций.

 Обязательно необходим Makefile в котором каждому артефакту прописаны правильные зависимости (а не просто один таргет main с gcc \*.c).

Выполнение:

struct list {

int64\_t value;

struct list\* next;

};

struct list\* node\_create( int64\_t value ) {

struct list\* const list = malloc( sizeof( struct list ) );

\*list = (struct list) { .value = value, .next = NULL };

return list;

}

// Вывести список

void list\_print(const struct list\* l) {

while (l) {

printf("%" PRId64 " ", l->value);

l = l->next;

}

}

// Освободить память под весь список

void list\_destroy( struct list\* list\_d ) {

while(list\_d){

struct list\* tmp=list\_d->next;

free(list\_d);

list\_d=tmp;

}

}

// добавить элемент в начало списка

// необходимо передавать указатель на указатель, т.к.

// список задаётся указателем на первый элемент

// Добавление элемента в начало это перенаправление

// указателя на свежий элемент.

void list\_add\_front( struct list\*\* old, int64\_t value ) {

struct list \*tmp=malloc(sizeof(struct list));

tmp->value=value;

tmp->next=\*old;

\*old=tmp;

}

// Последний элемент списка

struct list\* list\_last( struct list\* list ) {

struct list\* tmp=list;

while(tmp->next){

tmp=tmp->next;

}

return tmp;

}

// Добавить элемент в конец

// Используйте list\_last

void list\_add\_back( struct list\*\* old, int64\_t value ) {

struct list \*tmp=malloc(sizeof(struct list));

tmp->next=NULL;

tmp->value=value;

list\_last(\*old)->next=tmp;

}

// Посчитать длину списка

size\_t list\_length( struct list const\* list ) {

size\_t size=1;

struct list\* tmp=list->next;

while(tmp){

size++;

tmp=tmp->next;

}

return size;

}

// указатель на структуру по индексу или NULL

struct list\* list\_node\_at( struct list\* list, size\_t idx ) {

struct list\* tmp=list;

for(size\_t i=0;i<idx;i++){

if(tmp->next==NULL) return NULL;

tmp=tmp->next;

}

return tmp;

}

// Вернуть элемент по индексу или none\_int64 если индекс слишком большой

struct maybe\_int64 list\_at(struct list\* list,size\_t idx) {

struct list\* tmp=list\_node\_at(list,idx);

if(tmp==NULL) return none\_int64;

else return some\_int64(tmp->value);

}

// Сумма элементов в списке

struct maybe\_int64 list\_sum( struct list const\* list ) {

int64\_t sum=list->value;

struct list\* tmp=list->next;

while(tmp){

sum+=tmp->value;

tmp=tmp->next;

}

return some\_int64(sum);

}

// Возвращает перевернутую копию списка

struct list\* list\_reverse( const struct list\* list ) {

struct list \*tmp=list->next;

struct list \*list1=malloc( sizeof( struct list ) );

list1->next=NULL;

list1->value=list->value;

while (tmp->next){

list\_add\_front(&list1,tmp->value);

tmp=tmp->next;

}

list\_add\_front(&list1,tmp->value);

return list1;

}

// Прочитать список со входа

struct list\* list\_read() {

struct list\* l;

for (;;) {

const struct maybe\_int64 i = read\_int64();

if (i.valid) {

list\_add\_front( &l, i.value );

}

else {

break;

}

}

struct list\* rev = list\_reverse( l );

list\_destroy( l );

return rev;

}

Вывод: В ходе выполнения данной лабораторной работы была освоена работа со структурами и указателями в языке C, написание макросов на языке препроцессора C, вынесение исходного кода во внешние файлы и использование заголовочных файлов для этого.