**Отчет по лабораторной работе №** 2по курсу “Фундаментальная информатика”

Студент группы М80-103Б-21 Ершова Станислава Григорьевича, № по списку 8

Контакты e-mail, telegram: [stas.ershov57@gmail.com](mailto:stas.ershov57@gmail.com) , @stas\_orel

Работа выполнена: «4» мая 2022г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

Отчет сдан «4» мая 2022 г., итоговая оценка \_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Тема:** **Линейные списки**

1. **Цель работы:**
2. **Задание** (*вариант №* **8**)**: Дополнить список значениями m до размера k**
3. **Оборудование** (студента):

Процессор *Intel Pentium N4200 1.1 ГГц* с ОП *8* Гб, SSD *128* Гб. Монитор *1920x1080*

1. **Программное обеспечение (**студента**):**

Операционная система семейства: *linux*, наименование: *ubuntu*, версия *20.04.3 LTS*

интерпретатор команд: *bash* версия *5.0.17*

Система программирования -- версия --**,** редактор текстов *nano* версия *4.8*

Утилиты операционной системы --

Прикладные системы и программы --

Местонахождение и имена файлов программ и данных на домашнем компьютере --

**6. Идея, метод, алгоритм:**

**7. Сценарий выполнения работы** [план работы,первоначальный текст программы в черновике(можно на отдельном листе)итесты либо соображения по тестированию].

**8. Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем).

MAIN.C:

#include <stdio.h>

#include "circular\_list.h"

int main() {

circular\_list\* l = create\_list();

append\_list(l, 10);

print\_list(l);

delete\_list(l, 10);

print\_list(l);

append\_list(l, 1);

append\_list(l, 2);

append\_list(l, 3);

append\_list(l, 4);

append\_list(l, 5);

print\_list(l);

// delete\_list(l, 1);

delete\_list(l, 5);

// delete\_list(l, 3);

print\_list(l);

insert\_list(l, 1, 8);

insert\_list(l, 2022, 2022);

print\_list(l);

}

CIRCULAR\_LIST.C:

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

#include "circular\_list.h"

circular\_list\* create\_list()

{

circular\_list\* l = malloc(sizeof(circular\_list));

l->genesis\_item = malloc(sizeof(list\_item));

l->genesis\_item->prev = l->genesis\_item;

l->genesis\_item->next = l->genesis\_item;

l->genesis\_item->val = 1991;

l->size = 0;

return l;

}

void append\_list(circular\_list\* l, int val) {

if (l->size == 0)

l->genesis\_item->val = val;

else {

list\_item\* new\_item = malloc(sizeof(list\_item));

new\_item->val = val;

new\_item->next = l->genesis\_item;

if (l->size > 1) {

new\_item->prev = l->genesis\_item->prev;

l->genesis\_item->prev->next = new\_item;

}

else

new\_item->prev = l->genesis\_item;

l->genesis\_item->prev = new\_item;

if (l->size == 1)

l->genesis\_item->next = new\_item;

}

l->size++;

}

void insert\_list(circular\_list\* l, int index, int val) {

if (index >= l->size - 1) {

append\_list(l, val);

return;

}

list\_item\* item = l->genesis\_item;

for (int i = 0; i < size\_of\_list(l); ++i) {

if (i == index) {

list\_item\* new\_item = malloc(sizeof(list\_item));

new\_item->val = val;

new\_item->prev = item;

new\_item->next = item->next;

item->next->prev = new\_item;

item->next = new\_item;

l->size++;

break;

}

item = item->next;

}

}

void delete\_list(circular\_list\* l, int val) {

list\_item\* item = l->genesis\_item;

for (int i = 0; i < l->size; ++i) {

if (item->val == val) {

if (item == l->genesis\_item && l->size == 1) {

l->size = 0;

item->val = 1991;

} else {

item->prev->next = item->next;

item->next->prev = item->prev;

if (item == l->genesis\_item) {

l->genesis\_item = item->next;

}

l->size--;

free(item);

}

break;

}

item = item->next;

}

}

void print\_list(circular\_list\* l) {

int size = l->size;

list\_item\* item = l->genesis\_item;

for (int i = 0; i < size; ++i) {

printf("%d ", item->val);

item = item->next;

}

printf("\n");

}

int size\_of\_list(circular\_list\* l) {

return l->size;

}

void dopolnenie(circular\_list\* l, int val, int k) {

if (size\_of\_list(l) >= k)

return;

else

for (int i = size\_of\_list(l); i < k; ++i)

append\_list(l, val);

}

CIRCULAR\_LIST.H:

#ifndef STACK\_H

#define STACK\_H

#include <stdlib.h>

#include <stdbool.h>

typedef struct list\_item {

int val;

struct list\_item\* next;

struct list\_item\* prev;

} list\_item;

typedef struct circular\_list {

list\_item\* genesis\_item;

int size;

} circular\_list;

circular\_list\* create\_list();

void append\_list(circular\_list\* l, int val);

void insert\_list(circular\_list\* l, int index, int val);

void delete\_list(circular\_list\* l, int val);

void dopolnenie(circular\_list\* l, int val, int k);

int size\_of\_list(circular\_list\* l);

void print\_list(circular\_list\* l);

#endif

**9.Дневник отладки** должен содержать дату и время сеансов отладки и основные события(ошибки в сценарии и программе,нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Замечания автора** по существу работы
2. **Выводы: Узнал, что такое двусвязные списки, до выполнения КП про них не слышал.**

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_