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

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

Контакты e-mail, telegram: 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**

2. **Цель работы:**

3. **Задание** (*вариант №* **8**)**: Дополнить список значениями m до размера k**

4. **Оборудование** (студента):
   Процессор *Intel Pentium N4200 1.1 ГГц* с ОП *8* Гб, SSD *128* Гб. Монитор *1920x1080*

5. **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) и тесты либо соображения по тестированию].

8. Распечатка протокола (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

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. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

11. Выводы: Узнал, что такое двусвязные списки, до выполнения КП про них не слышал.

Подпись студента

_____