Отчет по лабораторной работе № 25,26 по курсу по курсу “Практикум на ЭВМ”

Студент группы **М8О-107Б-20** Чекменев Вячеслав Алексеевич, № по списку 27

Контакты e-mail: chekmenev031@gmail.com, telegram: @suraba03

Работа выполнена: «21» мая 2021 г.

Преподаватель: каф. 806 Найденов Иван Евгеньевич

Отчет сдан « » _____20 ___ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Абстрактные типы данных. Рекурсия. Модульное программирование на языке Си.

2. **Цель работы:** Составить и отладить модуль определений и модуль реализации по заданной схеме модуля определений для абстрактного типа данных. Создать MakeFile

3. **Задание:** (**Вариант 4 и 3**) конкатенация двух списков. Быстрая сортировка Хоара

4. **Оборудование** (студента):

   Процессор *Intel Core i5-8265U* с ОП 7851 Мб, НМД 256 Гб. Монитор *1920x1080*

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

   Операционная система семейства UNIX: linux, наименование: manjaro, версия: 20.1 Mikah
   интерпретатор команд: bash, версия: 5.0.18.
   текстовый редактор: atom, версия: 5.2
   Утилиты операционной системы --
   Прикладные системы и программы --
   Местонахождение и имена файлов программ и данных –

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

1) возьму и слегка изменю файлы list.h и list.c из кп8. Это файлы с функциями для работы со списками на векторе.

2) интерфейс также возьму из кп8 и слегка изменю под 26 ЛР.

3) напишу файлы Qsort.h Qsort.c испольняющие алгоритм быстрой сортировки Хоара.

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

Тестирование (листинг терминала):

тест 1:

```
[suraba04@asusx512fl lab25_26]$ make -f MakeFile run
gcc -c client.c
gcc -c Qsort.c
gcc -c list.c
gcc client.o Qsort.o list.o -o full
./full
Напишите '?' для получения помощи в использовании программы:
?
Набор команд:
c - создать списочек.
b - добавить элемент в конец списка (введите число).
p - напечатать список.
d - удалить элемент по индексу.
a - добавить элемент по индексу.
k - количество элементов в списке.
f - увеличить список до K элементов введенным элементом.
```

l - конкатенация двух списков и вывод отсортированного списка конкатенаций.
e - закончить сеанс.
l1
введите `1`, если хотите создать новые списки.
введите `2`, если хотите использовать свои списки.
введите колво элементов l1: 4
введите элементы списка: 1 4 2 5
введите колво элементов вектора l2: 5
введите элементы списка: 1 24 -1 -353 2
сконкатенированный список:
[1]->[4]->[2]->[5]->[1]->[24]->[-1]->[-353]->[2]
отсортированный список:
[-353]->[-1]->[1]->[1]->[2]->[2]->[4]->[5]->[24]
готово
e
все на сегодня…

тест 2:

[suraba04@asusx512fl lab25_26]$ make -f MakeFile run
gcc -c client.c
gcc -c Qsort.c
gcc -c list.c
gcc client.o Qsort.o list.o -o full
./full
Напишите '?' для получения помощи в использовании программы:
/
?
Набор команд:
c - создать списочек.
b - добавить элемент в конец списка (введите число).
p - напечатать список.
d - удалить элемент по индексу.
a - добавить элемент по индексу.
k - количество элементов в списке.
f - увеличить список до K элементов введенным элементом.
l - конкатенация двух списков и вывод отсортированного списка конкатенаций.
e - закончить сеанс.
l1
введите `1`, если хотите создать новые списки.
введите `2`, если хотите использовать свои списки.
введите колво элементов l1: 4
введите элементы списка: 26 26 -135 135
введите колво элементов вектора l2: 6
введите элементы списка: 135 15 0 395 2
-24
сконкатенированный список:
[26]->[26]->[-135]->[135]->[135]->[15]->[0]->[395]->[2]->[-24]
отсортированный список:
[-135]->[-24]->[0]->[2]->[15]->[26]->[26]->[135]->[135]->[395]
готово
e
все на сегодня...

Тест 3: для пустых списков

[suraba04@asusx512fl lab25_26]$ make -f MakeFile run
./full
Напишите '?' для получения помощи в использовании программы:
l1
введите `1`, если хотите создать новые списки.
введите `2`, если хотите использовать свои списки.
введите колво элементов l1: 0
введите элементы списка: введите колво элементов вектора l2: 0
введите элементы списка: сконкатенированный список:
Список пуст((
отсортированный список:
Список пуст((
готово
e
все на сегодня…

Тест 4: для пустого первого списка

```
[suraba04@asusx512fl lab25_26]$ make -f MakeFile run
./full
Напишите '?' для получения помощи в использовании программы:
l1
введите `1`, если хотите создать новые списки.
введите `2`, если хотите использовать свои списки.
введите колво элементов l1: 0
введите элементы списка: введите колво элементов вектора l2: 2
введите элементы списка: 4 -1
сконкатенированный список:
[4]->[-1]
отсортированный список:
[-1]->[4]
готово
e
все на сегодня...
```

Тест 5: для пустого второго списка

```
[suraba04@asusx512fl lab25_26]$ make -f MakeFile run
./full
Напишите '?' для получения помощи в использовании программы:
l1
введите `1`, если хотите создать новые списки.
введите `2`, если хотите использовать свои списки.
введите колво элементов l1: 4
введите элементы списка: 2 -1 24 2
введите колво элементов вектора l2: 0
введите элементы списка: сконкатенированный список:
[2]->[-1]->[24]->[2]
отсортированный список:
[-1]->[2]->[2]->[24]
готово
e
все на сегодня...
```

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

list.h – заголовочный файл с функциями для работы со списками

```
#ifndef list_h
#define list_h

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

typedef struct list {
    int number_of_elements;
    int capacity;
    int *elements;
} list;

void create_list(list *l, int capacity);
int is_list_empty(list *l);
int size(list *l);
void resize(list *l);
void push_back(list *l, int act);
void print_list(list *l);
void delete_element_by_index(list *l);
void add_by_index(list *l);
int number_of_elements(list *l);
void add_to_K_elements(list *l, int act, int k);
void list_input(list *l);
list *lists_concat(list *l1, list *l2);
```

```
#endif
```

list.c – функции для работы со списками

```
#include <stdlib.h>
#include <stdio.h>
#include "list.h"

void create_list(list *l, int capacity) // нужно
{
    l->capacity = capacity;
    l->number_of_elements = 0;
    l->elements = malloc(sizeof(int) * l->capacity);
}

int is_list_empty(list *l)
{
    if (l->number_of_elements == 0) {
        return 1;
    } else {
        return 0;
    }
}

int size(list *l)
{
    return l->capacity;
}

void resize(list *l)
{
    l->capacity++;
    l->elements = realloc(l->elements, sizeof(int) * l->capacity);
}

void push_back(list *l, int numb) // нужно
{
    if (l->number_of_elements == l->capacity) {
        resize(l);
    }
    l->elements[l->number_of_elements] = numb;
    l->number_of_elements++;
}

void print_list(list *l) // нужно
{
    printf("\n");

    if (is_list_empty(l)) {
        printf("Список пуст((\n");
    }
    for (int i = 0; i < l->number_of_elements; i++) {
        if (i == l->number_of_elements - 1) {
            printf("[%d]\n", l->elements[i]);
        } else {
            printf("[%d]->", l->elements[i]);
        }
    }
}

void delete_element_by_index(list *l) // нужно
{
    int ind;

    printf("Введите индекс элемента, который хотите удалить: ");
    scanf("%d", &ind);

    if (l->number_of_elements == 0) {
        printf("Нечего удалять, список пуст((\n");
        return;
    }
```

```
    while ((ind >= l->number_of_elements) || (ind < 0)) {
        printf("Слишком большой или маленький индекс, введите поменбше/поболбше...\n");
        scanf("%d", &ind);
    }
    l->elements[ind] = 0;

    for (int i = 0; i < l->number_of_elements - 1 - ind; i++) {
        l->elements[i + ind] = l->elements[i + ind + 1];
    }
    l->number_of_elements--;
    printf("удаляем элемент с индексом %d...\n", ind);
}

void add_by_index(list *l) // нужно
{
    if ((l->capacity < l->number_of_elements + 1) && (l->number_of_elements != 0)) {
        resize(l);
    }
    int index, flag = 0;
    int numb, tmp_1, tmp_2;

    printf("Введите какое-нибудь число, которое вы хотите добавить: ");
    scanf("%d", &numb);
    printf("введите индекс, куда хотите добавить: ");
    scanf("%d", &index);

    if (index == l->number_of_elements) {
        push_back(l, numb);
        flag = 1;
    } else {
        while ((index > l->number_of_elements - 1) || (index < 0)) {
            printf("Слишком большой индекс, введите поменбше. максимально возможный индекс = %d\n", l-
>number_of_elements);
            scanf("%d", &index);
            if (index == l->number_of_elements && flag != 1) {
                push_back(l, numb);
                flag == 1;
                return;
            }
        }
        tmp_2 = l->elements[index];
        tmp_1 = numb;

        l->number_of_elements++;

        for (int i = 0; i < l->number_of_elements - index + 1; i++) {
            if (l->capacity < l->number_of_elements + 1) {
                resize(l);
            }
            l->elements[index + i] = tmp_1;
            tmp_1 = tmp_2;
            tmp_2 = l->elements[index + i + 1];
        }
    }
}

int number_of_elements(list *l) // нужно
{
    return l->number_of_elements;
}

void add_to_K_elements(list *l, int numb, int k) // нужно
{
    if (l->number_of_elements < k) {
        while (l->number_of_elements < k) {
            push_back(l, numb);
        }
    }
}

list *lists_concat(list *l1, list *l2)
{
```

```
    for (int i = 0; i < l2->number_of_elements; i++) {
        push_back(l1, l2->elements[i]);
    }
    return l1;
}

void list_input(list *l)
{
    int n, a;
    scanf("%d", &n);
    printf("введите элементы списка: ");

    for (int i = 0; i < n; i++) {
        scanf("%d", &a);
        push_back(l, a);
    }
}
```

Qsort.h – заголовочный файл с процедурами для алгоритма quick sort

---

```
#ifndef _QSORT_
#define _QSORT_

void swap(int* a, int* b);
void qs(int* s_arr, int first, int last);

#endif
```

Qsort.c – процедуры для алгоритма quick sort

```
#include "Qsort.h"
#include <stdio.h>

void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

void qs(int* s_arr, int first, int last)
{
    int i = first, j = last, x = s_arr[(first + last) / 2];

    do {
        while (s_arr[i] < x) i++;
        while (s_arr[j] > x) j--;

        if (i <= j) {
            if (s_arr[i] > s_arr[j]) {
                swap(&s_arr[i], &s_arr[j]);
            }
            i++;
            j--;
        }
    } while (i <= j);

    if (i < last)
        qs(s_arr, i, last);
    if (first < j)
        qs(s_arr, first, j);
}
```

client.c – интерфейс

```
 #include <stdio.h>
#include <stdlib.h>
#include "list.h"
```

```
#include "Qsort.h"

int main()
{
    list l, l1, l2;
    int numb;
    char c;
    int k, flag = 0;

    printf("Напишите '?' для получения помощи в использовании программы:\n");

    while ((c = getchar()) != EOF) {
        if (c == '?') {
            printf("Набор команд:\n");
            printf("c - создать списочек.\n");
            printf("b - добавить элемент в конец списка (введите число).\n");
            printf("p - напечатать список.\n");
            printf("d - удалить элемент по индексу.\n");
            printf("a - добавить элемент по индексу.\n");
            printf("k - количество элементов в списке.\n");
            printf("f - увеличить список до K элементов введенным элементом.\n");
            printf("l - конкатенация двух списков и вывод отсортированного списка конкатенаций.\n");
            printf("e - закончить сеанс.\n");
        }
        else if (c == 'c') {
            printf("создаем списочек\n");
            create_list(&l, 0);
            flag = 1;
            printf("все, список создан, вводите следующую команду\n");
        }
        else if (c == 'b' && flag != 0) {
            printf("введите число: ");
            scanf("%d", &numb);
            printf("добаляем элемент [%d] в конец списка\n", numb);
            push_back(&l, numb);
            printf("все, элемент добавлен в конец, вводите следующую команду\n");
        }
        else if (c == 'p' && flag != 0) {
            printf("печатаем списочек\n");
            print_list(&l);
            printf("все, список напечатан, вводите следующую команду\n");
        }
        else if (c == 'd' && flag != 0) {
            printf("удаляем элемент по индексу\n");
            delete_element_by_index(&l);
            printf("все, элемент удален, вводите следующую команду\n");
        }
        else if (c == 'a' && flag != 0) {
            printf("добавляем элемент по индексу\n");
            add_by_index(&l);
            printf("все, элемент добавлен, вводите следующую команду\n");

        }
        else if (c == 'e') {
            printf("все на сегодня...\n");
            return 0;
        }
        else if (c == 'k' && flag != 0) {
            printf("выводим количество элементов в списке\n");
            printf("%d\n", number_of_elements(&l));
            printf("все, готово, вводите следующую команду\n");
        }
        else if (c == 'f' && flag != 0) {
            printf("увеличиваем список\n");
            printf("введите число: ");
            scanf("%d", &numb);
            printf("введите новое количество элементов: ");
            scanf("%d", &k);
            add_to_K_elements(&l, numb, k);
            printf("все, элементы добавлен, вводите следующую команду\n");
        }
        else if (c == 'l') {
```

```
            printf("введите `1`, если хотите создать новые списки.\n");
            printf("введите `2`, если хотите использовать свои списки.\n");
            char ch;
            scanf("%c", &ch);
            if (ch == '1') {
                create_list(&l1, 0);
                create_list(&l2, 0);
                printf("введите колво элементов l1: ");
                list_input(&l1);
                printf("введите колво элементов вектора l2: ");
                list_input(&l2);
                printf("сконкатенированный список: ");
                list *L = lists_concat(&l1, &l2);
                print_list(L);
                qs(L->elements, 0, L->number_of_elements - 1);
                printf("отсортированный список:");
                print_list(L);
            } else {
                printf("здесь сконкатенированный список: ");
                print_list(lists_concat(&l1, &l2));
                printf("\n");
            }
            printf("готово\n");
        }
        else if (flag == 0 && (c == 'c' || c == 'b' || c == 'p' || c == 'd' || c == 'a' || c == 'e' || c == 'k' || c == 'f' || c == '?') && (c != ' '
&& c != '\n' && c != '\t')) {
            printf("список не создан, используйте команду _c_\n");
        }
    }
    return 0;

}
```

MakeFile

```
all: full

full: client.o Qsort.o list.o
	gcc client.o Qsort.o list.o -o full

client.o: client.c
	gcc -c client.c

Qsort.o: Qsort.c
	gcc -c Qsort.c

list.o: list.c
	gcc -c list.c

run: full
	./full

clean:
	rm -rf *.o full
```

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
25 26	дома	26.05	1: 00	закончил	никаких	

10. Замечания автора по существу работы

11. Выводы

В лабораторной работе я познакомился с быстрой сортировкой Хоара и MakeFile, второе – удобный инструмент для запуска и сборки исхдников. Алгоритм данной сортировки несложный для понимая, написания и исполнения компьютером. Работа была несложной, так как структура и фунции списка у меня остались с КП8. Сортировки имеют огромное значение для программирования в целом. Я уверен, что полученные знания мне пригодятся в будущем.

Подпись студента: