**Отчёт по лабораторной работе №25-26**

по курсу «практикум программирования».

Выполнил студент группы **М8О-114БВ-24**: **Дылдин Сергей Владиславович** № по списку **10**.

Контакты: [**dsergey010206@gmail.com**](mailto:dsergey010206@gmail.com)

Работа выполнена: «28» мая 2025 г.

Преподаватель: **каф. 806 Никулин Сергей Петрович**

Входной контроль знаний с оценкой: **\_\_\_\_\_**

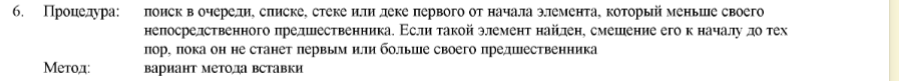
Отчет сдан «30» мая 2025 г.

Итоговая оценка: \_\_\_\_\_

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

1. **Тема:** динамические структуры данных, обработка деревьев.

2. **Цель работы:** составить программу на языке Си для построения и обработки списка, содержащего элементы типа int и создать Makefile.

3. **Задание: 4.Линейный список** .

4. **Оборудование**: *Оборудование ПЭВМ студента, если использовалось:*

Процессор **Intel core i5 12400H**, ОП **16** ГБ, SSD **1**TB.

5. **Программное обеспечение:** *Программное обеспечение ПЭВМ студента, если использовалось****:***

Операционная система семейства **Ubuntu**, наименование версия **VirtualBox Ubuntu 20.04.3 LTS**,

интерпретатор команд **bash** версия **5.0.17**. Система программирования **C**.Редактор текстов **VI** версия **8.1**

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

*Составление программы на Си, выполняющей следующие задачи:*Инициализация списка:  
 При запуске программы вызывается функция list\_init, которая устанавливает указатель на начало списка (head) в NULL, тем самым создавая пустой список.

**Метод:**

* **Добавление и удаление:** При добавлении нового элемента производится сдвиг существующих элементов вправо, чтобы освободить нужную позицию. При удалении — сдвиг элементов влево для заполнения образовавшегося «пробела».
* **Сортировка вставками:** Для упорядочивания списка применяется алгоритм сортировки вставками, при котором элементы перемещаются в нужное место для поддержания возрастающей последовательности.

**Алгоритм сортировки вставками:**

1. **Поиск первого нарушающего порядок элемента:** Функция find\_first\_unsorted проходит по массиву, начиная с заданного индекса, и находит первый элемент, у которого ключ меньше, чем у предыдущего элемента.
2. **Перемещение элемента:** Функция move\_element сохраняет найденный элемент, затем сдвигает элементы с более высокими значениями вправо до тех пор, пока не будет найдено место для вставки сохранённого элемента.
3. **Повторение:** Основная функция insertion\_sort последовательно перебирает элементы списка и, при обнаружении нарушения порядка, вызывает функцию перемещения.

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

linked\_list.h

#ifndef \_\_STATIC\_LIST\_H\_\_

#define \_\_STATIC\_LIST\_H\_\_

#define MAX\_SIZE 100

#define STRING\_LENGTH 50

typedef int key\_type;

typedef char value\_type[STRING\_LENGTH];

typedef struct {

key\_type key;

value\_type value;

} data\_type;

typedef struct {

data\_type data[MAX\_SIZE];

int size;

} StaticList;

void init\_list(StaticList \*list);

int empty(const StaticList \*list);

void insert(StaticList \*list, key\_type key, const char \*value, int index);

void erase(StaticList \*list, int index);

void print\_list(const StaticList \*list);

void clear\_list(StaticList \*list);

int find\_first\_unsorted(const StaticList \*list, int start);

void move\_element(StaticList \*list, int from\_index);

#endif

**oper.c**

#include <stdio.h>

#include <string.h>

#include "linked\_list.h"

void init\_list(StaticList \*list) {

list->size = 0;

}

int empty(const StaticList \*list) {

return (list->size == 0);

}

void insert(StaticList \*list, key\_type key, const char \*value, int index) {

if (index < 0 || index > list->size) {

printf("Invalid index!\n");

return;

}

if (list->size >= MAX\_SIZE) {

printf("List is full!\n");

return;

}

for (int i = list->size; i > index; i--) {

list->data[i] = list->data[i - 1];

}

list->data[index].key = key;

strncpy(list->data[index].value, value, STRING\_LENGTH - 1);

list->data[index].value[STRING\_LENGTH - 1] = '\0';

list->size++;

}

void erase(StaticList \*list, int index) {

if (index < 0 || index >= list->size) {

printf("Invalid index!\n");

return;

}

for (int i = index; i < list->size - 1; i++) {

list->data[i] = list->data[i + 1];

}

list->size--;

}

int find\_first\_unsorted(const StaticList \*list, int start) {

for (int i = start; i < list->size; i++) {

if (i > 0 && list->data[i].key < list->data[i - 1].key) {

return i;

}

}

return -1;

}

data\_type get\_element(const StaticList \*list, int index) {

return list->data[index];

}

void move\_element(StaticList \*list, int from\_index) {

data\_type temp = get\_element(list, from\_index);

erase(list, from\_index);

int pos = 0;

while (pos < list->size && get\_element(list, pos).key <= temp.key) {

pos++;

}

insert(list, temp.key, temp.value, pos);

}

void insertion\_sort(StaticList \*list) {

for (int i = 1; i < list->size; i++) {

int wrong\_index = find\_first\_unsorted(list, i);

if (wrong\_index != -1) {

move\_element(list, wrong\_index);

}

}

}

void print\_list(const StaticList \*list) {

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

printf("(%d: %s) ", list->data[i].key, list->data[i].value);

}

printf("\n");

}

void clear\_list(StaticList \*list) {

list->size = 0;

}

**main.c**

#include "linked\_list.h"

#include <stdio.h>

void insertion\_sort(StaticList \*list);

void showMenu() {

printf("\nМеню:\n");

printf("1. Добавить элемент по индексу\n");

printf("2. Удалить элемент по индексу\n");

printf("3. Отсортировать список\n");

printf("4. Выход\n");

printf("Выберите действие: ");

}

int main() {

StaticList myList;

init\_list(&myList);

int option, key, index;

char value[STRING\_LENGTH];

while (1) {

showMenu();

if (scanf("%d", &option) != 1) break;

switch (option) {

case 1:

printf("Введите ключ, значение и индекс: ");

scanf("%d %s %d", &key, value, &index);

insert(&myList, key, value, index);

print\_list(&myList);

break;

case 2:

printf("Введите индекс для удаления: ");

scanf("%d", &index);

erase(&myList, index);

print\_list(&myList);

break;

case 3:

insertion\_sort(&myList);

print\_list(&myList);

break;

case 4:

clear\_list(&myList);

return 0;

break;

default:

printf("Неверный выбор!\n");

}

}

clear\_list(&myList);

return 0;

}

**Makefile**

CC = gcc

CFLAGS = -Wall -Wextra -std=c11

SRC = main.c oper.c

OBJ = main.o oper.o

EXEC = program.exe

all: $(EXEC)

$(EXEC): $(OBJ)

$(CC) $(CFLAGS) -o $(EXEC) $(OBJ)

main.o: main.c linked\_list.h

$(CC) $(CFLAGS) -c main.c -o main.o

oper.o: oper.c linked\_list.h

$(CC) $(CFLAGS) -c oper.c -o oper.o

clean:

-del /F /Q $(OBJ) $(EXEC) 2> NUL

*Пункты 1-7 отчета составляются строго до начала лабораторной работы.*

*Допущен к выполнению работы.*  **Подпись преподавателя \_\_\_\_\_\_\_**

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

PS D:\apps\labi\2sem\laba4> make

gcc -Wall -Wextra -std=c11 -c main.c -o main.o

gcc -Wall -Wextra -std=c11 -o program.exe main.o oper.o

PS D:\apps\labi\2sem\laba4> ./program

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 1

Введите ключ, значение и индекс: 3 string1 0

(3: string1)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 1

Введите ключ, значение и индекс: 1 string2 1

(3: string1) (1: string2)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 1

Введите ключ, значение и индекс: 5 string3 0

(5: string3) (3: string1) (1: string2)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 3

(1: string2) (3: string1) (5: string3)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 2

Введите индекс для удаления: 1

(1: string2) (5: string3)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 2

Введите индекс для удаления: 1

(1: string2)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 2

Введите индекс для удаления: 0

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 1

Введите ключ, значение и индекс: 2 test2 0

(2: test2)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 1

Введите ключ, значение и индекс: 3 test2 0

(3: test2) (2: test2)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 1

Введите ключ, значение и индекс: 4 test 0

(4: test) (3: test2) (2: test2)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 1

Введите ключ, значение и индекс: 1 tsdf 3

(4: test) (3: test2) (2: test2) (1: tsdf)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 3

(1: tsdf) (2: test2) (3: test2) (4: test)

Меню:

1. Добавить элемент по индексу

2. Удалить элемент по индексу

3. Отсортировать список

4. Выход

Выберите действие: 4

PS D:\apps\labi\2sem\laba4>

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: замечания отсутствуют.

11. **Выводы:** научился работать со структурой данных - линейный список и с сортировкой вставкой

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