**Отчёт по лабораторной работе**№25-26 по курсу 1

студента группы М8О-108Б Попова Матвея, № по списку 18

Адреса www, e-mail, jabber, skype popov.m4tvei@yandex.ru

Работа выполнена: "28" апреля 2020г.

Преподаватель: Трубченко Никита Михайлович.

Входной контроль знаний с оценкой

Отчёт сдан "28" апреля 2021 г., итоговая оценка

Подпись преподавателя

- **Тема**: типы данных

- **Цель работы**: научиться реализовывать типы данных
- **Задание (вариант 18)**: Линейный список, сортировка простой вставкой, вставка элемента с сохранением порядка
- **Оборудование** (*лабораторное*):
  ЭВМ _____, процессор _____, имя узла сети _____ с ОП _____
  МБ
  НМД _____ ГБ. Терминал _____ адрес _____. Принтер _____
  Другие устройства _____

  *Оборудование ПЭВМ студента, если использовалось:*
  Процессор AMD Ryzen 7 4800U, ОП 16 ГБ, НМД 1 ТБ. Монитор _____
  Другие устройства _____

- **Программное обеспечение** (*лабораторное*):
  Операционная система семейства _____, наименование _____ версия _____
  Интерпретатор команд _____ версия _____
  Система программирования _____ версия _____
  Редактор текстов _____ версия _____
  Утилиты операционной системы _____
  Прикладные системы и программы _____
  Местонахождения и имена файлов программ и данных _____

  *Программное обеспечение ЭВМ студента, если использовалось:*
  Операционная система семейства _____, наименование _____ версия
  Интерпретатор команд _____ версия _____
  Система программирования _____ версия _____

Редактор текстов _____ версия _____
Утилиты операционной системы _____
_____
Прикладные системы и программы _____
Местонахождения и имена файлов программ и данных _____
_____

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

main.c

```
#include <stdio.h> //linear list, procedure & method -- 2
#include <stdlib.h>
#include "list.h"
#include "sort.h"

int main()
{
   printf("Operations:\n");
   printf("an to add element n to the list\n");
   printf("fn to find element n in the list\n");
   printf("dn to delete element n from the list\n");
   printf("$n to add element n in the right order\n");
   printf("e to check the list if it is empty\n");
   printf("p to print the list\n");
   printf("s to sort the list\n");
   printf("# to finish.\n");
   char op = 'a';
   int e = 1;
   struct list *L = NULL;
   while (op != '#')
   {
      scanf("%c", &op);
      if (op == 'a')
      {
         scanf("%d", &e);
         L = add(L, e);
      }
      if (op == 'f')
      {
         scanf("%d", &e);
         int r = find(L, e, 0);
         if (r == 0)
            printf("Not found\n");
         if (r == 1)
            printf("Found 1 element\n");
         if (r > 1)
            printf("Found %d elements\n", r);
      }
      if (op == 'd')
      {
         scanf("%d", &e);
         int r = find(L, e, 0);
         for (int i = 0; i < r; i++)
```

```
            L = delete(L, e);
        }
        if (op == '$')
        {
            scanf("%d", &e);
            L = add(L, e);
            L = sort(L, length(L, 0) - 1);
        }
        if (op == 'e')
        {
            if (empty(L) == 0)
                printf("List is empty\n");
            else
                printf("List is not empty\n");
        }
        if (op == 'p')
        {
            output(L);
            printf("\n");
        }
        if (op == 's')
        {
            int f = length(L, 0);
            L = sort(L, f-1);
        }
    }
    L = rem(L);
    return 0;
}
```

list.c

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

/*struct list
{
    int k;
    struct list *next;
}; */

struct list *add(struct list *l, int n)
{
    if (l == NULL)
    {
        struct list *l = malloc(sizeof(struct list));
        l->k = n;
        l->next = NULL;
        return l;
    }
    l->next = add(l->next, n);
}
```

```
int find(struct list *l, int n, int r)
{
   if (l == NULL)
      return r;
   if (l->k == n)
      r++;
   find(l->next, n, r);
}

struct list *delete(struct list *l, int n)
{
   if (l->k == n)
      return l->next;
   if (l == NULL)
      return l;
   if ((l->next)->k == n)
   {
      l->next = (l->next)->next;
      return l;
   }
   delete(l->next, n);
   return l;
}

short int empty(struct list *l)
{
   if (l == NULL)
      return 0;
   else
      return 1;
}

void output(struct list *l)
{
   if (l == NULL)
      return;
   printf("%d ", l->k);
   output(l->next);
}

struct list *rem(struct list *l)
{
   if (l == NULL)
      return l;
   if (l->next == NULL)
   {
      free(l);
      return NULL;
   }
   if (l->next != NULL)
      l->next = rem(l->next);
}
```

list.h

```
#ifndef list_h
#define list_h

struct list
{
   int k;
   struct list *next;
};

struct list *add(struct list *l, int n);

int find(struct list *l, int n, int r);

struct list *delete(struct list *l, int n);

short int empty(struct list *l);

void output(struct list *l);

struct list *rem(struct list *l);

#endif
```

sort.c

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

int length(struct list *l, int r)
{
   if (l == NULL)
      return r;
   r++;
   length(l->next, r);
}

int inf(struct list *l, int min)
{
   if (l == NULL)
      return min;
   if (l->k < min)
      min = l->k;
   inf(l->next, min);
}

struct list *replace(struct list *l, int min, int d)
{
   if (l == NULL)
```

```
        return l;
    if (l->k == min)
    {
        l->k = d;
        return l;
    }
    replace(l->next, min, d);
    return l;
}

struct list *sort(struct list *l, int n)
{
    if (n == 0)
        return l;
    int min = inf(l, l->k);
    l = replace(l, min, l->k);
    l->k = min;
    n--;
    sort(l->next, n);
    return l;
}
```

sort.h

```
#ifndef sort_h
#define sort_h

int length(struct list *l, int r);

int inf(struct list *l, int min);

struct list *replace(struct list *l, int min, int d);

struct list *sort(struct list *l, int n);

#endif
```

Makefile

```
CC ?= gcc
CFLAGS ?= -Werror -pedantic
main: main.o list.o sort.o
	$(CC) -o main main.o list.o sort.o

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

list.o: list.c list.h
	$(CC) $(CFLAGS) -c list.c

sort.o: sort.c sort.h
```

```
	$(CC) $(CFLAGS) -c sort.c

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

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

Допущен к выполнению работы. Подпись преподавателя _____

- **Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)

- **Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) 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 |
|---|---|---|---|---|---|---|
| | | | | | | |

- Замечание автора по существу работы

_____

- Выводы : реализовал тип данных и сортировку
-

Недочеты, допущенные при выполнении задания, могут быть устранены следующим образом _____

Подпись студента _____