Отчет по лабораторной работе № 25, 26 по курсу “Фундаментальная информатика”

Студент группы М80-103Б-21
Хайруллина Ясмин Алмазовна, № по списку 23

Контакты e-mail: jasmin.almazovna@rambler.ru

Работа выполнена: «16» апреля 2022 г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

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 для заданного абстрактного типа данных.

3. **Задание:** Поиск в линейном списке первого от начала элемента, который меньше своего непосредственного предшественника. Если такой элемент найден, то смещение его к началу до тех пор, пока он не станет первым или больше своего предшественника.

4. **Оборудование** (студента):
   AMD Ryzen 7 4700U 2400MHz/15.6"/1920x1080/16G/1T HDD +256G SSD/AMD Radeon Graphics

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

   Операционная система семейства: Linux, наименование: Ubuntu, версия 20.04.3 LTS
   Интерпретатор команд: GNU Bash версия 5.0.17(1)
   Система программирования -- версия --
   Редактор текстов: GNU Emacs версия 26.3
   Утилиты операционной системы: Gnuplot
   Прикладные системы и программы --
   Местонахождение и имена файлов программ и данных на домашнем компьютере --

6. Идея, методы и алгоритмы

Используя различные сторонние источники, написать программу для решения поставленной задачи.

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

Makefile

```
laba: list.o main.o
	gcc list.o main.o
list.o : list.h list.c
	gcc -c list.c
main.o : list.h main.c
	gcc -c main.c
```

list.h

```
#ifndef list_h
#define list_h

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

struct list *add(struct list *l, int n);
struct list *find(struct list *l, int n);
struct list *delete (struct list *l);
int empty(struct list *l);
void print(struct list *l);
struct list *rem(struct list *l);
struct list *replace(struct list *l, int min);
void insert(struct list *after_node, int value);
int minim(struct list *l);
struct list *init(int a);
struct list *deletehead(struct list *root);
void pushFront(struct list *lst, int data);

#endif
```

list.c

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

struct list *deletehead(struct list *root)
{
   struct list *temp;
   temp = root->next;
   temp->prev = NULL;
   free(root); // освобождение памяти текущего корня
   root = NULL;
   return (temp); // новый корень списка
}

struct list *add(struct list *lst, int number) //указатель на узел, после которого происходит добавление, и значение
{
   if (lst->next == NULL)
   {
      struct list *temp, *p;
      temp = (struct list *)malloc(sizeof(struct list));
      p = lst->next;    // сохранение указателя на следующий узел
```

```
        lst->next = temp; // предыдущий узел указывает на создаваемый
        temp->k = number; // сохранение поля данных добавляемого узла
        temp->next = p;   // созданный узел указывает на следующий узел
        temp->prev = lst; // созданный узел указывает на предыдущий узел
        if (p != NULL)
            p->prev = temp;
        return (temp);
    }
    add(lst->next, number);
}
struct list *init(int a) // a- значение первого узла
{
    struct list *lst;
    // выделение памяти под корень списка
    lst = (struct list *)malloc(sizeof(struct list));
    lst->k = a;
    lst->next = NULL; // указатель на следующий узел
    lst->prev = NULL; // указатель на предыдущий узел
    return (lst);
}

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

struct list *delete (struct list *l) //указатель на удаляемый узел
{
    struct list *prev, *next;
    prev = l->prev; // узел, предшествующий lst
    next = l->next; // узел, следующий за lst
    if (prev != NULL)
        prev->next = l->next; // переставляем указатель
    if (next != NULL)
        next->prev = l->prev; // переставляем указатель
    free(l);                  // освобождаем память удаляемого элемента
    return (prev);
}

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

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

```
    struct list *p;
    p = lst;
    do
    {
        printf("%d ", p->k); // вывод значения элемента p
        p = p->next;         // переход к следующему узлу
    } while (p != NULL);     // условие окончания обхода
}

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);
}

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

int minim(struct list *l)
{
    if (l != NULL)
    {
        if (l->k <= l->next->k)
        {
            minim(l->next);
        }
        else
        {
            return l->next->k;
        }
    }
}

void insert(struct list *after_node, int value)
{
    struct list *inserted = (struct list *)malloc(sizeof(struct list));
    inserted->k = value;
    inserted->next = after_node->next;
    after_node->next = inserted;
    inserted->prev = after_node;
    inserted->next->prev = inserted;
}

struct list *replace(struct list *l, int min)
```

```
{
    if (l == NULL)
    {
        return l;
    }
    //struct list *t = find(l, min)->prev;
    // struct list *s = find(l, min)->prev;
    if ((min >= l->prev->k) && (min <= l->next->k) && (l->prev != NULL))
    {
        struct list *t = l->prev;
        delete (find(l, min));
        insert(t, min);
        return l;
    }
    /*if ((min <= l->next->k) && (l->prev == NULL))
    {
        struct list *lst = (struct list *)malloc(sizeof(struct list));
        lst->k = min;
        lst->next = l;    // указатель на следующий узел
        lst->prev = NULL; // указатель на предыдущий узел
        return (lst);
    }*/
    replace(l->prev, min);
    return l;
}
```

main.c

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

int main()
{
    printf("Operations:\n");
    printf("a - to add element n to the list\n");
    printf("d - to delete element n from the list\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);
            if (L == NULL)
            {
                L = init(e);
            }
            else
            {
                add(L, e);
            }
        }
        if (op == 'd')
```

```
        {
            scanf("%d", &e);
            struct list *d = find(L, e);
            if (d->prev == NULL)
            {
                deletehead(d);
            }
            if (d->prev != NULL)
            {
                delete (d);
            }
        }
        if (op == 'e')
        {
            if (empty(L) == 0)
            {
                printf("List is empty\n");
            }
            else
            {
                printf("List is not empty\n");
            }
        }
        if (op == 'p')
        {
            print(L);
            printf("\n");
        }
        if (op == 's')
        {
            if (L != NULL)
            {
                int min = minim(L);
                replace(find(L, min), min);
            }
        }
    }

    L = rem(L);
    return 0;
}
```

8. **Замечания автора** по существу работы

Замечания отсутствуют.

9. **Выводы**

В ходе данной лабораторной работы я познакомилась с новым для меня абстрактным типом данных — линейным списком. Написала программу для сортировки его заданным методом. Работа была полезной, уверена, что полученные знания пригодятся мне в дальнейшем.

Подпись студента

_____