Московский Авиационный Институт
(Национальный Исследовательский Институт)

Факультет прикладной математики

КУРСОВАЯ РАБОТА

по курсам
«Инструментальные средства UNIX, алгоритмы и структуры данных»

Задание VIII
«Линейные списки»

Студент: Пивницкий Д.
Группа: М8о-101Б-19
Преподаватель: Сорокин С.
Оценка: _____
Дата: _____

Москва, 2019

Оглавление

Введение

Составить и отладить программу на языке Си для обработки линейного списка заданной организации с отображением списка на динамические структуры. Навигацию по списку следует реализовать с применением итераторов. Предусмотреть выполнение одного нестандартного и четырех стандартных действий.

Общий метод решения

У нас кольцевой двунаправленный список.
В начале мы задаем список вводя каждый элемент списка.
Можем использовать разные возможности операций над линейным списком (добавление в начало или конец, печать, обновление указанного элемента, удаление элемента или удаление заданного диапазона)
Заданный диапазон удаляется как:

Алгоритм решения задачи выглядит простым. Достаточно перебрать элементы списка и удалить те, которые удовлетворяют условию. Однако при удалении элемента на его место становится следующий, но поскольку мы переходим к следующему элементу, то пропускаем проверку того, что стал на место удаленного. Цикл for использовать нельзя, т. к. меняется количество элементов списка.

Общие сведения о программе

Аппаратное обеспечение: ноутбук Lenovo Z570
ОС: Linux MINT Tessa
Язык и система программирования: GNU C
Стандарт языка: C99
Число строк программы: ~399
Компиляция программы в терминале: gcc main8.c -o kp8
Вызов программы: ./kp8

Функциональное назначение

Задача программы состоит в удалении определенного указанного диапазона значений в кольцевом двунаправленном списке

Описание программы

1. Подключаем библиотеки **stdio.h, stdlib.h**

2. Определим структуру кольцевого двунаправленного списка

3. Ввод с клавиатуры элементов списка

4. Удаление диапазона происходит способом перебора элементов списка и удалить те, которые удовлетворяют условию.

5. Отображаем список

Описание функций программы

Функция/выражение	Описание
add_node();	Добавить узел
insert_at_first();	Вставить в начало
insert_at_end();	Вставить в конец
insert_at_position();	Вставить в позицию
delete_node_position();	Удалить узел
delete_range();	Удалить диапазон значений
update();	Обновить элемент
search();	Найти элемент
display_from_beg();	Отобразить с начала
display_in_rev();	Отобразить с конца

Используемые переменные

Переменная	Тип	Значение
ch	int	Выбор в меню
info	int	Ввод с клавиатуры
pos	int	Позиция
i	int	Индекс, чтобы проходиться по списку
valF	int	Диапазон ОТ
valT	int	Диапазон ДО
count	int	счётчик

Входные значения

Вводятся непосредственно в программе с помощью команд Добавления узла в список, тк исходный список пуст

Протокол

```
(py37) ➜  ~ cd kursach8
(py37) ➜  kursach8 ls
kp8  main8.c
(py37) ➜  kursach8 cat main8.c
#include <stdio.h>
#include <stdlib.h>

struct node
{
   int val;
   struct node *next;
   struct node *prev;
};
typedef struct node n;

n* create_node(int);
void add_node();
void insert_at_first();
void insert_at_end();
void insert_at_position();
void delete_node_position();
void delete_range();
void update();
void search();
void display_from_beg();
void display_in_rev();

n *new, *ptr, *prev;
n *first = NULL, *last = NULL;
int number = 0;

void main()
{
   int ch;

   printf("\nКОЛЬЦЕВОЙ ДВУНАПРАВЛЕННЫЙ СПИСОК\n");
   printf("1.Вставить в начало \n2.Вставить в конец \n3.Вставить в позицию \n 4.Удалить
диапазон значений \n5.Удалить элемент из списка \n6.Обновить элемент \n7.Найти
элемент \n8.Отобразить с начала списка \n9.Отобразить с конца списка \n 10.ВЫХОД
");

   while (1)
```

```
    {

        printf("\nВведите операцию:");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1 :
            insert_at_first();
            break;
        case 2 :
            insert_at_end();
            break;
        case 3 :
            insert_at_position();
            break;
        case 4 :
            delete_range();
            break;
        case 5 :
            delete_node_position();
            break;
        case 6 :
            update();
            break;
        case 7 :
            search();
            break;
        case 8 :
            display_from_beg();
            break;
        case 9 :
            display_in_rev();
            break;
        case 10 :
            exit(0);
        case 11 :
            add_node();
            break;
        default:
            printf("\nНеверная операция");
        }
    }
}

n* create_node(int info)
{
    number++;
    new = (n *)malloc(sizeof(n));
    new->val = info;
    new->next = NULL;
    new->prev = NULL;
```

```
    return new;
}

void add_node()
{

    int info;

    printf("\n   Введите значение которое хотите добавить:");
    scanf("%d", &info);
    new = create_node(info);

    if (first == last && first == NULL)
    {

        first = last = new;
        first->next = last->next = NULL;
        first->prev = last->prev = NULL;
    }
    else
    {
        last->next = new;
        new->prev = last;
        last = new;
        last->next = first;
        first->prev = last;
    }
}

void insert_at_first()
{

    int info;

    printf("\n   Введите значение которое хотите вставить первым:");
    scanf("%d",&info);
    new = create_node(info);

    if (first == last && first == NULL)
    {
        printf("\nИсходный список пуст, вставка произведена");
        first = last = new;
        first->next = last->next = NULL;
        first->prev = last->prev = NULL;
    }
    else
    {
        new->next = first;
        first->prev = new;
        first = new;
        first->prev = last;
```

```
        last->next = first;
        printf("\n3начение вставлено в начало");
    }
}

void insert_at_end()
{

    int info;

    printf("\n   Введите значение которое хотите вставить последним:");
    scanf("%d", &info);
    new = create_node(info);

    if (first == last && first == NULL)
    {
        printf("\nИсходный список пуст и новая вставка была в начало");
        first = last = new;
        first->next = last->next = NULL;
        first->prev = last->prev = NULL;
    }
    else
    {
        last->next = new;
        new->prev = last;
        last = new;
        first->prev = last;
        last->next = first;
    }
}

void insert_at_position()
{
    int info, pos, len = 0, i;
    n *prevnode;

    printf("\n   Введите значение которое хотите вставить:");
    scanf("%d", &info);
    printf("\n   Введите позицию в которую хотите вставить:");
    scanf("%d", &pos);
    new = create_node(info);

    if (first == last && first == NULL)
    {
        if (pos == 1)
        {
            first = last = new;
            first->next = last->next = NULL;
            first->prev = last->prev = NULL;
        }
        else
```

```
        {
            printf("\nСписок пуст, вы не можете вставить в конкретной позиции");
            free(new);
            number--;
        }
    }
    else
    {
        if (number < pos)
        {
            printf("\nЭлемент не может быть вставлен, так как позиция превышает длину
связанного списка");
            free(new);
            number--;
        }

        else
        {
            for (ptr = first, i = 1;i <= number;i++)
            {
                prevnode = ptr;
                ptr = ptr->next;
                if (i == pos-1)
                {
                    prevnode->next = new;
                    new->prev = prevnode;
                    new->next = ptr;
                    ptr->prev = new;
                    printf("\nВставка в позицию %d произведена", pos);
                    break;
                }
            }
        }
    }
}

void delete_range()
{
    int valF, valT, count = 0, i;
    n *temp, *prevnode;

    printf("\n   Введите значение ОТ: ");
    scanf("%d", &valF);

    printf("\n   Введите значение ДО: ");
    scanf("%d", &valT);

    if (first == last && first == NULL)
    {
        printf("\nСписок пуст, невозможно удалить");
        return;
```

```
    }

    n *cur, *next;
    while(first && first != last && first->val >= valF && first->val <= valT)
    {
        last->next = first->next;
        first->next->prev = last;
        cur = first;
        first = first->next;
        free(cur);
        number--;
    }
    while(last && first != last && last->val >= valF && last->val <=valT)
    {
        first->prev = last->prev;
        last->prev->next = first;
        cur = last;
        last = last->prev;
        free(cur);
        number--;
    }
    cur = first->next;
    while(cur && cur != last)
    {
        if(cur->val < valF || cur->val > valT)
        {
            cur = cur->next;
            continue;
        }

        next = cur->next;
        cur->prev->next = next;
        next->prev = cur->prev;
        free(cur);
        cur = next;
        number--;
    }
}

void delete_node_position()
{
    int pos, count = 0, i;
    n *temp, *prevnode;

    printf("\n   Введите позицию которую хотите удалить:");
    scanf("%d", &pos);

    if (first == last && first == NULL)
        printf("\nСписок пуст, невозможно удалить");

    else
```

```
    {
        if (number < pos)
            printf("\nЭлемент не может быть удален, так как позиция превышает длину
связанного списка");

        else
        {
            for (ptr = first,i = 1;i <= number;i++)
            {
                prevnode = ptr;
                ptr = ptr->next;
                if (pos == 1)
                {
                    number--;
                    last->next = prevnode->next;
                    ptr->prev = prevnode->prev;
                    first = ptr;
                    printf("Элемент %d удален", prevnode->val);
                    free(prevnode);
                    break;
                }
                else if (i == pos - 1)
                {
                    number--;
                    prevnode->next = ptr->next;
                    ptr->next->prev = prevnode;
                    printf("Элемент %d удален", ptr->val);
                    free(ptr);
                    break;
                }
            }
        }
    }
}

void update()
{
    int oldval, newval, i, f = 0;
    printf("\n   Введите старое значение:");
    scanf("%d", &oldval);
    printf("\n   Введите новое значение:");
    scanf("%d", &newval);
    if (first == last && first == NULL)
        printf("\nСписок пуст, нет элементов для обновления");
    else
    {
        for (ptr = first, i = 0;i < number;ptr = ptr->next,i++)
        {
            if (ptr->val == oldval)
            {
                ptr->val = newval;
```

```
            printf("Элемент обновлен -> %d", ptr->val);
            f = 1;
         }
      }
      if (f == 0)
         printf("\n no such old value to be get updated");
   }
}

void search()
{
   int count = 0, key, i, f = 0;

   printf("\nenter the value to be searched:");
   scanf("%d", &key);

   if (first == last && first == NULL)
      printf("\nlist is empty no elemnets in list to search");
   else
   {
      for (ptr = first,i = 0;i < number;i++,ptr = ptr->next)
      {
         count++;
         if (ptr->val == key)
         {
            printf("\n the value is found at position at %d", count);
            f = 1;
         }
      }
      if (f == 0)
         printf("\n the value is not found in linkedlist");
   }
}

void display_from_beg()
{
   int i;
   if (first == last && first == NULL)
      printf("\nlist is empty no elemnts to print");
   else
   {
      printf("\n%d number of nodes are there", number);
      for (ptr = first, i = 0;i < number;i++,ptr = ptr->next)
         printf("\n %d", ptr->val);
   }
}

void display_in_rev()
{
   int i;
   if (first == last && first == NULL)
```

```
        printf("\nlist is empty there are no elments");
    else
    {
        for (ptr = last, i = 0;i < number;i++,ptr = ptr->prev)
        {
            printf("\n%d", ptr->val);
        }
    }
}%                                                    (py37) ➜  kursach8 ./kp8
```

КОЛЬЦЕВОЙ ДВУНАПРАВЛЕННЫЙ СПИСОК
1.Вставить в начало
2.Вставить в конец
3.Вставить в позицию
 4.Удалить диапазон значений
5.Удалить элемент из списка
6.Обновить элемент
7.Найти элемент
8.Отобразить с начала списка
9.Отобразить с конца списка
 10.ВЫХОД
Введите операцию:8

list is empty no elemnts to print
Введите операцию:9

list is empty there are no elments
Введите операцию:1

  Введите значение которое хотите вставить первым:3

Исходный список пуст, вставка произведена
Введите операцию:2

  Введите значение которое хотите вставить последним:5

Введите операцию:2

  Введите значение которое хотите вставить последним:7

Введите операцию:2

  Введите значение которое хотите вставить последним:9

Введите операцию:2

  Введите значение которое хотите вставить последним:11

Введите операцию:2

  Введите значение которое хотите вставить последним:18

Введите операцию:2

Введите значение которое хотите вставить последним:25

Введите операцию:8

7 number of nodes are there
3
5
7
9
11
18
25
Введите операцию:4

Введите значение ОТ: 6

Введите значение ДО: 15

Введите операцию:8

4 number of nodes are there
3
5
18
25
Введите операцию:4

Введите значение ОТ: 8

Введите значение ДО: 17

Введите операцию:8

4 number of nodes are there
3
5
18
25
Введите операцию:10
(py37) ➜ kursach8

Заключение

Мы научились работе с линейными списками в языке Си и получили дополнительный опыт в работе с файлами и написанием инструкций для компилятора. В целом после выполнения данной работы я нашёл для себя новые способы работы с файлами с большей эффективностью, поэтому считаю выполненную работу полезной для становления программистом.