Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа**

**по курсу «Языки и методы программирования»**

**ll Семестр**

**Задание 8**

**Линейные списки**

|  |  |
| --- | --- |
| Студент: | Жерлыгин М.А |
| Группа: | М8О-108Б-18 |
| Преподаватель: | Поповкин А.В. |
|  |  |
| Оценка: |  |
| Дата: |  |

# Содержание

[Содержание 2](#_Toc513648674)

[Задание 3](#_Toc513648675)

[Общий метод решения 3](#_Toc513648676)

[Общие сведения о программе 4](#_Toc513648677)

[Функциональное назначение 4](#_Toc513648678)

[Текст программы 4](#_Toc513648679)

[Описание переменных и функций 7](#_Toc513648680)

[Распечатка протокола 7](#_Toc513648681)

[Заключение 10](#_Toc513648683)

[Список литературы 10](#_Toc513648684)

# 1. Задание

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:

1. Печать списка.
2. Вставка нового элемента в список.
3. Удаление элемента из списка.
4. Подсчет длины списка.

*Вариант*

N=5

*Тип элемента списка*: *(определяется как номер\_группы%8+1)*

Целый

*Вид списка:(определяется как (N div 2)%6+1)*

Кольцевой однонаправленный

*Нестандартное действие:(определяется как N%15+1)*

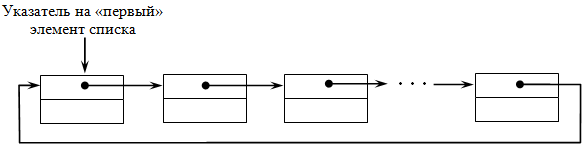
Выполнить попарный обмен значениями элементов списка

# 2. Общий метод решения

Кольцевой список – это структура данных, представляющая собой последовательность элементов, последний элемент которой содержит указатель на первый элемент списка, а первый (в случае двунаправленного списка) – на последний. Основная особенность такой организации состоит в том, что в этом списке нет элементов, содержащих пустые указатели, и, следовательно, нельзя выделить крайние элементы.

Циклический однонаправленный список похож на линейный однонаправленный список, но его последний элемент содержит указатель, связывающий его с первым элементом.

Для полного обхода такого списка достаточно иметь указатель на произвольный элемент, а не на первый, как в линейном однонаправленном списке. Понятие "первого" элемента здесь достаточно условно и не всегда требуется. Хотя иногда бывает полезно выделить некоторый элемент как "первый" путем установки на него специального указателя. Это требуется, например, для предотвращения "зацикливания" при просмотре списка.



# **3.** **Общие сведения о программе**

**Язык и система программирования:** GNU C

**Местонахождение файлов: ~**/2sem/kursach8

# 4. Функциональное назначение

Программа представляет собой программную реализацию такой последовательной структуры данных как кольцевые списки. Кроме того, показаны основные функции для полноценной работы со списками:

* Генерация списка
* Вставка элемента в конец списка
* Удаление элемента
* Удаление списка
* Печать списка

Также, реализовано нестандартное действие - обмен местами произвольных элементов списка.

# 5. Текст программы

***main.c:***

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <math.h>  
   
int size = 0;  
   
typedef struct ring {  
 int value;  
 struct ring\* next;  
} list;  
   
list\* find(list\* X, int index)  
{  
 list\* temp = X;  
 for (int i = 0; i != index; i++)  
 temp = temp->next;  
 return temp;  
}  
   
list\* initList(int val)  
{  
 size++;  
 list\* X = (list\*)malloc(sizeof(list));  
 X->value = val;  
 X->next = X;  
 return X;  
}  
   
list\* addElement(list\* X, int val)  
{  
 if (!X) return initList(val);  
 size++;  
 list \*temp, \*point;  
 temp = (list\*)malloc(sizeof(list));  
 point = X->next;  
 X->next = temp;  
 temp->value = val;  
 temp->next = point;  
 return temp;  
}  
   
list\* delElement(list\* X)  
{  
 size--;  
 list\* temp = X;  
 while (temp->next != X)  
 temp = temp->next;  
 temp->next = X->next;  
 free(X);  
 return temp;  
}  
   
void prnt(list \*X)  
{  
 printf("List:\t");  
 list\* temp = X;  
 do  
 {  
 printf("%d ", temp->value);  
 temp = temp->next;  
 } while (X != temp) ;  
 puts("");  
 return;  
}  
   
void clear(list\* X)  
{  
 if(--size > 0)  
 clear(X->next);  
 if (X) free(X);  
 return;  
}  
   
void change(list\* X, int first, int second)  
{  
 list \*f = X, \*s = X;  
 int temp;  
 f = find(f, first);  
 s = find(s, second);  
   
 temp = f->value;  
 f->value = s->value;  
 s->value = temp;  
 return;  
}  
   
int main()  
{  
 list\* X = NULL;  
 int input, first, second;  
 char q[10];  
 printf(">> ");  
   
 while(scanf("%s %d", q, &input) == 2)  
 {  
 if (q[0] == '+')  
 {  
 X = addElement(X, input);  
 prnt(X->next);  
 }  
 else if (q[0] == '-')  
 {  
 if (size == 0)  
 {  
 puts("List is empty");  
 printf(">> ");  
 continue;  
 }  
 if (X != X->next)  
 {  
 X = delElement(find(X, input));  
 X = find(X, abs(size - input + 1));  
 prnt(X->next);  
 }  
 else  
 {  
 free(X);  
 X = NULL;  
 size = 0;  
 }  
 }  
 else if (q[0] == '~')  
 {  
 if (size == 0)  
 {  
 puts("List is empty");  
 printf(">> ");  
 continue;  
 }  
 puts("Input index to change");  
 scanf("%d %d", &first, &second);  
 change(X, first, second);  
 prnt(X->next);  
 }  
 else break;  
 printf(">> ");  
 }  
   
 clear(X);  
 return 0;  
}

# 6. Описание переменных и функций

***main.c:***

|  |  |  |
| --- | --- | --- |
| **Название и тип**  **функции** | **Аргументы и их тип** | **Описание** |
| ***глобальные переменные*** | **int size** | **Количество элементов списка** |
| ***list\* find*** | **list\* X**  **int index** | **Функция для получения указателя, сдвинутого относительно *X* на**  ***index* позиций** |
| ***list\* initList*** |  | **Функция инициализации списка** |
|  | **int val** | Значение, присваиваемое первому элементу списка |
| ***list\* addElement*** |  | Добавляет элемент в уже существующий список |
|  | **list\* X** | Указатель на список |
| **int index** | Добавляемый элемент |
| ***list\* delElement*** |  | Удаляет элемент из списка |
|  | **list\* X** | Указатель на список |
| ***void prnt*** |  | Выводит все элементы списка на экран |
|  | **list\* X** | Указатель на список |
| ***void clear*** |  | Освобождает память при завершении работы программы |
|  | **list\* X** | Указатель на список |
| ***void change*** |  | Осуществляет нестандартное действие: перестановку элементов |
|  | **list\* X** | Указатель на список |
| **int first** | Позиция первого перемещаемого элемента(относительно главного) |
| **int second** | Позиция второго перемещаемого элемента(относительно главного) |
| ***int main*** |  | Основная функция |
|  | **list\* X** | Список |
| **int input** | Вспомогательная переменная для добавления и удаления элементов |
| **int first** | Вспомогательная переменная для перестановки элементов местами |
| **int second** | **Вспомогательная переменная для перестановки элементов местами** |
| **char\* q** | Вспомогательная переменная для добавления и удаления элементов |

# Распечатка протокола

mmaxim2710@DESKTOP-RDPBU3D:~/2sem/kursach8$ ./main   
>> + 1  
List: 1  
>> + 2  
List: 1 2  
>> - 1  
List: 2  
>> + 3  
List: 2 3  
>> ~ 0  
Input index to change  
1 2  
List: 3 2  
>> - 3  
List: 2  
>> + 2  
List: 2 2  
>> + 4  
List: 2 2 4  
>> a 0  
mmaxim2710@DESKTOP-RDPBU3D:~/2sem/kursach8$ ./main   
>> - 100  
List is empty  
>> + 1  
List: 1  
>> ~ 0 1 2  
Input index to change  
List: 1  
>> ~ 0  
Input index to change  
0 0  
List: 1  
>> - 10  
>> - 2  
List is empty  
>> ~ 0  
List is empty  
>> 1 20  
mmaxim2710@DESKTOP-RDPBU3D:~/2sem/kursach8$ ./main   
>> + 1  
List: 1  
>> + 2  
List: 1 2  
>> + 3  
List: 1 2 3  
>> + 4  
List: 1 2 3 4  
>> + 5  
List: 1 2 3 4 5  
>> - 1  
List: 2 3 4 5  
>> + 1  
List: 2 3 4 5 1  
>> ~ 0  
Input index to change  
1 2  
List: 3 2 4 5 1  
>> ~ 0  
Input index to change  
2 3  
List: 3 4 2 5 1  
>> ~ 0  
Input index to change  
3 4  
List: 3 4 5 2 1  
>> ~ 0  
Input index to change  
1 2  
List: 4 3 5 2 1  
>> ~ 0 2 3  
Input index to change  
List: 4 5 3 2 1  
>> ~ 0  
Input index to change  
1 2  
List: 5 4 3 2 1  
>> ~ 0  
Input index to change  
1 2  
List: 4 5 3 2 1  
>> ~ 0  
Input index to change  
1 1  
List: 4 5 3 2 1  
>> - 2  
List: 4 3 2 1  
>> - 1  
List: 3 2 1  
>> a 0

# 7. Заключение

Списки бывают линейные и кольцевые, с барьерным элементом и без, однонаправленные и двунаправленные.

*Достоинства списков:*

* эффективное (за константное время) добавление и удаление элементов
* размер ограничен только объёмом памяти компьютера и разрядностью указателей.
* динамическое добавление и удаление элементов.

*Недостатки:*

Недостатки связных списков вытекают из их главного свойства — последовательного доступа к данным:

* сложность прямого доступа к элементу, а именно определения физического адреса по его индексу (порядковому номеру) в списке
* на поля-указатели (указатели на следующий и предыдущий элемент) расходуется дополнительная память (в массивах, например, указатели не нужны)
* некоторые операции со списками медленнее, чем с массивами, так как к произвольному элементу списка можно обратиться, только пройдя все предшествующие ему элементы
* над связными списками, по сравнению с массивами, гораздо труднее (хоть и возможно) производить параллельные векторные операции, такие, как вычисление суммы: накладные расходы на перебор элементов снижают эффективность распараллеливания

# Список используемых источников

1. Виртуальная энциклопедия Linux по-русски [Электронный ресурс]. –

<http://rus-linux.net/>

1. Агрегатор гиковских статей [Электронный ресурс]. –

<https://habr.com/post/117050/>

1. Электронная энциклопедия [Электронный ресурс]. –

<https://ru.wikipedia.org/wiki/Dd>

1. Linux Open Source Software Technologies [Электронный ресурс]. –

<https://losst.ru/komanda-dd-linux>