Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский Авиационный Институт»
Национальный Исследовательский Университет

Факультет №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

КУРСОВОЙ ПРОЕКТ

По дисциплине «Практикум на ЭВМ»

Задание: «Линейные списки»

Студент: Хренникова А. С.
Группа: М80-108-19
Преподаватель: Поповкин А. В.
Подпись:
Оценка:
Дата:

Москва, 2020

Содержание

Задача

Составить и отладить программу ну языке Си для обработки линейного списка заданной организации с отображением на массив (только с индексным доступом, без применения ссылок и указателей). Навигацию по списку реализовать с применением итераторов. Предусмотреть выполнение одного нестандартного и четырёх стандартных действий:

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

Тип элемента списка: **литерный**

Вид списка: **линейный двунаправленный с барьерным элементом**

Нестандартное действие: **исключить из списка последние k элементов. Если в списке менее k элементов, то не менять его**.

Общие сведения о списках

Каждый узел двунаправленного (двусвязного) циклического списка (ДЦС) содержит два поля указателей — на следующий и на предыдущий узлы. Указатель на предыдущий узел корня списка содержит адрес последнего узла. Указатель на следующий узел последнего узла содержит адрес корня списка.

Инициализация ДЛС:

Инициализация списка предназначена для создания корневого узла списка, у которого поля указателей на следующий и предыдущий узлы содержат нулевое значение.

Добавление узла в ДЛС включает в себя следующие этапы:

- создание узла добавляемого элемента и заполнение его поля данных;
- переустановка указателя «следующий» узла, предшествующего добавляемому, на добавляемый узел;
- переустановка указателя «предыдущий» узла, следующего за добавляемым, на добавляемый узел;
- установка указателя «следующий» добавляемого узла на следующий узел (тот, на который указывал предшествующий узел);
- установка указателя «предыдущий» добавляемого узла на узел, предшествующий добавляемому (узел, переданный в функцию).

Возвращаемым значением функции является адрес добавленного узла.

Удаление узла ДЛС включает в себя следующие этапы:

- установка указателя «следующий» предыдущего узла на узел, следующий за удаляемым;
- установка указателя «предыдущий» следующего узла на узел, предшествующий удаляемому;
- освобождение памяти удаляемого узла.

Общие сведения о программе

Необходимое программное и аппаратное обеспечение: ОС семейства UNIX (Linux Ubuntu), среда программирования Си (язык Си, компилятор gcc), процессор с 64-битной архитектурой (как на лабораторном компьютере).

Система программирования: GUN C.

Местонахождение файлов на домашнем компьютере: /home/lina_tucha/dir/kp8/kurs8.c. Сам файл компилируется с помощью написания «gcc kurs8.c -o 123» в командной строке интерпретатора команд.

Общий метод решения

Программа должна выводить меню, и в зависимости от запроса, выдавать результат в интерактивном режиме.

Метод решения заключается в написании регулярного структурного типа данных «список». На нём с помощью функций должны быть реализованы действия. Так как по заданию необходимо использовать отображение списка на массив, причём доступ только индексный – без применения ссылок, нужно предусмотреть функцию создания структуры list. В list должно быть шесть полей – для индекса на текущий элемент, для индексов на предыдущий и следующий элементы, для индексов на первый и последний элементы, для размера списка.

Подразумеваются функции вставки элемента в конец списка и перед определенным элементом, удаления элемента из списка по натуральному числу, введенному пользователем, функции печати, очищения и вывода размера списка. Также должна существовать функция для выполнения задания – удаление последних k элементов.

Описание переменных, функций

Так как заданию требуется реализовать список без использования указателей, то в структуре храним не ссылки на элементы, а индексы (следующего/предыдущего).

Основные функции:

Для инициализации списка используется функция

list_create (структура список), которая обнуляет все элементы и индексы внутри структуры.

list_update (список, размер) устанавливает связи в списке или обновляет их, если был добавлен новый элемент.

list_insert_elem (список, позиция, элемент) вставляет элемент в список на указанную позицию.

list_delete_elem(список, позиция) удаляет элемент на указанной позиции.

list_print (список) печатает список.

list_destroy (список) обнуляет все элементы списка.

list_delete_k_elem (список, k) исключает из списка последние k элементов. Если в списке менее k элементов, то не меняет его.

Итераторы:

list_next_elem - индекс следующего элемента.

list_prev_elem - индекс предыдущего элемента.

list_fetch - дает значение элемента по индексу.

list_store - присваивает значение элемента по индексу.

list_first_elem - индекс первого элемента.

list_last_elem - индекс последнего элемента.

list_size - длина списка.

list_empty - проверка на пустоту списка.

Для удаления элемента из списка и вставки элемента необходима функция **list_update**

Функция **list_update** проходится по всему списку и каждому элементу присваивает индексы (для следующего и предыдущего). Если элемент первый,

то она присваивает ему индекс **first_elem**, если последний, то **last_elem**. Также функция подсчитывает одновременно длину списка. **list_update** вызывается автоматически при любом изменении списка, чтобы обновлять данные в структуре.

Переменные:

Таблица 1 – Описание переменных программы

Имя	Тип	Начальное значение	Значение переменной
c	int	0	Счетчик размера списка
i	int	0	Номер элемента(индекс)
max_size	int	Задается пользователем	Максимальный размер списка
act	int		Вводимый пункт меню
position	int		Позиция, на которую вставить элемент(с которой удалить символ)
k	int		Количество элементов, которое нужно удалить
elem	char		Вводимый символ

Входные данные

Программа должна получать с клавиатуры максимальный размер списка, а затем целые числа в зависимости от пунктов меню. Взаимодействие с программой происходит за счёт меню. Ввод производится с клавиатуры, команды меню имеют целочисленные значения (при неправильном вводе (любых других символов, кроме указанных в меню) программа уведомит о несуществовании данной команды).

Входные данные для значений элементов списка должны быть представлены литерным типом. Для нумерации элементов списка должны использоваться натуральные числа.

Выходные данные

Результат работы программы может меняться в зависимости от выбранного пунктов меню:

1) Вставить элемент в конец списка;

2) Вставка элемента перед определенным элементом;

3) Удаление элемента из списка;

4) Исключить из списка последние k элементов. Если в списке менее k элементов, то не менять его;

5) Печать списка;

6) Длина списка;

7) Очистить список.

Листинг программы

```
#include <stdio.h>
#include <stdbool.h>
#include <errno.h>
#include <math.h>
#include <malloc.h>
#include <windows.h>
int max_size;

typedef struct
{
   char num;
   int next_index;
   int prev_index;
   int last_index;
   int first_index;
   int size;
} list;
int list_create(list* l) //Объявление списка
{
   for (int i = 0; i < max_size; i++)
      l[i].num=0;
   l->size=0;
   l->last_index=l->first_index=0;
}
int list_update(list *l, int size)  //Обновление длины списка(если был добавлен какой-либо элемент или удалён, то используется эта функция для обновления связей и длины списка)
{
   int c=0;
   for (int i = 0; i < size; i++)
   {
      c++;
      if (size==1)
      {
         l[i].next_index=l[i].prev_index=i;
         l->first_index=i;
         l->last_index=i;
      }
      else if (i==0)
      {
         l[i].next_index=i+1;
         l[i].prev_index=size-1;
         l->first_index=i;
      }
      else if (i==size-1)
      {
         l[i].next_index=l->first_index;
         l[i].prev_index=i-1;
         l->last_index=i;
      }
      else
      {
         l[i].next_index=i+1;
         l[i].prev_index=i-1;
      }
   }
```

```
   l->size=c;
}
int list_next_elem(list* l, int i) //Итератор следующего элемента
{
   i=l[i].next_index;
   return i;
}
int list_prev_elem(list* l, int i) // Итератор предыдущего элемента
{
   i=l[i].prev_index;
   return i;
}
int list_fetch(list* l, int i) // Итератор, дающий значение по индексу
{
   return l[i].num;
}
void list_store(list* l, int i, char t) // Итератор, присваивающий значение по индексу
{
   l[i].num=t;
}
int list_first_elem(list* l) // Итератор, возвращающий первый элемент
{
   return l->first_index;
}
int list_last_elem(list* l) // Итератор, возвращающий последний элемент
{
   return l->last_index;
}
int list_size(list* l) //Итератор, возвращающий длину списка
{
   return l->size;
}
bool list_empty(list* l) //Итератор, проверяющий на пустоту список
{
   if (l->size==0)
      return true;
   return false;
}
bool list_insert_elem(list *l, int i, char t) //Вставка элемента в любую часть списка(не эффективно, но
просто вставляем элемент в список, на указанное место, а остальное копируется вправо)
{
   if (list_size(l)==max_size)
   {
      printf("\nПривышен максимальный размер списка!\n");
      return false;
   }
   if (i>list_size(l))
   {
      printf("\nЭлемента с таким индексном не существует!\n");
      return false;
   }
   list_update(l, list_size(l)+1);
   if (i==0)
      list_store(l, list_size(l)-1, t);
   else
   {
         int c=i-1;
```

```
            for (int p=list_size(l)-1; p>=i; p--)
               list_store(l, p, list_fetch(l, list_prev_elem(l,p)));
            list_store(l,c,t);
      }
      return true;
}
bool list_delete_elem(list* l, int i) //Удаление элемента из списка (весь список копируется влево на этот элемент)
{
      if (i>list_size(l))
      {
         printf("\nЭлемента с таким индексном не существует!\n");
         return false;
      }
      for (i; i<list_size(l); i++)
         list_store(l,i,list_fetch(l,list_next_elem(l,i)));
      list_store(l, list_size(l)-1, 0);
      list_update(l, list_size(l)-1);
      return true;
}
void list_print(list* l) //Печать списка
{
      printf("\n");
      for (int i = 0; i < list_size(l); i++)
      {
         printf(" %c ", list_fetch(l,i));
      }
      printf("\n");

}

void list_destroy(list *l) //очистить список
{
      for (int i=0; i<max_size; i++)
         list_store(l, i, 0);
      l->size=0;
}
bool list_delete_k_elem(list *l, int k)
{
      if (k>=list_size(l))
      {
         printf("\nСписок слишком маленький\n!");
         return false;
      }
      else
         for (k; k>0; k--)
            list_delete_elem(l, list_last_elem(l));
      return true;
}

void print_menu()
{
      printf("\n1.Вставить элемент в конец списка \n2.Вставка элемента перед определенным элементом.\n3.Удаление элемента из списка.\n4.Удалить последние k элементов.\n5.Печать списка.\n6.Длина списка.\n7.Очистить список.\n\nВыберите действие: ");
}
int main()
```

```
{
   printf("\nmax_size: ");
   scanf("%d", &max_size);
   list l[max_size];
   int act, position, k;
   char elem;
   list_create(l);
   print_menu();
   while(scanf("%d", &act)!=EOF)
   {
      switch(act)
      {
         case 1:
               printf("\nЭлемент: ");
               scanf(" %c", &elem);
               list_insert_elem(l, 0, elem);
              break;
         case 2:
               printf("\nВыберите перед каким элементов вставить нужный(от 1): ");
               scanf("%d", &position);
               printf("\nЭлемент: ");
               scanf(" %c", &elem);
               list_insert_elem(l, position, elem);
         break;
         case 3:
            printf("\nВыберите на какой позици удалить элемент(от 1): ");
            scanf("%d", &position);
            list_delete_elem(l, position-1);
         break;
         case 4:
            printf("\nk:");
            scanf("%d", &k);
            list_delete_k_elem(l, k);
         break;
         case 5:
            list_print(l);
         break;
          case 6:
            printf("\n%d\n",list_size(l));
         break;
         case 7:
            list_destroy(l);
         break;
      }
      print_menu();
   }
   printf("\n");
}
```

Пример работы программы

```
lina_tucha@LAPTOP-44CRFC1U:~/dir/kp8$ gcc kurs8.c -o 123
lina_tucha@LAPTOP-44CRFC1U:~/dir/kp8$ ./123

max_size: 45

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: 4

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: 8

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: g

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: 0

1.Вставить элемент в конец списка
```

2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: z

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: o

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 2

Выберите перед каким элементов вставить нужный(от 1): 3

Элемент: k

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: i

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: 8

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: 8

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 5

4 8 k g 0 z o i 8 8

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 4

k:4

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 5

4 8 k g 0 z

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.

6.Длина списка.
7.Очистить список.

Выберите действие: 3

Выберите на какой позици удалить элемент(от 1): 2

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: 7

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: y

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: L

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 1

Элемент: 9

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.

3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 5

4 k g 0 z 7 y L 9

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 6

9

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 4

k:7

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 5

4 k

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Выберите действие: 6

2

1.Вставить элемент в конец списка
2.Вставка элемента перед определенным элементом.
3.Удаление элемента из списка.
4.Удалить последние k элементов.
5.Печать списка.
6.Длина списка.
7.Очистить список.

Заключение

Цель задания достигнута – реализованы списковая структура для хранения элементов литерного типа. Написаны процедуры и функции для работы с регулярным структурным типом «список». Создана процедура, выполняющее задание – удаление последних k элементов. Действия над списком сопровождаются возможностью вывода его и на экран. Также существует возможность напечатать размер списка. Реализация подобных структур помогает улучшить понимания представления, хранения и обработки данных в ЭВМ.

Список использованных источников

1. РосДиплом, Оформление таблиц в дипломной работе, особенности и требования ГОСТ/Электронный диплом/Режим доступа: https://www.rosdiplom.ru/rd/pubdiplom/view.aspx?id=288
2. Диплом Журнал, Оформление курсовой работы по ГОСТу 2019(образец)/Электронный диплом/Режим доступа: https://journal.duplom.ru/kursovaya/oformlenie-kursovoj-raboty-po-gostu-2019-obrazec/
3. Vyuchit.work – универсальная методичка/Электронный диплом/Режим доступа: https://vyuchit.work/samorazvitie/sekretyi/oformlenie-risunkov-po-gostu.html
4. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования С, 2-е издание. :Пер. с англ. – М. : Издательский дом «Вильямс», 2009. – 304 с. : ил. – Парал. тит. англ.
5. Программирование, двусвязный линейный список/Электронный диплом/Режим доступа: https://prog-cpp.ru/data-dls/