Федеральное государственное бюджетное образовательное учреждение
высшего образования «Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт № 8
«Информационные технологии и прикладная математика»

Курсовой проект

по курсу «Вычислительные системы»

Семестр 2

Задание 8

Студент: Хайруллина Ясмин Алмазовна

Группа: М8О-103Б-21

Руководитель: Севастьянов Виктор Сергеевич

Дата сдачи: 14.05.22

Москва, 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная курсовая работа направлена на изучение обработки линейного списка на языке программирования Си. В ходе работы полученные с помощью предоставленной информации и самостоятельного изучения материала будет составлена программа для решения предложенной задачи.

ЗАДАЧИ

1. Изучить материал по данной теме, поискать дополнительную информацию в сторонних источниках.
2. Составить программу обработки линейного списка заданной организации с отображением на динамические структуры.
3. Производить работу в режиме меню.
4. Предоставить отчет.

ФОРМУЛИРОВКА ЗАДАНИЯ

Составить программу на языке Си для обработки линейного списка заданной организации с отображением на динамические структуры. Навигацию по списку следует организовать с помощью итераторов. Предусмотреть выполнение одного нестандартного действия и четырех стандартных:

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

Нестандартное действие: дополнить список копиями заданного значения до указанной длины k, если в списке уже имеется k элементов, то не менять его.

Тип элемента списка: строковый.

Вид списка: линейный двунаправленный список с барьерным элементом.

АЛГОРИТМ ПРОГРАММЫ

1. Реализовать итератор и линейный двунаправленный список с барьерным элементом.
2. Написать функции для вставки, удаления элементов списка.
3. Написать функцию для вывода списка.
4. Написать функцию для подсчета длины списка.
5. Написать функцию для выполнения нестандартного действия.

ОПИСАНИЕ ПРОГРАММЫ

Выходные данные: список.

Используемые структуры:

1. list – структура списка с указателем на head
2. listIterator – структура итератора
3. list_node – структура узла списка

Файл Makefile

```
laba: list.o kp8.o
	gcc list.o kp8.o
list.o : list.h list.c
	gcc -c list.c
kp8.o : list.h kp8.c
	gcc -c kp8.c
```

Файл list.h

```
#ifndef LIST_H
#define LIST_H

typedef int item;
typedef struct _list_node list_node;
typedef struct _list_node
{
   char data[100];
   list_node *next;
   list_node *prev;
} list_node;
typedef struct listIterator
{
   list_node *node;
} listIterator;

typedef struct
{
   list_node *head;
} list;

list *listCreate();
void listPrint(list *l);
void listInsert(list *lst, char *data);
void listRemove(list *lst, char *data);
int listLen(list *lst);
void listInsertToK(list *lst, int k, char *data);
listIterator *iteratorCreate(list *lst);
void iteratorNext(listIterator *it);
```

```
list_node *iteratorGet(listIterator *it);
void iteratorSet(list_node *l, listIterator *it);
void listInsertb(list *lst, char *data);

#endif
```

Файл list.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list7.h"

listIterator *iteratorCreate(list *lst)
{
   if (lst != NULL)
   {
      listIterator *it = (listIterator *)malloc(sizeof(listIterator));
      it->node = lst->head;
      return it;
   }
   else
      return NULL;
}

void iteratorNext(listIterator *it)
{
   if (it->node == NULL)
   {
      printf("Конец списка\n");
   }
   else
   {
      it->node = it->node->next;
   }
}

list_node *iteratorGet(listIterator *it)
{
   return it->node;
}
```

```
void iteratorSet(list_node *lst, listIterator *it)
{
   it->node = lst;
}

list *listCreate()
{
   list *lst = (list *)malloc(sizeof(list));
   lst->head = (list_node *)malloc(sizeof(list_node));
   lst->head->next = NULL;
   lst->head->prev = NULL; //
   strcpy(lst->head->data, "BARRIER");
   return lst;
}

void listPrint(list *lst)
{
   listIterator *it = iteratorCreate(lst);
   if (it->node)
   {
      while (it->node != NULL)
      {
         if (it->node->data != "BARRIER")
         {
            printf("%s", it->node->data);
            if (it->node->next != NULL)
               printf(" -> ");
         }
         //  enum_out(it->node->data);
         it->node = it->node->next;
      }
      // printf("BARRIER");
      printf("\n");
   }
}

void listInsert(list *lst, char *data) // в конец листа
{
   listIterator *it = iteratorCreate(lst);
   if (it->node)
   {
      while (it->node->next)
      {
```

```
            iteratorNext(it);
        }
        list_node *tail = (list_node *)malloc(sizeof(list_node));
        tail->next = NULL;
        strcpy(tail->data, "BARRIER");
        strcpy(it->node->data, data);
        it->node->next = tail;
        tail->prev = it->node;
    }
    else
        printf("Список не существует\n");
    free(it);
}

void listInsertb(list *lst, char *data) // в конец листа
{
    listIterator *it = iteratorCreate(lst);
    if (it->node)
    {
        list_node *tail = (list_node *)malloc(sizeof(list_node));
        tail->next = it->node;
        tail->prev = NULL;
        strcpy(tail->data, data);
        it->node->prev = tail;
        lst->head = tail;
    }
    else
        printf("Список не существует\n");
    free(it);
}

/*list *find(list *lst, char *data)
{
    listIterator *it = iteratorCreate(lst);
    if (strncmp(it->node->data, data, 10) == 0)
    {
        return lst;
    }
    find(lst->head->next, data);
}*/

void listRemove(list *lst, char *data) // удаление узла листа
{
```

```
    int flag = 0;
    listIterator *it = iteratorCreate(lst);
    list_node *prew;
    char *f;
    if (it->node)
    {
        if (strncmp(it->node->data, data, 10) == 0) // если значение узла = значению
удаляемого узла
        {
            lst->head = it->node->next;
            free(it->node);
            it->node = NULL;
        }
        else
        {
            while (it->node->next) // пока существует следующий узел
            {
                if (strncmp(it->node->next->data, data, 10) == 0) // если значение этого
самого следующего узла = искомому
                {
                    prew = it->node;
                    iteratorNext(it);
                    flag = 1;
                    if (it->node->next != NULL)
                    {
                        prew->next = it->node->next;
                        it->node->next->prev = prew->next;
                    }
                    free(it->node);
                    it->node = NULL;
                    break;
                }
                iteratorNext(it);
            }
            if (!flag)
                printf("Элемента нет в списке\n");
            else
                printf("Элемент удалён\n");
        }
    }
    else
        printf("Список не существует\n");
    free(it);
```

```
}

int listLen(list *lst) // длина списка
{
   int count = 0;
   listIterator *it = iteratorCreate(lst);
   iteratorSet(lst->head, it);
   if (it->node)
   {
      while (it->node->next && it->node->next->data != "BARRIER")
      {
         count++;
         iteratorNext(it);
      }
   }
   else
      printf("The list doesn't exist!\n");
   free(it);
   //count++;
   return count;
}

void listInsertToK(list *lst, int k, char *data) // функция сама
{
   while (listLen(lst) < k)
   {
      listInsert(lst, data);
   }
}
```

Файл kp8.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list7.h"

void menu()
{
   printf("\nМеню:\n");
   printf("0) Добавить элемент в начало списка\n");
   printf("1) Добавить элемент в конец списка\n");
```

```
    printf("2) Удалить элемент из списка\n");
    printf("3) Вывести список\n");
    printf("4) Вывести длину списка\n");
    printf("5) Добавить в список копии одного элемента до длины списка k\n");
    printf("6) Меню\n");
    printf("7) Выход\n");
}

int main(void)
{

    char *data;
    list *lst = listCreate();
    listIterator *it = iteratorCreate(lst);
    char c;
    menu();
    while (1)
    {
        scanf("%c", &c);
        iteratorSet(lst->head, it);
        if (c == '\n' || c == ' ')
            continue;
        switch (c)
        {
        case '0':
            printf("Для того, чтобы добавить элемент, введите:\n");
            scanf("%s", data);
            listInsertb(lst, data);
            printf("Элемент добавлен\n");
            break;
        case '1':
            printf("Для того, чтобы добавить элемент, введите:\n");
            scanf("%s", data);
            listInsert(lst, data);
            printf("Элемент добавлен\n");
            break;
        case '2':
            printf("Для того, чтобы удалить элемент, введите:\n");
            scanf("%s", data);
            listRemove(lst, data);
            break;
        case '3':
            listPrint(lst);
```

```
            break;
        case '4':
            printf("Длина списка: %d\n", listLen(lst));
            break;
        case '5':
            printf("Введите k - длину итогового списка:\n");
            int k;
            scanf("%d", &k);
            printf("Введите элемент, которым вы хотите дополнить список до длины
%d:\n", k);
            scanf("%s", data);
            listInsertToK(lst, k, data);
            printf("Нужное количество элементов добавлено\n");
            break;
        case '6':
            menu();
            break;
        case '7':
            free(it);
            free(lst);
            return 0;
        default:
            printf("Действия с таким номером не существует, выберите команду из
меню\n");
            menu();
            break;
        }
    }
}
```

РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

```
jasmin@ubuntu:~$ make
gcc list.o kp8.o
jasmin@ubuntu:~$ ./a.out

Меню:
0) Добавить элемент в начало списка
1) Добавить элемент в конец списка
2) Удалить элемент из списка
3) Вывести список
4) Вывести длину списка
5) Добавить в список копии одного элемента до длины списка k
6) Меню
7) Выход
1
Для того, чтобы добавить элемент, введите:
a
Элемент добавлен
1
Для того, чтобы добавить элемент, введите:
b
Элемент добавлен
1
Для того, чтобы добавить элемент, введите:
c
Элемент добавлен
0
Для того, чтобы добавить элемент, введите:
s
Элемент добавлен
3
```

s -> a -> b -> c -> BARRIER

2

Для того, чтобы удалить элемент, введите:

c

Элемент удалён

3

s -> a -> b -> BARRIER

4

Длина списка: 3

5

Введите k - длину итогового списка:

6

Введите элемент, которым вы хотите дополнить список до длины 6:

h

Нужное количество элементов добавлено

3

s -> a -> b -> h -> h -> h -> BARRIER

4

Длина списка: 6

7

jasmin@ubuntu:~$

ЗАКЛЮЧЕНИЕ

В ходе данной работы была составлена программа на языке Си для обработки линейного двунаправленного списка с барьерным элементом. Вновь проработаны знания о данной структуре. Проделана работа с итераторами. Полученные знания, практика и опыт в поиски нужной информации в сторонних источниках привнесли большой вклад в мое развитие и будут помогать мне в дальнейшей моей работе.