МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Московский авиационный институт

(национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра 806: «Вычислительная математика и программирование»

Курсовой проект

по курсу «Вычислительные системы»

I I семестр

Задание 8. «Линейные списки.»

|  |  |
| --- | --- |
| Группа: | М8о-107б-18 |
| Студент: | Тояков Артем |
| Преподаватель: | Ридли Александра Николаевна |
| Оценка: |  |
| Дата: |  |

Оглавление

[1.Постановка задачи 3](#_Toc12250953)

[2.Структура программы 3](#_Toc12250954)

[3.Описание программы 3](#_Toc12250955)

[4.Результат работы 7](#_Toc12250956)

[5.Вывод 9](#_Toc12250957)

[6.Список литературы 9](#_Toc12250958)

# 1.Постановка задачи

Составить программу на языке СИ с функциями или процедурами для обработки линейного списка.

Вариант: 1,9.

Тип данных: int;

Тип списка: Кольцевой односвязный список.

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

# 2.Структура программы

В данной работе моя программа состоит из четырех модулей:

1. main.c
2. Makefile
3. list.c
4. list.h

В list.c содержатся функции для работы со списком. В list.h содержатся структуры и заголовки функций. Main.c предлагает меню для пользователя и выполняет функции, прописанные в list.c.

# 

# 3.Описание программы

3.1 list.h

В данном разделе хранится структура моего списка.

#ifndef \_LIST\_H\_

#define \_LIST\_H\_

#include <stdio.h>

#include <stdlib.h>

typedef struct list {

struct list \*next;

int info;

} list;

void list\_init(list \*\*head, int info);

void list\_push(list \*\*head, int info);

void list\_pop(list \*\*head);

void list\_print(list \*head);

int list\_size(list \*head);

void list\_delete\_k\_elements(list \*\*head, int count\_of\_elemnts);

#endif //\_LIST\_H\_

3.2 Makefile

Довольно простой Makefile состоящий из опций make и make clean.

Используются флаги:-g и -Wall

CC = gcc

CCFLAGS = -g -Wall

OBJ = list.o main.o

.SUFFIXES: .c .o

main: $(OBJ)

@$(CC) $(CCFLAGS) $(OBJ) -o main

main.o: list.o

$(CC) -c $(CCFLAGS) main.c

list.o: list.c

$(CC) -c $(CCFLAGS) list.c

clean:

@rm -f \*.o \*.gch main

3.3 Main.c

Main.c содержит в себе меню switch, в котором присутствует пять 7 операций.Вначале я инициализирую структуру своего списка,а потом запускается диалоговое окно. Из основных присутствуют такие операции как:

добавление, удаление элемента в список.

#include <stdio.h>

#include <stdlib.h>

#include "list.h"

void menu() {

printf("0-menu, 1-push, 2-pop, 3-print, 4-size, 5-function, 6-exit\n");

}

int main() {

int info, temp, count\_of\_elements;

printf("enter info of first elem: ");

scanf("%d", &info);

list \*root;

list\_init(&root, info);

while(temp != 6) {

printf("\nEnter your option: ");

scanf("%d", &temp);

printf("\n");

switch (temp) {

case 0:

menu();

break;

case 1:

printf("Enter the info of the node: ");

scanf("%d", &info);

if (list\_size(root) > 0) {

list\_push(&root, info);

} else {

list\_init(&root, info);

}

break;

case 2:

list\_pop(&root);

break;

case 3:

if (list\_size(root) == 0) {

printf("The list is empty!\n");

break;

}

list\_print(root);

printf("\n");

break;

case 4:

printf("Size of list = %d\n", list\_size(root));

break;

case 5:

printf("Enter count:");

scanf("%d", &count\_of\_elements);

list\_delete\_k\_elements(&root, count\_of\_elements);

break;

case 6:

printf("Exiting...\n");

return 0;

break;

}

}

return 0;

}

3.4 List.c

В данном файле содержатся функции, работающие с самим списком. В первую очередь присутствует инициализация самого списка, а также его удаление. Из стандартных присутствуют добавление и удаление. Я реализовал добавление и удаление последнего элемента, т. к. это удобно для выполнения моей функции, которую я также написал в этом файле.

#include "list.h"

void list\_init(list \*\*root, int info) {

\*root = (list\*) malloc (sizeof(list));

(\*root)->info = info;

(\*root)->next = \*root;

}

void list\_push(list \*\*root, int info) {

list \*temp\_add, \*temp\_go\_to\_end;

temp\_add = (list\*) malloc(sizeof(list));

temp\_go\_to\_end = \*root;

while(temp\_go\_to\_end->next != \*root) {

temp\_go\_to\_end = temp\_go\_to\_end->next;

}

temp\_go\_to\_end->next = temp\_add;

temp\_add->info = info;

temp\_add->next = \*root;

}

void list\_pop(list \*\*root) {

list \*temp\_go\_to\_end, \*temp\_popping;

temp\_go\_to\_end = \*root;

while(temp\_go\_to\_end->next->next != \*root) {

temp\_go\_to\_end = temp\_go\_to\_end->next;

}

temp\_popping = temp\_go\_to\_end->next;

temp\_go\_to\_end->next = \*root;

if (\*root == temp\_popping) {

\*root = NULL;

}

free(temp\_popping);

}

void list\_print(list \*root) {

if (root == NULL) {

printf("\n");

return;

} else {

printf("%d ", root->info);

list \*temp\_go\_to\_end;

temp\_go\_to\_end = root;

while(temp\_go\_to\_end->next != root) {

temp\_go\_to\_end = temp\_go\_to\_end->next;

printf("%d ", temp\_go\_to\_end->info);

}

}

}

int list\_size(list \*root) {

if (root == NULL) {

return 0;

} else {

int size = 1;

list \*temp\_go\_to\_end;

temp\_go\_to\_end = root;

while(temp\_go\_to\_end->next != root) {

temp\_go\_to\_end = temp\_go\_to\_end->next;

size++;

}

return size;

}

}

void list\_delete\_k\_elements(list \*\*root, int count\_of\_elements) {

if (list\_size(\*root) < count\_of\_elements) {

return;

}

for (int i = 0; i < count\_of\_elements; i++) {

list\_pop(root);

}

}

# 4.Результат работы

➜ kp8 git:(master) make

gcc -c -g -Wall list.c

gcc -c -g -Wall main.c

➜ kp8 git:(master) ✗ ./main

enter info of first elem: 3

Enter your option: 0

0-menu, 1-push, 2-pop, 3-print, 4-size, 5-function, 6-exit

Enter your option: 1

Enter the info of the node: 23

Enter your option: 1

Enter the info of the node: 6

Enter your option: 1

Enter the info of the node: 12

Enter your option: 1

Enter the info of the node: 0

Enter your option: 0

0-menu, 1-push, 2-pop, 3-print, 4-size, 5-function, 6-exit

Enter your option: 3

3 23 6 12 0

Enter your option: 4

Size of list = 5

Enter your option: 5

Enter count:3

Enter your option: 3

3 23

Enter your option: 5

Enter count:3

Enter your option: 3

3 23

Enter your option: 2

Enter your option: 2

Enter your option: 3

The list is empty!

Enter your option: 6

Exiting...

# 5.Вывод

Линейный список имеет как плюсы, так и минусы. Также количество видов линейного списка насчитывает порядка шести разных представлений. Основным плюсом конечно же является эффективное добавление и удаление элементов. Также, размер ограничен только объёмом памяти компьютера и разрядностью указателей. Но основные минусы списка вытекают из одного свойства: последовательный доступ к данным. В динамическом массиве, например, можно получить любой элемент гораздо быстрее чем в линейном списке. Исходя из этого и выявляются такие минусы как:

1. сложность прямого доступа к элементу, а именно определения физического адреса по его индексу (порядковому номеру) в списке
2. на поля-указатели (указатели на следующий и предыдущий элемент) расходуется дополнительная память (в массивах, например, указатели не нужны)
3. некоторые операции со списками медленнее, чем с массивами, так как к произвольному элементу списка можно обратиться, только пройдя все предшествующие ему элементы и т. д.

# 6.Список литературы

1.Гайсарян С.С., Зайцев В.Е. «Курс информатики» Москва, Издательство