|  |  |
| --- | --- |
| + | **Отчёт по лабораторной работе** №VIII  по курсу 1 фундаментальная информатика  студента группы М8О-105Б-21 Бондаревой Елены Евгеньевны, № по списку 1  Контакты www, e-mail, icq, skype : lena\_bondareva\_03@mail.ru  Работа выполнена: «3» мая 2022г.  Преподаватель: каф.806 В.К.Титов  Входной контроль знаний с оценкой  Отчёт сдан , итоговая оценка  Подпись преподавателя |

**1. Тема**: «Линейные списки»

**2. Цель работы**: Составить и отладить программу на языке Си для обработки линейного списка заданной организации с отображением списка на динамические структуры. Предусмотреть выполнение одного нестандартного и стандартных действий. (станд. действия: печать списка, вставка нового элемента в список, удаление элемента из списка, подсчет длины списка..).

**3. Задание** (вариант: 1*)*

Вид списка: кольцевой однонаправленный

Тип элемента списка: вещественный

Нестандартное действие: очистить список, если в нем есть элемент, равный заданному значению.

**4. Оборудование** (*лабораторное*):

ЭВМ -, процессор -, имя узла сети-с ОП -МБ

НМД -ГБ. Терминал - адрес -. Принтер -.

Другие устройства -.

*Оборудование ПЭВМ студента, если использовалось:*

Процессор Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz , ОП 6 ГБ, НМД 240 ГБ. Монитор IPS 1920x1080

Другие устройства -.

**5. Программное обеспечение** (*лабораторное*):

Операционная система семейства -, наименование - версия -

Интерпретатор команд - версия -

Система программирования -версия -

Редактор текстов - версия -

Утилиты операционной системы -

Прикладные системы и программы -

Местонахождения и имена файлов программ и данных-

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства UNIX/GNU , наименование Ubuntu версия x86\_64

Интерпретатор команд bash

Редактор текстов emax

Утилиты операционной системы cat

Прикладные системы и программы VTM-diagram

Местонахождения и имена файлов программ и данных -

**6. Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями)

Необходимо составить и отладить программу на Си с выполнением стандартных и нестандартного действия. Для начала создадим menu, благодаря которому будем выбирать необходимое действие, которое будет осуществлять операцию. Отдельно создадим функции под печать списка, вставку нового элемента в начало списка, вставку нового элемента в конец списка, удаление элемента из списка, подсчет элементов в списке, генерацию элементов списка, а также выполнения нестандартного действия: очистить список, если в нем есть элемент, равный заданному значению.

Пользователь в любом порядке может проводить действия со списком. Если список пуст, то пользователю будет выведено на экран сообщение: list empty.

**7.Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <math.h>

#include <locale.h>

#include <time.h>

struct point {

float data;

struct point\* next;

};

typedef struct point node;

void print\_list(node\* tail) {

if (tail == NULL) {

printf("list empty\n");

}

else {

node\* p = tail->next;

while (p != tail)

{

printf("%.2f ", p->data);

p = p->next;

}

printf("%.2f\n", tail->data);

}

}

node\* add\_head(node\* tail, float data) {

node\* p = (node\*)malloc(sizeof(node));

p->data = data;

if (tail == NULL) {

tail = p;

tail->next = p;

}

else {

p->next = tail->next;

tail->next = p;

}

return tail;

}

node\* add\_last(node\* tail, float data) {

node\* p = (node\*)malloc(sizeof(node));

p->data = data;

if (tail == NULL) {

tail = p;

tail->next = p;

}

else {

p->next = tail->next;

tail->next = p;

tail = p;

}

return tail;

}

int count\_items(node\* tail) {

if (tail == NULL)

return 0;

node\* p = tail->next;

int count = 1;

while (p != tail) {

count++;

p = p->next;

}

return count;

}

int equal(float a, float b) {

if (fabs(a-b)<0.0001)

return 1;

else

return 0;

}

node\* delete\_item(node\* tail, float data) {

node\* p = tail->next;

node\* pred = tail;

while (p != tail) {

if (equal(p->data, data))

break;

pred = p;

p = p->next;

}

if (p != tail) {

pred->next = p->next;

free(p);

}

else if (equal(tail->data, data)) {

//если в списке стоит один элемент

if (tail->next == tail) {

free(tail);

return NULL;

}

else {

pred->next = tail->next;

free(tail);

tail = pred;

}

}

return tail;

}

node\* generate\_list(node\* tail, int n) {

for (int i = 0; i < n; i++) {

float b = (float)rand() / RAND\_MAX \* (rand() % 100);

b = round(b \* 100) / 100;

tail = add\_last(tail, b);

}

return tail;

}

int find\_element(node\* tail, float data) {

node\* p = tail->next;

while (p!=tail) {

if (p->data == data)

return 1;

p=p->next;

}

if (tail->data==data)

return 1;

return 0;

}

void free\_list(node\* tail) {

node\* p=tail->next;

while(p!=tail) {

node\* t=p->next;

free(p);

p=t;

}

free(tail);

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

node\* tail = NULL;

float data;

while (1)

{

printf("1.Print list\n");

printf("2.Add to begin\n");

printf("3.Add to end\n");

printf("4.Delete item\n");

printf("5.Count items\n");

printf("6.Generate random items\n");

printf("7.No\_standart\_action\n");

printf("0.Exit\n");

int action;

scanf("%d", &action);

if (action == 0)

break;

if (action == 1)

print\_list(tail);

else if (action == 2) {

printf("Input data: ");

scanf("%f", &data);

tail = add\_head(tail, data);

}

else if (action == 3) {

printf("Input data: ");

scanf("%f", &data);

tail = add\_last(tail, data);

}

else if (action == 4) {

printf("Input data: ");

scanf("%f", &data);

tail = delete\_item(tail, data);

}

else if (action == 5) {

int count = count\_items(tail);

printf("Count = %d\n", count);

}

else if (action == 6) {

int n;

printf("n = ");

scanf("%d", &n);

tail = generate\_list(tail, n);

}

else if (action == 7) {

printf("Input data: ");

scanf("%f", &data);

int check= find\_element(tail, data);

if (check) {

free\_list(tail);

tail=NULL;

}

}

}

}

*Пункты 1-7 отчёта составляются* ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя

**8.Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)

elena@elena-Aspire-A315-53G:~$ cat tit.txt

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~ ~

~ Лабораторная работа № VIII ~

~ Линейные списки ~

~ ~

~ Бондарева Елена ~

~ М8О-105Б-21 ~

~ ~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

elena@elena-Aspire-A315-53G:~$ cat 8.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <math.h>

#include <locale.h>

#include <time.h>

struct point {

float data;

struct point\* next;

};

typedef struct point node;

void print\_list(node\* tail) {

if (tail == NULL) {

printf("list empty\n");

}

else {

node\* p = tail->next;

while (p != tail)

{

printf("%.2f ", p->data);

p = p->next;

}

printf("%.2f\n", tail->data);

}

}

node\* add\_head(node\* tail, float data) {

node\* p = (node\*)malloc(sizeof(node));

p->data = data;

if (tail == NULL) {

tail = p;

tail->next = p;

}

else {

p->next = tail->next;

tail->next = p;

}

return tail;

}

node\* add\_last(node\* tail, float data) {

node\* p = (node\*)malloc(sizeof(node));

p->data = data;

if (tail == NULL) {

tail = p;

tail->next = p;

}

else {

p->next = tail->next;

tail->next = p;

tail = p;

}

return tail;

}

int count\_items(node\* tail) {

if (tail == NULL)

return 0;

node\* p = tail->next;

int count = 1;

while (p != tail) {

count++;

p = p->next;

}

return count;

}

int equal(float a, float b) {

if (fabs(a-b)<0.0001)

return 1;

else

return 0;

}

node\* delete\_item(node\* tail, float data) {

node\* p = tail->next;

node\* pred = tail;

while (p != tail) {

if (equal(p->data, data))

break;

pred = p;

p = p->next;

}

if (p != tail) {

pred->next = p->next;

free(p);

}

else if (equal(tail->data, data)) {

//если в списке стоит один элемент

if (tail->next == tail) {

free(tail);

return NULL;

}

else {

pred->next = tail->next;

free(tail);

tail = pred;

}

}

return tail;

}

node\* generate\_list(node\* tail, int n) {

for (int i = 0; i < n; i++) {

float b = (float)rand() / RAND\_MAX \* (rand() % 100);

b = round(b \* 100) / 100;

tail = add\_last(tail, b);

}

return tail;

}

int find\_element(node\* tail, float data) {

node\* p = tail->next;

while (p!=tail) {

if (p->data == data)

return 1;

p=p->next;

}

if (tail->data==data)

return 1;

return 0;

}

void free\_list(node\* tail) {

node\* p=tail->next;

while(p!=tail) {

node\* t=p->next;

free(p);

p=t;

}

free(tail);

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

node\* tail = NULL;

float data;

while (1)

{

printf("1.Print list\n");

printf("2.Add to begin\n");

printf("3.Add to end\n");

printf("4.Delete item\n");

printf("5.Count items\n");

printf("6.Generate random items\n");

printf("7.No\_standart\_action\n");

printf("0.Exit\n");

int action;

scanf("%d", &action);

if (action == 0)

break;

if (action == 1)

print\_list(tail);

else if (action == 2) {

printf("Input data: ");

scanf("%f", &data);

tail = add\_head(tail, data);

}

else if (action == 3) {

printf("Input data: ");

scanf("%f", &data);

tail = add\_last(tail, data);

}

else if (action == 4) {

printf("Input data: ");

scanf("%f", &data);

tail = delete\_item(tail, data);

}

else if (action == 5) {

int count = count\_items(tail);

printf("Count = %d\n", count);

}

else if (action == 6) {

int n;

printf("n = ");

scanf("%d", &n);

tail = generate\_list(tail, n);

}

else if (action == 7) {

printf("Input data: ");

scanf("%f", &data);

int check= find\_element(tail, data);

if (check) {

free\_list(tail);

tail=NULL;

}

}

}

}

elena@elena-Aspire-A315-53G:~$ g++ 8.cpp

elena@elena-Aspire-A315-53G:~$ ./a.out

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

6

n = 8

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

1

6.61 6.51 28.75 7.20 19.05 34.15 34.15 1.35

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

2

Input data: 4.45

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

1

4.45 6.61 6.51 28.75 7.20 19.05 34.15 34.15 1.35

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

3

Input data: 56.78

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

1

4.45 6.61 6.51 28.75 7.20 19.05 34.15 34.15 1.35 56.78

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

5

Count = 10

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

4

Input data: 7.20

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

5

Count = 9

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

1

4.45 6.61 6.51 28.75 19.05 34.15 34.15 1.35 56.78

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

7

Input data: 19.05

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

1

list empty

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

6

n = 7

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

3

Input data: 27.28

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

1

1.67 1.01 21.77 16.65 61.61 37.55 10.25 27.28

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

5

Count = 8

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

7

Input data: 10.15

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

1

1.67 1.01 21.77 16.65 61.61 37.55 10.25 27.28

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

7

Input data: 16.65

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

1

list empty

1.Print list

2.Add to begin

3.Add to end

4.Delete item

5.Count items

6.Generate random items

7.No\_standart\_action

0.Exit

0

**9. Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб.  или  дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
| 1 | дом | 2.05.  2022 | 11:15 | В main не сделала перенос строки при написании menu. | Внимательно писать код. |  |

**10**. Замечание автора по существу работы

**11.** Выводы

В результате выполнения работы я составила и отладила программу на языке Си для обработки линейного списка заданной организации с отображением списка на динамические структуры, предусмотрев выполнение одного нестандартного(очистить список, если в нем есть элемент, равный заданному значению) и стандартных действий(печать списка, вставка нового элемента в список, удаление элемента из списка, подсчет длины списка).

Недочеты, допущенные при выполнении задания, могут быть устранены следующим образом

Подпись студента