Московский Авиационный

Институт

(Национальный Исследовательский Университет)

Институт №8

«Фундаментальная информатика и информационные технологии»

КУРСОВОЙ ПРОЕКТ

по курсу «Практикум на ЭВМ»

II семестр

«Линейные списки»

Выполнил: студент

группы M8O-113Б-21

Сыткин Даниил Сергеевич

Преподаватель:

Довженко Анастасия Александровна

Работа сдана

«\_\_»\_\_\_\_\_\_\_\_\_\_\_2022

Оценка \_\_\_\_\_\_

**Тема:**  Линейные списки

**Цель работы:** Составить и отладить программу на языке Си для обработки линейного списка заданной организации с отображением списка на динамические структуры. Реализовать стандартные действия со списком и одно нестандартное по варианту.

**Задание:**

1) Тип элемента списка: целый(№1)

2) Вид списка: кольцевой однонаправленный(№1)

3) Нестандартное действие: переставить элементы списка в обратном порядке(№11)

**Идея, метод, алгоритм решения задачи**

Структура кольцевого однонаправленного списка:

1) поле body типа item(в моем случае int)

2) поле next типа link(указатель на след элемент списка)

Последний элемент указывает на первый, так как он кольцевой.

Для удобства работы со списком создам два указателя этой структуры tail(указатель на первый элемент списка) и head(указатель на последний элемент списка).

Рассмотрим *стандартные действия* со списком:

1) add(item m) - Добавление элемента в конец списка. Если список пустой, то создаются 2 указателя tail и head, с указателем на себя и с элементом m. Если не пустой, создаем временный указатель структуры списка t(next = 0, body = m), указатель на след элемент head будет равен t(head->next=t), двигаем head вперед(head = head->next) и head->next = tail(так как кольцевой). Если указатель на след элемент tail = tail(это может быть только если мы добавляем 2 элемент), то tail->next = t.

2) addFirst(item m) - Добавление элемента в начало списка. Если список пустой, инициализируем также 2 указателя tail и head. Если не пустой, создаем указатель структуры t(next = tail, body = m). tail меняем на t. И если указатель tail равен tail(добавляем 2 элемент), то t->next = head. Не забываем обновить указатель на tail для head. head->next = tail.

3) genlist(int n) - Генерация списка с кол-вом n случайных элементов. С помощью генератора рандома получаем число и записываем в список с помощью функции add(описана выше).

4) printList() - Печать списка. Записываем в переменную t указатель на первый элемент(t = tail). Выводим body t на экран, двигаем t вперед до тех пор, пока t != tail(в силу того, что список кольцевой)(t=t->next).

5) sizeList() - Возвращает кол-во элементов списка. Используется такая же техника при печати списка(printList())

6) deleting(item m) - Удаляет выбранный элемент. Рассмотрим случай, когда нужно будет удалить первый элемент. Проверим указатель на след элемент, если он указывает на tail, то это случай, когда список становится пустым. Поэтому удаляем этот элемент, tail = 0, head = 0. Если он не пустой, то двигаем tail вперед и удаляем этот элемент.

Теперь рассмотрим случай, когда нужно удалить не первый элемент. Проходим по списку и проверяем, если значение след элемента = m, то записываем след элемент в переменную f, если f - это последний элемент, то двигаем head назад и удаляем f, если это не последний элемент, то указатель текущего элемента next будет равен f next, f удаляем. Если элемент, который нужно удалить, не будет найден, то сообщим об этом пользователю.

7) erase() - Удалить все элемента списка(сделать его пустым). Проходим по списку как в методе printList(), при этом удаляя элементы списка по очереди.

8) insert(item m1, item m2) - Вставить элемент m1 в список после элемента со значением m2. Проходим по списку, если текущий элемент будет равен m2, то проверяем, если след элемента не существует, то это значит, что нужно просто добавить в конец списка элемент m1, сделаем это с помощью метода add(m2). Если след элемент существует. То создаем указатель на переменную структуры f, f->next = t->next(будет на след элемент). Текущий элемент будет указывать на f. Если элемент в списке с значением m2 не будет найден, сообщим об этом пользователю.

Рассмотрим *нестандартное действие*: переставить элементы списка в обратном порядке(№11)

Алгоритм:

1. запоминаем текущий хвост(tail, записываем во временную переменную), и обнуляем указатели tail, head(создаем новый список)
2. обходим старый список используя временную переменную, при этом добавляя в начало нового списка(чтобы перевернуть) и удаляя их.

Работа со списком будет реализована с помощью меню.

**Сценарий выполнения работы**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

struct ls;

typedef ls \*link;

typedef int item; //тип элемента целый

// 11. переставить элементы списка в обратном порядке;

struct ls { // кольцевой однонаправленный список

item body;

link next;

}\*tail, \*head, \*t;

void randomize() {

long a = time(0);

srand(a);

}

void add(item m) {

if (!tail) { // если элементов в списке 0

tail = new ls;

tail->body = m;

tail->next = tail;

head = new ls;

head->body = m;

head->next = tail;

}

else {

t = new ls;

t->body = m;

t->next = 0;

if (tail->next == tail) {

tail->next = t;

}

head->next = t;

head = head->next;

head->next = tail;

// голова передвигается

}

}

void addFirst(item m) { // добавление элемента в начало

if (!head) { // если элементов в списке 0

tail = new ls;

tail->body = m;

tail->next = tail;

head = new ls;

head->body = m;

head->next = tail;

}

else {

t = new ls;

t->body = m;

if (tail->next == tail) {

t->next = head;

}

else {

t->next = tail;

}

tail = t;

head->next = tail;

}

}

void genList(int n) { // генерация списка с кол-вом случайных элементов n

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

item m = rand() % 155;

add(m);

}

}

int sizeList() {

t = tail;

int n = 0;

do {

n++;

t = t->next;

} while (t != tail);

return n;

}

void printList() {

if (!tail) { // список пустой

printf("\nList is empty\n");

}

else {

printf("\nList:\n[ ");

t = tail;

do {

printf("%d ", t->body);

t = t->next;

} while (t != tail);

// printf("%d ]\n", t->body);

printf("]\n");

}

}

void deleting(item m) { // удаление элемента из списка

t = tail;

int flag = 0;

ls \*f;

if (t->body == m) { // случай если первый элемент нужно удалить

if (t->next == tail) { //список станет пустым

delete t;

tail = 0;

head = 0;

return;

}

else { // удаляем текущий элемент и двигаем tail

f = t;

t = t->next;

tail = t;

head->next = tail; // последний указывает на хвост

delete f;

return;

}

}

do {

if (t->next != tail) {

if (t->next->body == m) {

flag = 1;

f = t->next;

if (f->next != tail){

t->next = f->next;

}

else {

t->next = tail; // конец списка изменяется!!!

head = t;

break;

}

delete f;

}

}

t = t->next;

} while (t != tail);

if (!flag) {

printf("\nItem with this value not exists!\n");

}

}

void erase() {

t = tail;

ls \*f;

if (t) {

do {

f = t;

t = t->next;

delete f;

} while (t != tail);

}

tail = 0;

head = 0;

}

void insert(item m1, item m2) {

t = tail;

int flag = 0;

ls \*f;

do {

if (t->body == m2) {

flag = 1;

if (t->next == tail) { // нужно просто добавить в конец списка число

add(m1);

return;

}

else {

f = new ls;

f->body = m1;

f->next = t->next;

t->next = f;

return;

}

}

t = t->next;

} while (t != tail);

if (!flag) {

printf("\nItem with this value not exists!\n");

}

}

void action() {

if (!tail) { // список пустой -> ничего не делаем

return;

}

ls \*temp\_tail;

temp\_tail = tail;

tail = 0;

head = 0;

ls \*temp\_t;

temp\_t = temp\_tail;

ls \*f;

if (temp\_t) {

do {

f = temp\_t;

addFirst(f->body);

temp\_t = temp\_t->next;

delete f; // очищение старых элементов

} while (temp\_t != temp\_tail);

}

}

int main() {

int k = 10;

randomize();

for (;;) {

if (k == 10) {

printf("\nInput from 1 to 10 or 0 for actions:\n"

"0. Exit.\n"

"1. Generation list.\n"

"2. Print list.\n"

"3. Addition new item in begin of list.\n"

"4. Addition new item in end of list.\n"

"5. Erase list.\n"

"6. Deleting from list.\n"

"7. Length of list.\n"

"8. Inserting in list.\n"

"9. Main Action(reverse list 11 variant). \n"

"10. Menu.\n");

}

else if (!k) {

break;

}

else if (k == 1) {

int n;

printf("\nInput number of new items of list: n=");

scanf("%d", &n);

genList(n);

}

else if (k == 2) {

printList();

}

else if (k == 3) {

item m;

printf("\nInput value of adding item: m=");

scanf("%d", &m);

addFirst(m);

}

else if (k == 4) {

item m;

printf("\nInput value of adding item: m=");

scanf("%d", &m);

add(m);

}

else if (k == 5) {

erase();

}

else if (k == 6) {

if (!tail) {

printf("\nList is empty!\n");

}

else {

printf("\nInput value of deleting item: m=");

item m;

scanf("%d", &m);

deleting(m);

}

}

else if (k == 7) {

int n = 0;

if (head) {

n = sizeList();

}

printf("\nLenght of list=%d\n", n);

}

else if (k == 8) {

if (!tail) {

printf("\nList is empty!\n");

}

else {

int m1, m2;

printf("\nInput value inserting item: m1=");

scanf("%d", &m1);

printf("\nInput value item after which to insert: m2=");

scanf("%d", &m2);

insert(m1, m2);

}

}

else if (k == 9) {

if (!tail) {

printf("\nList is empty!\n");

}

else {

action();

}

}

else {

printf("Have no such number of MENU\n");

}

printf("\nInput number of MENU: k=");

scanf("%d", &k);

}

return 0;

}

Тестирование:

Создам случайный список. Проверю стандартные и нестандартные действия.

**Распечатка протокола** 

~/Р/mai\_labs/2/б/кп8(1-1-11) main !2 ?3 g++ main.cpp ✔

~/Р/mai\_labs/2/б/кп8(1-1-11) main !2 ?3 ./a.out ✔

Input from 1 to 10 or 0 for actions:

0. Exit.

1. Generation list.

2. Print list.

3. Addition new item in begin of list.

4. Addition new item in end of list.

5. Erase list.

6. Deleting from list.

7. Length of list.

8. Inserting in list.

9. Main Action(reverse list 11 variant).

10. Menu.

Input number of MENU: k=1

Input number of new items of list: n=5

Input number of MENU: k=2

List:

[ 118 83 59 0 149 ]

Input number of MENU: k=3

Input value of adding item: m=2

Input number of MENU: k=3

Input value of adding item: m=1

Input number of MENU: k=2

List:

[ 1 2 118 83 59 0 149 ]

Input number of MENU: k=4

Input value of adding item: m=34

Input number of MENU: k=4

Input value of adding item: m=35

Input number of MENU: k=4

Input value of adding item: m=36

Input number of MENU: k=2

List:

[ 1 2 118 83 59 0 149 34 35 36 ]

Input number of MENU: k=6

Input value of deleting item: m=34

Input number of MENU: k=6

Input value of deleting item: m=118

Input number of MENU: k=2

List:

[ 1 2 83 59 0 149 35 36 ]

Input number of MENU: k=6

Input value of deleting item: m=36

Input number of MENU: k=2

List:

[ 1 2 83 59 0 149 35 ]

Input number of MENU: k=7

Lenght of list=7

Input number of MENU: k=8

Input value inserting item: m1=444

Input value item after which to insert: m2=0

Input number of MENU: k=2

List:

[ 1 2 83 59 0 444 149 35 ]

Input number of MENU: k=5

Input number of MENU: k=2

List is empty

Input number of MENU: k=1

Input number of new items of list: n=10

Input number of MENU: k=2

List:

[ 39 117 61 40 97 65 73 46 113 91 ]

Input number of MENU: k=9

Input number of MENU: k=2

List:

[ 91 113 46 73 65 97 40 61 117 39 ]

Input number of MENU: k=5

Input number of MENU: k=4

Input value of adding item: m=1

Input number of MENU: k=4

Input value of adding item: m=2

Input number of MENU: k=4

Input value of adding item: m=3

Input number of MENU: k=4

Input value of adding item: m=4

Input number of MENU: k=2

List:

[ 1 2 3 4 ]

Input number of MENU: k=9

Input number of MENU: k=2

List:

[ 4 3 2 1 ]

Input number of MENU: k=5

Input number of MENU: k=1

Input number of new items of list: n=2

Input number of MENU: k=2

List:

[ 67 88 ]

Input number of MENU: k=9

Input number of MENU: k=2

List:

[ 88 67 ]

Input number of MENU: k=6

Input value of deleting item: m=67

Input number of MENU: k=2

List:

[ 88 ]

Input number of MENU: k=9

Input number of MENU: k=2

List:

[ 88 ]

Input number of MENU: k=0

**Вывод:** В ходе курсовой работы я изучил такую структуру данных как списки. Сравнил ее с массивом.