МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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

На тему

«Линейные списки»

Курс / Семестр:	1 / 2
Группа:	М8О-108Б-19
ФИО студента:	Горохов М.А.
ФИО преподавателя:	Поповкин А.В.
Подпись:	
Оценка:	
Дата сдачи:	
Дата проверки:	

Москва

2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В этом курсовом проекте я реализовал кольцевой двунаправленный список.

Базовая динамическая структура данных в информатике, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

Реализация была на динамических структурах, то есть с применением указателей, выделением и очищением памяти.

1. ЦЕЛИ ПРОЕКТА

Изучить способы хранения и обработки связанного списка в компьютере.

2. ПОСТАНОВКА ЗАДАЧИ

Мой ToDo list по данному курсовому проекту, который является структурой проекта:

На динамический структурах. Тип данных целый. Кольцевой двунаправленный.

Дополнить список копиями числа n до кол-ва k элементов в списке. Если в списке не менее k элементов - не добавлять

[+] Реализовать стурктуру итератор (он уже узел)

[+] Реализовать структуру список

[+] Реализовать вывод списка

[+] Реализовать добавление и удаление элемента через push и pop

[+] Реализовать изменение, добавление и удаление элемента по индексу. Если просят удалить элемент вне списка - напечатать ошибку и ничего не делать.

[+] Реализовать нестандартное действие

[ ] Подготовить к сдаче .doc файл

3. СТРУКТУРА ПРОЕКТА

Проект состоит из 2 файлов: main.c - главный файл, list.h - библиотека,

Пример работы программы ***main.c***:

```
[revammark@Raft kp8]$ ./main
1 value - Push Back
2 value - Push Front
3 - Pop Back
4 - Pop Front
5 index value - Push Index
6 index - Pop Index
7 - Erase List
8 size value - Push Until List is Full
9 - print List
10 - print Size
0 - Exit programm

Action: 10
Size of list: 0
Action: 1 1
Action: 1 2
Action: 1 4
Action: 1 5
Action: 1 7
Action: 9
1 2 4 5 7
Action: 2 -19
Action: -87
Getted: -87
It is strange..
Action: 2 -87
Action: 9
-87 -19 1 2 4 5 7
Action: 10
Size of list: 7
Action: 5 7 1000
Action: 9
-87 -19 1 2 4 5 7 1000
Action: 5 7 2000
Action: 9
-87 -19 1 2 4 5 7 2000 1000
Action: 5 7 0
Action: 9
```

-87 -19 1 2 4 5 7 0 2000 1000
Action: 5 0 -1
Action: 9
-1 -87 -19 1 2 4 5 7 0 2000 1000
Action: 5 1000 1000
Out of Range
Index is 1000 but size of list is 11
Action: 9
-1 -87 -19 1 2 4 5 7 0 2000 1000
Action: 5 -11 9
Out of Range
Index is -11 but size of list is 11
Action: 9
-1 -87 -19 1 2 4 5 7 0 2000 1000
Action: 6 3
Poped: 1
Action: 6 3
Poped: 2
Action: 6 2
Poped: -19
Action: 9
-1 -87 4 5 7 0 2000 1000
Action: 8 20 -13
Action: 9
-1 -87 4 5 7 0 2000 1000 -13 -13 -13 -13 -13 -13 -13 -13 -13 -13 -13 -13
Action: 7
Erased!
Action: 9

Action: 1 123456789
Action: 9
123456789
Action: 0
[revammark@Raft kp8]$ date
Ср 06 мая 2020 22:53:14 MSK

4. ПРОГРАММНЫЙ КОД И ТЕСТОВЫЕ ДАННЫЕ

Код программы ***main.c***:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    List list;
    createList(&list);

    short action;
    int index;
    T value;

    printf("1 value - Push Back\n");
    printf("2 value - Push Front\n");
    printf("3 - Pop Back\n");
    printf("4 - Pop Front\n");
    printf("5 index value - Push Index\n");
    printf("6 index - Pop Index\n");
    printf("7 - Erase List\n");
    printf("8 size value - Push Until List is Full\n");
    printf("9 - print List\n");
    printf("10 - print Size\n");
    printf("0 - Exit programm\n\n");
    printf("Action: ");


    while((scanf("%d", &action) != EOF) && action) {
        switch (action) {
        case 1:
            scanf("%ld", &value);
            pushBack(&list, value);
            break;
        case 2:
            scanf("%ld", &value);
            pushFront(&list, value);
            break;
        case 3:
            printf("Poped: %ld\n", popBack(&list));
            break;
        case 4:
            printf("Poped: %ld\n", popFront(&list));
            break;
        case 5:
            scanf("%d", &index);
            scanf("%ld", &value);
            pushIndex(&list, index, value);
            break;
        case 6:
            scanf("%d", &index);
            printf("Poped: %ld\n", popIndex(&list, index));
            break;
```

```
        case 7:
            eraseList(&list);
            printf("Erased!\n");
            break;
        case 8:
            // index === size
            scanf("%d", &index);
            scanf("%ld", &value);
            pushUtilListFull(&list, index, value);
            break;
        case 9:
            printList(&list);
            break;
        case 10:
            printf("Size of list: %d\n", list.size);
            break;
        case 0:
            break;

        default:
        printf("Getted: %d\nIt is strange..\n", action);
            break;
        }
        printf("Action: ");
    }


    return 0;
}
```

Код библиотеки ***list.c***:

```
#include <stdio.h>
#include <stdlib.h>
typedef long T;
typedef struct Node {
    T value;
    struct Node* prev;
    struct Node* next;
} Node;
typedef struct {
    int size;
    Node* first;
    Node* last;
} List;

Node* makeNode(T value) {
    Node* node = malloc(sizeof(Node));
    node->value = value;
}
void destroyNode(Node* node) {
    free(node);
}
void printNode(Node* node) {
```

```
    printf("%li ", node->value);
}
void createList(List* list) {
    list->size=0;
    list->first = NULL;
    list->last = NULL;
}
void printList(List* list) {
    Node* current = list->first;
    for(int i=0; i<list->size; i++) {

        printNode(current);
        current = current->next;
    }
    printf("\n");
}
void pushBack(List* list, T value) {
    Node* newNode = makeNode(value);
    if(list->size >= 1) {
        newNode->prev = list->last;
        newNode->next = list->first;
        list->last->next = newNode;
        list->first->prev = newNode;
        list->last = newNode;
    }
    else {
        newNode->next = newNode;
        newNode->prev = newNode;
        list->first = newNode;
        list->last = newNode;
    }
    list->size++;
}
void pushFront(List* list, T value) {
    Node* newNode = makeNode(value);
    if(list->size >= 1) {
        newNode->prev = list->last;
        newNode->next = list->first;
        list->last->next = newNode;
        list->first->prev = newNode;
        list->first = newNode;
    }
    else {
        newNode->next = newNode;
        newNode->prev = newNode;
        list->first = newNode;
        list->last = newNode;
    }
    list->size++;
}
T popFront(List* list) {
    if(list->size >= 1) {
        T value = list->first->value;
        Node* oldfirst = list->first;
        list->first = list->first->next;
        list->first->prev = list->last;
        list->last->next = list->first;
```

```
        list->size--;
        destroyNode(oldfirst);
        return value;
    }
    else if(list->size == 1) {
        T value = list->last->value;
        Node* oldfirst = list->first;

        list->first = NULL;
        list->last = NULL;

        list->size--;
        destroyNode(oldfirst);
        return value;
    }
    else {
        printf("There is no Elements. Returned 0");
        return 0;
    }
}
T popBack(List* list) {
    if(list->size >= 2) {
        T value = list->last->value;
        Node* oldlast = list->last;
        list->last = list->last->prev;
        list->last->next = list->first;
        list->first->prev = list->first;

        list->size--;
        destroyNode(oldlast);
        return value;
    }
    else if(list->size == 1) {
        T value = list->last->value;
        Node* oldlast = list->last;

        list->first = NULL;
        list->last = NULL;

        list->size--;
        destroyNode(oldlast);
        return value;
    }
    else {
        printf("There is no Elements. Returned 0");
        return 0;
    }
}
void eraseList(List* list) {
    while(list->size != 0) {
        popBack(list);
    }
}
void pushIndex(List* list, unsigned int index, T value) {
    if(index <= list->size) {
        Node* insertNode = makeNode(value);
```

```
        Node* nextNode = list->first;
        if(index < list->size/2) {
            for(int i=0; i < index; i++) nextNode = nextNode->next;
        }
        else if(index != list->size) {
            for(int i=list->size; i > index; i--) nextNode = nextNode->prev;

        }
        insertNode->prev = nextNode->prev;
        insertNode->next = nextNode;
        insertNode->prev->next = insertNode;
        insertNode->next->prev = insertNode;

        if(index == list->size) list->last = insertNode;
        if(index == 0) list->first = insertNode;

        list->size++;
    }
    else {
        printf("Out of Range\nIndex is %i but size of list is %i\n", index, list->size);
    }
}
T popIndex(List* list, unsigned int index) {
    if(index < list->size) {
        Node* node;
        if(index < list->size/2) {
            node = list->first;
            for(int i=0; i < index; i++) node = node->next;
        }
        else {
            node = list->last;
            for(int i=list->size-1; i > index; i--) node = node->prev;
        }

        T value = node->value;
        node->next->prev = node->prev;
        node->prev->next = node->next;

        if(index == list->size-1) list->last = node->prev;
        if(index == 0) list->first = node->next;

        destroyNode(node);
        list->size--;

        return value;
    }
    else {
        printf("Out of Range\nIndex is %i but size of list is %i\nReturned 0", index, list->size);
        return 0;
    }

}
void pushUtilListFull(List* list, unsigned int size, T value) {
    while(list->size < size) {
        pushBack(list, value);
    }
}
```

5. ЗАКЛЮЧЕНИЕ

Я составил кольцевые двунаправленные списки в СП Си. Реализовал программу, работающую со списком.

Реализованная программа умеет: добавлять элемент в начало и в конец списка, добавлять элемент по индексу, получать элемент из начала, конца списка, получать элемент по индексу, выводить содержимое списка, выводить размер списка, заполнять список заданными значениями до нужного размера, и, конечно же, очищать список.

Я полность выполнил поставленное задание

Я освоил работу со структурами в Си, с указателями и управлением памяти в СИ.

6. ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. Керниган Б., Ритчи Д. Язык программирования Си = The C programming language. — 2-е изд. — М.: Вильямс, 2007. — С. 304. — ISBN 0-13-110362-8.

2. Эндрю Таненбаум, *Structured Computer Organization,* ISBN 0-13-148521-0

3. В.Е. Зайцев, Конспект Лекций по курсу “Фундаментальная Информатика и Языки и методы программирования”.