Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра вычислительной математики и программирования

Курсовая работа

По курсу «Языки и методы программирования»

II семестр

Задание №8 «Линейные списки»

Выполнил студент

1-го курса, 105-ой группы

Махмудов О. С.

_____

(подпись)

Научный руководитель

Доцент кафедры 806

Сластушенский Ю. В.

_____

(подпись)

Работа защищена

«__»_____ 2019

Оценка_____

ОГЛАВЛЕНИЕ

ЦЕЛЬ

Составить и отладить программу на языке Си с процедурами и/или функциями для обработки линейного списка заданной организации (линейный двунаправленный с барьерным элементом) с отображением списка на динамическую структуру. Навигацию по списку следует реализовать с применением итераторов. Функции программы:

1) Вставка нового элемента

2) Удаление элемента из списка

3) Печать списка

4) Подсчет длины списка

5)Проверка на упорядоченность

Отмечу, тип данных для элементов списка – комплексный. Для этого я использовал дополнительную структуру с полями для целой и мнимой части.

ПРОГРАММА

Функции

spisok *barrier1(spisok *node)	Создает первый барьерный элемент
spisok *barrier2(spisok *barrier1, spisok *node)	Создает второй барьерный элемент
void push_node(spisok *node, int iter, complex key)	Вставляет новый элемент в список
void pop_node(spisok *node, int iter)	Удаляет элемент из списка
void print_spisok(spisok *node)	Печатает список
int counter(spisok *node)	Считает количество элементов списка
int task(spisok *node)	Проверяет список на упорядоченность
void menu()	Выводит меню

Код программы

```
#include "stdio.h"
#include "stdlib.h"
#include "malloc.h"

typedef struct {
	int re;
	int im;
} complex;

typedef struct node {
	complex key;
	struct node *prev;
	struct node *next;
}spisok;

spisok *barrier1(spisok *node) {
	node = (spisok*)malloc(sizeof(spisok));
	node->key.im = 0;
	return node;
}

spisok *barrier2(spisok *barrier1, spisok *node) {
	node = (spisok*)malloc(sizeof(spisok));
	node->key.im = 0;
	barrier1->next = node;
	node->prev = barrier1;
	return node;
}

void push_node(spisok *node, int iter, complex key) {
	if (iter < 1)
		iter = 1;
	while ((iter > 1) && (node->next->key.im != 0)) {
		node = node->next;
		iter--;
	}
	spisok *newnode = malloc(sizeof(spisok));
	newnode->key.re = key.re;
	newnode->key.im = key.im;
	newnode->next = node->next;
	newnode->prev = node;
	node->next->prev = newnode;
	node->next = newnode;
}

void pop_node(spisok *node, int iter) {
	while (iter > 0) {
		node = node->next;
		iter--;
	}
	node->next->prev = node->prev;
	node->prev->next = node->next;
	free(node);
```

```
}

void print_spisok(spisok *node) {
	node = node->next;
	if (node->key.im == 0)
		return;
	else {
		printf("%d", node->key.re);
		printf("+i*");
		printf("%d  ", node->key.im);
		print_spisok(node);
	}
}

int counter(spisok *node) {
	int k = 0;
	node = node->next;
	while (node->key.im != 0) {
		k++;
		node = node->next;
	}
	return k;
}

int task(spisok *node) {
	int k = 0;
	node = node->next;
	while (node->next->key.im != 0) {
		if (node->key.re <= node->next->key.re) {
			k++;
			node = node->next;
		}
		else {
			k = 0;
			return 0;
			break;
		}
	}
	if (k != 0)
		return 1;
}

void menu() {
	printf("==========================\n");
	printf("||   1-Push node       ||\n");
	printf("||   2-Pop node        ||\n");
	printf("||   3-Print spisok    ||\n");
	printf("||   4-Count the length ||\n");
	printf("||   5-Curry to task   ||\n");
	printf("||   6-Menu            ||\n");
	printf("||   0-End             ||\n");
	printf("==========================\n");
	printf("\n");
}

int main() {
	spisok *a = NULL;
```

```
	spisok *b = NULL;
	int i, x, c;
	char ch = '9';
	complex n;
	a = barrier1(a);
	b = barrier2(a, b);
	menu();
	while (ch != '0') {
		printf("=> ");
		scanf("%s", &ch);
		switch (ch) {
		case '1':
			printf("Enter the index of the node: ");
			scanf(" %d", &i);
			printf("Enter the real part: ");
			scanf(" %d", &n.re);
			printf("Enter the imaginary part: ");
			scanf(" %d", &n.im);
			push_node(a, i, n);
			break;
		case '2':
			printf("Enter the index of the node: ");
			scanf(" %d", &i);
			pop_node(a, i);
			break;
		case '3':
			print_spisok(a);
			printf("\n");
			break;
		case '4':
			c = counter(a);
			printf("%d\n", c);
			break;
		case '5':
			x = task(a);
			if (x == 0)
				printf("False\n");
			else
				printf("True\n");
			break;
		case '6':
			menu();
			break;
		}
	}
	return 0;
}
```

Тесты

```
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ gcc kursach8.c
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ ./a.out
==========================
||  1-Push node           ||
||  2-Pop node            ||
||  3-Print spisok        ||
||  4-Count the length    ||
||  5-Curry to task       ||
||  6-Menu                ||
||  0-End                 ||
==========================

=> 1
Enter the index of the node: 2
Enter the real part: 3
Enter the imaginary part: 4
=> 3
3+i*4
=> 1
Enter the index of the node: 1
Enter the real part: 2
Enter the imaginary part: 3
=> 3
2+i*3  3+i*4
=> 4
2
=> 5
True
=> 1
Enter the index of the node: 1
Enter the real part: 9
```

Enter the imaginary part: 8

=> 3

9+i*8 2+i*3 3+i*4

=> 4

3

=> 5

False

=> 2

Enter the index of the node: 1

=> 3

2+i*3 3+i*4

=> 4

2

=> 5

True

ЗАКЛЮЧЕНИЕ

В процессе выполнения данной курсовой работы, я улучшил навыки работы с линейным списком и его отображением на динамические структуры, узнал много полезных функций для этого на языке Си и понял общую концепцию динамических структур - списков.