### 

Выполнил

студент группы КТбо1-2 Д. Д. Карасенко

Принял

доцент кафедры САиТ В. С. Лапшин

Таганрог 2024

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

Кафедра системного анализа и телекоммуникаций

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине «Основы алгоритмизации и программирования»

**«ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ, СПИСКИ»**

Вариант 5

Оглавление

[1](#_Toc163560090)

[**Техническое задание** 3](#_Toc163560091)

[**Цель задания** 3](#_Toc163560092)

[**Задача** 3](#_Toc163560093)

[**Ход работы** 4](#_Toc163560094)

[**Алгоритм** 4](#_Toc163560095)

[**Написание программы** 4](#_Toc163560096)

[**Пример работы программы** 5](#_Toc163560097)

[**Вывод** 5](#_Toc163560098)

[**Листинг** 6](#_Toc163560099)

**Техническое задание**

## **Цель задания**

Целью данной лабораторной работы является освоение студентами способов представления и обработки данных в виде списков. Для успешного выполнения лабораторной работы необходимо владеть теоретическими сведениями по таким понятиям, как списки, указатели, способы выделения памяти, функции, отвечающие за выделение памяти, функции malloc() calloc() free(), структуры данных, информационная и адресная составляющие списка.

## **Задача**

Задание. Написать программы для работы с односвязным и двусвязным списками в соответствии с выданным вариантом задания. Предусмотреть в программах следующие функции:

1. Включить новый элемент в конец списка.
2. Включить новый элемент на заданное пользователем место списка.
3. Включить новый элемент после элемента с заданной информационной частью.
4. Включить новый элемент перед элементом с заданной информационной частью.
5. Включить новый элемент в середину списка.
6. Исключить элемент из середины списка.
7. Исключить элемент с заданной информационной частью.
8. Исключить элемент из конца списка.
9. Исключить элемент из заданного пользователем места списка.
10. Исключить элемент из головы списка.

Задание моего варианта: обработать структуру данных, содержащую в себе телефонную книгу.

# **Ход работы**

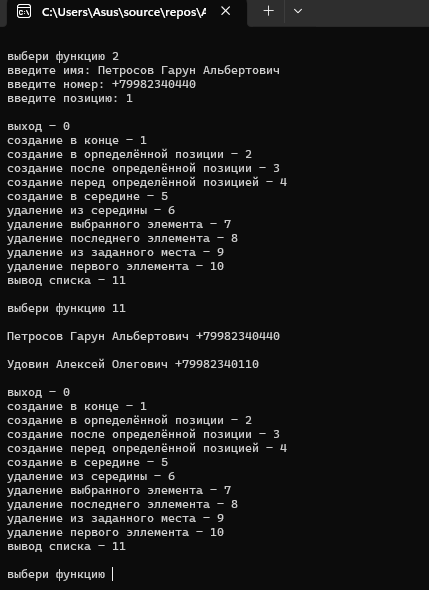
## **Алгоритм**

1. Создаём структуру, которая в дальнейшем будет служить односвязным списком.
2. Прописываем функции создания, удаления и вывода элементов списка.
3. Создаём бесконечный цикл, в котором пользователь сможет работать с ранее написанными функциями

## **Написание программы**

Пишем структуру состоящую из двух массивов строк для имя и номера, а так же указателя на другую структуру. Большинство функций для создания принимают начальный элемент списка, имя и номер телефона введённые пользователем. Дальше создаётся структура и мы изменяем указатель в элементе введённым в функцию и с помощью strcpy\_s() вставляем значения строк. В случае функций удаления освобождаем место с помощью функции free() и изменяем указатель предыдущего элемента на NULL. Пишем цикл с выводом перечня функции работы со списком и используем функцию switch() для того, чтобы пользователь мог работать со списком.

## **Пример работы программы**



# **Вывод**

Подводя итоги, можно сказать, что я создал программу для работы с односвязными списками, которая имеет множество функций для изменения списков. В ходе работы я ознакомился с понятием списков, функциями malloc() и free(), воспользовался структурами и указателями.

# **Листинг**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#include <windows.h>

struct Node

{

char name[1000];

char num[1000];

struct Node\* next;

};

//создание в конец 1

void CreateInEnd(struct Node\*\* head, char newName[], char newNum[])

{

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

struct Node\* last = \*head;

strcpy\_s(newNode->name,sizeof(newNode->name), newName);

strcpy\_s(newNode->num, sizeof(newNode->num), newNum);

newNode->next = NULL;

if (\*head == NULL)

{

\*head = newNode;

return;

}

while (last->next != NULL)

{

last = last->next;

}

last->next = newNode;

}

// создание в орпеделённой позиции 2

void CreateInPos(struct Node\*\* head, char newName[],char newNum[], int pos)

{

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

strcpy\_s(newNode->name, sizeof(newNode->name), newName);

strcpy\_s(newNode->num, sizeof(newNode->num), newNum);

if (pos == 0)

{

newNode->next = \*head;

\*head = newNode;

}

else

{

struct Node\* current = \*head;

for (int i = 0; current != NULL && i < pos-1 ; i++)

{

current = current->next;

}

newNode->next = current->next;

current->next = newNode;

}

}

// создание после определённой позиции 3

void CreateAfterTarget(struct Node\*\* head, char targetData[], char newName[],char newNum[])

{

struct Node\* current = \*head;

while (current != NULL && strcmp(current->name, targetData) != 0)

{

current = current->next;

}

if (current == NULL)

{

printf("Data not founded");

return;

}

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

strcpy\_s(newNode->name, sizeof(newNode->name), newName);

strcpy\_s(newNode->num, sizeof(newNode->num), newNum);

newNode->next = current->next;

current->next = newNode;

}

// создание перед определённой позицией 4

void CreateBeforeTarget(struct Node\*\* head, char targetData[], char newName[], char newNum[])

{

if (strcmp((\*head)->name, targetData))

{

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

strcpy\_s(newNode->name, sizeof(newNode->name), newName);

strcpy\_s(newNode->num, sizeof(newNode->num), newNum);

newNode->next = \*head;

\*head = newNode;

return;

}

struct Node\* current = \*head;

while (current->next != NULL && strcmp(current->next->name, targetData) != 0)

{

current = current->next;

}

if (current->next == NULL)

{

printf("Data not founded");

return;

}

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

strcpy\_s(newNode->name, sizeof(newNode->name), newName);

strcpy\_s(newNode->num, sizeof(newNode->num), newNum);

newNode->next = current->next;

current->next = newNode;

}

//создание в середине 5

void CreateInMiddle(struct Node\*\* head, char newName[],char newNum[])

{

struct Node\* slow = \*head;

struct Node\* fast = \*head;

struct Node\* prev = NULL;

while (fast != NULL && fast->next != NULL)

{

fast = fast->next->next;

prev = slow;

slow = slow->next;

}

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

strcpy\_s(newNode->name, sizeof(newNode->name), newName);

strcpy\_s(newNode->num, sizeof(newNode->num), newNum);

newNode->next = slow;

if (prev != NULL)

{

prev->next = newNode;

}

else

{

\*head = newNode;

}

}

//удаление из середины 6

void DeleteInMiddle(struct Node\*\* head)

{

struct Node\* slow = \*head;

struct Node\* fast = \*head;

struct Node\* prev = NULL;

while (fast != NULL && fast->next != NULL)

{

fast = fast->next->next;

prev = slow;

slow = slow->next;

}

if (prev == NULL)

{

free(\*head);

\*head = NULL;

}

else

{

prev->next = slow->next;

free(slow);

}

}

//удаление выбранного элемента 7

void DeleteInTarget(struct Node\*\* head, char target[]) {

struct Node\* temp = \*head;

struct Node\* prev = \*head;

if (temp != NULL && strcmp(temp->name, target) == 0) {

\*head = temp->next;

free(temp);

return;

}

while (temp != NULL && strcmp(temp->name, target) != 0) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) {

printf("Элемент с именем %s не найден в списке.\n", target);

return;

}

prev->next = temp->next;

free(temp);

}

//удаление последнего эллемента 8

void DeleteInEnd(struct Node\*\* head)

{

if ((\*head)->next == NULL)

{

free(\*head);

\*head = NULL;

return;

}

struct Node\* current = \*head;

while (current->next->next != NULL)

{

current = current->next;

}

free(current->next);

current->next = NULL;

}

//удаление из заданного места 9

void DeleteInPos(struct Node\*\* head, int pos)

{

if (pos == 0)

{

struct Node\* temp = \*head;

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

free(temp);

return;

}

struct Node\* current = \*head;

for (int i = 0; current != NULL && i < pos - 1; i++)

{

current = current->next;

}

if (current == NULL || current->next == NULL)

{

printf("Position is greater");

return;

}

struct Node\* temp = current->next;

current->next = current->next->next;

free(temp);

}

//удаление первого эллемента 10

void DeleteFirst(struct Node\*\* head)

{

Node\* temp = \*head;

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

free(temp);

}

// Функция для отображения элементов списка 11

void displayList(struct Node\* node)

{

while (node != NULL) {

printf("\n%s %s", node->name, node->num);

node = node->next;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

struct Node\* test = NULL;

char name[100];

char num [15];

char target[100];

int pos = 0;

int choice = 15;

while (choice != 0)

{

printf("\nвыход - 0");

printf("\nсоздание в конце - 1");

printf("\nсоздание в орпеделённой позиции - 2");

printf("\nсоздание после определённой позиции - 3");

printf("\nсоздание перед определённой позицией - 4");

printf("\nсоздание в середине - 5");

printf("\nудаление из середины - 6");

printf("\nудаление выбранного элемента - 7");

printf("\nудаление последнего эллемента - 8");

printf("\nудаление из заданного места - 9");

printf("\nудаление первого эллемента - 10");

printf("\nвывод списка - 11");

printf("\n\nвыбери функцию ");

scanf\_s("%d", &choice);

getchar();

switch (choice)

{

case 1:

printf("введите имя: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = '\0';

printf("введите номер: ");

fgets(num, sizeof(num), stdin);

CreateInEnd(&test, name, num);

break;

case 2:

printf("введите имя: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = '\0';

printf("введите номер: ");

fgets(num, sizeof(num), stdin);

printf("введите позицию: ");

scanf\_s("%d", &pos);

CreateInPos(&test,name,num,pos-1);

break;

case 3:

printf("введите имя после которого вставить ");

fgets(target, sizeof(target), stdin);

target[strcspn(target, "\n")] = '\0';

printf("введите имя: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = '\0';

printf("введите номер: ");

fgets(num, sizeof(num), stdin);

CreateAfterTarget(&test, target, name, num);

break;

case 4:

printf("введите имя перед которым вставить ");

fgets(target, sizeof(target), stdin);

target[strcspn(target, "\n")] = '\0';

printf("введите имя: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = '\0';

printf("введите номер: ");

fgets(num, sizeof(num), stdin);

CreateBeforeTarget(&test, target, name, num);

break;

case 5:

printf("введите имя: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = '\0';

printf("введите номер: ");

fgets(num, sizeof(num), stdin);

CreateInMiddle(&test, name, num);

break;

case 6:

DeleteInMiddle(&test);

break;

case 7:

printf("введите имя элемента, который хотите удалить ");

fgets(target, sizeof(target), stdin);

target[strcspn(target, "\n")] = '\0';

DeleteInTarget(&test, target);

break;

case 8:

DeleteInEnd(&test);

break;

case 9:

printf("введите позицию, которую хотите удалить: ");

scanf\_s("%d", &pos);

DeleteInPos(&test, pos-1);

break;

case 10:

DeleteFirst(&test);

case 11:

displayList(test);

}

}

}