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 2

Выполнил

студент группы КТбо1-2 И. Н. Акимов

Принял

доцент кафедры САиТ В. С. Лапшин

Таганрог 2024

# Содержание

Содержание 2

Техническое задание 3

Цель задания 3

Задача 4

Ход работы 5

Алгоритм 5

Пример работы программы 6

Вывод 7

Листинг 8

# 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: Карточка студента и его успеваемости деканата.

# Ход работы

## Алгоритм

Алгоритм моей программы следующий:

1.вывод списка предложенных функций

2.предложение пользователю выбрать нужную

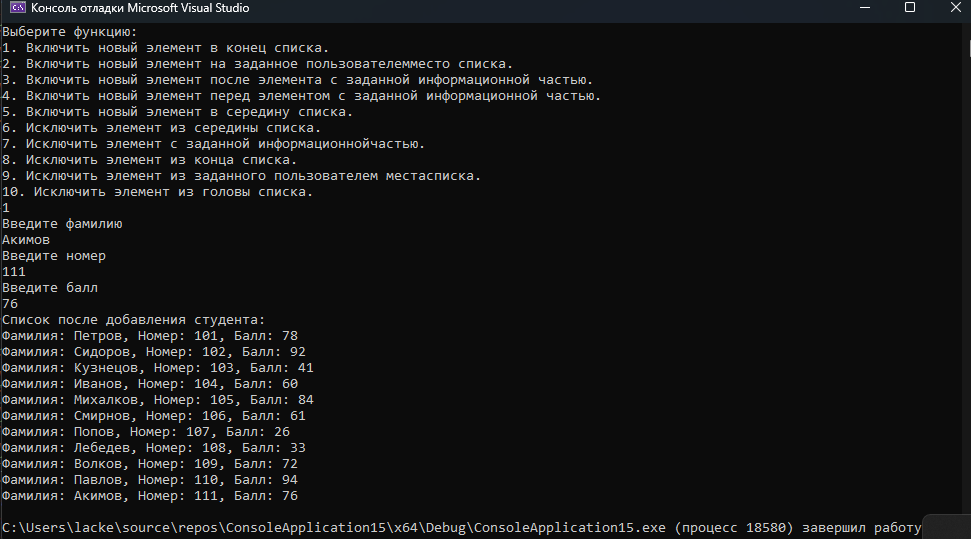
3.запрос данных для выполнения при необходимости

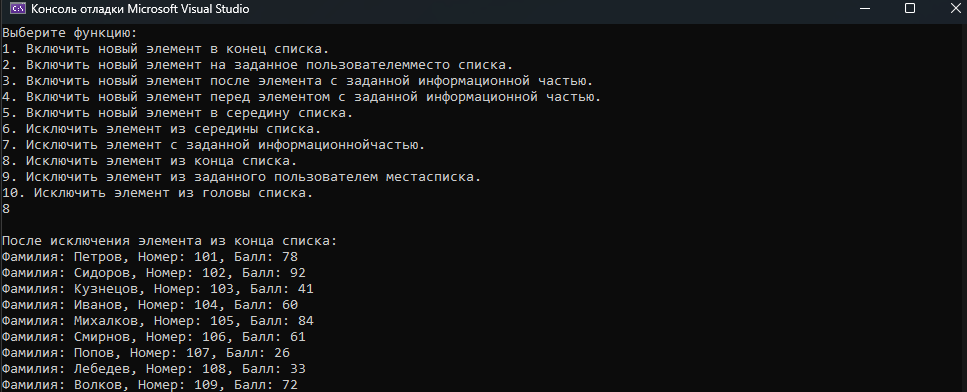
4.выполнение функции

5.вывод списка после выполнения

запись, или потом снова попытаться ввести).

## Пример работы программы





# Вывод

Итогом работы стала программа, работающая с односвязным списком адресов. Она динамически может создавать новые узлы, а также удалять их без утечки памяти, сравнивать данных в узлах, и выводить список.

В ходе работы я ознакомился с функциями calloc() и free(), воспользовался структурами и нумерованным списком, а также ознакомился с возможностями использования указателей в подобных задачах.

# Листинг

## main.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <locale.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

// Определение структуры для односвязного списка

struct Student {

char surname[50];

int student\_number;

int score;

struct Student\* next;

};

//1. Функция для добавления студента в конец списка

void append(struct Student\*\* head, const char\* surname, int student\_number, int score) {

struct Student\* new\_student = (struct Student\*)malloc(sizeof(struct Student));

strcpy(new\_student->surname, surname);

new\_student->student\_number = student\_number;

new\_student->score = score;

new\_student->next = NULL;

if (\*head == NULL) {

\*head = new\_student;

return;

}

struct Student\* last = \*head;

while (last->next != NULL) {

last = last->next;

}

last->next = new\_student;

}

//2. функция для добавления студента на заданную позицию

void insert\_at\_position(struct Student\*\* head, const char\* surname, int student\_number, int score, int position) {

if (position < 0) {

printf("Позиция должна быть неотрицательным числом.\n");

return;

}

struct Student\* new\_student = (struct Student\*)malloc(sizeof(struct Student));

strcpy(new\_student->surname, surname);

new\_student->student\_number = student\_number;

new\_student->score = score;

if (position == 0) {

new\_student->next = \*head;

\*head = new\_student;

return;

}

struct Student\* current = \*head;

for (int i = 0; i < position - 1; ++i) {

if (current == NULL) {

printf("Недопустимая позиция.\n");

return;

}

current = current->next;

}

if (current == NULL) {

printf("Недопустимая позиция.\n");

return;

}

new\_student->next = current->next;

current->next = new\_student;

}

//3. Функция для включения нового элемента после элемента с заданной информационной частью

void insert\_after(struct Student\* head, const char\* surname, const char\* newSurname, int newStudentNumber, int newScore) {

struct Student\* current = head;

while (current != NULL) {

if (strcmp(current->surname, surname) == 0) {

struct Student\* new\_student = (struct Student\*)malloc(sizeof(struct Student));

strcpy(new\_student->surname, newSurname);

new\_student->student\_number = newStudentNumber;

new\_student->score = newScore;

new\_student->next = current->next;

current->next = new\_student;

break;

}

current = current->next;

}

}

//4. Функция для добавления студента перед элементом с заданной информационной частью

void insert\_before(struct Student\* head, char\* target\_surname, char\* surname, int student\_number, int score) {

struct Student\* new\_student = (struct Student\*)malloc(sizeof(struct Student));

strcpy(new\_student->surname, surname);

new\_student->student\_number = student\_number;

new\_student->score = score;

if (strcmp((head)->surname, target\_surname) == 0) {

new\_student->next = head;

head = new\_student;

return;

}

struct Student\* current = head;

while (current->next != NULL) {

if (strcmp(current->next->surname, target\_surname) == 0) {

new\_student->next = current->next;

current->next = new\_student;

return;

}

current = current->next;

}

printf("Студент с фамилией %s не найден в списке.\n", target\_surname);

}

//5. Функция для добавления студента в середину списка

void insert\_in\_middle(struct Student\* head, char\* surname, int student\_number, int score) {

struct Student\* new\_student = (struct Student\*)malloc(sizeof(struct Student));

strcpy(new\_student->surname, surname);

new\_student->student\_number = student\_number;

new\_student->score = score;

int count = 0;

struct Student\* current = head;

while (current != NULL) {

count++;

current = current->next;

}

int middle\_position = count / 2;

current = head;

for (int i = 0; i + 1 < middle\_position; ++i) {

current = current->next;

}

new\_student->next = current->next;

current->next = new\_student;

}

//6. Исключить элемент из середины списка.

void delete\_from\_middle(struct Student\* head)

{

int count = 0;

struct Student\* current = head;

struct Student\* prev = NULL;

while (current != NULL) {

count++;

current = current->next;

}

int middle\_position = count / 2;

current = head;

for (int i = 0; i < middle\_position; ++i) {

prev = current;

current = current->next;

}

prev->next = current->next;

return;

}

//7. Функция для удаления студента с заданной информационной частью

void delete\_by\_surname(struct Student\*\* head, char\* surname) {

if (\*head == NULL) {

printf("Список пуст.\n");

return;

}

struct Student\* current = \*head;

struct Student\* prev = NULL;

while (current != NULL) {

if (strcmp(current->surname, surname) == 0) {

if (prev == NULL) {

\*head = current->next;

}

else {

prev->next = current->next;

}

free(current);

printf("Студент с фамилией %s удален из списка.\n", surname);

return;

}

prev = current;

current = current->next;

}

printf("Студент с фамилией %s не найден в списке.\n", surname);

}

//8. Функция для исключения элемента из конца списка

void delete\_from\_tail(struct Student\* head) {

if (head == NULL) {

printf("Список пуст.\n");

return;

}

if (head->next == NULL) {

free(head);

return;

}

struct Student\* current = head;

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

current = current->next;

}

free(current->next);

current->next = NULL;

return;

}

//9. функция для удаления студента на заданной позиции

void delete\_by\_position(struct Student\*\* head, int position) {

if (\*head == NULL) {

printf("Список пуст.\n");

return;

}

if (position == 0) {

printf("Недопустимая позиция.\n");

return;

}

struct Student\* current = \*head;

struct Student\* prev = NULL;

int count = 0;

while (current != NULL) {

if (count == position) {

prev->next = current->next;

free(current);

return;

}

prev = current;

current = current->next;

count++;

}

printf("Недопустимая позиция.\n");

}

// Функция для отображения списка студентов (10)

void display\_10(struct Student\* newHead) {

setlocale(LC\_ALL, "rus");

struct Student\* current = newHead;

while (current != NULL) {

printf("Фамилия: %s, Номер: %d, Балл: %d\n", current->surname, current->student\_number, current->score);

current = current->next;

}

}

//10 . Функция для исключения элемента из головы списка

void delete\_from\_head(struct Student\* head) {

setlocale(LC\_ALL, "rus");

if (head == NULL) {

printf("Список пуст.\n");

return;

}

struct Student\* newHead = head->next;

free(head);

display\_10(newHead);

free(newHead);

return;

}

// Функция для отображения списка студентов

void display(struct Student\* head) {

setlocale(LC\_ALL, "rus");

struct Student\* current = head;

while (current != NULL) {

printf("Фамилия: %s, Номер: %d, Балл: %d\n", current->surname, current->student\_number, current->score);

current = current->next;

}

}

int main() {

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

struct Student\* head = NULL;

int number, score, position;

char surname[30], target\_surname[30];

append(&head, "Петров", 101, 78);

append(&head, "Сидоров", 102, 92);

append(&head, "Кузнецов", 103, 41);

append(&head, "Иванов", 104, 60);

append(&head, "Михалков", 105, 84);

append(&head, "Смирнов", 106, 61);

append(&head, "Попов", 107, 26);

append(&head, "Лебедев", 108, 33);

append(&head, "Волков", 109, 72);

append(&head, "Павлов", 110, 94);

//Выбор функции

int vibor;

printf("Выберите функцию:\n1. Включить новый элемент в конец списка.\n2. Включить новый элемент на заданное пользователемместо списка.\n3. Включить новый элемент после элемента с заданной информационной частью.\n4. Включить новый элемент перед элементом с заданной информационной частью.\n5. Включить новый элемент в середину списка.\n6. Исключить элемент из середины списка.\n7. Исключить элемент с заданной информационнойчастью.\n8. Исключить элемент из конца списка.\n9. Исключить элемент из заданного пользователем местасписка.\n10. Исключить элемент из головы списка.\n");

scanf("%d", &vibor);

switch (vibor)

{

// Пример добавления в конец списка

case(1):

printf("Введите фамилию\n");

scanf("%s", &surname);

printf("Введите номер\n");

scanf("%d", &number);

printf("Введите балл\n");

scanf("%d", &score);

append(&head, surname, number, score);

printf("Список после добавления студента:\n");

display(head);

while (head != NULL) {

struct Student\* temp = head;

head = head->next;

free(temp);

}

break;

// Пример добавления на заданное место списка

case(2):

printf("Введите фамилию\n");

scanf("%s", &surname);

printf("Введите номер\n");

scanf("%d", &number);

printf("Введите балл\n");

scanf("%d", &score);

printf("Введите позицию\n");

scanf("%d", &position);

insert\_at\_position(&head, surname, number, score, position);

printf("Список после добавления студента:\n");

display(head);

while (head != NULL) {

struct Student\* temp = head;

head = head->next;

free(temp);

}

break;

// Включение нового элемента после элемента с заданной информационной частью

case(3):

printf("Введите фамилию, после которой вставить новую\n");

scanf("%s", &target\_surname);

printf("Введите фамилию\n");

scanf("%s", &surname);

printf("Введите номер\n");

scanf("%d", &number);

printf("Введите балл\n");

scanf("%d", &score);

insert\_after(head, target\_surname, surname, number, score);

printf("Список после включение нового элемента:\n");

display(head);

while (head != NULL) {

struct Student\* temp = head;

head = head->next;

free(temp);

}

break;

// Включение нового элемента перед элемента с заданной информационной частью

case(4):

printf("Введите фамилию, перед которой вставить новую\n");

scanf("%s", &target\_surname);

printf("Введите фамилию\n");

scanf("%s", &surname);

printf("Введите номер\n");

scanf("%d", &number);

printf("Введите балл\n");

scanf("%d", &score);

insert\_before(head, target\_surname, surname, number, score);

printf("Список после добавления студента:\n");

display(head);

while (head != NULL) {

struct Student\* temp = head;

head = head->next;

free(temp);

}

break;

// Включить новый элемент в середину списка.

case(5):

printf("Введите фамилию\n");

scanf("%s", &surname);

printf("Введите номер\n");

scanf("%d", &number);

printf("Введите балл\n");

scanf("%d", &score);

insert\_in\_middle(head, surname, number, score);

printf("Список после добавления студента в середину:\n");

display(head);

while (head != NULL) {

struct Student\* temp = head;

head = head->next;

free(temp);

}

break;

// Исключить элемент из середины списка.

case(6):

delete\_from\_middle(head);

printf("Список после удаления студента из середины:\n");

display(head);

while (head != NULL) {

struct Student\* temp = head;

head = head->next;

free(temp);

}

break;

// Исключить элемент с заданной информационной частью.

case(7):

printf("Введите фамилию\n");

scanf("%s", &target\_surname);

delete\_by\_surname(&head, target\_surname);

printf("\nПосле исключения заданного студента:\n");

display(head);

while (head != NULL) {

struct Student\* temp = head;

head = head->next;

free(temp);

}

break;

// Исключение элемента из конца списка

case(8):

delete\_from\_tail(head);

printf("\nПосле исключения элемента из конца списка:\n");

display(head);

while (head != NULL) {

struct Student\* temp = head;

head = head->next;

free(temp);

}

break;

// Исключить элемент из заданного пользователем места списка.

case(9):

printf("Введите позицию\n");

scanf("%d", &position);

delete\_by\_position(&head, position);

printf("Список после удаления студента на %d позиции:\n", position);

display(head);

while (head != NULL) {

struct Student\* temp = head;

head = head->next;

free(temp);

}

break;

// Исключить элемент из головы списка.

case(10):

printf("Список после удаления студента из головы списка:\n");

delete\_from\_head(head);

break;

}

}