**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

Московский институт электроники и математики Национального

исследовательского университета "Высшая школа экономики"

Департамент прикладной математики

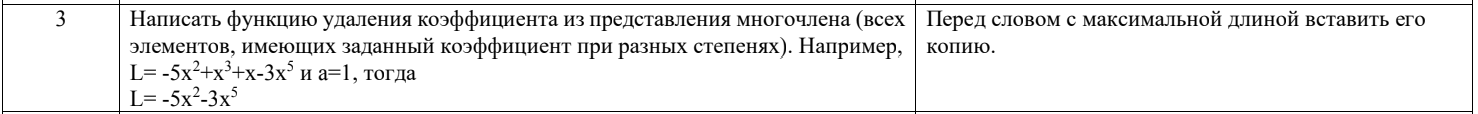
**ОТЧЕТ**

**По лабораторной работе № 7**

**По курсу «Алгоритмизация и программирование»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | ФИО студента | | Номер группы | Дата |
| Бедный Дмитрий Ильич | БПМ 245 | 18 июня 2025 |
|  |
|  |
|  |

**Москва – 2025 г.**



#include <stdio.h>

#include <stdlib.h>

#include "headlines.h"

// Удаляет все элементы с заданным коэффициентом

void removeByCoefficient(Node\*\* head, int coeff) {

    Node\* curr = \*head;

    Node\* prev = NULL;

    while (curr != NULL) {

        if (curr->coefficient == coeff) {

            Node\* toDelete = curr;

            if (prev == NULL) {

                \*head = curr->next;

            } else {

                prev->next = curr->next;

            }

            curr = curr->next;

            free(toDelete);

        } else {

            prev = curr;

            curr = curr->next;

        }

    }

}

// Пример добавления элемента (в начало)

void addTerm(Node\*\* head, int degree, int coefficient) {

    //if (coefficient == 0) return;

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

    newNode->degree = degree;

    newNode->coefficient = coefficient;

    newNode->next = \*head;

    \*head = newNode;

}

// Функция для печати многочлена

void printPoly(Node\* head) {

    Node\* curr = head;

    while (curr) {

        printf("%dx^%d ", curr->coefficient, curr->degree);

        if (curr->next) printf("+ ");

        curr = curr->next;

    }

    printf("\n");

}

// Структура для элемента многочлена

typedef struct Node{

    int degree;         // Степень X

    int coefficient;    // Коэффициент (ненулевой)

    struct Node\* next;

} Node;

void removeByCoefficient(Node\*\* head, int coeff);

void addTerm(Node\*\* head, int degree, int coefficient);

void printPoly(Node\* head);

==================================

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include "headlines.h"

int main() {

    system("chcp 65001");

    Node\* poly = NULL;

    int n, degree, coefficient;

    printf("Введите количество членов многочлена: ");

    scanf("%d", &n);

    for (int i = 0; i < n; ++i) {

        printf("Введите степень и коэффициент для члена %d (через пробел): ", i + 1);

        scanf("%d %d", &degree, &coefficient);

        addTerm(&poly, degree, coefficient);

    }

    printf("Исходный многочлен: ");

    printPoly(poly);

    int coeffToRemove;

    printf("Введите коэффициент для удаления: ");

    scanf("%d", &coeffToRemove);

    removeByCoefficient(&poly, coeffToRemove);

    printf("После удаления коэффициента %d: ", coeffToRemove);

    printPoly(poly);

    // Очистка памяти

    while (poly) {

        Node\* tmp = poly;

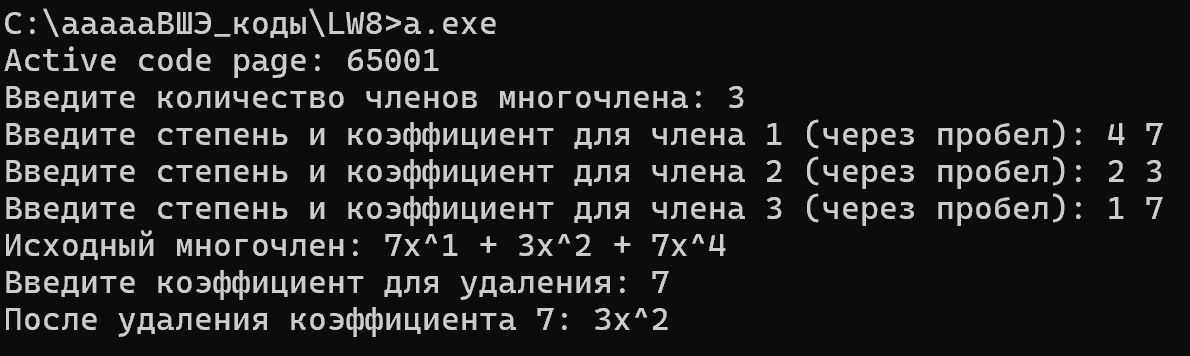
        poly = poly->next;

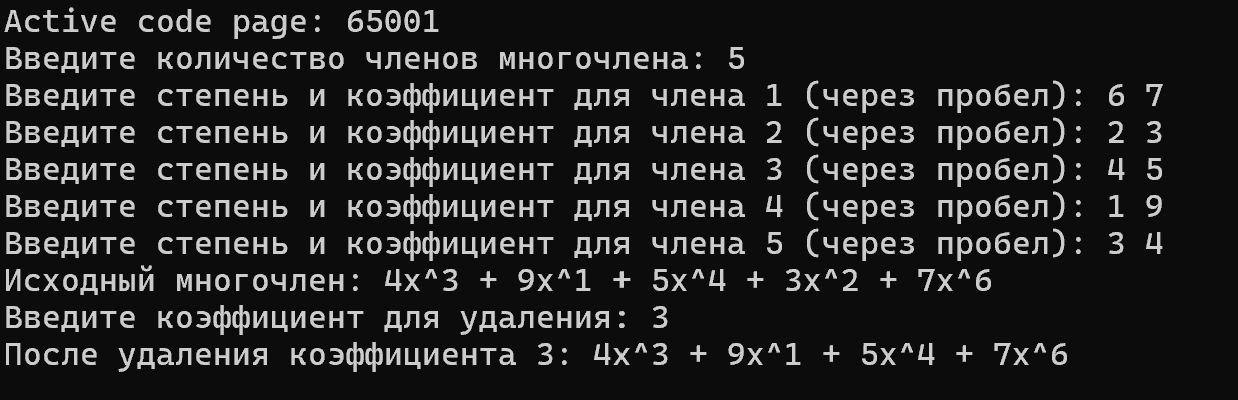
        free(tmp);

    }

    return 0;

}





#include <stdio.h>

#include "headlines.h"

#include <stdlib.h>

#include <string.h>

// Создание нового узла

Node\* create\_node(const char \*word) {

    Node \*node = (Node\*)malloc(sizeof(Node));

    node->word = strdup(word);

    node->prev = node->next = NULL;

    return node;

}

// Добавление узла в конец списка

void append(Node \*\*head, Node \*\*tail, const char \*word) {

    Node \*node = create\_node(word);

    if (!\*head) {

        \*head = \*tail = node;

    } else {

        node->prev = \*tail;

        (\*tail)->next = node;

        \*tail = node;

    }

}

// Вывод списка

void print\_list(Node \*head) {

    for (Node \*cur = head; cur; cur = cur->next) {

        printf("%s", cur->word);

        if (cur->next) printf(" ");

    }

    printf(".\n");

}

// Освобождение памяти

void free\_list(Node \*head) {

    while (head) {

        Node \*tmp = head;

        head = head->next;

        free(tmp->word);

        free(tmp);

    }

}

// Вставка копии перед данным узлом

void insert\_copy\_before(Node \*\*head, Node \*node) {

    Node \*copy = create\_node(node->word);

    copy->next = node;

    copy->prev = node->prev;

    if (node->prev) node->prev->next = copy;

    node->prev = copy;

    if (\*head == node) \*head = copy;

}

// Поиск максимальной длины слова

int max\_word\_length(Node \*head) {

    int max = 0;

    for (Node \*cur = head; cur; cur = cur->next) {

        int len = strlen(cur->word);

        if (len > max) max = len;

    }

    return max;

}

// Вставка копии перед словом максимальной длины

void insert\_before\_max(Node \*\*head) {

    int max = max\_word\_length(\*head);

    for (Node \*cur = \*head; cur; cur = cur->next) {

        if ((int)strlen(cur->word) == max) {

            insert\_copy\_before(head, cur);

            // Пропустить только что вставленную копию

            cur = cur->next;

        }

    }

}

void build\_list(Node \*\*head, Node \*\*tail, const char \*input) {

    char word[101];

    int i = 0, j = 0;

    while (input[i]) {

        if (isalpha((unsigned char)input[i])) {

            word[j] = (unsigned char)input[i];

            j++;

        } else if ((input[i] == ' ' || input[i] == '.') && j > 0) {

            word[j] = '\0';

            append(head, tail, word);

            j = 0;

        }

        if (input[i] == '.') break;

        i++;

    }

}

-----------------------------------

typedef struct Node {

    char \*word;

    struct Node \*prev, \*next;

} Node;

Node\* create\_node(const char \*word);

void append(Node \*\*head, Node \*\*tail, const char \*word);

void print\_list(Node \*head);

void free\_list(Node \*head);

void insert\_copy\_before(Node \*\*head, Node \*node);

int max\_word\_length(Node \*head);

void insert\_before\_max(Node \*\*head);

void build\_list(Node \*\*head, Node \*\*tail, const char \*input);

----------------

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include "headlines.h"

int main() {

    system("chcp 65001"); // Установка кодировки UTF-8 для консоли

    char input[1024];

    printf("Введите строку: ");

    fgets(input, sizeof(input), stdin);

    Node \*head = NULL, \*tail = NULL;

    build\_list(&head, &tail, input);

    printf("Исходный список:\n");

    print\_list(head);

    insert\_before\_max(&head);

    printf("Модифицированный список:\n");

    print\_list(head);

    free\_list(head);

    return 0;

}

