МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЕТ

по лабораторной работе № 10
по дисциплине «Программирование»
Тема: «Кольцевые списки»

Студент гр. 3311 _____ Шарпинский Д. А.

Преподаватель _____ Хахаев И. А.

Санкт-Петербург
2024

Цель работы.

Научиться работать с кольцевыми списками в языке С.

Задание (вариант 4)

Общее задание:

Для выбранной предметной области создать динамический массив структур, содержащих характеристики объектов предметной области.

Обязательный набор полей:

- динамический массив символов, включая пробелы (name)
- произвольный динамический массив символов
- числовые поля типов int и float (не менее двух полей каждого типа)
- поле с числовым массивом.

Задание по варианту:

Разработать подалгоритм создания односвязного кольцевого списка с обратным расположением элементов по отношению к полученному односвязному списку, но без элемента, номер которого получен. В случае отсутствия элемента с таким номером вывести сообщение.

Постановка задачи и описание решения

Предметная область: Профессии

Целью данной работы является разработка программы на языке С для создания односвязного кольцевого списка на основе существующего односвязного списка профессий. Новый список должен содержать элементы в обратном порядке по отношению к исходному списку, за исключением элемента с указанным номером. Если элемент с таким номером отсутствует, программа должна вывести соответствующее сообщение.

Структура Profession, используемая для хранения информации о профессиях, представлена следующим образом:

Название поля	Тип	Описание
id	int	Идентефикатор элемента
name	char*	Название профессии
next	Profession*	Указатель на следующую профессию

Также важной частью программы является структура ProfessionHead, которая служит для управления списком профессий:

Название поля	Тип	Описание
count	int	Количество пользователей
first	Profession*	Указатель на первую профессию в списке
last	Profession*	Указатель на последнюю профессию в списке

Программа строится на основе структур данных, представляющих собой профессии в виде односвязного кольцевого списка. Каждая профессия характеризуется уникальным идентификатором и названием. Отдельная структура ProfessionHead используется для управления списком профессий, содержа указатели на первый и последний элементы списка, а также общее количество элементов.

Работа с программой начинается с инициализации данных профессий из файла CSV. Пользователю предоставляется меню с опциями для управления списком, включая просмотр всех профессий, создание нового списка с обратным расположением элементов без указанного элемента и проверку кольцевой структуры списка.

Для создания нового списка с обратным расположением элементов без указанного элемента используется функция makeReversedListWithNoID. Эта функция принимает исходный список профессий и идентификатор элемента, который нужно исключить из нового списка. Функция создает новый список,

добавляя элементы из исходного списка в обратном порядке, пропуская элемент с указанным идентификатором.

Функция listCarouselGUI предоставляет возможность проверки кольцевой структуры списка. Пользователь может последовательно просматривать элементы списка, переходя от одного элемента к другому по кольцу, пока не будет нажата кнопка выхода.

Программа также включает в себя вспомогательные функции для создания и управления списками профессий, чтения данных из файла, поиска элементов по идентификатору, а также функции для пользовательского интерфейса, такие как вывод меню, заголовков таблиц и элементов списка.

Описание переменных

Функция main()

№	Имя переменной	Тип	Назначение
1	pHead	ProfessionHead*	Указатель на начало списка профессий, инициализируется в NULL

Функция reverseListGUI()

№	Имя переменной	Тип	Назначение
1	pHead	ProfessionHead*	Указатель на голову списка профессий

Функция findProfessionById()

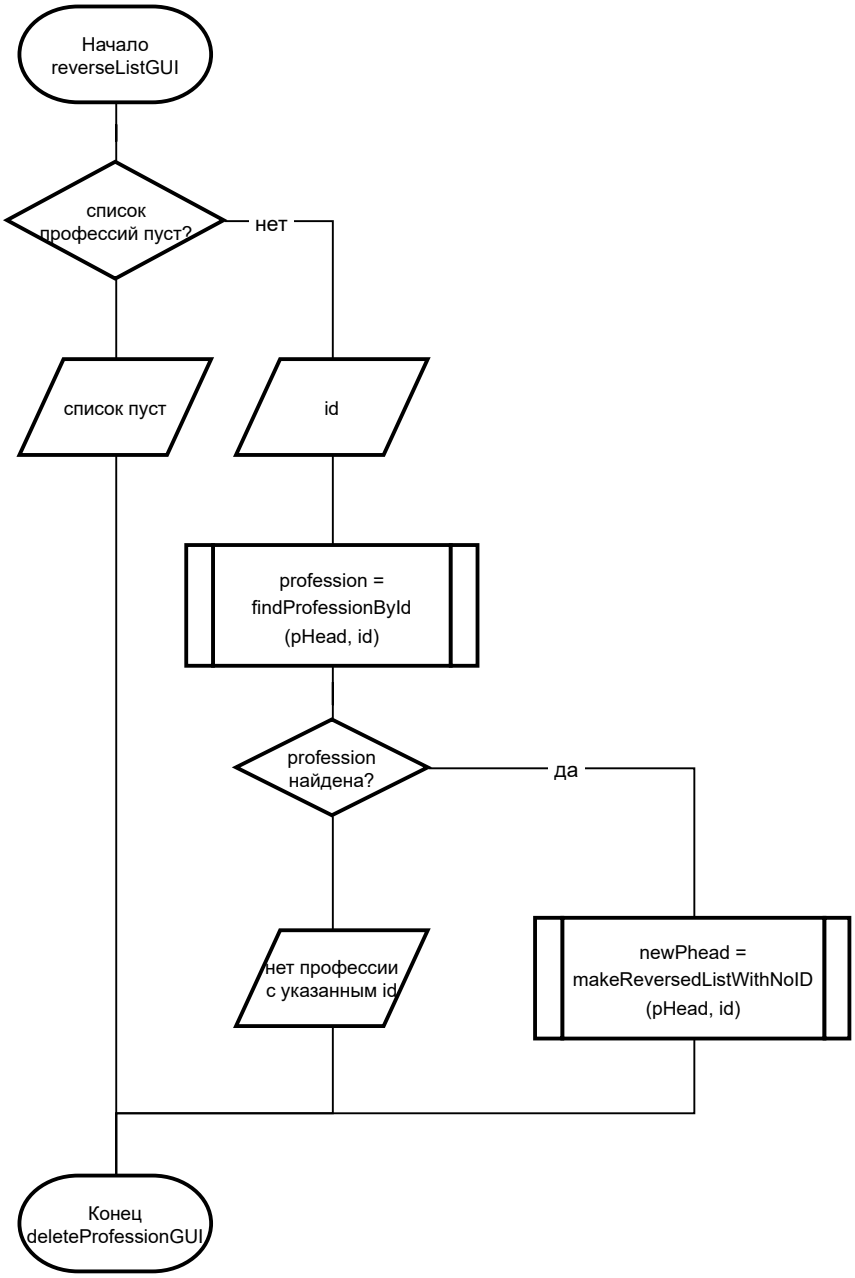
№	Имя переменной	Тип	Назначение
1	head	ProfessionHead*	Указатель на голову списка профессий
2	id	int	Идентификатор профессии, которую нужно найти

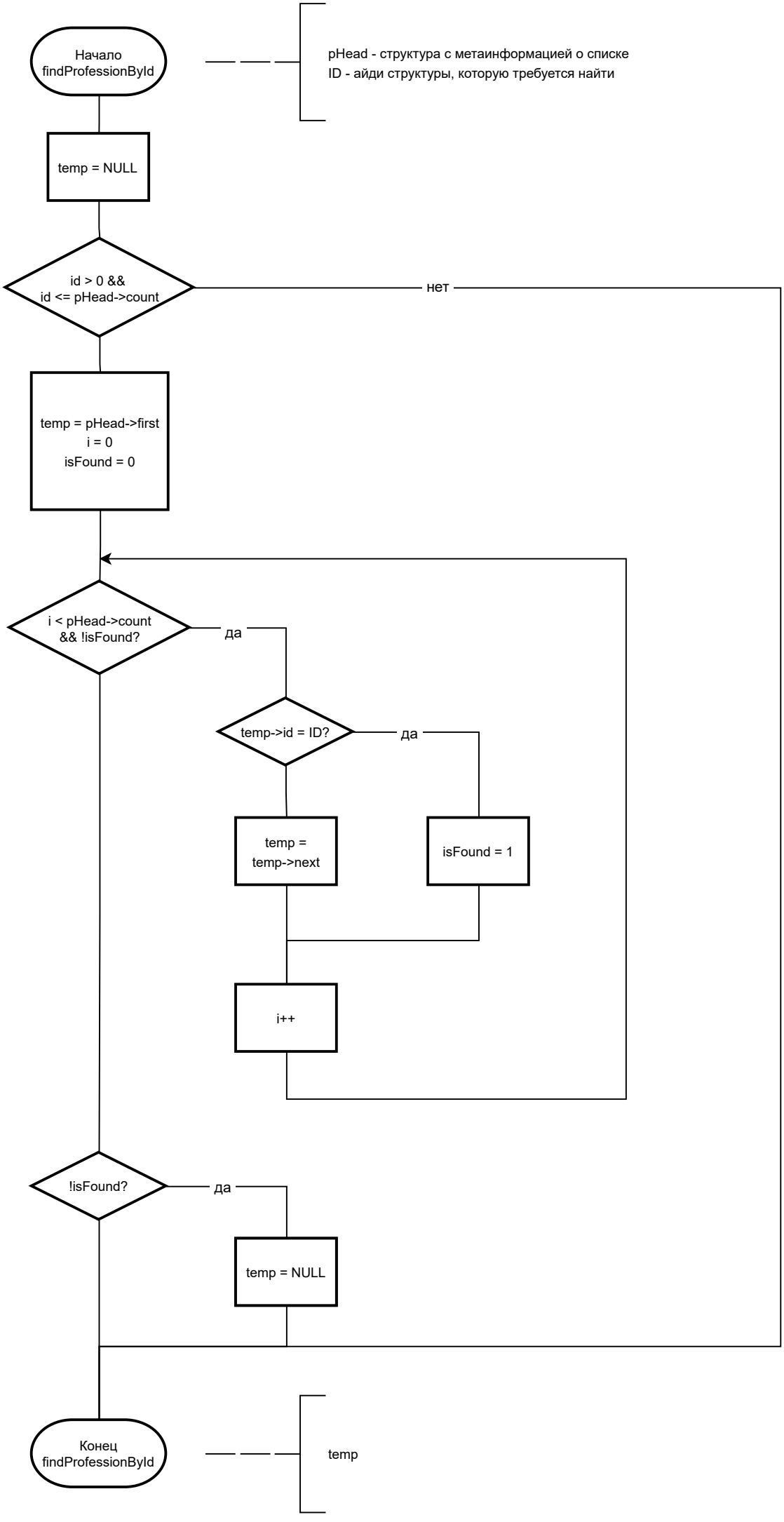
Функция makeReversedListWithNoID()

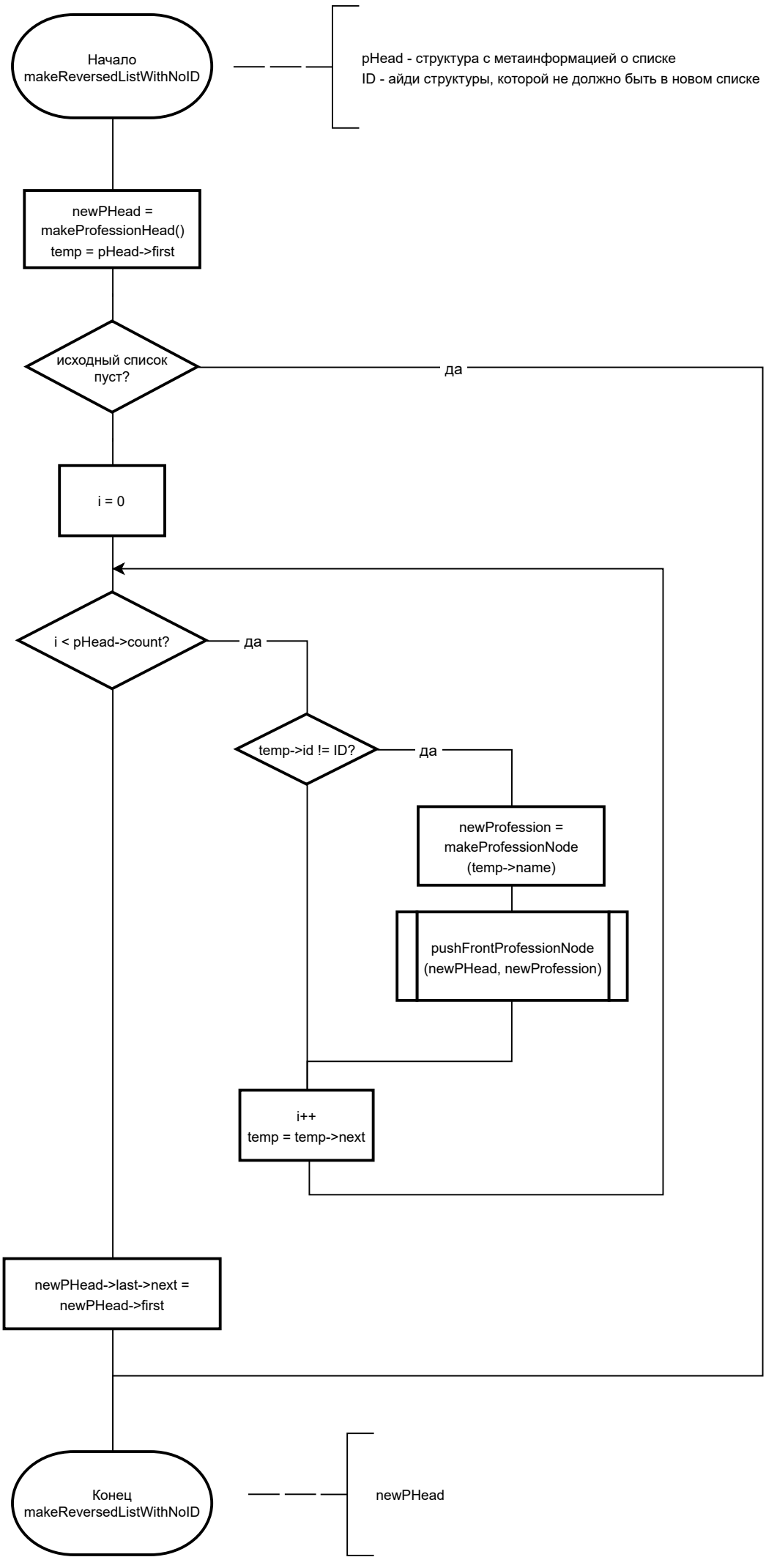
№	Имя переменной	Тип	Назначение
1	pHead	ProfessionHead*	Указатель на голову списка профессий
2	id	int	Идентификатор профессии, которой не должно быть в списке

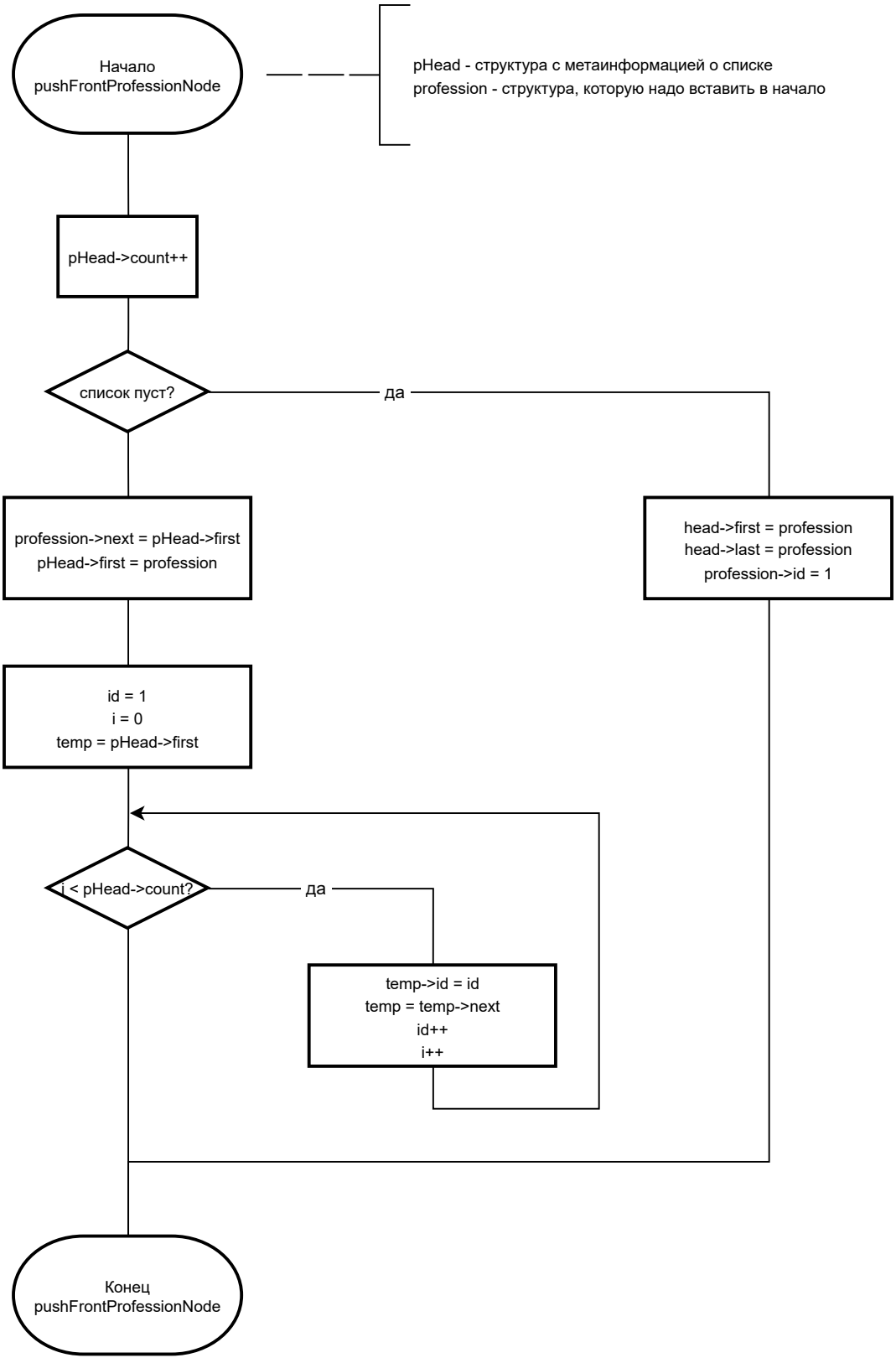
Функция pushFrontProfessionNode()

№	Имя переменной	Тип	Назначение
1	pHead	ProfessionHead*	Указатель на голову списка профессий
2	profession	Profession*	Структура, которую необходимо добавить в начало списка


Начало
reverseListGUI
список
профессий пуст?
нет
список пуст
id
profession =
findProfessionById
(pHead, id)
profession
найдена?
да
нет профессии
с указанным id
newPhead =
makeReversedListWithNoID
(pHead, id)
Конец
deleteProfessionGUI


Начало
findProfessionById
pHead - структура с метаинформацией о списке
ID - айди структуры, которую требуется найти
temp = NULL
id > 0 &&
id <= pHead->count
нет
temp = pHead->first
i = 0
isFound = 0
i < pHead->count
&& !isFound?
да
temp->id = ID?
да
temp =
temp->next
isFound = 1
i++
!isFound?
да
temp = NULL
Конец
findProfessionById
temp


Начало
makeReversedListWithNoID
pHead - структура с метаинформацией о списке
ID - айди структуры, которой не должно быть в новом списке
newPHead =
makeProfessionHead()
temp = pHead->first
исходный список
пуст?
да
i = 0
i < pHead->count?
да
temp->id != ID?
да
newProfession =
makeProfessionNode
(temp->name)
pushFrontProfessionNode
(newPHead, newProfession)
i++
temp = temp->next
newPHead->last->next =
newPHead->first
Конец
makeReversedListWithNoID
newPHead


Начало
pushFrontProfessionNode
pHead - структура с метаинформацией о списке
profession - структура, которую надо вставить в начало
pHead->count++
список пуст?
да
profession->next = pHead->first
pHead->first = profession
head->first = profession
head->last = profession
profession->id = 1
id = 1
i = 0
temp = pHead->first
i < pHead->count?
да
temp->id = id
temp = temp->next
id++
i++
Конец
pushFrontProfessionNode

Исходный текст программы

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

#define MAXLEN 256
#define OPTION_1 "Print all profession"
#define OPTION_2 "Select profession and reverse list"
#define OPTION_3 "List carousel"

typedef struct professionStruct {
    int id;
    char name[MAXLEN];
    struct professionStruct* next;
} Profession;

typedef struct professionHeadStruct {
    Profession* first;
    Profession* last;
    int count;
} ProfessionHead;

void appOption(ProfessionHead* pHead, int option);
void appGUI(ProfessionHead* pHead);
void reverseListGUI(ProfessionHead* pHead);
void listCarouselGUI(ProfessionHead* pHead);

ProfessionHead* makeProfessionHead();
Profession* makeProfessionNode(char name[MAXLEN]);
void pushBackProfessionNode(ProfessionHead* head, Profession* profession);
void freeProfessionList(ProfessionHead* head);
void readProfessions(char* filename, ProfessionHead* head);
void pushFrontProfessionNode(ProfessionHead* head, Profession* profession);
ProfessionHead* makeReversedListWithNoID(ProfessionHead* head, int id);
Profession* findProfessionById(ProfessionHead* head, int id);

void trim(char str[MAXLEN]);
void clearStdin();

void printMenu();
```

```
void printProfessionHeader();
void printAllProfessions(ProfessionHead* head);
void printOptionHeader(const char* optionDescription);
void pressEnterToContinue();
void clearConsole();
void trimForDisplay(char *output, const char *input, int maxLength);
void printProfession(Profession *profession);
void printShortLine();
void printListSize(ProfessionHead *pHead);

int main() {
    ProfessionHead* pHead = NULL;

    pHead = makeProfessionHead();

    if (pHead != NULL) {
        appGUI(pHead);
        freeProfessionList(pHead);
    } else {
        perror("Memory allocation failed");
    }

    return 0;
}

void appGUI(ProfessionHead* pHead) {
    int option;

    readProfessions("professions.csv", pHead);

    do {
        clearConsole();
        printMenu();
        scanf("%d", &option);
        clearStdin();
        if (option != 0) {
            appOption(pHead, option);
        } else {
            clearConsole();
        }
```

```
    } while (option != 0);
}


void appOption(ProfessionHead* professionHead, int option) {
    clearConsole();
    switch (option) {
        case 1:
            printOptionHeader(OPTION_1);
            printAllProfessions(professionHead);
            break;
        case 2:
            printOptionHeader(OPTION_2);
            reverseListGUI(professionHead);
            break;
        case 3:
            listCarouselGUI(professionHead);
            break;
        default:
            clearConsole();
            printf("\nFailed: invalid option\n");
            break;
    }
    pressEnterToContinue();
}


void reverseListGUI(ProfessionHead* pHead) {
    ProfessionHead* newPhead = NULL;
    Profession* profession = NULL;
    int id, option;


    if (pHead->first != NULL) {
        printAllProfessions(pHead);
        printf("\nSelect the ID (after this a new list will be created that will contain all the elements
of the original list\n");
        printf("except the element whose ID you specify. The order of the elements will be inverse in this
list)\n\n");
        printf("ID: ");
        scanf("%d", &id);
        clearStdin();
        profession = findProfessionById(pHead, id);
        if (profession == NULL) {
```

```
            printf("\nFailed: there is no profession with id %d\n", id);
        } else {
            printf("\nProfession with id %d:\n", id);
            printProfessionHeader();
            printProfession(profession);
            printShortLine();
            newPhead = makeReversedListWithNoID(pHead, id);
            if (newPhead != NULL) {
                printf("\nReversed list:\n");
                printAllProfessions(newPhead);
                option = 2;
                printf("Do you want to make sure that this list is circular?\n");
                printf("1. Yes\n");
                printf("2. No\n");
                printf("Option: ");
                scanf("%d", &option);
                clearStdin();
                if (option == 1) {
                    listCarouselGUI(newPhead);
                }
                freeProfessionList(newPhead);
            } else {
                printf("\nFailed: memory allocation failed\n");
            }
        }
    } else {
        printf("There are no profession in the list\n");
    }
}


void listCarouselGUI(ProfessionHead* pHead) {
    Profession* temp;
    int option;

    if (pHead->first != NULL) {
        temp = pHead->first;
        do {
            clearConsole();
            printf("This program uses circular linked lists! You can verify this by using the \"carousel\"
to endlessly scroll the list\n\n");
            option = 0;
```

```
            printProfessionHeader();
            printProfession(temp);
            printShortLine();
            printf("\nPress ENTER to see the next profession\n");
            printf("Press 0 to exit\n");
            option = getchar();
            if (option == '\n') {
                temp = temp->next;
            } else {
                clearStdin();
            }
        } while (option != '0');
    } else {
        printf("\nThere are no professions in the list\n");
    }
}

ProfessionHead* makeProfessionHead() {
    ProfessionHead* head = NULL;

    head = (ProfessionHead*)malloc(sizeof(ProfessionHead));
    if (head != NULL) {
        head->count = 0;
        head->first = NULL;
        head->last = NULL;
    } else {
        perror("Memory allocation failed");
    }

    return head;
}

Profession* makeProfessionNode(char name[MAXLEN]) {
    Profession* profession = NULL;

    profession = (Profession*)malloc(sizeof(Profession));

    if (profession != NULL) {
        profession->id = 0;
        strcpy(profession->name, name);
```

```
        profession->next = NULL;
    }

    return profession;
}

void pushBackProfessionNode(ProfessionHead* head, Profession* profession) {
    head->count++;

    if (head->first == NULL) {
        head->first = profession;
        head->last = profession;
        profession->id = 1;
    } else {
        profession->id = head->last->id + 1;
        head->last->next = profession;
        head->last = profession;
    }
    profession->next = head->first;
}

void freeProfessionList(ProfessionHead* head) {
    Profession *temp1, *temp2;
    int i;

    temp1 = head->first;
    for (i = 0; i < head->count; i++) {
        temp2 = temp1->next;
        free(temp1);
        temp1 = temp2;
    }

    free(head);
}

void readProfessions(char* filename, ProfessionHead* head) {
    FILE* file;
    Profession* profession;
    int n, count, i;
    char temp[MAXLEN] = {0};
```

```
    profession = NULL;
    n = count = 0;
    file = fopen(filename, "r");

    if (file != NULL) {
        while ((fgets(temp, MAXLEN, file)) != NULL) n++;
        rewind(file);

        for (i = 0; i < n; i++) {
            fgets(temp, MAXLEN, file);
            trim(temp);
            profession = makeProfessionNode(temp);
            if (profession != NULL) {
                pushBackProfessionNode(head, profession);
                count++;
            }
        }
        fclose(file);
    } else {
        perror("Failed to open file");
    }

    if (count != n) {
        perror("Failed to read from file");
        freeProfessionList(head);
    }
}

void pushFrontProfessionNode(ProfessionHead* pHead, Profession* profession) {
    Profession* temp;
    int i, id;

    pHead->count++;

    if (pHead->first == NULL) {
        pHead->first = profession;
        pHead->last = profession;
        profession->id = 1;
    } else {
```

```
        profession->next = pHead->first;
        pHead->first = profession;

        id = 1;
        temp = pHead->first;
        for (i = 0; i < pHead->count; i++, id++) {
            temp->id = id;
            temp = temp->next;
        }
    }
}

ProfessionHead* makeReversedListWithNoID(ProfessionHead* pHead, int id) {
    ProfessionHead* newPHead = NULL;
    Profession* temp = pHead->first;
    Profession* newProfession;
    int i, errorFlag;

    newPHead = makeProfessionHead();

    if (newPHead != NULL && temp != NULL) {
        errorFlag = 0;
        for (i = 0; i < pHead->count && !errorFlag; i++) {
            if (temp->id != id) {
                newProfession = makeProfessionNode(temp->name);
                if (newProfession != NULL) {
                    pushFrontProfessionNode(newPHead, newProfession);
                } else {
                    errorFlag = 1;
                }
            }
            temp = temp->next;
        }
        if (!errorFlag) {
            newPHead->last->next = newPHead->first;
        }
    }

    return newPHead;
}
```

```
Profession* findProfessionById(ProfessionHead* head, int id) {
    Profession* temp = NULL;
    int i, isFound = 0;

    if (id > 0 && id <= head->count) {
        temp = head->first;
        for (i = 0; i < head->count && !isFound; i++) {
            if (temp->id == id) {
                isFound = 1;
            } else {
                temp = temp->next;
            }
        }
        if (!isFound) {
            temp = NULL;
        }
    }

    return temp;
}

void trim(char str[MAXLEN]) {
    int i, flag = 0;
    str[MAXLEN - 1] = '\0';
    for (i = 0; str[i] != '\0' && !flag; i++) {
        if (str[i] == '\n' || str[i] == '\r') {
            str[i] = '\0';
            flag = 1;
        }
    }
}

void clearStdin() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF) { }
}

void printMenu() {
    printShortLine();
```

```
    printf("|                  Choose an option               |\n");
    printf("|-------------------------------------------------|\n");
    printf("| 0. Exit                                         |\n");
    printf("| 1. Print all professions                        |\n");
    printf("| 2. Select profession and reverse list           |\n");
    printf("| 3. List carousel                                |\n");
    printShortLine();
    printf("Option: ");
}

void printProfessionHeader() {
    printShortLine();
    printf("| ID |                  Name                       |\n");
    printf("|----|---------------------------------------------|\n");
}

void printAllProfessions(ProfessionHead* head) {
    Profession *q;
    int i;

    printListSize(head);

    printProfessionHeader();
    q = head->first;
    for (i = 0; i < head->count; i++) {
        printProfession(q);
        q = q->next;
    }
    printShortLine();
}

void printProfession(Profession *profession) {
    char trimmedProfessionName[32];
    trimForDisplay(trimmedProfessionName, profession->name, 31);
    printf("| %-2d | %-43s |\n", profession->id, trimmedProfessionName);
}

void printShortLine() {
    printf("===================================================\n");
}
```

```
void printOptionHeader(const char* optionDescription) {
    printShortLine();
    printf("| Option: %-40s |\n", optionDescription);
    printShortLine();
    printf("\n");
}

void pressEnterToContinue() {
    printf("\nPress ENTER to continue ");
    clearStdin();
    clearConsole();
}

void clearConsole() {
    #if defined(_WIN32) || defined(_WIN64)
        system("cls");
    #else
        system("clear");
    #endif
}

void trimForDisplay(char *output, const char *input, int maxLength) {
    if (strlen(input) > maxLength) {
        strncpy(output, input, maxLength - 3);
        output[maxLength - 3] = '\0';
        strcat(output, "...");
    } else {
        strcpy(output, input);
    }
}

void printListSize(ProfessionHead *pHead) {
    printShortLine();
    printf("| List size: %-37d |\n", pHead->count);
    printShortLine();
    printf("\n");
}
```

Контрольные примеры

Пример 1:

```
==================================================
|                  Choose an option              |
|------------------------------------------------|
| 0. Exit                                        |
| 1. Print all professions                       |
| 2. Select profession and reverse list          |
| 3. List carousel                               |
==================================================
Option: 2

==================================================
| Option: Select profession and reverse list     |
==================================================
==================================================
| List size: 8                                   |
==================================================
==================================================
| ID |                  Name                      |
|----|--------------------------------------------|
| 1  | pilot                                      |
| 2  | engineer                                   |
| 3  | teacher                                    |
| 4  | driver                                     |
| 5  | dentist                                    |
| 6  | actor                                      |
| 7  | writer                                     |
| 8  | musician                                   |
==================================================
Select the ID (after this a new list will be created that will contain all the elements of the
original list
except the element whose ID you specify. The order of the elements will be inverse in this
list)
ID: 4
Profession with id 4:
==================================================
| ID |                  Name                      |
|----|--------------------------------------------|
| 4  | driver                                     |
==================================================
Reversed list:
==================================================
| List size: 7                                   |
==================================================
==================================================
| ID |                  Name                      |
|----|--------------------------------------------|
| 1  | musician                                   |
| 2  | writer                                     |
| 3  | actor                                      |
| 4  | dentist                                    |
| 5  | teacher                                    |
| 6  | engineer                                   |
| 7  | pilot                                      |
==================================================
```

Пример 2:

```
=================================================
| Option: Select profession and reverse list    |
=================================================
=================================================
| List size: 8                                  |
=================================================
=================================================
| ID |                  Name                     |
|----|-------------------------------------------|
| 1  | pilot                                     |
| 2  | engineer                                  |
| 3  | teacher                                   |
| 4  | driver                                    |
| 5  | dentist                                   |
| 6  | actor                                     |
| 7  | writer                                    |
| 8  | musician                                  |
=================================================
Select the ID (after this a new list will be created that will contain all the
elements of the original list
except the element whose ID you specify. The order of the elements will be inverse in
this list)
ID: 10
Failed: there is no profession with id 10
Press ENTER to continue
```

Примеры выполнения программы

```
=====================================================
|                   Choose an option                |
|---------------------------------------------------|
| 0. Exit                                           |
| 1. Print all professions                          |
| 2. Select profession and reverse list             |
| 3. List carousel                                  |
=====================================================
Option: 2
```

```
| Option: Select profession and reverse list       |
================================================

================================================
| List size: 8                                 |
================================================

================================================
| ID |                Name                     |
|----|-----------------------------------------|
| 1  | pilot                                   |
| 2  | engineer                                |
| 3  | teacher                                 |
| 4  | driver                                  |
| 5  | dentist                                 |
| 6  | actor                                   |
| 7  | writer                                  |
| 8  | musician                                |
================================================

Select the ID (after this a new list will be created that will contain all the elements of the original list
except the element whose ID you specify. The order of the elements will be inverse in this list)

ID: 4

Profession with id 4:
================================================
| ID |                Name                     |
|----|-----------------------------------------|
| 4  | driver                                  |
================================================

Reversed list:
================================================
| List size: 7                                 |
================================================

================================================
| ID |                Name                     |
|----|-----------------------------------------|
| 1  | musician                                |
| 2  | writer                                  |
| 3  | actor                                   |
| 4  | dentist                                 |
| 5  | teacher                                 |
| 6  | engineer                                |
| 7  | pilot                                   |
================================================
Do you want to make sure that this list is circular?
1. Yes
2. No
Option: 2
```

```
================================================
| Option: Select profession and reverse list   |
================================================


================================================
| List size: 8                                 |
================================================

================================================
| ID |                 Name                    |
|----|-----------------------------------------|
| 1  | pilot                                   |
| 2  | engineer                                |
| 3  | teacher                                 |
| 4  | driver                                  |
| 5  | dentist                                 |
| 6  | actor                                   |
| 7  | writer                                  |
| 8  | musician                                |
================================================

Select the ID (after this a new list will be created that will contain all the elements of the original list
except the element whose ID you specify. The order of the elements will be inverse in this list)

ID: 10

Failed: there is no profession with id 10

Press ENTER to continue
```

Выводы.

В результате выполнения работы были получены практические навыки работы с кольцевыми односвязными списками в языке С.