Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий
институт
Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Лабораторная работа №3. Использование механизма виртуальной памяти в программах для ОС GNU/Linux
тема

Преподаватель _____ А.С. Кузнецов
подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 _____ Е.В. Железкин
номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2019

СОДЕРЖАНИЕ

1 Цель работы

Изучение работы механизма виртуальной памяти в программах для ОС GNU/Linux.

2 Краткие теоретические сведения

```
void *malloc(size_t size);
```

Функция **malloc()** возвращает адрес на первый байт области памяти размером size байт, которая была выделена из кучи. Если памяти недостаточно, чтобы удовлетворить запрос, функция **malloc()** возвращает нулевой указатель. Очень важно всегда проверять возвращаемое значение на его равенство NULL, прежде чем пытаться использовать этот указатель. Попытка использования нулевого указателя обычно влечет крах системы.

```
void free(void *ptr);
```

Функция **free()** возвращает память, на которую указывает параметр ptr, назад в кучу. В результате эта память может выделяться снова. Обязательным условием использования функции **free()** является то, что освобождаемая память должна была быть предварительно выделена с использованием одной из следующих функций: **malloc()**, **realloc()** или **calloc()**. Использование неверного указателя при вызове этой функции обычно ведет к разрушению механизма управления памятью и краху системы.

3 Задача работы

Требуется разработать программу в виде Linux-приложения, позволяющую манипулировать многоэлементными абстрактными структурами данных. Обязательны к реализации функции по добавлению одной структуры,

модификации структуры (значения всех полей или только части из них), удаления структуры, чтения одной структуры, отображения содержимого всех структур (или части). Должен использоваться интерфейс командной строки (CLI).

Вариант 12. Структура данных: кафедра; количество преподавателей; количество профессоров. Создать два запроса, позволяющих определить кафедры, где нет профессоров, и кафедры, в которых их доля максимальна.

4 Ход работы

Листинг 1 – содержание файла functions.h проекта SP_3:

```
#ifndef SP_3_FUNCTIONS_H
#define SP_3_FUNCTIONS_H

#include "structs.h"

int AddElement(struct department** mainStruct, char tString[],
               int tCount, int pCount);
int ModifyElement(struct department** currentStruct, int type, void* data);
int DeleteStruct(struct department** mainStruct);
int DeleteAllStruct(struct department** mainStruct);

#endif //SP_3_FUNCTIONS_H
```

Листинг 2 – содержание файла structs.h проекта SP_3:

```
#ifndef SP_3_STRUCTS_H
#define SP_3_STRUCTS_H
#define MAX_LENGTH 50

/*! \struct department
 *  \brief Структура, отвечающая за хранение кафедр.
 */
struct department
{
    /*! Название кафдеры. */
    char* name;
    /*! Число учителей. */
    double countOfTeachers;
    /*! Число профессоров. */
    double countOfProfessors;

    /*! Ссылка на следующий элемент структуры. */
    struct department* next;
    /*! Ссылка на предыдущий элемент структуры. */
    struct department* prev;
};
```

```c
/*! \struct pair
 *  \brief Структура, для работы с парой связных переменных.
 */
struct pair
{
    /*! Дробное значние в пары. */
    double value;
    /*! Название пары. */
    char name[MAX_LENGTH];
};

#endif //SP_3_STRUCTS_H
```

Листинг 3 – содержание файла InteractiveFunctions.h проекта SP_3:

```c
#ifndef SP_3_PRINTFUNCTIONS_H
#define SP_3_PRINTFUNCTIONS_H

#include "structs.h"

int InputChecker(char* strForCheck);
void InteractiveForAddElement (struct department** mainStruct);
void InteractiveForModifyElement(struct department** mainStruct);
void ShowInfoByNumber(struct department** mainStruct);
void ShowAllInfo(struct department** mainStruct);
void InteractiveForDeleteStruct(struct department** mainStruct);
void InteractiveForFirstRequest(struct department** mainStruct);
void InteractiveForSecondRequest(struct department** mainStruct);

#endif //SP_3_PRINTFUNCTIONS_H
```

Листинг 4 – содержание файла main.c проекта SP_3:

```c
#include <stdio.h>
#include <string.h>
#include "functions.h"
#include "InteractiveFunctions.h"
#include "structs.h"
#define INFINITY_LOOP for(;;)
#define MAX_LENGTH 50
#define ERROR -1
#define SUCCESS 0
#define CHOOSE_ADD 1
#define CHOOSE_SHOW_ONE 2
#define CHOOSE_SHOW_ALL 3
#define CHOOSE_MODIFY 4
#define CHOOSE_DELETE_ONE 5
#define CHOOSE_FIRST_REQUEST 6
#define CHOOSE_SECOND_REQUEST 7
#define CHOOSE_EXIT 8

/*! \brief Главная функция.
 *  @return Код завершения программы.
 */
int main()
```

```
{
    char input[MAX_LENGTH];
    int number = 0;

    struct department* dataStructure = NULL;

    INFINITY_LOOP
    {
        printf("Меню:\n"
               "\"1\" - Добавить новую кафедру в структуру;\n"
               "\"2\" - Вывести информацию о кафедре по её номеру;\n"
               "\"3\" - Вывести инфаормацию о всех кафедрах;\n"
               "\"4\" - Изменить информацию о кафедрах;\n"
               "\"5\" - Удалить кафедру по её номеру;\n"
               "\"6\" - Запрос, позволяющий определить кафедры, на которых "
               "нет профессоров;\n"
               "\"7\" - Запрос, позволяющий определить кафедру с наибольшим "
               "процентов профессоров на ней;\n"
               "\"8\" - Выход.\n");

        INFINITY_LOOP
        {
            scanf("%s", input);
            number = InputChecker(input);
            if (number < CHOOSE_ADD || number > CHOOSE_EXIT)
                printf("Введены некорректные данные, исправьте, "
                       "пожалуйста: ");
            else break;
        }

        switch (number)
        {
        case CHOOSE_ADD:
            InteractiveForAddElement(&dataStructure);
            break;
        case CHOOSE_SHOW_ONE:
            ShowInfoByNumber(&dataStructure);
            break;
        case CHOOSE_SHOW_ALL:
            ShowAllInfo(&dataStructure);
            break;
        case CHOOSE_MODIFY:
            InteractiveForModifyElement(&dataStructure);
            break;
        case CHOOSE_DELETE_ONE:
            InteractiveForDeleteStruct(&dataStructure);
            break;
        case CHOOSE_FIRST_REQUEST:
            InteractiveForFirstRequest(&dataStructure);
            break;
        case CHOOSE_SECOND_REQUEST:
            InteractiveForSecondRequest(&dataStructure);
            break;
        case CHOOSE_EXIT:
            if (DeleteAllStruct(&dataStructure) != SUCCESS)
                printf("Ошибка удаления структуры!\n");
            return 0;
            break;
        default:
```

```
            return 0;
        }

    }

}
```

Листинг 4 – содержание файла functions.c проекта SP_3:

```
#include <stddef.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "structs.h"
#define MAX_LENGTH 50
#define ERROR -1
#define SUCCESS 0
#define ASCII_CONST 48
#define NEXT_DIGIT 10
#define INFINITY_LOOP for(;;)
#define CHANGE_NAME 1
#define CHANGE_COUNT_OF_TEACHERS 2
#define CHANGE_COUNT_OF_PROFESSORS 3


/*! \brief Функция, добавляющая элемент в структуру.
 *  @param mainStruct Ссылка на первый элемент структуры.
 *  @param tString Строка с названием кафедры.
 *  @param tCount Количество учителей в добавляемой структуре.
 *  @param pCount Количество профессоров в добавляемой структуре.
 *  @return Код завершения функции.
 */
int AddElement(struct department** mainStruct, char tString[],
        int tCount, int pCount)
{
    struct department* temp = NULL;

    temp = (struct department*) malloc(sizeof(struct department));
    if (temp == NULL)
    {
        perror("malloc");
        return ERROR;
    }

    temp->name = (char*) malloc(sizeof(char) * (strlen(tString) + 1));
    if (temp->name == NULL)
    {
        perror("malloc");
        return ERROR;
    }
    strcpy(temp->name, tString);

    temp->countOfTeachers = tCount;

    temp->countOfProfessors = pCount;

    if (*mainStruct == NULL)
    {
        *mainStruct = temp;
```

```
        return SUCCESS;
    }

    struct department* temp2 = *mainStruct;

    while (temp2->next != NULL)
    {
        temp2 = temp2->next;
    }

    temp2->next = temp;
    temp->prev = temp2;

    return SUCCESS;
}

/*! \brief Функция, изменяющая элемент структуры.
 *  @param mainStruct Ссылка на первый элемент структуры.
 *  @param type Флаг, указывающий поле, которое необходимо изменить.
 *  @param data Ссылка на область памяти с новой информацией для
 *  изменяемого поля.
 *  @return Код завершения функции.
 */
int ModifyElement(struct department** currentStruct, int type, void* data)
{

    if (*currentStruct == NULL)
        return ERROR;

    if (type == CHANGE_NAME)
    {
        char* input = data;

        if ((*currentStruct)->name != NULL) free((*currentStruct)->name);

        (*currentStruct)->name = (char*) malloc(sizeof(char) *
                                                (strlen(input) + 1));
        if ((*currentStruct)->name == NULL)
        {
            perror("malloc");
            return ERROR;
        }

        strcpy((*currentStruct)->name, input);
        free(data);
        return SUCCESS;
    }

    if (type == CHANGE_COUNT_OF_TEACHERS)
    {
        int value = *(int*)data;
        (*currentStruct)->countOfTeachers = value;
        free(data);
        return SUCCESS;
    }

    if (type == CHANGE_COUNT_OF_PROFESSORS)
    {
        int value = *(int*)data;
        (*currentStruct)->countOfProfessors = value;
```

```
        free(data);
        return SUCCESS;
    }

    return ERROR;

}

/*! \brief Функция, удаляющая элемент структуры.
 *  @param mainStruct Ссылка на первый элемент структуры.
 *  @return Код завершения функции.
 */
int DeleteStruct(struct department** mainStruct)
{

    if (*mainStruct == NULL)
        return ERROR;

    if ((*mainStruct)->next == NULL)
    {
        (*mainStruct)->prev->next = NULL;
        free((*mainStruct)->name);
        free((*mainStruct));
        return SUCCESS;
    }

    if ((*mainStruct)->prev == NULL)
    {
        (*mainStruct)->next->prev = NULL;
        free((*mainStruct)->name);
        free((*mainStruct));
        return SUCCESS;
    }

    (*mainStruct)->prev->next = (*mainStruct)->next;
    (*mainStruct)->next->prev = (*mainStruct)->prev;
    free((*mainStruct)->name);
    free((*mainStruct));

    return SUCCESS;

}

/*! \brief Функция, удаляющая структуру.
 *  @param mainStruct Ссылка на первый элемент структуры.
 *  @return Код завершения функции.
 */
int DeleteAllStruct(struct department** mainStruct)
{

    if (*mainStruct == NULL) return ERROR;

    struct department* temp = NULL;

    while (*mainStruct != NULL)
    {
        temp = *mainStruct;
        *mainStruct = (*mainStruct)->next;
        free(temp->name);
        free(temp);
```

```
    }

    return SUCCESS;

}
```

Листинг 4 – содержание файла InteractiveFunctions.c проекта SP_3:

```
#include <stddef.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "structs.h"
#include "functions.h"
#define MAX_LENGTH 50
#define ERROR -1
#define SUCCESS 0
#define ASCII_CONST 48
#define NEXT_DIGIT 10
#define INFINITY_LOOP for(;;)
#define CHANGE_NAME 1
#define CHANGE_COUNT_OF_TEACHERS 2
#define CHANGE_COUNT_OF_PROFESSORS 3

/*! \brief Функция, обрабатывающая ввод.
 *  @param parameters Строка считанная с клавиатуры.
 *  @return number Число, содержащееся в строке, либо ошибка.
 */
int InputChecker(char* strForCheck)
{
    int number = 0;

    for (unsigned long i = 0; i < strlen(strForCheck); i++)
    {
        if (strForCheck[i] == '0' || strForCheck[i] == '1' ||
            strForCheck[i] == '2' || strForCheck[i] == '3' ||
            strForCheck[i] == '4' || strForCheck[i] == '5' ||
            strForCheck[i] == '6' || strForCheck[i] == '7' ||
            strForCheck[i] == '8' || strForCheck[i] == '9') number =
                    number * NEXT_DIGIT + (strForCheck[i] - ASCII_CONST);
        else
            return ERROR;
    }

    return number;
}

/*! \brief Функция, обработки ввода-вывода для функции AddElement.
 *  @param mainStruct Ссылка на первый элемент структуры.
 */
void InteractiveForAddElement (struct department** mainStruct)
{

    char tString[MAX_LENGTH];
    printf("Введите название кафедры: ");
    scanf("%s", tString);

    int tCount = 0;
    int pCount = 0;
```

```
    char input[MAX_LENGTH];
    printf("Введите количество учителей: ");
    INFINITY_LOOP {
        scanf("%s", input);
        tCount = InputChecker(input);
        if (tCount == ERROR)
            printf("Введены некорректные данные, исправьте, "
                   "пожалуйста: ");
        else break;
    }

    printf("Введите количество профессоров: ");
    INFINITY_LOOP {
        scanf("%s", input);
        pCount = InputChecker(input);
        if (pCount == ERROR)
            printf("Введены некорректные данные, исправьте, "
                   "пожалуйста: ");
        else break;
    }

    if (AddElement(mainStruct, tString, tCount, pCount) != SUCCESS)
        printf("Ошибка добавления элемента!\n");

}

/*! \brief Функция, обработки ввода-вывода для функции ModifyElement.
 *  @param mainStruct Ссылка на первый элемент структуры.
 */
void InteractiveForModifyElement(struct department** mainStruct)
{
    struct department* temp = *mainStruct;
    int tCount = 0;
    int flag = 0;
    char input[MAX_LENGTH];

    INFINITY_LOOP {

        if (flag == 1)
        {
            printf("Указанный элемент отсутствует!\n");
        }
        flag = 0;

        printf("Введите номер структуры, которую "
               "необходимо изменить: ");
        INFINITY_LOOP {
            scanf("%s", input);
            tCount = InputChecker(input);
            if (tCount == ERROR || tCount == 0)
                printf("Введены некорректные данные, исправьте, "
                       "пожалуйста: ");
            else break;
        }

        if (temp == NULL)
        {
            printf("Структура пуста!\n");
            return;
        }
```

```
    for (int i = 1; i < tCount; i++)
    {
        if (temp->next == NULL && i + 1 == tCount)
        {
            flag = 1;
            break;
        }
        temp = temp->next;
    }

    if (flag == 0) break;
}

INFINITY_LOOP
{
    printf("Выберите поле, которое необходимо изменить:\n"
           "1 - Название кафедры;\n"
           "2 - Количество учителей на кафедре;\n"
           "3 - Количество профессоров на кафедре.\n");
    INFINITY_LOOP {
        scanf("%s", input);
        tCount = InputChecker(input);
        if (tCount < 1 || tCount > 3)
            printf("Введены некорректные данные, исправьте, "
                   "пожалуйста: ");
        else break;
    }

    if (tCount == CHANGE_NAME)
    {
        printf("Введите новое название кафедры: ");
        scanf("%s", input);
        void* par = (char*) malloc(sizeof(char) * strlen(input));
        strcpy(par, input);
        if (ModifyElement(&temp, 1, par) != SUCCESS)
            printf("Ошибка изменения имени!\n");
        return;
    }

    if (tCount == CHANGE_COUNT_OF_TEACHERS)
    {
        printf("Введите новое число учителей: ");
        INFINITY_LOOP {
            scanf("%s", input);
            tCount = InputChecker(input);
            if (tCount == ERROR)
                printf("Введены некорректные данные, исправьте, "
                       "пожалуйста: ");
            else break;
        }

        void* par = malloc(sizeof(int));
        *(int*)par = tCount;
        if (ModifyElement(&temp, 2, par) != SUCCESS)
            printf("Ошибка изменения числа учителей!\n");

        return;
    }
```

```
        if (tCount == CHANGE_COUNT_OF_PROFESSORS)
        {
            printf("Введите новое число профессоров: ");
            INFINITY_LOOP {
                scanf("%s", input);
                tCount = InputChecker(input);
                if (tCount == ERROR)
                    printf("Введены некорректные данные, исправьте, "
                           "пожалуйста: ");
                else break;
            }

            void* par = malloc(sizeof(int));
            *(int*)par = tCount;
            if (ModifyElement(&temp, 3, par) != SUCCESS)
                printf("Ошибка изменения числа профессоров!\n");

            return;
        }
    }

}

/*! \brief Функция, выводящая элемент по его номеру.
 *  @param mainStruct Ссылка на первый элемент структуры.
 */
void ShowInfoByNumber(struct department** mainStruct)
{
    if (*mainStruct == NULL)
    {
        printf("Структура пуста!\n");
        return;
    }

    int number;
    char input[MAX_LENGTH];
    struct department* temp = *mainStruct;

    INFINITY_LOOP {
        scanf("%s", input);
        number = InputChecker(input);
        if (number == ERROR || number == 0)
            printf("Введены некорректные данные, исправьте, "
                   "пожалуйста: ");
        else break;
    }

    for (int i = 1; i < number; i++)
    {
        if (temp->next == NULL)
        {
            printf("Отсутствует указанный элемент!\n");
            return;
        }
        temp = temp->next;
    }

    printf("Название кафедры: %s\nКоличество учителей:"
           " %g\nКоличество профессоров: %g\n",
           temp->name, temp->countOfTeachers, temp->countOfProfessors);
```

```
}

/*! \brief Функция, выводящая все элементы структуры.
 *  @param mainStruct Ссылка на первый элемент структуры.
 */
void ShowAllInfo(struct department** mainStruct)
{
    if (*mainStruct == NULL)
    {
        printf("Структура пуста!\n");
        return;
    }

    struct department* temp = *mainStruct;

    while (temp != NULL)
    {
        printf("Название кафедры: %s;   Количество учителей:"
               " %g;   Количество профессоров: %g\n",
               temp->name, temp->countOfTeachers, temp->countOfProfessors);
        temp = temp->next;
    }
}

/*! \brief Функция, обработки ввода-вывода для функции DeleteStruct.
 *  @param mainStruct Ссылка на первый элемент структуры.
 */
void InteractiveForDeleteStruct(struct department** mainStruct)
{
    int number;
    char input[MAX_LENGTH];

    INFINITY_LOOP {
        printf("Введите номер удаляемого элемента: ");
        scanf("%s", input);
        number = InputChecker(input);
        if (number == ERROR || number == 0)
            printf("Введены некорректные данные, исправьте, "
                   "пожалуйста: ");
        else break;
    }

    if (*mainStruct == NULL)
    {
        printf("Структура пуста!\n");
        return;
    }

    struct department* temp = *mainStruct;

    for (int i = 1; i < number; i++)
    {
        if (temp->next == NULL)
        {
            printf("Отсутствует указанный элемент!\n");
            return;
        }
        temp = temp->next;
    }
```

```
    if (DeleteStruct(&temp) != SUCCESS)
        printf("Ошибка удаления элемента структуры!\n");

}

/*! \brief Функция, обрабатывающая запрос 1.
 *  @param mainStruct Ссылка на первый элемент структуры.
 */
void InteractiveForFirstRequest(struct department** mainStruct)
{
    if (*mainStruct == NULL)
    {
        printf("Структура пуста!\n");
        return;
    }

    struct department* temp = *mainStruct;
    int flag = 0;

    while(temp != NULL)
    {
        if (temp->countOfProfessors == 0) {
            printf("На кафедре \"%s\" нет профессоров!\n",
                   temp->name);
            flag++;
        }
        temp = temp->next;
    }

    if (flag == 0)
    {
        printf("На всех кафедрах есть профессора!\n");
    }

}

void free(void *ptr);

/*! \brief Функция, обрабатывающая запрос 2.
 *  @param mainStruct Ссылка на первый элемент структуры.
 */
void InteractiveForSecondRequest(struct department** mainStruct)
{
    struct pair max = {0, 0};
    struct department* temp = *mainStruct;

    while (temp != NULL)
    {
        if (max.value < temp->countOfProfessors /
                        (temp->countOfProfessors + temp->countOfTeachers))
        {
            max.value = temp->countOfProfessors / (temp->countOfProfessors
                                                   + temp->countOfTeachers);
            strcpy(max.name, temp->name);
        }
        temp = temp->next;
    }

    temp = *mainStruct;
```

```
    while (temp != NULL)
    {
        if (max.value == temp->countOfProfessors /
                         (temp->countOfProfessors + temp->countOfTeachers))
        {
            printf("На кафедре \"%s\" доля профессоров "
                   "максимальна\n", temp->name);
        }
        temp = temp->next;
    }

}
```

1) Примеры работы:

```
/mnt/c/Users/super/Documents/Stud/SP/SP_3/cmake-build-debug/SP_3
Меню:
"1" - Добавить новую кафедру в структуру;
"2" - Вывести информацию о кафедре по её номеру;
"3" - Вывести инфаормацию о всех кафедрах;
"4" - Изменить информацию о кафедрах;
"5" - Удалить кафедру по её номеру;
"6" - Запрос, позволяющий определить кафедры, на которых нет профессоров;
"7" - Запрос, позволяющий определить кафедру с наибольшим процентов философов на ней;
"8" - Выход.
1
1
Введите название кафедры: assaf
assaf
Введите количество учителей: 5
5
Введите количество профессоров: 3
3
```

Рисунок 1 – результат работы с пунктом меню «1»

```
Меню:
"1" - Добавить новую кафедру в структуру;
"2" - Вывести информацию о кафедре по её номеру;
"3" - Вывести инфаормацию о всех кафедрах;
"4" - Изменить информацию о кафедрах;
"5" - Удалить кафедру по её номеру;
"6" - Запрос, позволяющий определить кафедры, на которых нет профессоров;
"7" - Запрос, позволяющий определить кафедру с наибольшим процентов философов на ней;
"8" - Выход.
3
3
Название кафедры: assaf;   Количество учителей: 5;   Количество профессоров: 3
Название кафедры: qweqwrq;   Количество учителей: 5;   Количество профессоров: 0
Название кафедры: ijeoirjoier;   Количество учителей: 2;   Количество профессоров: 2
```

Рисунок 2 – результат работы с пунктом меню «3»

```
Меню:
"1" - Добавить новую кафедру в структуру;
"2" - Вывести информацию о кафедре по её номеру;
"3" - Вывести инфаормацию о всех кафедрах;
"4" - Изменить информацию о кафедрах;
"5" - Удалить кафедру по её номеру;
"6" - Запрос, позволяющий определить кафедры, на которых нет профессоров;
"7" - Запрос, позволяющий определить кафедру с наибольшим процентов философов на ней;
"8" - Выход.
6
6
На кафедре "qweqwrq" нет профессоров!
```

Рисунок 3 - результат работы с пунктом меню «6»

```
Меню:
"1" - Добавить новую кафедру в структуру;
"2" - Вывести информацию о кафедре по её номеру;
"3" - Вывести инфаормацию о всех кафедрах;
"4" - Изменить информацию о кафедрах;
"5" - Удалить кафедру по её номеру;
"6" - Запрос, позволяющий определить кафедры, на которых нет профессоров;
"7" - Запрос, позволяющий определить кафедру с наибольшим процентов философов на ней;
"8" - Выход.
7
7
На кафедре "ijeoirjoier" доля профессоров максимальна
```

Рисунок 4 – результат работы с пунктом меню «7»

```
Меню:
"1" - Добавить новую кафедру в структуру;
"2" - Вывести информацию о кафедре по её номеру;
"3" - Вывести инфаормацию о всех кафедрах;
"4" - Изменить информацию о кафедрах;
"5" - Удалить кафедру по её номеру;
"6" - Запрос, позволяющий определить кафедры, на которых нет профессоров;
"7" - Запрос, позволяющий определить кафедру с наибольшим процентов философов на ней;
"8" - Выход.
5
5
Введите номер удаляемого элемента: 2
2
Меню:
"1" - Добавить новую кафедру в структуру;
"2" - Вывести информацию о кафедре по её номеру;
"3" - Вывести инфаормацию о всех кафедрах;
"4" - Изменить информацию о кафедрах;
"5" - Удалить кафедру по её номеру;
"6" - Запрос, позволяющий определить кафедры, на которых нет профессоров;
"7" - Запрос, позволяющий определить кафедру с наибольшим процентов философов на ней;
"8" - Выход.
3
3
Название кафедры: assaf;   Количество учителей: 5;   Количество профессоров: 3
Название кафедры: ijeoirjoier;   Количество учителей: 2;   Количество профессоров: 2
```

Рисунок 5 – результат работы с пунктом меню «5»

5 Вывод

В ходе данной лабораторной работы были изучены азы работы механизма виртуальной памяти в программах для ОС GNU/Linux.