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

ОТЧЕТ

по практической работе №1

по дисциплине «Программирование»

Тема: Типы данных, определяемые пользователем. Структуры.

Студент гр. 0323 _____ Балашевич К.Д.

Преподаватель _____ Глущенко А.Г.

Санкт-Петербург

2021

Цель работы.

Изучение и организация структур; получение практических навыков работы со структурами; определение преимуществ и недостатков использования структур.

Основные теоретические положения.

В языке C++ структуры представляют собой частный случай класса, сохраняя при этом синтаксис, унаследованный от языка си. Определение структуры осуществляется с использованием ключевого слова struct и выглядит следующим образом:

```
struct ИМЯ_СТРУКТУРЫ
{
....//Поля структуры
} СПИСОК_ОПИСАТЕЛЕЙ ;
```

Структурную переменные возможно объявлять как при определении структуры, так и в других участках кода. В последнем случае - с использованием следующего синтаксиса:

```
ИМЯ_СТРУКТУРЫ ИМЯ_ПЕРЕМЕННОЙ;
```

Возможно и объявление массивов структурных пременных. При этом используется синтаксис, аналогичный обычным переменных.

Доступ к полям структурам осуществляется с использованием следующего синтаксиса:

```
ИМЯ_ПЕРЕМЕННОЙ.ИМЯ_ПОЛЯ
```

Поля структуры представляют собой обычные переменные, располагаемые в памяти ЭВМ последовательно (с учётом выравнивания по минимальный блокам памяти, зависящим от платформы и реализации).

Последовательное расположение полей структуры в памяти позволяет обращаться к полям структуры с испорльзованием указателей, при этом используется следующий синтаксис:

```
УКАЗАТЕЛЬ_НА_ПЕРЕМЕННУЮ->ИМЯ_ПОЛЯ
```

В языке C++ сохраняется унаследованный от языка Си синтаксис определения структур с использованием ключевого слова typedef, однако никаких преимуществ перед определением структур без этого слова данный вариант не имеет.

В качестве поля структуры возможно и нерекурсивное использование других структур.

Структуры могут быть переданы в функцию как по значению, так и по ссылке (указателю), а также и возвращены ею. Это позволяет, в частности передавать по значению в функции массивы, а также возвращать их из функции, «упаковав» их в структуру.

Постановка задачи.

Создать массив структур, содержащий информацию о студентах: ФИО, пол, номер группы, номер в списке группы, оценки за прошедшую сессию (всего 3 экзамена и 5 дифференцированных зачетов), форма обучения, отметка времени о внесении или изменении данных. Ввод и изменение данных обо всех студентах должен осуществляться в файл students.

Написать функции, реализующие операции со структурами (ввод данных с клавиатуры):

1) Создание новой записи о студенте.
2) Внесение изменений в уже имеющуюся запись.
3) Вывод всех данных о студентах.
4) Вывод информации обо всех студентах группы N. N – инициализируется пользователем.
5) Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.
6) Вывод количества студентов мужского и женского пола.
7) Определение количества студентов, которые будут получать стипендию (стипендия начисляется, если у студента нет троек и очная форма обучения).
8) Вывод данных о студентах, которые не получают стипендию; учатся

только на «хорошо» и «отлично»; учатся только на «отлично»;

9) Вывод данных о студентах, имеющих номер в списке – k.

10) Вывод всех записей, сделанных в день, который введет пользователь. Вывод всех записей, сделанных после полудня. Вывод всех записей, сделанных до полудня.

Сделать вывод по проделанной работе.

Выполнение работы.

Для решения поставленной задачи была написана программа на языке C++. Итоговый код программы представлен в приложении А.

Для хранения данных во внешнем файле был использован следующий формат:

```
ИМЯ_СТУДЕНТА
ПОЛ_СТУДЕНТА (1-мужской, 2-женский)
ГРУППА_СТУДЕНТА
НОМЕР_СТУДЕНТА_В_СПИСКЕ_ГРУППЫ
ОЦЕНКИ_СТУДЕНТА (каждая оценка представлена цифрой на новой строке)
ФОРМА_ОБУЧЕНИЯ_СТУДЕНТОВ (1-очная, 2-очно-заочная, 3-заочная)
ВРЕМЯ_СОЗДАНИЯ/ИЗМЕНЕНИЯ_ЗАПИСИ (В формате Unix-time)
```

Было проведено тестирование программы с использование различных ЭВМ и компиляторов. Результаты тестирования представлены в приложении Б. При этом были получены результаты, соответствующие расчётным.

Выводы.

Структуры являются довольно удобным для использования эллементом языка C++, хоть и уступают пости во всём классическим классам и служат, больше, для переносимости программ, написаных на языке Си.

ПРИЛОЖЕНИЕ А
ПОЛНЫЙ КОД ПРОГРАММЫ

```
////////////////////////////////////////////////////////////////////
//                              ВНИМАНИЕ                          //
////////////////////////////////////////////////////////////////////
//Для компиляции под ОС Windows с использованием компилятора MSVC//
//              раскомментируйте следующую строку:                //
#define CLEARSCREEN "cls"                                         //
////////////////////////////////////////////////////////////////////
//         Для компиляции под POSIX-совместимой ОС или ОС         //
//          Windows с использованием компилятора GCC/G++          //
//               раскомментируйте следующую строку:               //
//#define CLEARSCREEN "clear"                                     //
////////////////////////////////////////////////////////////////////

#include <memory>
#include <iostream>
#include <fstream>
#include <cstring>
#include <ctime>
#include <limits>

struct student
  {
    char Name[101];
    short Sex;
    char Group[11];
    short Number;
    short Ekz1;
    short Ekz2;
    short Ekz3;
    short Diff1;
    short Diff2;
    short Diff3;
    short Diff4;
    short Diff5;
    short Form;
    time_t modificationTime;
  };

student* mainMenu (student *List);
student* addStudentFromTerminal (student *List);
```

```
int getNumericAnswer (int minRange, int maxRange, const char
*question);
student* addStudentsFromFile (student *List);
student*    dismissStudent    (student    *List,    unsigned    int
studentNumber);
void studentList (student *List);
void studentMenu (student *List, unsigned int studentNumber);
void printStudentInfo (student *List, unsigned int studentNumber);
void dumpListToFile (student *List);
void studentGradesMenu (student *Student);
void changeStudentGrades (student *Student, short request);
void changeStudentName (student *Student, bool isNew);
void changeStudentSex (student *Student, bool isNew);
void changeStudentGroup (student *Student, bool isNew);
void changeStudentNumber (student *Student, bool isNew);
void changeStudentForm (student *Student, bool isNew);
void listFilterChoise (student *List);
void listFilter (student *List, bool length);
void listGroup (student *List, bool length, const char *Group);
void listTop (student *List, bool length);
void listSex (student *List, bool length);
void listStipend (student *List, bool length);
void listStipendLess (student *List, bool length);
void listFourFive (student *List, bool length);
void listFive (student *List, bool length);
void listNumber (student *List, bool length, int Number);
void listTime (student *List, bool length, tm Begin, int Noon);
tm getTmDate ();

unsigned int listSize;

int main ()
  {
    setlocale(0, "");
    listSize=0;
    student *List=NULL;
    List=mainMenu (List);
    free(List);
    return 0;
  }

student* mainMenu (student *List)
  {
    system (CLEARSCREEN);
```

```
     std::cout << "Главное меню:" << std::endl << std::endl << "1) Вывести полный список студентов." << std::endl << "2) Добавить студента из терминала." << std::endl << "3) Добавить студентов из файла." << std::endl << "4) Вывести список студентов по фильтру." << std::endl << "5) Вывод всех данных в файл." << std::endl << "0) Выход из программы." << std::endl << std::endl;
    switch (getNumericAnswer (0, 5, "Введите номер желаемого варианта: "))
      {
        case 1:
          studentList(List);
          List=mainMenu(List);
          break;
        case 2:
          List=addStudentFromTerminal(List);
          List=mainMenu(List);
          break;
        case 3:
          List=addStudentsFromFile(List);
          List=mainMenu(List);
          break;
        case 4:
          listFilterChoise(List);
          List=mainMenu(List);
          break;
        case 5:
          dumpListToFile(List);
          List=mainMenu(List);
          break;
        default:
          break;
      }
    return List;
  }

student* addStudentFromTerminal (student *List)
  {
    system (CLEARSCREEN);
    std::cout << "Добавление студента:" << std::endl << std::endl;
    listSize++;
    List=(student*)realloc(List, listSize*sizeof(student));
    student *Student=List+listSize-1;
    changeStudentName(Student, 1);
    std::cout << std::endl;
    changeStudentSex(Student, 1);
```

```
    std::cout << std::endl;
    changeStudentGroup(Student, 1);
    std::cout << std::endl;
    changeStudentNumber(Student, 1);
    std::cout << std::endl;
    changeStudentGrades(Student, 1);
    std::cout << std::endl;
    changeStudentGrades(Student, 2);
    std::cout << std::endl;
    changeStudentGrades(Student, 3);
    std::cout << std::endl;
    changeStudentGrades(Student, 4);
    std::cout << std::endl;
    changeStudentGrades(Student, 5);
    std::cout << std::endl;
    changeStudentGrades(Student, 6);
    std::cout << std::endl;
    changeStudentGrades(Student, 7);
    std::cout << std::endl;
    changeStudentGrades(Student, 8);
    std::cout << std::endl;
    changeStudentForm(Student, 1);
    Student->modificationTime=time (NULL);
    return List;
  }

int getNumericAnswer (int minRange, int maxRange, const char
*Question)
  {
    int answer;
    std::cout << Question;
    std::cin >> answer;
    if ((std::cin.fail())||(answer<minRange)||(answer>maxRange))
      {
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
        std::cout << "Некорректный ввод!" << std::endl;
        answer=getNumericAnswer (minRange, maxRange, Question);
        return answer;
      }
    else
      {
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
```

```
        return answer;
      }
  }

student* addStudentsFromFile (student *List)
  {
    char fileName[261];
    std::cout << std::endl << "Введите имя (или путь) файла (по
умолчанию - students): ";
    std::cin.getline (fileName, 261);
    if (fileName[0]=='\0') strcpy(fileName, "students");
    std::ifstream File;
    File.open (fileName);
    if (File.is_open())
      {
        while (!File.eof())
          {
            listSize++;
            List=(student*)realloc(List, listSize*sizeof(student));
            student *newStudent;
            newStudent=List+listSize-1;
            File.getline (newStudent->Name, 101);
            File >> newStudent->Sex;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File.getline (newStudent->Group, 11);
            File >> newStudent->Number;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File >> newStudent->Ekz1;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File >> newStudent->Ekz2;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File >> newStudent->Ekz3;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File >> newStudent->Diff1;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File >> newStudent->Diff2;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File >> newStudent->Diff3;
```

```
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File >> newStudent->Diff4;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File >> newStudent->Diff5;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File >> newStudent->Form;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            File >> newStudent->modificationTime;
            File.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
          }
        File.close();
      }
    else
      {
        std::cout << "Некорректный ввод!" << std::endl;
        List=addStudentsFromFile(List);
      }
    return List;
  }

void studentList (student *List)
  {
    system (CLEARSCREEN);
    std::cout << "Список студентов в базе:" << std::endl <<
std::endl;
    for (unsigned int i=0; i<listSize; i++)
      {
        std::cout << i+1 << ". " << (List+i)->Name << std::endl;
      }
    std::cout << std::endl;
   int answer=getNumericAnswer(0, listSize, "Введите номер студента
в списке, карточку которого вы хотите посмотреть (для возврата в
главное меню введите 0): ");
    if (answer)
      {
        studentMenu (List, answer);
        studentList (List);
      }
  }
```

```
void studentMenu (student *List, unsigned int studentNumber)
  {
    system (CLEARSCREEN);
    printStudentInfo (List, studentNumber);
    std::cout << std::endl << std::endl << "Меню работы с данными студента:" << std::endl << std::endl << "1) Изменить сведения о ФИО студента." << std::endl << "2) Изменить сведения о поле студента." << std::endl << "3) Изменить сведения о группе студента." << std::endl << "4) Изменить сведения о номере студента в списке группы." << std::endl << "5) Изменить сведения об оценках студента." << std::endl << "6) Изменить сведения о форме обучения студента." << std::endl << "0) Вернуться в главное меню." << std::endl << std::endl;
   student *Student=List+studentNumber-1;
    switch (getNumericAnswer (0, 6, "Введите номер желаемого варианта: "))
      {
        case 1:
          changeStudentName (Student, 0);
          Student->modificationTime=time(NULL);
          studentMenu (List, studentNumber);
          break;
        case 2:
          changeStudentSex (Student, 0);
          Student->modificationTime=time(NULL);
          studentMenu (List, studentNumber);
          break;
        case 3:
          changeStudentGroup (Student, 0);
          Student->modificationTime=time(NULL);
          studentMenu (List, studentNumber);
          break;
        case 4:
          changeStudentNumber (Student, 0);
          Student->modificationTime=time(NULL);
          studentMenu (List, studentNumber);
          break;
        case 5:
          studentGradesMenu (Student);
          studentMenu (List, studentNumber);
          break;
        case 6:
          changeStudentForm (Student, 0);
          Student->modificationTime=time(NULL);
          studentMenu (List, studentNumber);
```

```
          break;
        default:
          break;
      }
  }

void printStudentInfo (student *List, unsigned int studentNumber)
  {
    student *Student=List+studentNumber-1;
    std::cout << "Студент " << studentNumber << std::endl << std::endl << "ФИО студента: " << Student->Name << std::endl << "Пол студента:";
    switch (Student->Sex)
      {
        case 1:
          std::cout << " мужской.";
          break;
        default:
          std::cout << " женский.";
          break;
      }
    std::cout << std::endl << "Номер группы студента: " << Student->Group << std::endl << "Номер студента в списке группы: " << Student->Number << std::endl << "Оценка студента за первый экзамен: " << Student->Ekz1 << std::endl << "Оценка студента за второй экзамен: " << Student->Ekz2 << std::endl << "Оценка студента за третий экзамен: " << Student->Ekz3 << std::endl << "Оценка студента за первый зачёт: " << Student->Diff1 << std::endl << "Оценка студента за второй зачёт: " << Student->Diff2 << std::endl << "Оценка студента за третий зачёт: " << Student->Diff3 << std::endl << "Оценка студента за четвёртый зачёт: " << Student->Diff4 << std::endl << "Оценка студента за пятый зачёт: " << Student->Diff5 << std::endl << "Форма обучения студента: ";
    switch (Student->Form)
      {
        case 1:
          std::cout << "очная.";
          break;
        case 2:
          std::cout << "очно-заочная.";
          break;
        default:
          std::cout << "заочная.";
          break;
      }
```

```
       std::cout << std::endl << "Дата внесения изменений в запись:
" << asctime(localtime(&Student->modificationTime));
  }

void changeStudentName (student *Student, bool isNew)
  {
     if (!isNew) std::cout << std::endl << "Текущие сведения о ФИО
студента: " << Student->Name << std::endl;
    std::cout << "Введите ФИО студента: ";
    std::cin.getline (Student->Name, 101);
  }

void changeStudentSex (student *Student, bool isNew)
  {
    if (!isNew)
      {
       std::cout << std::endl << "Текущие сведения о поле студента:
";
       switch (Student->Sex)
         {
           case 1:
             std::cout << "мужской." << std::endl;
             break;
           default:
             std::cout << "женский." << std::endl;
             break;
         }
      }
     std::cout << "Выберите пол студента:" << std::endl << "1)
мужской пол." << std::endl << "2) женский пол." << std::endl;
     Student->Sex=getNumericAnswer (1, 2, "Введите номер желаемого
варианта: ");
  }

void changeStudentGroup (student *Student, bool isNew)
  {
      if (!isNew) std::cout << std::endl << "Текущие сведения о
группе студента: " << Student->Group << std::endl;
    std::cout << "Введите номер группы студента: ";
    std::cin.getline (Student->Group, 11);
  }

void changeStudentNumber (student *Student, bool isNew)
  {
```

```
   if (!isNew) std::cout << std::endl << "Текущие сведения о номере студента в списке группы: " << Student->Number << std::endl;
   Student->Number=getNumericAnswer(0, 100, "Введите номер студента в списке группы: ");
  }

void studentGradesMenu (student *Student)
  {
    system (CLEARSCREEN);
    std::cout << "Оценки студента: " << std::endl << std::endl << "Оценка студента за первый экзамен: " << Student->Ekz1 << std::endl << "Оценка студента за второй экзамен: " << Student->Ekz2 << std::endl << "Оценка студента за третий экзамен: " << Student->Ekz3 << std::endl << "Оценка студента за первый зачёт: " << Student->Diff1 << std::endl << "Оценка студента за второй зачёт: " << Student->Diff2 << std::endl << "Оценка студента за третий зачёт: " << Student->Diff3 << std::endl << "Оценка студента за четвёртый зачёт: " << Student->Diff4 << std::endl << "Оценка студента за пятый зачёт: " << Student->Diff5 << std::endl << std::endl << "Вы хотите изменить сведения о зачёте или экзамене?" << std::endl << "1) Изменить оценку за зачёт." << std::endl << "2) Изменить оценку за экзамен." << std::endl << "0) Вернуться в предыдущее меню." << std::endl;
       switch (getNumericAnswer(0, 2, "Введите номер желаемого варианта: "))
      {
        case 1:
          switch (getNumericAnswer(1,5,"Введите номер зачёта, оценку за который вы желаете изменить: "))
            {
              case 1:
                changeStudentGrades(Student, 4);
                break;
              case 2:
                changeStudentGrades(Student, 5);
                break;
              case 3:
                changeStudentGrades(Student, 6);
                break;
              case 4:
                changeStudentGrades(Student, 7);
                break;
              default:
                changeStudentGrades(Student, 8);
                break;
```

```
            }
          Student->modificationTime=time (NULL);
          studentGradesMenu (Student);
          break;
       case 2:
        switch (getNumericAnswer(1,3,"Введите номер экзамена, оценку
за который вы желаете изменить: "))
            {
              case 1:
                changeStudentGrades(Student, 1);
                break;
              case 2:
                changeStudentGrades(Student, 2);
                break;
              default:
                changeStudentGrades(Student, 3);
                break;
            }
          Student->modificationTime=time (NULL);
          studentGradesMenu (Student);
          break;
        default:
          break;
      }
  }

void changeStudentGrades (student *Student, short request)
  {
    switch (request)
      {
        case 1:
           Student->Ekz1=getNumericAnswer(2, 5, "Пожалуйста, введите
оценку студента за первый экзамен: ");
          break;
        case 2:
           Student->Ekz2=getNumericAnswer(2, 5, "Пожалуйста, введите
оценку студента за второй экзамен: ");
          break;
        case 3:
           Student->Ekz3=getNumericAnswer(2, 5, "Пожалуйста, введите
оценку студента за третий экзамен: ");
          break;
        case 4:
                 Student->Diff1=getNumericAnswer(2,  5,  "Пожалуйста,
введите оценку студента за первый зачёт: ");
```

```
          break;
        case 5:
          Student->Diff2=getNumericAnswer(2, 5, "Пожалуйста, введите
оценку студента за второй зачёт: ");
          break;
        case 6:
          Student->Diff3=getNumericAnswer(2, 5, "Пожалуйста, введите
оценку студента за третий зачёт: ");
          break;
        case 7:
          Student->Diff4=getNumericAnswer(2, 5, "Пожалуйста, введите
оценку студента за четвёртый зачёт: ");
          break;
        case 8:
          Student->Diff5=getNumericAnswer(2, 5, "Пожалуйста, введите
оценку студента за пятый зачёт: ");
          break;
      }
  }

void changeStudentForm (student *Student, bool isNew)
  {
    if (!isNew)
      {
       std::cout << std::endl << "Текущие сведения о форме обучения
студента: ";
        switch (Student->Form)
          {
            case 3:
              std::cout << "заочная форма." << std::endl;
              break;
            case 2:
              std::cout << "очно-заочная форма." << std::endl;
              break;
            default:
              std::cout << "очная форма." << std::endl;
              break;
          }
      }
     std::cout << "Выберите форму обучения студента:" << std::endl
<< "1) Очная форма." << std::endl << "2) Очно-заочная форма." <<
std::endl << "3) Заочная форма." << std::endl;
     Student->Form=getNumericAnswer (1, 3, "Введите номер желаемого
варианта: ");
  }
```

```
void dumpListToFile (student *List)
  {
    char fileName[261];
     std::cout << std::endl << "Введите имя (или путь) файла (по
умолчанию - students): ";
    std::cin.getline (fileName, 261);
    if (fileName[0]=='\0') strcpy(fileName, "students");
    std::ofstream File;
    File.open (fileName, std::ios_base::trunc);
    if (File.is_open())
      {
        for (unsigned int i=0; i<listSize; i++)
          {
            student *newStudent;
            newStudent=List+i;
            File << newStudent->Name << "\n";
            File << newStudent->Sex << "\n";
            File << newStudent->Group << "\n";
            File << newStudent->Number << "\n";
            File << newStudent->Ekz1 << "\n";
            File << newStudent->Ekz2 << "\n";
            File << newStudent->Ekz3 << "\n";
            File << newStudent->Diff1 << "\n";
            File << newStudent->Diff2 << "\n";
            File << newStudent->Diff3 << "\n";
            File << newStudent->Diff4 << "\n";
            File << newStudent->Diff5 << "\n";
            File << newStudent->Form << "\n";
            if (i==listSize-1)
              {
                File << newStudent->modificationTime;
              }
            else
              {
                File << newStudent->modificationTime << "\n";
              }
          }
        File.close();
      }
    else
      {
        std::cout << "Некорректный ввод!" << std::endl;
        List=addStudentsFromFile(List);
      }
```

```
  }

void listFilterChoise (student *List)
  {
     bool answer=getNumericAnswer(1, 2, "\nВыберите желаемый формат списков:\n1)Краткий\n2)Длинный\n\nВведите желаемый вариант: ")-1;
    listFilter(List, answer);
  }

void listFilter (student *List, bool length)
  {
    system (CLEARSCREEN);
        std::cout << "Вывести список студентов, соответствующих критериям:" << std::endl << std::endl << "1) Вывести список студентов конкретной группы." << std::endl << "2) Вывести топ студентов с наивысшим средним баллом за прошедшую сессию." << std::endl << "3) Вывод списков студентов мужского и женского полов." << std::endl << "4) Вывести список студентов, которые будут получать стипендию по итогам прошедшей сессии." << std::endl << "5) Вывести список студентов, которые не будут получают стипендию." << std::endl << "6) Вывести список студентов, которые учатся только на \"хорошо\" и \"отлично\"." << std::endl << "7) Вывести список студентов, которые учатся только на \"отлично\"." << std::endl << "8) Выводести список всех студентов, имеющих конкретный номер в списке группы." << std::endl << "9) Вывод всех записей, сделанных/изменённых в конкретный промежуток времени." << std::endl << "0) Возврат в главное меню." << std::endl << std::endl;
       switch (getNumericAnswer (0, 9, "Введите номер желаемого варианта: "))
      {
        case 1:
          {
            std::cout << "Введите номер группы: ";
            char Group[11];
            std::cin.getline(Group, 11);
            listGroup(List, length, Group);
            listFilter(List, length);
            break;
          }
        case 2:
          listTop(List, length);
          listFilter(List, length);
          break;
        case 3:
          listSex(List, length);
```

```
          listFilter(List, length);
          break;
        case 4:
          listStipend(List, length);
          listFilter(List, length);
          break;
        case 5:
          listStipendLess(List, length);
          listFilter(List, length);
          break;
        case 6:
          listFourFive(List, length);
          listFilter(List, length);
          break;
        case 7:
          listFive(List, length);
          listFilter(List, length);
          break;
        case 8:
          {
              int number=getNumericAnswer(0, 9999, "Введите номер
студента в списке группы: ");
            listNumber(List, length, number);
            listFilter(List, length);
            break;
          }
        case 9:
          {
            tm Day=getTmDate();
                int answer=getNumericAnswer(0, 2, "Вывести список
записей, сделанных:\n\n0) в течении всего дня.\n1) до полудня.\n2)
после полудня.\n\nВведите желаемы вариант: ");
            listTime(List, length, Day, answer);
            listFilter(List, length);
            break;
          }
        default:
          break;
      }
  }

void listGroup (student *List, bool length, const char *Group)
  {
    system (CLEARSCREEN);
```

```
     std::cout << "Список студентов в группе " << Group << ":" <<
std::endl << std::endl;
    int studentNumber=0;
    for (unsigned int i=0; i<listSize; i++)
      {
        if (!strcmp((List+i)->Group, Group))
          {
            if (length)
              {
                printStudentInfo(List, i+1);
                std::cout << std::endl;
              }
            else
              {
                  std::cout << i+1 << ". " << (List+i)->Name <<
std::endl;
              }
            studentNumber++;
          }
      }
     std::cout << std::endl << "Всего студентов в группе " << Group
<< ": " << studentNumber << "." << std::endl << std::endl;
         int answer=getNumericAnswer(0, listSize, "Введите номер
студента в списке, карточку которого вы хотите посмотреть (для
возврата в главное меню введите 0): ");
    if (answer)
      {
        studentMenu (List, answer);
        listGroup (List, length, Group);
      }
  }

void listTop (student *List, bool length)
  {
    system (CLEARSCREEN);
      std::cout << "Топ студентов с наивысшим средним баллом за
прошедшую сессию:" << std::endl << std::endl;
    struct sortList
      {
        unsigned int number;
        double score;
      };
   sortList *SortList=(sortList*)malloc(sizeof(sortList)*listSize);
    for (unsigned int i=0; i<listSize; i++)
      {
```

```
        (SortList+i)->number=i;
(SortList+i)->score=static_cast<float>((((List+i)->Diff1)+((List+i)
->Diff2)+((List+i)->Diff3)+((List+i)->Diff4)+((List+i)->Diff5)+((Li
st+i)->Ekz1)+((List+i)->Ekz2)+((List+i)->Ekz3))/8);
      }
    int sorted=0;
    int currentMaxUnsorted=listSize-1;
    while (!sorted)
      {
        sorted=1;
        for (int i=0; i<currentMaxUnsorted; i++)
          {
            if ((SortList+i)->score-(SortList+i+1)->score>0)
              {
                sortList Buffer;
                Buffer.number=(SortList+i)->number;
                Buffer.score=(SortList+i)->score;
                (SortList+i)->number=(SortList+i+1)->number;
                (SortList+i)->score=(SortList+i+1)->score;
                (SortList+i+1)->number=Buffer.number;
                (SortList+i+1)->score=Buffer.score;
                sorted=0;
              }
          }
        currentMaxUnsorted--;
      }
    for (unsigned int i=listSize-1, j=0; i>=0&&j<10; i--, j++)
      {
        if (length)
              {
                printStudentInfo(List, (SortList+i)->number+1);
                std::cout << std::endl;
              }
            else
              {
                std::cout << ((SortList+i)->number)+1 << ". " <<
                             (List+((SortList+i)->number))->Name <<
std::endl;
              }
      }
    std::cout << std::endl << std::endl;
        int  answer=getNumericAnswer(0,  listSize,  "Введите  номер
студента в списке, карточку которого вы хотите посмотреть (для
возврата в главное меню введите 0): ");
    if (answer)
```

```
      {
        studentMenu (List, answer);
        listTop (List, length);
      }
    free(SortList);
  }

void listSex (student *List, bool length)
  {
    system (CLEARSCREEN);
    std::cout << "Список студентов мужского пола:" << std::endl;
    int studentNumber=0;
    for (unsigned int i=0; i<listSize; i++)
      {
        if ((List+i)->Sex==1)
          {
            if (length)
              {
                printStudentInfo(List, i+1);
                std::cout << std::endl;
              }
            else
              {
                    std::cout << i+1 << ". " << (List+i)->Name <<
std::endl;
              }
            studentNumber++;
          }
      }
     std::cout << std::endl << "Всего студентов мужского пола: " <<
studentNumber
               << "." << std::endl << std::endl << "Список студентов
женского по"
              "ла:" << std::endl;
    studentNumber=0;
    for (unsigned int i=0; i<listSize; i++)
      {
        if ((List+i)->Sex==2)
          {
            if (length)
              {
                printStudentInfo(List, i+1);
                std::cout << std::endl;
              }
            else
```

```
              {
                    std::cout << i+1 << ". " << (List+i)->Name <<
std::endl;
              }
            studentNumber++;
          }
      }
     std::cout << std::endl << "Всего студентов женского пола: " <<
studentNumber
              << "." << std::endl << std::endl;
        int  answer=getNumericAnswer(0,  listSize,  "Введите  номер
студента в списке, карточку которого вы хотите посмотреть (для
возврата в главное меню введите 0): ");
    if (answer)
      {
        studentMenu (List, answer);
        listSex (List, length);
      }
  }

void listStipend (student *List, bool length)
  {
    system (CLEARSCREEN);
       std::cout  <<  "Список  студентов,  получающих  стипендию:"  <<
std::endl <<
                 std::endl;
    int studentNumber=0;
    for (unsigned int i=0; i<listSize; i++)
      {
                                                                    if
(((((List+i)->Diff1)-3)>0)&&((((List+i)->Diff2)-3)>0)&&((((List+i)-
>Diff3)-3)>0)&&((((List+i)->Diff4)-3)>0)&&((((List+i)->Diff5)-3)>0)
&&((((List+i)->Ekz1)-3)>0)&&((((List+i)->Ekz2)-3)>0)&&((((List+i)->
Ekz3)-3)>0)&&((List+i)->Form==1))
          {
            if (length)
              {
                printStudentInfo(List, i+1);
                std::cout << std::endl;
              }
            else
              {
                    std::cout << i+1 << ". " << (List+i)->Name <<
std::endl;
              }
```

```
            studentNumber++;
          }
      }
        std::cout << std::endl << "Всего студентов, получающих
стипендию: " <<
                 studentNumber << "." << std::endl << std::endl;
        int answer=getNumericAnswer(0, listSize, "Введите номер
студента в списке, карточку которого вы хотите посмотреть (для
возврата в главное меню введите 0): ");
    if (answer)
      {
        studentMenu (List, answer);
        listStipend (List, length);
      }
  }

void listStipendLess (student *List, bool length)
  {
    system (CLEARSCREEN);
     std::cout << "Список студентов, не получающих стипендию:" <<
std::endl <<
                 std::endl;
    int studentNumber=0;
    for (unsigned int i=0; i<listSize; i++)
      {
                                                                  if
(((((List+i)->Diff1)-3)<=0)||((((List+i)->Diff2)-3)<=0)||((((List+i
)->Diff3)-3)<=0)||((((List+i)->Diff4)-3)<=0)||((((List+i)->Diff5)-3
)<=0)||((((List+i)->Ekz1)-3)<=0)||((((List+i)->Ekz2)-3)<=0)||((((Li
st+i)->Ekz3)-3)<=0)||(!((List+i)->Form==1)))
          {
            if (length)
              {
                printStudentInfo(List, i+1);
                std::cout << std::endl;
              }
            else
              {
                     std::cout << i+1 << ". " << (List+i)->Name <<
std::endl;
              }
            studentNumber++;
          }
      }
```

```
      std::cout << std::endl << "Всего студентов, не получающих
стипендию: " <<
                studentNumber << "." << std::endl << std::endl;
        int answer=getNumericAnswer(0, listSize, "Введите номер
студента в списке, карточку которого вы хотите посмотреть (для
возврата в главное меню введите 0): ");
    if (answer)
      {
        studentMenu (List, answer);
        listStipendLess (List, length);
      }
  }

void listFourFive (student *List, bool length)
  {
    system (CLEARSCREEN);
       std::cout << "Список студентов, которые учатся только на
\"хорошо\" и \"отл"
                "ично\":" << std::endl << std::endl;
    int studentNumber=0;
    for (unsigned int i=0; i<listSize; i++)
      {
                                                                  if
(((((List+i)->Diff1)-3)>0)&&((((List+i)->Diff2)-3)>0)&&((((List+i)-
>Diff3)-3)>0)&&((((List+i)->Diff4)-3)>0)&&((((List+i)->Diff5)-3)>0)
&&((((List+i)->Ekz1)-3)>0)&&((((List+i)->Ekz2)-3)>0)&&((((List+i)->
Ekz3)-3)>0))
          {
            if (length)
              {
                printStudentInfo(List, i+1);
                std::cout << std::endl;
              }
            else
              {
                    std::cout << i+1 << ". " << (List+i)->Name <<
std::endl;
              }
            studentNumber++;
          }
      }
      std::cout << std::endl << "Всего студентов, которые учатся
только на \"хоро"
                 "шо\" и \"отлично\": " << studentNumber << "." <<
std::endl <<
```

```
                std::endl;
        int answer=getNumericAnswer(0, listSize, "Введите номер
студента в списке, карточку которого вы хотите посмотреть (для
возврата в главное меню введите 0): ");
    if (answer)
      {
        studentMenu (List, answer);
        listFourFive (List, length);
      }
  }

void listFive (student *List, bool length)
  {
    system (CLEARSCREEN);
       std::cout << "Список студентов, которые учатся только на
\"отлично\":" <<
                 std::endl << std::endl;
    int studentNumber=0;
    for (unsigned int i=0; i<listSize; i++)
      {
                                                                  if
(((((List+i)->Diff1)-4)>0)&&((((List+i)->Diff2)-4)>0)&&((((List+i)-
>Diff3)-4)>0)&&((((List+i)->Diff4)-4)>0)&&((((List+i)->Diff5)-4)>0)
&&((((List+i)->Ekz1)-4)>0)&&((((List+i)->Ekz2)-4)>0)&&((((List+i)->
Ekz3)-4)>0))
          {
            if (length)
              {
                printStudentInfo(List, i+1);
                std::cout << std::endl;
              }
            else
              {
                     std::cout << i+1 << ". " << (List+i)->Name <<
std::endl;
              }
            studentNumber++;
          }
      }
      std::cout << std::endl << "Всего студентов, которые учатся
только на \"отли"
                  "чно\": " << studentNumber << "." << std::endl <<
std::endl;
```

```
        int  answer=getNumericAnswer(0,  listSize,  "Введите  номер
студента в списке, карточку которого вы хотите посмотреть (для
возврата в главное меню введите 0): ");
    if (answer)
      {
        studentMenu (List, answer);
        listFive (List, length);
      }
  }

void listNumber (student *List, bool length, int Number)
  {
    system (CLEARSCREEN);
     std::cout << "Список студентов, имеющих номер " << Number << "
в списке гру"
                  "ппы:" << std::endl << std::endl;
    int studentNumber=0;
    for (unsigned int i=0; i<listSize; i++)
      {
        if ((List+i)->Number==Number)
          {
            if (length)
              {
                printStudentInfo(List, i+1);
                std::cout << std::endl;
              }
            else
              {
                    std::cout << i+1 << ". " << (List+i)->Name <<
std::endl;
              }
            studentNumber++;
          }
      }
     std::cout << std::endl << "Всего студентов, имеющих номер " <<
Number <<
                   " в списке группы: " << studentNumber << "." <<
std::endl <<
                std::endl;
        int  answer=getNumericAnswer(0,  listSize,  "Введите  номер
студента в списке, карточку которого вы хотите посмотреть (для
возврата в главное меню введите 0): ");
    if (answer)
      {
        studentMenu (List, answer);
```

```
        listNumber (List, length, Number);
      }
  }

void listTime (student *List, bool length, tm Begin, int Noon)
  {
    system (CLEARSCREEN);
      std::cout << "Список студентов, записи о которых сделаны в
указанный промеж"
                "уток времени:" << std::endl << std::endl;
    tm End;
    End.tm_sec=Begin.tm_sec;
    End.tm_min=Begin.tm_min;
    End.tm_hour=Begin.tm_hour;
    End.tm_mday=Begin.tm_mday;
    End.tm_mon=Begin.tm_mon;
    End.tm_year=Begin.tm_year;
    End.tm_isdst=Begin.tm_isdst;
    switch (Noon)
      {
        case 0:
          End.tm_sec=59;
          End.tm_min=59;
          End.tm_hour=23;
          break;
        case 1:
          End.tm_sec=59;
          End.tm_min=59;
          End.tm_hour=11;
          break;
        case 2:
          End.tm_sec=59;
          End.tm_min=59;
          End.tm_hour=23;
          Begin.tm_sec=00;
          Begin.tm_min=00;
          Begin.tm_hour=12;
          break;
      }
    time_t beginTime=mktime(&Begin);
    time_t endTime=mktime(&End);
    int studentNumber=0;
    for (unsigned int i=0; i<listSize; i++)
      {
```

```
                                                                    if
((((List+i)->modificationTime)>=beginTime)&&(((List+i)->modifica-
tionTime)<=endTime))
          {
            if (length)
              {
                printStudentInfo(List, i+1);
                std::cout << std::endl;
              }
            else
              {
                     std::cout << i+1 << ". " << (List+i)->Name <<
std::endl;
              }
            studentNumber++;
          }
      }
     std::cout << std::endl << "Всего студентов, записи о которых
сделаны в указ"
                  "анный промежуток времени: " << studentNumber <<
"." <<
                 std::endl << std::endl;
        int  answer=getNumericAnswer(0,  listSize,  "Введите  номер
студента в списке, карточку которого вы хотите посмотреть (для
возврата в главное меню введите 0): ");
    if (answer)
      {
        studentMenu (List, answer);
        listTime (List, length, Begin, Noon);
      }
  }

tm getTmDate ()
  {
    tm Day;
    std::cout << "Введите дату (формат: ДД.ММ.ГГГГ): ";
    char date[11];
    std::cin.getline(date, 11);
    Day.tm_sec=0;
    Day.tm_min=0;
    Day.tm_hour=0;
    Day.tm_mday=(date[0]-48)*10+(date[1]-48);
    Day.tm_mon=(date[3]-48)*10+(date[4]-48)-1;
```

```
Day.tm_year=(date[6]-48)*1000+(date[7]-48)*100+(date[8]-48)*10+(dat
e[9]-48)-1900;
    Day.tm_isdst=-1;
    if (mktime(&Day)<0)
      {
        std::cout << "Некорректный ввод!" << std::endl;
        Day=getTmDate();
      }
    return Day;
  }
```

ПРИЛОЖЕНИЕ Б
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Тестирование программы осуществлялось на компьютере с процессором x64 под управлением ОС Windows 10 Pro (версия 20H2, 64-бит) с использованием следующих компиляторов:

- g++ (GCC) 10.2.0
- Оптимизирующий компилятор Microsoft (R) C/C++ версии 19.27.29112 для x64

А также на компьютере с процессором x64 под управлением ОС Linux Mint 20.1 XFCE (версия ядра 5.4.0-65-generic, x86_64) с использованием следующих компиляторов:

- g++ (GCC) 9.3.0

В ходе тестирования были получены идентичные результаты независимо от использованного компилятора и ОС.

Входные данные при тестировании (считываются из файла):

```
Ivanchuk Evgenia Vadimovna
2
9894
6
3
4
5
5
5
5
5
5
2
1582995944
Zhernakova Anna Arsenievna
2
6543
2
5
5
5
5
5
5
```

5
5
1
1582461339
Raskosnova Alisa Grigorievna
2
9876
10
4
4
5
4
5
4
5
4
2
1582578886
Ivanov Ivan Ivanovich
1
1000
5
5
5
5
5
5
5
5
5
3
1582998104
Sidorova Maria Alexandrovna
2
7575
12
3
4
5
4
4
4
5
5
1
1582995449
Alexandrov Vasily Alexandrovich
1
7676
1
4

4
4
4
4
4
4
4
1
1582995528
Zaytsev Yuri Yurievich
1
7676
4
4
5
4
4
4
4
4
4
1
1582995573
Lanova Olga Borisovna
2
9890
9
5
5
5
5
5
5
5
5
2
1582995664
Solovieva Natalia Dmitrievna
2
1000
11
4
5
4
4
5
5
5
5
3
1582995722

```
Mamontova Iskra Ivanovna
2
6543
9
4
4
5
4
5
4
5
4
1
1582998997
Ivanova Ivanna Ivanovna
2
2000
1
4
5
4
5
4
5
4
5
1
1583519337
```

1) Изменение данных конкретной записи:

Исходные данные:

```
Студент 2

ФИО студента: Балашевич Константин Дмитриевич
Пол студента: мужской.
Номер группы студента: 0323
Номер студента в списке группы: 2
Оценка студента за первый экзамен: 5
Оценка студента за второй экзамен: 5
Оценка студента за третий экзамен: 5
Оценка студента за первый зачёт: 5
Оценка студента за второй зачёт: 5
Оценка студента за третий зачёт: 5
Оценка студента за четвёртый зачёт: 5
Оценка студента за пятый зачёт: 5
Форма обучения студента: очно-заочная.
Дата внесения изменений в запись: Sun Jan 31 00:49:14 2021
```

Результирующие данные:

```
Студент 2

ФИО студента: Абракадабрин Абра Кадабрин
Пол студента: женский.
Номер группы студента: 3230
Номер студента в списке группы: 99
Оценка студента за первый экзамен: 2
Оценка студента за второй экзамен: 2
Оценка студента за третий экзамен: 2
Оценка студента за первый зачёт: 2
Оценка студента за второй зачёт: 2
Оценка студента за третий зачёт: 2
Оценка студента за четвёртый зачёт: 2
Оценка студента за пятый зачёт: 2
Форма обучения студента: заочная.
Дата внесения изменений в запись: Mon Feb 15 05:33:27 2021
```

2) Вывод списка студентов конкретной группы:

Исходные данные: номер группы - 0323.

Результирующие данные:

Список студентов в группе 0323:

```
Студент 1

ФИО студента: Атлас Мария Игоревна
Пол студента: женский.
Номер группы студента: 0323
Номер студента в списке группы: 1
Оценка студента за первый экзамен: 3
Оценка студента за второй экзамен: 3
Оценка студента за третий экзамен: 5
Оценка студента за первый зачёт: 4
Оценка студента за второй зачёт: 5
Оценка студента за третий зачёт: 5
Оценка студента за четвёртый зачёт: 5
Оценка студента за пятый зачёт: 5
Форма обучения студента: очно-заочная.
Дата внесения изменений в запись: Sun Jan 31 00:48:41 2021

Студент 3

ФИО студента: Балякина Вероника Владимировна
Пол студента: женский.
Номер группы студента: 0323
Номер студента в списке группы: 3
Оценка студента за первый экзамен: 2
Оценка студента за второй экзамен: 2
```

Оценка студента за третий экзамен: 5
Оценка студента за первый зачёт: 4
Оценка студента за второй зачёт: 5
Оценка студента за третий зачёт: 5
Оценка студента за четвёртый зачёт: 5
Оценка студента за пятый зачёт: 5
Форма обучения студента: очно-заочная.
Дата внесения изменений в запись: Sun Jan 31 00:50:17 2021

Студент 4

ФИО студента: Вологин Максим Павлович
Пол студента: мужской.
Номер группы студента: 0323
Номер студента в списке группы: 4
Оценка студента за первый экзамен: 5
Оценка студента за второй экзамен: 5
Оценка студента за третий экзамен: 5
Оценка студента за первый зачёт: 5
Оценка студента за второй зачёт: 5
Оценка студента за третий зачёт: 5
Оценка студента за четвёртый зачёт: 5
Оценка студента за пятый зачёт: 5
Форма обучения студента: очно-заочная.
Дата внесения изменений в запись: Sun Jan 31 00:50:55 2021

Студент 5

ФИО студента: Голубцов Владислав Владимирович
Пол студента: мужской.
Номер группы студента: 0323
Номер студента в списке группы: 5
Оценка студента за первый экзамен: 5
Оценка студента за второй экзамен: 5
Оценка студента за третий экзамен: 5
Оценка студента за первый зачёт: 3
Оценка студента за второй зачёт: 5
Оценка студента за третий зачёт: 5
Оценка студента за четвёртый зачёт: 5
Оценка студента за пятый зачёт: 5
Форма обучения студента: очно-заочная.
Дата внесения изменений в запись: Sun Jan 31 00:51:55 2021

Студент 6

ФИО студента: Гусляков Андрей Дмитриевич
Пол студента: мужской.
Номер группы студента: 0323
Номер студента в списке группы: 6
Оценка студента за первый экзамен: 4
Оценка студента за второй экзамен: 3

Оценка студента за третий экзамен: 5
Оценка студента за первый зачёт: 5
Оценка студента за второй зачёт: 2
Оценка студента за третий зачёт: 2
Оценка студента за четвёртый зачёт: 5
Оценка студента за пятый зачёт: 5
Форма обучения студента: очно-заочная.
Дата внесения изменений в запись: Sun Jan 31 00:52:51 2021

Студент 7

ФИО студента: Егоров Михаил Алексеевич
Пол студента: мужской.
Номер группы студента: 0323
Номер студента в списке группы: 7
Оценка студента за первый экзамен: 2
Оценка студента за второй экзамен: 2
Оценка студента за третий экзамен: 2
Оценка студента за первый зачёт: 2
Оценка студента за второй зачёт: 2
Оценка студента за третий зачёт: 2
Оценка студента за четвёртый зачёт: 2
Оценка студента за пятый зачёт: 5
Форма обучения студента: очно-заочная.
Дата внесения изменений в запись: Sun Jan 31 00:54:14 2021

Студент 8

ФИО студента: Землянский Дмитрий Анатольевич
Пол студента: мужской.
Номер группы студента: 0323
Номер студента в списке группы: 8
Оценка студента за первый экзамен: 4
Оценка студента за второй экзамен: 4
Оценка студента за третий экзамен: 5
Оценка студента за первый зачёт: 4
Оценка студента за второй зачёт: 5
Оценка студента за третий зачёт: 5
Оценка студента за четвёртый зачёт: 5
Оценка студента за пятый зачёт: 5
Форма обучения студента: очно-заочная.
Дата внесения изменений в запись: Sun Jan 31 00:55:04 2021

Студент 9

ФИО студента: Квачадзе Самсон Васильевич
Пол студента: мужской.
Номер группы студента: 0323
Номер студента в списке группы: 9
Оценка студента за первый экзамен: 5
Оценка студента за второй экзамен: 5

```
Оценка студента за третий экзамен: 5
Оценка студента за первый зачёт: 4
Оценка студента за второй зачёт: 5
Оценка студента за третий зачёт: 5
Оценка студента за четвёртый зачёт: 5
Оценка студента за пятый зачёт: 5
Форма обучения студента: очно-заочная.
Дата внесения изменений в запись: Sun Jan 31 00:55:49 2021

Студент 10

ФИО студента: Кольцов Кирилл Эдуардович
Пол студента: мужской.
Номер группы студента: 0323
Номер студента в списке группы: 10
Оценка студента за первый экзамен: 5
Оценка студента за второй экзамен: 5
Оценка студента за третий экзамен: 5
Оценка студента за первый зачёт: 5
Оценка студента за второй зачёт: 5
Оценка студента за третий зачёт: 5
Оценка студента за четвёртый зачёт: 5
Оценка студента за пятый зачёт: 5
Форма обучения студента: очно-заочная.
Дата внесения изменений в запись: Sun Jan 31 00:56:29 2021


Всего студентов в группе 0323: 9.
```

3) Вывод топа студентов по среднему баллу за прошедшую сессию:

```
18. Lanova Olga Borisovna
14. Ivanov Ivan Ivanovich
12. Zhernakova Anna Arsenievna
10. Кольцов Кирилл Эдуардович
4. Вологин Максим Павлович
21. Ivanova Ivanna Ivanovna
20. Mamontova Iskra Ivanovna
19. Solovieva Natalia Dmitrievna
17. Zaytsev Yuri Yurievich
16. Alexandrov Vasily Alexandrovich
```

4) Вывод списков студентов мужского и женского пола:

```
Список студентов мужского пола:
4. Вологин Максим Павлович
5. Голубцов Владислав Владимирович
6. Гусляков Андрей Дмитриевич
7. Егоров Михаил Алексеевич
8. Землянский Дмитрий Анатольевич
9. Квачадзе Самсон Васильевич
10. Кольцов Кирилл Эдуардович
```

```
14. Ivanov Ivan Ivanovich
16. Alexandrov Vasily Alexandrovich
17. Zaytsev Yuri Yurievich

Всего студентов мужского пола: 10.

Список студентов женского пола:
1. Атлас Мария Игоревна
2. Абракадабрин Абра Кадабрин
3. Балякина Вероника Владимировна
11. Ivanchuk Evgenia Vadimovna
12. Zhernakova Anna Arsenievna
13. Raskosnova Alisa Grigorievna
15. Sidorova Maria Alexandrovna
18. Lanova Olga Borisovna
19. Solovieva Natalia Dmitrievna
20. Mamontova Iskra Ivanovna
21. Ivanova Ivanna Ivanovna

Всего студентов женского пола: 11.
```

5) Вывод списка студентов, получающих стипендию:

```
Список студентов, получающих стипендию:

12. Zhernakova Anna Arsenievna
16. Alexandrov Vasily Alexandrovich
17. Zaytsev Yuri Yurievich
20. Mamontova Iskra Ivanovna
21. Ivanova Ivanna Ivanovna

Всего студентов, получающих стипендию: 5.
```

6) Вывод списка записей, сделанных в конкретный день:

Исходные данные: 31.01.2021, весь день

Результирующие данные:

```
Список студентов, записи о которых сделаны в указанный промежуток времени:

1. Атлас Мария Игоревна
3. Балякина Вероника Владимировна
4. Вологин Максим Павлович
5. Голубцов Владислав Владимирович
6. Гусляков Андрей Дмитриевич
7. Егоров Михаил Алексеевич
8. Землянский Дмитрий Анатольевич
9. Квачадзе Самсон Васильевич
10. Кольцов Кирилл Эдуардович
```

```
Всего студентов, записи о которых сделаны в указанный промежуток времени: 9.
```

В ходе тестирования остальных частей программы также были получены результаты, соответсвующие предполагаемым.