Министерство науки и высшего образования

Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ**

**РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ имени В.Ф. УТКИНА»**

Кафедра «Космические технологии»

Пояснительная записка к курсовому проекту по дисциплине «Основы компьютерных наук»

На тему: «Разработка программы, позволяющей провести тестирование студента по основам компьютерных наук»

Выполнил:

Студент группы 748

Чуйко А.А.

Проверил:

Доц. каф. КТ

Наумов Д. А.

Рязань 2019

Оглавление

[1. Постановка и анализ задачи 3](#_Toc11885484)

[2. Проектирование структуры данных 4](#_Toc11885485)

[3. Проектирование модульной структуры программы 5](#_Toc11885486)

[4. Проектирование алгоритмов 9](#_Toc11885487)

[5. Тестирование программы 12](#_Toc11885488)

[6. Программная реализация алгоритмов 13](#_Toc11885489)

[Заключение 14](#_Toc11885490)

[Библиографический список 15](#_Toc11885491)

[Приложение А (тестовые вопросы) 16](#_Toc11885492)

[Приложение Б (листинг программных модулей). 20](#_Toc11885493)

## 1. Постановка и анализ задачи

Целью курсового проекта является автоматизация процессов вхождения и регистрации в системе для проверки знаний студентов по дисциплине «Ос-новы компьютерных наук».

Задачей курсового проектирования является разработка программы мо-дульной структуры (далее – программа), позволяющей производить расчет сводной статистики по тесту и пользователю, предоставление статистики прохождения пользователем по тесту, предоставление статистики прохождения теста различными пользователями по различным темам дисциплины «Основы компьютерных наук».

Программа должна реализовывать вышеперечисленные функции предоставления статистики.

В рамках курсового проекта будет реализован модуль для проведения расчета (далее – модуль «Расчет»).

При расчете статистики, и её выводе, используются начальные данные, сохраненные в базе, после прохождения тестом пользователя. В модуле содержатся процедуры, осуществляющие вывод статистик отдельным по тестам и пользователям, и соответствующие расчеты, что позволяет осуществить более локальную статистическую выборку.

Для осуществления работы модуля необходимо наличие начальных данных прохождения теста, сохранение которых осуществляется при помощи предыдущего модуля сохранения статистики.

## 2. Проектирование структуры данных

Рассмотрим каждую из функций (Таблица 1). В столбце «Данные» пере-числены категории данных, которые использует соответствующая функция.

Таблица 1 – Функции системы

|  |  |  |
| --- | --- | --- |
| № | Функция | Данные |
| Ф10 | отображение статистики прохождения теста | статистика прохождения теста |
| Ф12 | сохранение статистики прохождения теста | статистика прохождения теста |

Таким образом, модуль «Расчет» будет работать с категорией данных «Статистика».

При расчете статистики в базе используется статистика прохождения теста (количество набранных баллов, имя теста, имя пользователя). Основные действия вычисления данных и предоставления статистики, будут осуществляться при работе с этими данными.

В целях улучшения эффективности работы с файлом, используется модуль, содержащий описание способа хранения данных в файле, который так же будет использоваться в модулях сохранения и предоставления статистики.

## 3. Проектирование модульной структуры программы

Модуль «Расчет»:

* выполняет функции расчета статистики по заданным запросам:
  + статистика прохождения заданного теста;
  + статистика прохождения теста пользователем;
  + сводная статистика по выбранным пользователям;
  + сводная статистика по выбранным тестам.

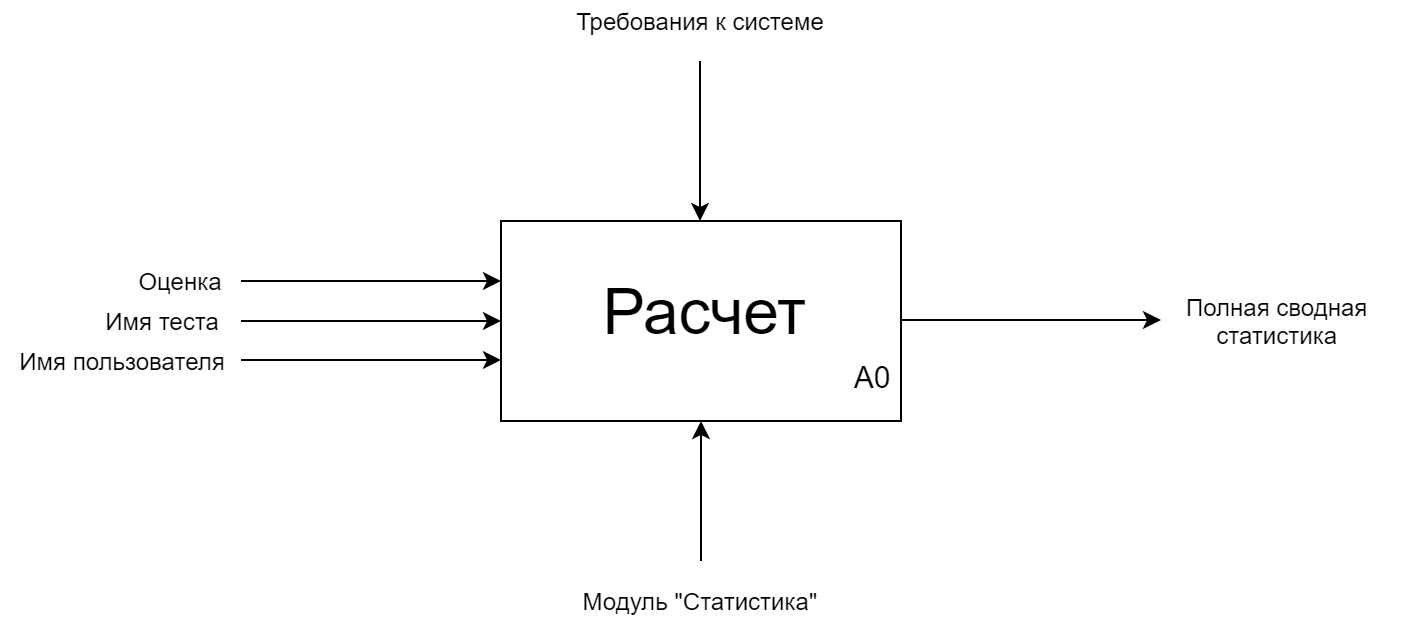


Рис. 1 – Контекстная диаграмма IDEF0

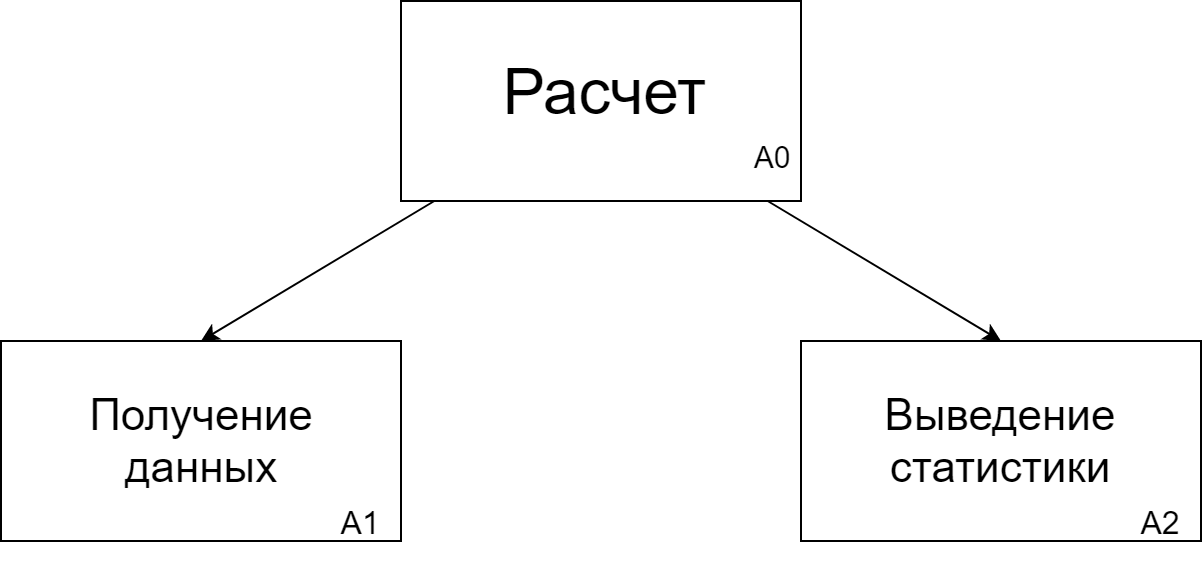


Рис. 2 – Диаграмма декомпозиции

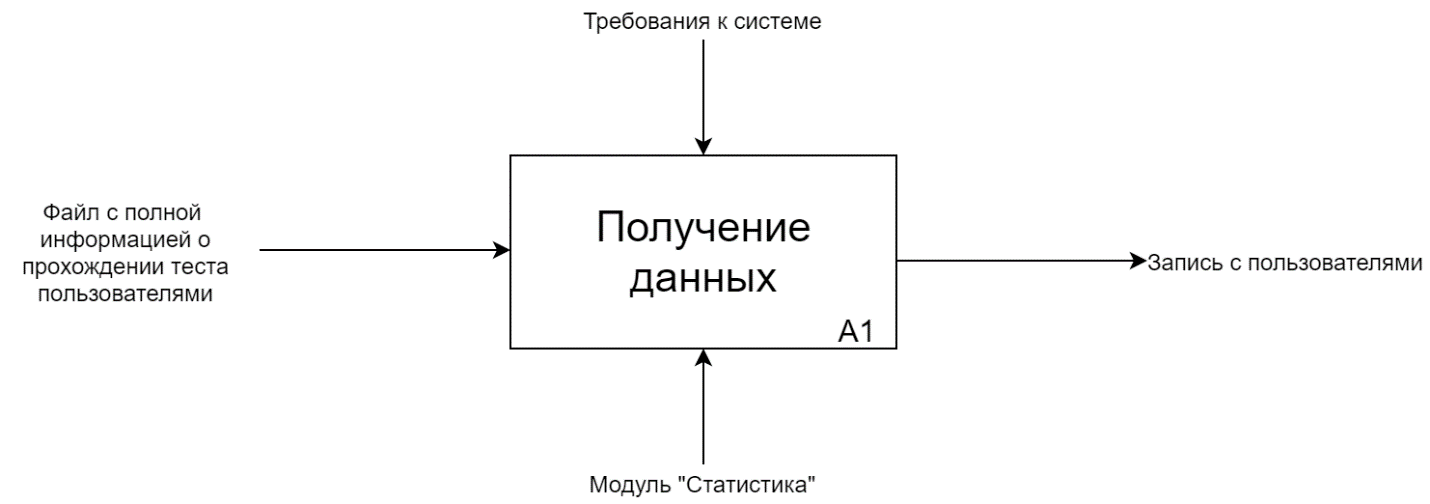


Рис. 3 – Блок получения данных

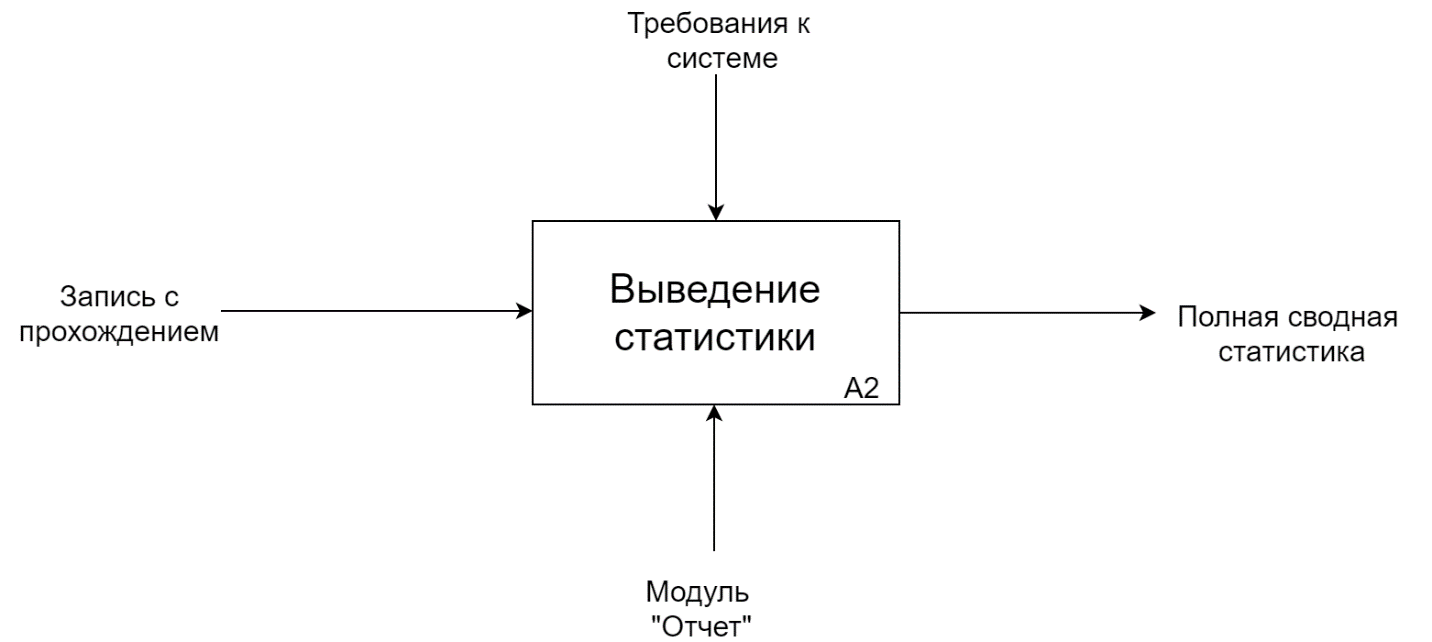


Рис. 4 – Блок выведения статистики

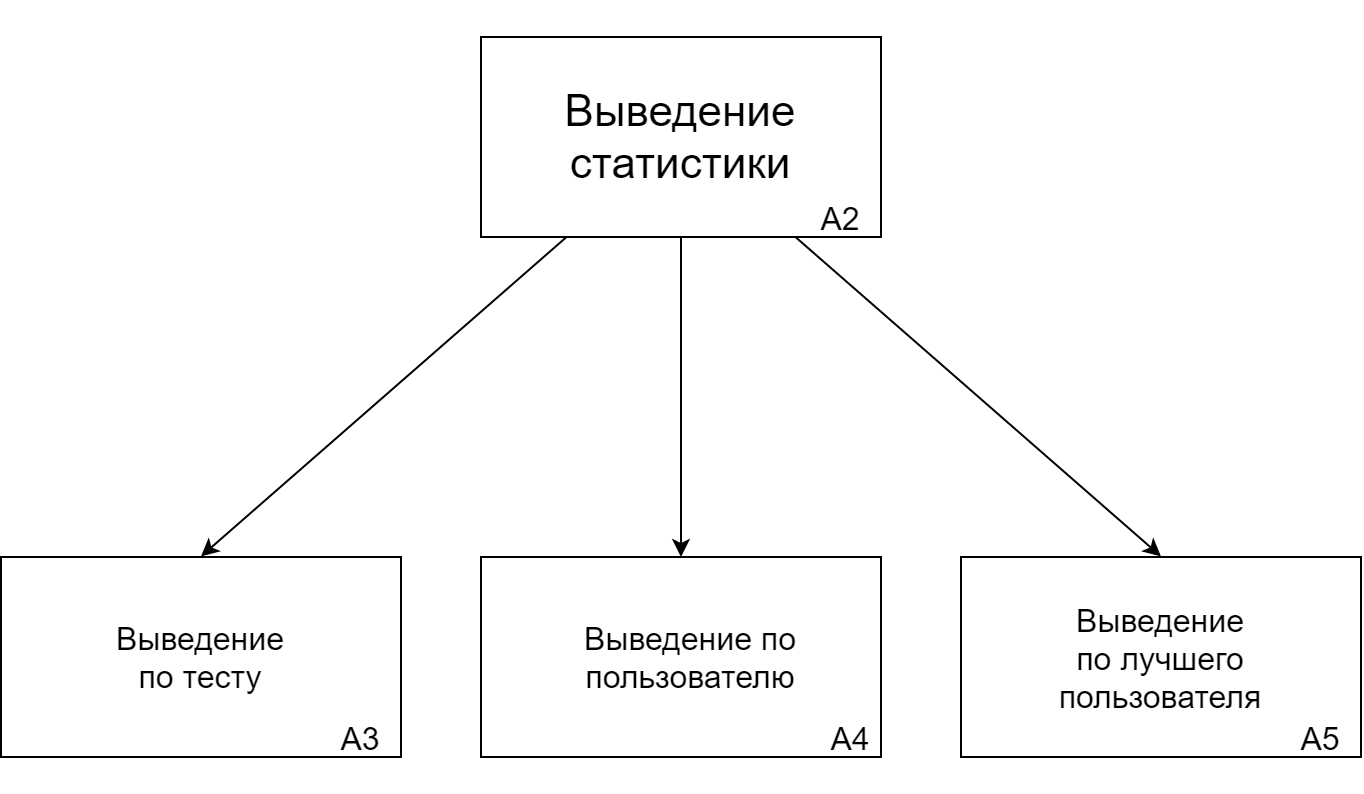


Рис. 5 – Диаграмма декомпозиции

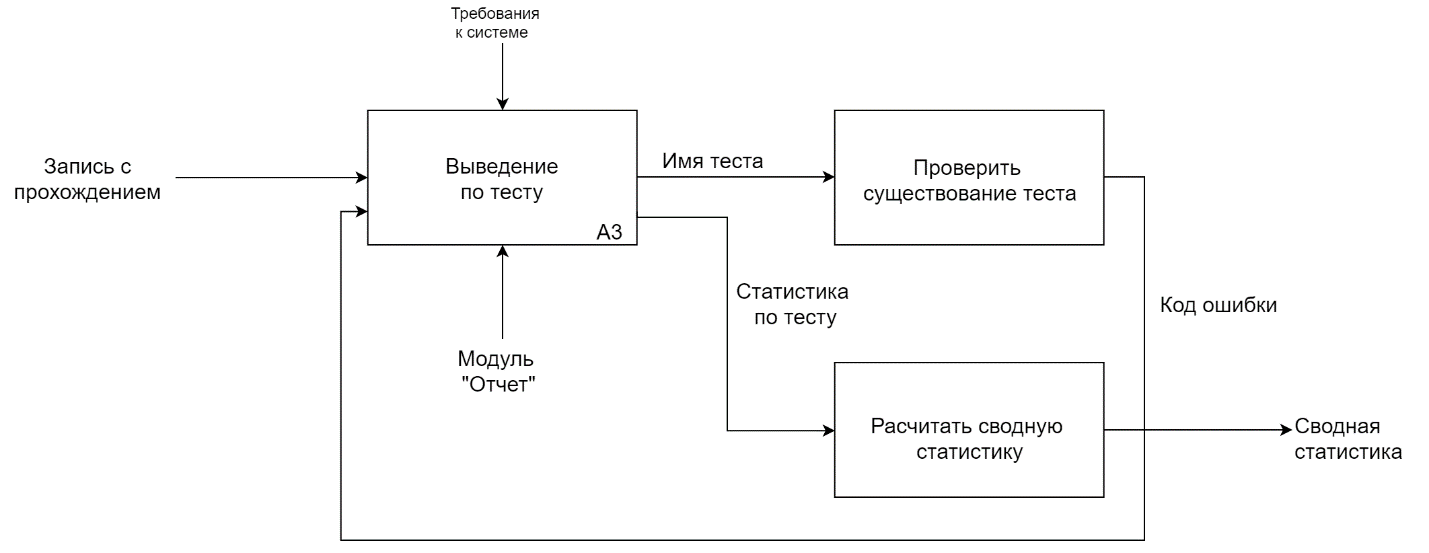


Рис. 5 – Блок выведения статистики по тесту

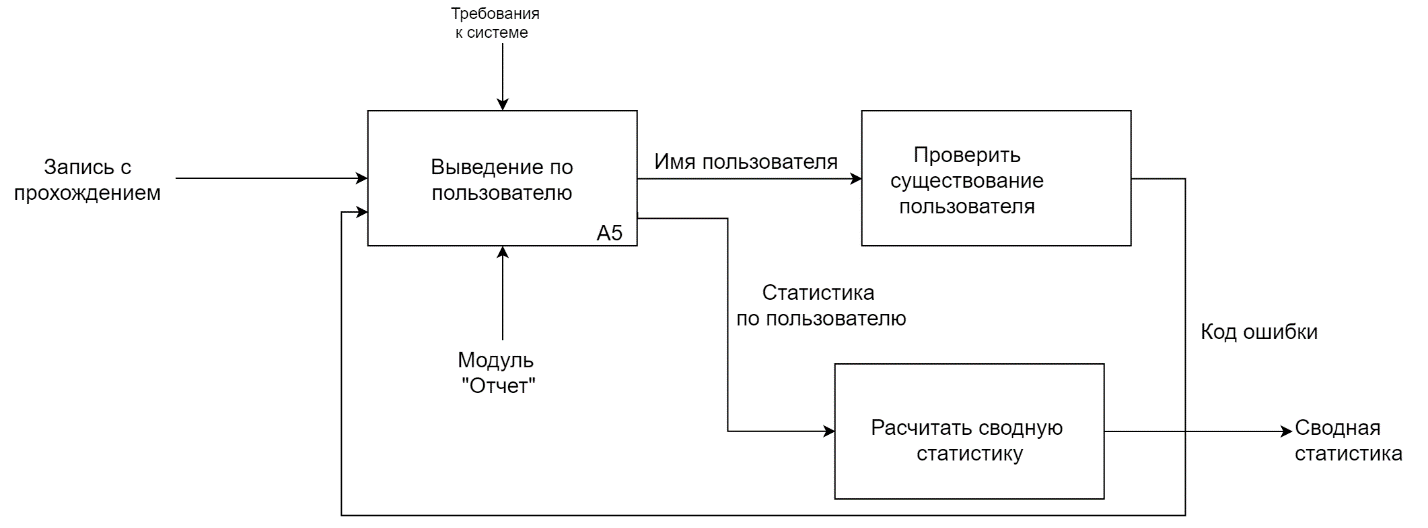


Рис. 6 – Блок выведения статистики по пользователю

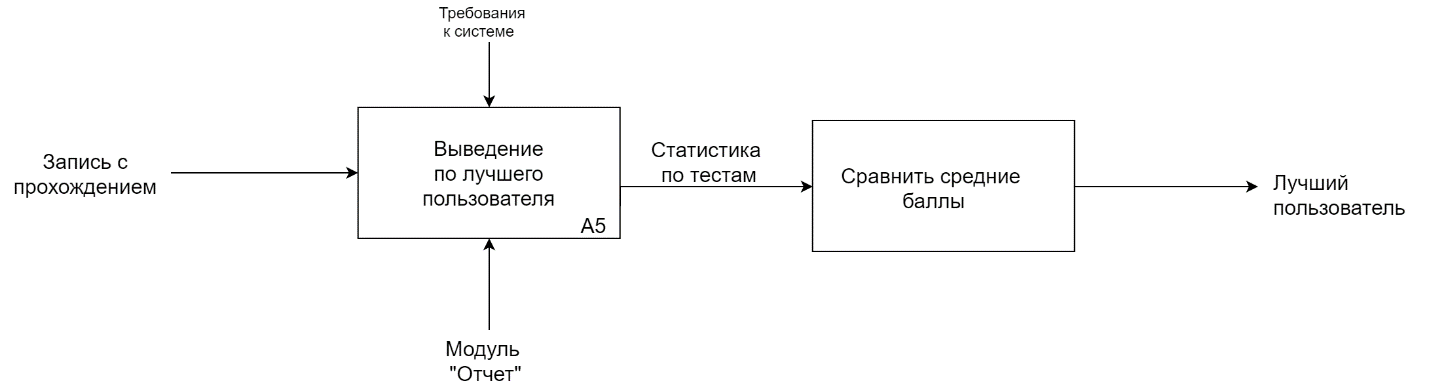


Рис. 7 – Блок определения лучшего пользователя

## 4. Проектирование алгоритмов

Модуль «Расчет» включает в себя подпрограммы:

* Процедура чтения статистики с типизированного файла;
* Процедура выведения статистики по пользователю;
* Процедура выведения статистики по тесту;
* Процедура вычисления сводной статистики по пользователю;
* Процедура вычисления сводной статистики по тесту;
* Процедура проверки наличия пользователя в базе;
* Процедура проверки наличия теста в базе;
* Процедура выведения лучшего пользователя;
* Процедура выведения лучшего пользователя за тест;

Процедура TakingDatas является основной в модуле, так как только по завершению её выполнения возможно выполнение последующих процедур расчета статистики. Данная функция обращается к типизированному файлу базы данных сохранения статистики и считывает первичные необходимые данные прохождения теста пользователем. После вызова данной процедуры возможно выведение и расчет остальных статистик.

Процедуры FullUserStatistic, FullTestStatistic осуществляют вывод статистики прохождения отдельного пользователя и теста соответственно в отдельную запись.

Процедуры SummaryStatUser, SummaryStatTest осуществляют вычисление и сохранение сводной статистики в отдельную запись.

Процедуры BestAllUser, BestTestUser осуществляют вычисление лучшего пользователя за все тесты и за отдельный тест соответственно.

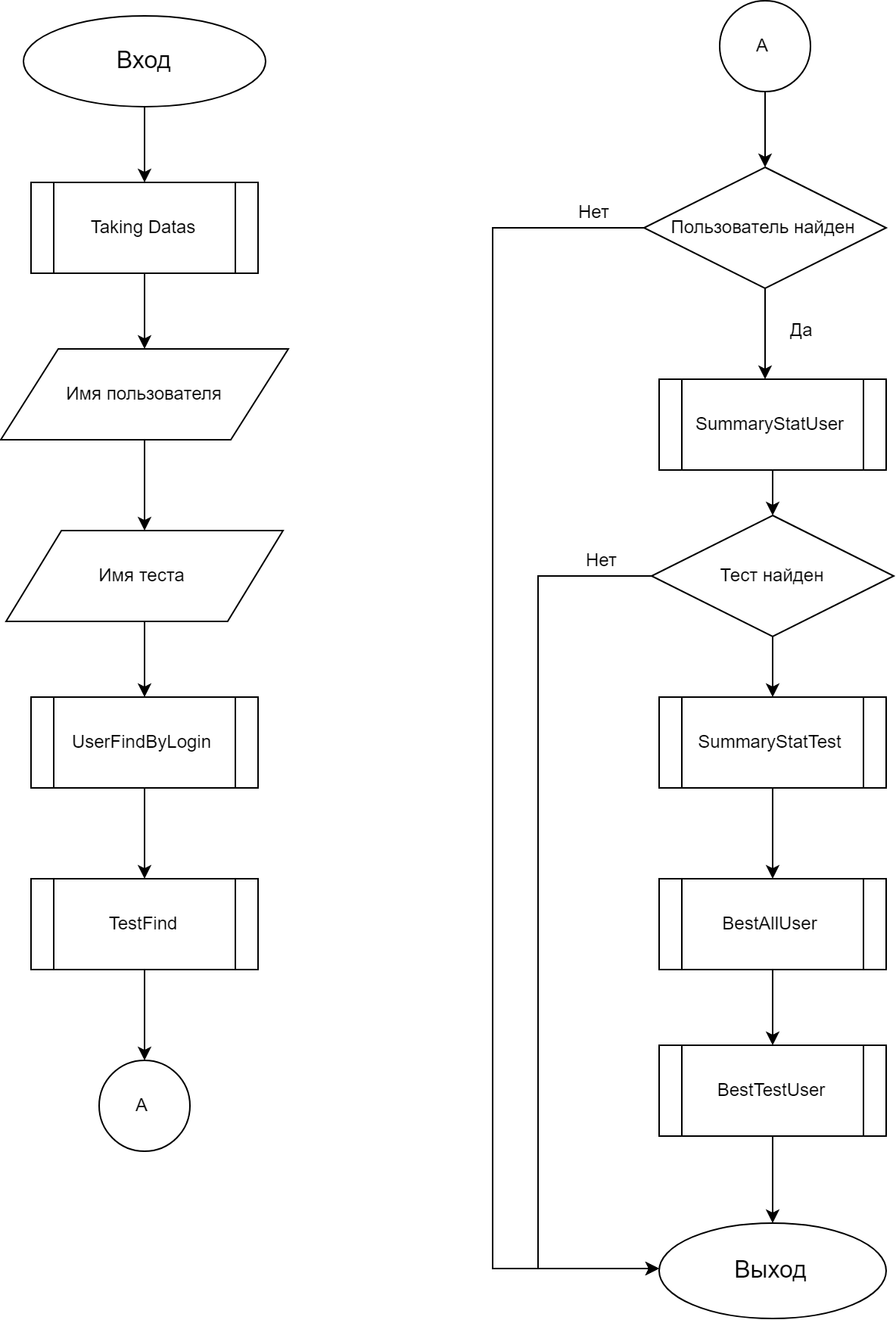


Рис. 8 – Блок схема модуля «Расчет»

## 5. Тестирование программы

Для тестирования модуля была создана программа, в которую в ручном зпорядке была занесена в типихированный файл информация о прохождении теста пользователем, затем применены соответствующие процедуры считывания информации и вынесении в отдельную запись вывода информации о пользователе.

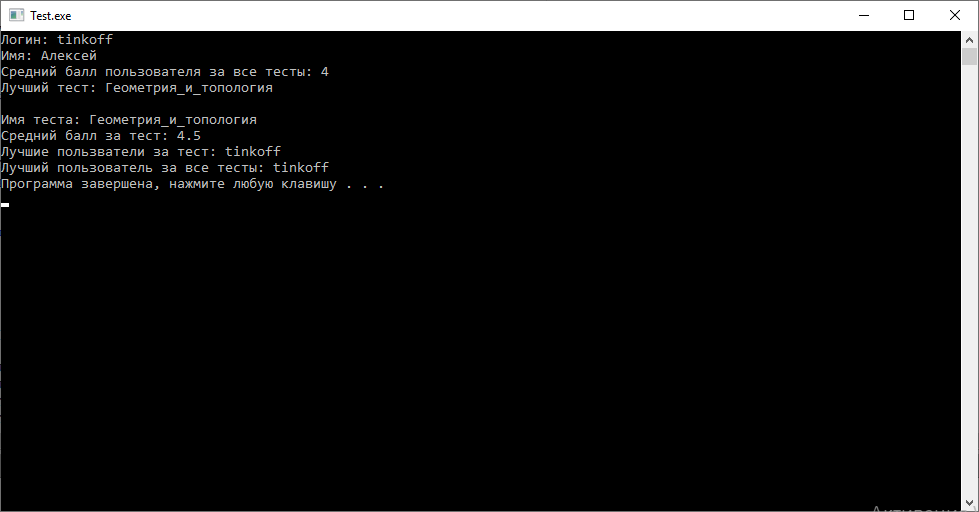


Рис.9 – Экран вывода программы, тестирующей модуль

## 6. Программная реализация алгоритмов

Модуль расчет является модулем без вывода меню, результатов, и какой либо информации на экран. Главный принцип реализации заключается в последовательности использования процедур. Первой обязательно должна быть процедура считывания первичной статистики с типизированного файла и сохранение статистики пользователя в запись. Данные файл предварительно будут занесены в модуле «Статистика». Стоит отметить, что для более стабильной работы с файлом и скоращения количества кода, при использовании модуля «Расчет» необходимо подключить модуль, в котором будет содержаться описание способа хранения данных. В данном случае будут использоваться записи. Модуль с описанием должен быть подключен как к модулю «Расчет», так и к модулю «Статистика» и «Отчет».

Далее должны использоваться процедуры либо вывода статистики по пользователю, либо по тесту. После чего будет доступная реализация вычисления сводной статистики по тесту или пользователю. Основным входным данным к этим процедурам послужит имя пользователя и имя теста. Первыми при обращении к процедуры вывода статистики выполнятся процедуры поиска пользователя или теста. Если тест/пользователь, будет найден, то запустится цикл поиска всех соответствующих запрашиваему имени тестов/пользователей. Для вывода пользователя и теста созданы отдельные записи и записи вывода присвоятся все значения прохождения тестов или пользователя, в зависимости от того, к какой операции происходит обращение.

После вычисления сводной статистики будут известны средние значения, и, сравнив их, будет возможным определение лучшего пользователя за каждый тест, и за все тесты соответственно.

## Заключение

В ходе выполнения курсовой работы были усвоены навыки основ составления баз данных, принципов расчета сводной статистики, а так же некоторые особенности платформы, на которой был выполнен курсовой проект. Был составлен модуль, осуществляющий работу с базой данных.

## Библиографический список

1.  Голицына, О. Л. Базы данных / О.Л. Голицына, Н.В. Максимов, И.И. Попов. - М.: Форум, **2015**. - 400 c.

2. Огнева М.В., Кудрина Е.В. О38 TurboPascal: первые шаги. Примеры и упражнения: Учеб.пособие. Изд.3-е, перераб. и доп. - Саратов: Изд-во “Научная книга”, 2008. - 100 с.

3. Павловская, Т.А. Паскаль. Программирование на языке высокого уровня/ Т.А. Павловская. – СПБ.: Питер, 2007. – 393с.

4. Черкасов, М.А. Практический курс программирования на Паскале/ М.А. Черкасов. – М.: МАИ, 2005. – 180с.

5. Бежанова М.М., Москвина Л.А. Практическое программирование. Приемы создания программ на языке Паскаль

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Приложение А (тестовые вопросы)  |  |  | | --- | --- | | **1.** | **База данных - это?** | |  | набор данных, собранных на одной дискете; | |  | данные, предназначенные для работы программы; | |  | совокупность взаимосвязанных данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и обработки данных; (Правильно) | |  | данные, пересылаемые по коммуникационным сетям. | |
| |  |  | | --- | --- | | **2.** | **Фактографическая база данных - это?** | |  | БД, которая содержит краткие сведения об описываемых объектах,  представленные в строго определенном формате; (Правильно) | |  | БД, которая содержит обширную информацию самого разного типа: текстовую, графическую, звуковую, мультимедийную; | |  | БД, которая содержит информацию определенной направленности; | |  | БД, которая содержит информацию отдельного пользователя ЭВМ. |  | |
| |  |  |  |  | | --- | --- | --- | --- | | **3.** | **Примером фактографической базы данных (БД) является БД, содержащая:** |  |  | |  | сведения о кадровом составе учреждения; (Правильно) |  |  | |  | законодательные акты; |  |  | |  | приказы по учреждению; |  |  | |  | нормативные финансовые документы. |  |  | |
| |  |  | | --- | --- | | **4.** | **Документальная база данных - это?** | |  | БД, которая содержит краткие сведения об описываемых объектах,  представленные в строго определенном формате; | |  | БД, которая содержит обширную информацию самого разного типа: текстовую, графическую, звуковую, мультимедийную; (Правильно) | |  | БД, которая содержит информацию определенной направленности; | |  | БД, которая содержит информацию отдельного пользователя ЭВМ. | |
| |  |  | | --- | --- | | **5.** | **Примером документальной базы данных является БД, содержащая:** | |  | законодательные акты; (Правильно) | |  | сведения о кадровом составе учреждения; | |  | сведения о финансовом состоянии учреждения; | |  | сведения о проданных билетах. | |
| |  |  | | --- | --- | | **6.** | **Ключами поиска в  системе управления базами данных называются:** | |  | диапазон записей файла БД, в котором осущест­вляется поиск; | |  | логические выражения, определяющие условия поиска; | |  | поля, по значению которых осуществляется поиск; | |  | номера записей, удовлетворяющих условиям по­ иска; (Правильно) | |  | номер первой по порядку записи, удовлетворяю­ щей условиям поиска. | |
| |  |  | | --- | --- | | **7.** | **Информационная система - это?** | |  | система, в которой хранится информация о состоянии сети INTERNET | |  | комплекс    аппаратно-программных средств, предназначенный для хранения й поиска информации; | |  | совокупность базы данных и всего комплекса аппаратно-программных средств для ее хранения, изменения и поиска информации, для взаимодействия с пользователем; (Правильно) | |  | система, которая предоставляет пользователю требуемую им информацию. | |
| |  |  | | --- | --- | | **8.** | **Иерархическая база данных - это?** | |  | БД, в которой информация организована в виде прямоугольных таблиц; | |  | БД, в которой элементы в записи упорядочены, т.е. один элемент считается главным, остальные подчиненными; (Правильно) | |  | БД, в которой записи расположена в произвольном порядке; | |  | БД, в которой существует возможность устанавливать дополнительно к вертикальным иерархическим связям горизонтальные связи. | |
| |  |  | | --- | --- | | **9.** | **В иерархической базе данных совокупность данных и связей между ними описывается:** | |  | таблицей; | |  | сетевой схемой; | |  | древовидной структурой; (Правильно) | |  | совокупностью таблиц. | |
| |  |  | | --- | --- | | **10.** | **Примером иерархической базы данных является:** | |  | страница классного журнала; | |  | каталог файлов, хранимых на диске; (Правильно) | |  | расписание поездов; | |

## Приложение Б (листинг программных модулей).

//Модуль расчета сводной статистики

**unit** SummaryStat;

**interface**

**uses** Type89;

**procedure** TakingDatas(**const** FileRead: **file of** TStatistic; **var** TotalInfo: TStatisticList; FileName: String);

**procedure** FullUserStatistic(Login: TCode; **const** Stat: TStatisticList;

**var** TotalStat: TUserStatistic; **var** ErrorCode: integer);

**procedure** SummaryStatUser(Login: String; Stat: TStatisticList; **var** SummStat: TUserStatistic);

**procedure** FullTestStatistic(TestName: TName; **const** Stat: TStatisticList;

**var** TotalStat: TTestStatistic; **var** ErrorCode: integer);

**procedure** SummaryStatTest(**const** TestName: TName; **var** SummStat: TTestStatistic; Stat: TStatisticList);

**procedure** bestAllUser(Stat:TStatisticList; SummStat: TUserStatistic; **var** bestUser:TUserInfo);

**procedure** BestTestUser(**var** TestInfo: TTestStatistic; **const** Stat: TStatisticList);

**implementation**

**procedure** TakingDatas(**const** FileRead: **file of** TStatistic; **var** TotalInfo: TStatisticList; FileName: String);

**var**

i:Integer;

**begin**

assign(FileRead, Filename);

Reset(FileRead);

i := 0;

**while not** (EOF(FileRead)) **do**

**begin**

Read(FileRead, TotalInfo[i]);

i := i + 1;

**end**;

close(FileRead);

**end**;

//Статистика прохождения теста пользователем

**procedure** UserFindByLogin(Login: TCode; **const** Stat: TStatisticList;

**var** UserInfo: TUserInfo; **var** ErrorCode: integer);

**var**

i, n: integer;

**begin**

i := 0;

n := Length(Stat);

**while** i <= n - 1 **do**

**begin**

**if** Stat[i].User.Login = Login **then**

**begin**

ErrorCode := 0;

UserInfo.Login := Stat[i].User.Login;

UserInfo.UserName := Stat[i].User.UserName;

**exit**;

**end**;

i := i + 1;

**end**;

ErrorCode := -1;//not found!

**end**;

**procedure** FullUserStatistic(Login: TCode; **const** Stat: TStatisticList;

**var** TotalStat: TUserStatistic; **var** ErrorCode: integer);

**var**

UserInfo: TUserInfo;

i, k, n: integer;

**begin**

//ищем пльзователя

UserFindByLogin(Login, Stat, UserInfo, ErrorCode);

**if** ErrorCode <> 0 **then**

**exit**;

//пользователь найден, рабоаем!

i := 0; k := -1;

n := Length(Stat);

TotalStat.User := UserInfo;

//заполняем статистику пользователя

**while** i <= n - 1 **do**

**begin**

**if** Stat[i].User.Login = Login **then**

**begin**

k := k + 1;

SetLength(TotalStat.TestsRun, k + 1);

TotalStat.TestsRun[k] := Stat[i];

**end**;

i := i + 1;

**end**;

**end**;

**procedure** SummaryStatUser(Login: String; Stat: TStatisticList; **var** SummStat: TUserStatistic);

**var**

j, i, Max, count: Integer;

**begin**

//avg score user

count := 0;

SummStat.AvgUserPerTest := 0;

**for** i := 0 **to** Length(SummStat.TestsRun) - 1 **do**

**if** SummStat.TestsRun[i].User.Login = Login **then**

**begin**

**for** j := 0 **to** Length(SummStat.TestsRun) - 1 **do**

**begin**

SummStat.AvgUserPerTest := SummStat.AvgUserPerTest + SummStat.TestsRun[i].TestRun.Score;

count := count + 1;

**end**;

**end**;

SummStat.AvgUserPerTest := SummStat.AvgUserPerTest / count;

Max := 0;

**for** i := 1 **to** Length(SummStat.TestsRun)-1 **do**

**begin**

**if** SummStat.TestsRun[i].TestRun.Score >= SummStat.TestsRun[Max].TestRun.Score **then**

**begin**

Max := i;

SummStat.BestTest := SummStat.TestsRun[Max].Test

**end**;

**end**;

**end**;

//статистика прохождения заданного теста;

**procedure** TestFind(TestName: TName; **const** Stat: TStatisticList;

**var** TestInfo: TTestInfo; **var** ErrorCode: integer);

**var**

i, n: integer;

**begin**

i := 0;

n := Length(Stat);

**while** i <= n - 1 **do**

**begin**

**if** Stat[i].Test.TestName = TestName **then**

**begin**

ErrorCode := 0;

TestInfo.TestName := Stat[i].Test.TestName;

TestInfo.TestFile := Stat[i].Test.TestFile;

**exit**;

**end**;

i := i + 1;

**end**;

ErrorCode := -1;//not found!

**end**;

**procedure** FullTestStatistic(TestName: TName; **const** Stat: TStatisticList; **var** TotalStat: TTestStatistic; **var** ErrorCode: integer);

**var**

TestInfo: TTestInfo;

//UserInfo: TUserInfo;

i, k, n: integer;

**begin**

//ищем пльзователя

TestFind(TestName, Stat, TestInfo, ErrorCode);

**if** ErrorCode <> 0 **then**

**exit**;

i := 0; k := -1;

n := Length(Stat);

TotalStat.Test := TestInfo;

//заполняем статистику ТЕСТА

**while** i <= n - 1 **do**

**begin**

**if** Stat[i].Test.TestName = TestName **then**

**begin**

k := k + 1;

SetLength(TotalStat.TestsRun, k + 1);

TotalStat.TestsRun[k] := Stat[i];

**end**;

i := i + 1;

**end**;

**end**;

**procedure** SummaryStatTest(**const** TestName: TName; **var** SummStat: TTestStatistic; Stat: TStatisticList);

**var**

i, count: Integer;

**begin**

//avg score per test

count := 0;

SummStat.AvgPerTest := 0;

**for** i := 0 **to** length(SummStat.TestsRun) - 1 **do**

**begin**

**if** SummStat.TestsRun[i].Test.TestName = TestName **then**

SummStat.AvgPerTest := SummStat.AvgPerTest + SummStat.TestsRun[i].TestRun.Score;

count := count + 1;

**end**;

SummStat.AvgPerTest := SummStat.AvgPerTest / count;

//avg score all tests

SummStat.AvgScoreTests := 0;

**for** i := 0 **to** length(stat) - 1 **do**

SummStat.AvgScoreTests := SummStat.AvgScoreTests + Stat[i].TestRun.Score;

SummStat.AvgScoreTests := SummStat.AvgScoreTests / (1);

**end**;

//лучшие пользователи за данный тест

**procedure** BestTestUser(**var** TestInfo: TTestStatistic; **const** Stat: TStatisticList);

**var**

k, i, Max: Integer;

**begin**

max:=0; k:=-1;

**for** i := 0 **to** length(TestInfo.TestsRun)-1 **do**

**begin**

**if** TestInfo.TestsRun[i].TestRun.Score >= TestInfo.TestsRun[Max].TestRun.Score **then**

**begin**

max := i;

**end**;

**end**;

**for** i := 0 **to** length(TestInfo.TestsRun)-1 **do**

**if** TestInfo.TestsRun[max].TestRun.Score = TestInfo.TestsRun[i].TestRun.Score **then**

**begin**

k:= k+1;

Setlength(TestInfo.BestUsers,k+1);

TestInfo.BestUsers[k]:= TestInfo.TestsRun[i].User;

**end**;

**end**;

//procedure SummaryStatUser(Login: String; Stat: TStatisticList; var SummStat: TUserStatistic);

//Вызовем эту процедуру для вывода среднего

**procedure** bestAllUser(Stat:TStatisticList; SummStat: TUserStatistic; **var** bestUser:TUserInfo);

**var**

maxNUser,i:Integer;

maxScore:real;

**begin**

maxScore := -1;

maxNUser := 0;

**for** i := 1 **to** length(Stat)-1 **do**

**begin**

SummaryStatUser(Stat[i].User.Login,Stat,SummStat);

**if** SummStat.AvgUserPerTest > MaxScore **then**

MaxScore := SummStat.AvgUserPerTest;

maxNUser:=i;

**end**;

SummaryStatUser(Stat[maxNUser].User.Login,Stat,SummStat);

bestUser := SummStat.User;

**end**;

**end**.

// Модуль описания хранения данных

**unit** Type89;

**Interface**

**type**

TCode = string[10];

TDate = string[20];

TName = string[80];

TUserInfo = **record**

Login: TCode;

UserName: TName;

**end**;

TTestInfo = **record**

TestName: TName;

TestFile: TName;

**end**;

TTestRun = **record**

StartDate, EndDate: TDate;

Score: integer;

Mark: TCode;

**end**;

TStatistic = **record**

User: TUserInfo;

Test: TTestInfo;

TestRun: TTestRun;

**end**;

TStatisticList = **array of** TStatistic;

TStatisticFile = **file of** TStatistic;

TUserStatistic = **record**

User: TUserInfo;

AvgUserPerTest:Real;

BestTest: TTestInfo;

TestsRun: **array of** TStatistic;

**end**;

TTestStatistic = **record**

Test: TTestInfo;

BestUsers: **array of** TUserInfo;

AvgScoreTests:Real;

AvgPerTest:Real;

TestsRun: **array of** TStatistic;

**end**;

**implementation**

**end**.

//Программа тестирования модуля

**program** Test;

**uses** SummaryStat, Type89, TestFileCreate,crt;

**var**

TestFile1, TestFile2, TestFile3, TestFile4: TStatisticFile;

AStat: TStatisticList;

ErrorCode: integer;

TotalStat: TUserStatistic;

TestStat: TTestStatistic;

bestUser:TUserInfo;

m, n, p, q: String;

**begin**

write('Имя файла: ');

readln(m);

readln(p);

readln(n);

readln(q);

SetLength(AStat, 4);

AStat[0].User.Login := 'smg';

AStat[0].User.UserName := 'Олег';

AStat[0].TestRun.Score := 5;

AStat[0].Test.TestName := 'ОКН';

AStat[1].User.Login := 'tinkoff';

AStat[1].User.UserName := 'Алексей';

AStat[1].TestRun.Score := 4;

AStat[1].Test.TestName := 'Геометрия\_и\_топология';

AStat[2].User.Login := 'tinkoff';

AStat[2].User.UserName := 'Алексей';

AStat[2].TestRun.Score := 3;

AStat[2].Test.TestName := 'Алгебра';

AStat[3].User.Login := 'tinkoff';

AStat[3].User.UserName := 'Алексей';

AStat[3].TestRun.Score := 5;

AStat[3].Test.TestName := 'Геометрия\_и\_топология';

Assign(TestFile1, m);

Assign(TestFile2, p);

Assign(TestFile3, n);

Assign(TestFile4, q);

rewrite(TestFile1);

rewrite(TestFile2);

rewrite(TestFile3);

rewrite(TestFile4);

write(TestFile1, AStat[0]);

write(TestFile1, AStat[1]);

write(TestFile1, AStat[2]);

write(TestFile1, AStat[3]);

TakingDatas(TestFile1, AStat, m);

TakingDatas(TestFile2, AStat, p);

TakingDatas(TestFile3, AStat, n);

TakingDatas(TestFile4, AStat, q);

FullUserStatistic('tinkoff', AStat, TotalStat, ErrorCode);

SummaryStatUser('tinkoff', AStat, TotalStat );

FullTestStatistic( 'Геометрия\_и\_топология', AStat, TestStat, ErrorCode);

SummaryStatTest('Геометрия\_и\_топология', TestStat, AStat);

BestTestUser(TestStat, AStat);

bestAllUser(AStat, TotalStat,bestUser);

ClrScr;

Writeln('Логин: ', TotalStat.User.Login);

Writeln('Имя: ', TotalStat.User.UserName);

Writeln('Средний балл пользователя за все тесты: ',TotalStat.AvgUserPerTest);

Writeln('Лучший тест: ', TotalStat.BestTest.TestName);

Writeln;

Writeln('Имя теста: ', TestStat.Test.TestName);

Writeln('Средний балл за тест: ', TestStat.AvgPerTest);

Write('Лучшие пользватели за тест: ');

**for var** i:Integer := 0 **to** length(TestStat.BestUsers)-1 **do**

Writeln(TestStat.BestUsers[i].Login);

writeln('Лучший пользователь за все тесты: ',bestUser.login);

**end**.