**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**«Новосибирский Государственный Технический Университет»**

**«Институт Социальных Технологий и Реабилитации»**

**Кафедра естественно-научных дисциплин**

**КУРСОВАЯ РАБОТА**

**На тему: «Разработка Win32 GUI-приложений на языке Free Pascal**

**в IDE Lazarus»**

Выполнили: студенты группы И-72

Кыхтева Мария и Джанкабулов Константин

Руководитель:

Преподаватель Устинов В.В.

Новосибирск 2021

Содержание

[Введение 3](#_Toc39501077)

[Среда программирования Lazarus 4](#_Toc39501078)

[Операторы, используемые для написания программы 8](#_Toc39501079)

[Компоненты Delphi используемые в программе. 11](#_Toc39501080)

[Типы данных, используемые для написания программы 13](#_Toc39501081)

[Операции и выражения, используемые для написания программы 14](#_Toc39501082)

[Функции, используемые для написания программы 14](#_Toc39501083)

[Описание программы 15](#_Toc39501084)

[Описание операций для пользователя 16](#_Toc39501085)

[Описание операций инженерного калькулятора для пользователя 19](#_Toc39501086)

[Алгоритм реализации программы «Стипендия» 23](#_Toc39501087)

[Заключение 29](#_Toc39501088)

[Листинг программы 1 30](#_Toc39501089)

[Листинг программы 2 41](#_Toc39501090)

# Введение

Первая программа была разработана в среде Lazarus. Pascal является наиболее универсальным и легко изучаемым языком, среди множества других. Lazarus прост в изучении при этом средства языка программирования Pascal позволяют использовать многие возможности программирования.

**Lazarus** — [открытая](https://ru.wikipedia.org/wiki/Открытое_программное_обеспечение) среда разработки программного обеспечения на языке Object Pascal для компилятора Free Pascal (часто используется сокращение FPC — Free Pascal Compiler, бесплатно распространяемый компилятор языка программирования Pascal). Интегрированная среда разработки предоставляет возможность кроссплатформенной разработки приложений в Delphi-подобном окружении.

Позволяет достаточно несложно переносить Delphi-программы с графическим интерфейсом в различные операционные системы: Linux, FreeBSD, Mac OS X, Microsoft Windows, Android. Начиная с Delphi XE2 в самом Delphi имеется возможность компиляции программ для Mac OS X, с версии XE4 — для iOS, с версии XE5 — для Android.

Среда разработки предоставляет доступ только к тем участкам кода, с которыми необходимо работать, скрывая основную часть программы, которая создается автоматически самой средой разработки.

**Delphi** - императивный структурированный объектно-ориентированный язык программирования со строгой статической типизацией переменных. Основная область использования — написание прикладного программного обеспечения.

Среда разработки предоставляет доступ только к тем участкам кода, с которыми необходимо работать, скрывая основную часть программы, которая создается автоматически самой средой разработки.

**Задачи данной курсовой работы:**

* Изучить среду программирования Lazarus
* Изучить возможности и инструменты среды разработки
* Написание программ
* Изучение возможностей среды Delphi

**Цель данной курсовой работы:**

Разработка Win32 GUI-приложений на языке Free Pascal в IDE Lazarus:

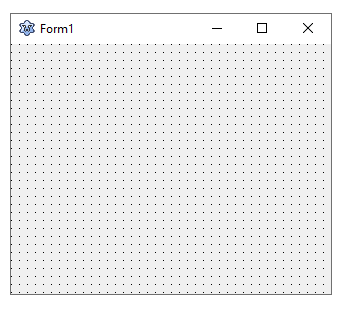
* «Калькулятор»
* «Стипендия»

# Среда программирования Lazarus

**Окна среды программирования Lazarus:**

**1.Окно формы**

— это проект интерфейса будущего программного продукта.



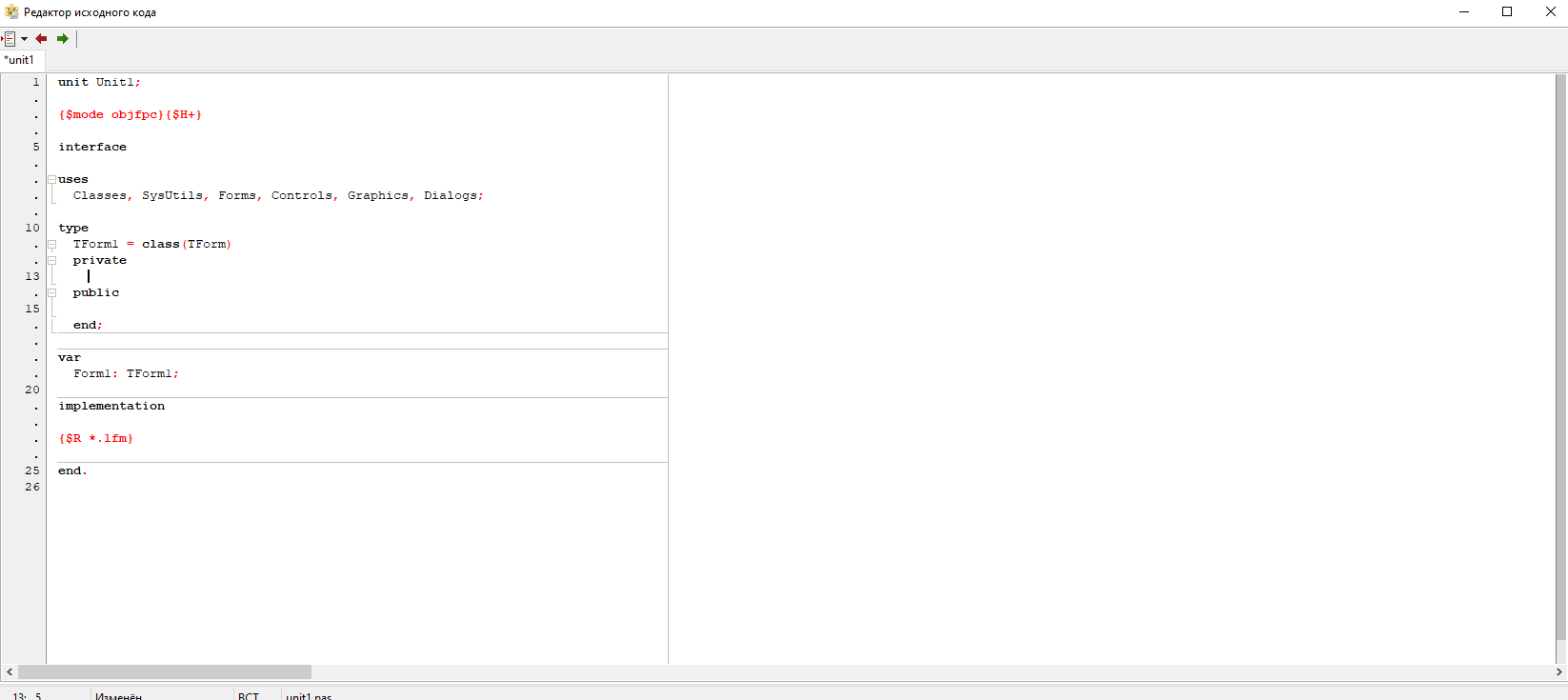
Изначально это окно содержит только стандартные элементы – строку заголовка и кнопки развертывания, свертывания и закрытия. Рабочая область окна заполнена точками координатной сетки. Координатная сетка отображается только на этапе создания программы.

Задача программиста – использовать панель компонентов, и заполнить форму различными элементами интерфейса, создавая тем самым внешний вид своей программы.

Команды настройки окна формы находятся на вкладке *Редактор форм* в меню *Окружение — Настройки окружения.*

**2.Окно редактора Lazarus**

Окно редактора тесно связано с окном формы и появляется вместе с ним при создании нового проекта.



Окно редактора предназначено для создания и редактирования текста программы, который создается по определенным правилам и описывает некий алгоритм. Если окно формы определяет внешний вид программный, то программный код, записанный в окне редактора, отвечает за ее поведение.

Изначально окно редактора содержит текст, обеспечивающий работу пустой формы. Этот код появляется в окне автоматически, а программист вносит дополнения, соответствующие задачам и функциям программы.

Окно редактора по умолчанию может находиться на первом плане и закрывает окно формы. Переключаться между этими окнами можно командой: *Просмотр* - *Переключить модуль/форму, клавишей* **F12** или просто щелчком мыши.

**3.Панель компонентов**

Панель компонентов расположена под главным меню. Она состоит из большого числа групп, в которых располагаются соответствующие компоненты.

*Компонент – это функциональный элемент интерфейса,*

обладающий определенными свойствами. Размещая компоненты на форме, программист создает внешний вид своей будущей программы – окна, кнопки, переключатели, поля ввода и т.п.

Для *внедрения нового компонента*на форму нужно сделать два щелчка мышкой:

•в панели компонентов, для выбора компонента;

•в рабочем пространстве формы, для указания положения левого

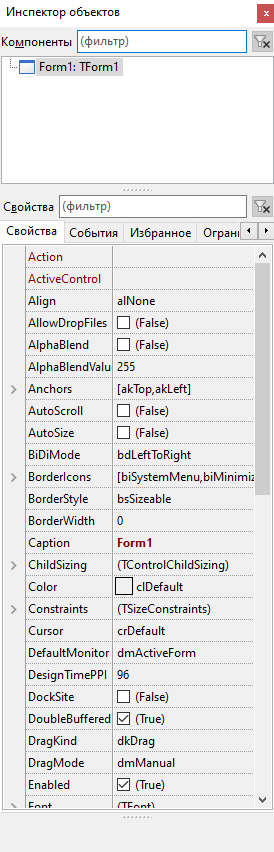
верхнего угла компонента.

Компоненты объединяются в группы по функциональному признаку. После создания проекта по умолчанию открывается список группы **Standard**, содержащий основные элементы диалоговых окон. Просмотреть другие группы можно, раскрывая их щелчком по соответствующей вкладке.

**4.Инспектор объектов**

–обеспечивает простой и удобный интерфейс для изменения свойств объектов Lazarus и управления событиями, на которые реагирует объект.

Окно Инспектора Объектов имеет две страницы. В верхней части окна имеется выпадающий список всех компонентов, размещенных на форме. В нем вы можете выбрать тот компонент, свойства и события которого вас интересуют.



Вкладки инспектора объектов представляют собой таблицу. В левой колонке расположены названия свойств или событий, в правой – конкретные значения свойств или имена подпрограмм, обрабатывающих события. Чтобы выбрать свойство или событие, необходимо щелкнуть левой кнопкой мыши по соответствующей строке. Свойства, отображенные в таблице, могут быть простыми или сложными. Простые свойства определены единственным значением.

Например, свойство Caption (Заголовок) определяется строкой символов, свойства Height (Высота) и Width (Ширина) – числом, свойство Enabled (Доступность) – значениями True (Истина) и False (Ложь).

Сложные свойства определяются совокупностью значений. Например, свойство Font (Шрифт). Слева от имени этого свойства стоит знак «+». Это означает, что свойство сложное, и при щелчке по значку «+» откроется список его составляющих. Активизировать значение любого свойства можно обычным щелчком мыши. При этом в конце строки может появиться либо кнопка с символом троеточия, либо кнопка со стрелкой, направленной вниз. Щелчок по троеточию откроет диалоговое окно для установки значений сложных свойств (например, Font). Обращение к стрелке приведет к раскрытию списка возможных значений простого свойства (например, Enabled).

# Операторы, используемые для написания программы

**1.Оператор присваивания**

В результате выполнения оператора присваивания значение переменной меняется – ей присваивается новое значение. Оператор присваивания состоит из двух символов: двоеточия и знака равенства. Символы **:=** всегда пишут слитно. Пробелы допускаются перед символом двоеточия и после символа равенства.

В общем случае оператор присваивания имеет вид:

**имя\_переменной := значение;**

где **значение** — это выражение, переменная, константа или функция. Выполняется оператор так. Сначала вычисляется значение выражения, указанного в правой части оператора, а затем его результат записывается в область памяти (переменную), имя которой указано слева. Например, запись **a:=b** означает, что переменной **а** присваивается значение выражения **b**.

Типы переменной **a** и выражения **b** должны совпадать или быть совместимыми для присваивания, то есть тип, к которому принадлежит **b**, должен находиться в границах типа переменной **а**.

Оператор присваивания, как и любой другой оператор в языке Free Pascal, отделяется от других точкой с запятой.

**2.Составной оператор**

Этот оператор еще называют операторными скобками, он начинается ключевым словом **Begin** и заканчивается словом **End**. Между этими словами можно записывать любое число операторов, но чисто внешне он воспринимается как один оператор.

**3.Условный оператор if**

При решении большинства задач порядок вычислений зависит от определённых условий, например от исходных данных или от промежуточных результатов, полученных на предыдущих шагах программы. Для организации вычислений в зависимости от какого-либо условия в языке Free Pascal используется условный оператор **if..then..else**, который в общем виде записывается так:

**if** условие **then** оператор\_1 **else** оператор\_2;

В этой записи зарезервированные слова выделены жирным шрифтом. Работает он следующим образом: если условие имеет значение «истина», то выполняется оператор\_1, иначе – оператор\_2.

Или если в задаче требуется, чтобы в зависимости от значения условия выполнялся не один оператор, а несколько, необходимо использовать составной оператор:

**if** условие **then**

**begin**

оператор\_1;

оператор\_2;

...

оператор\_n;

**end**

**else**

**begin**

оператор\_1;

оператор\_2;

...

оператор\_n;

**end**;

**4.Условный оператор case**

При осуществлении проверки состоящей из множества условий (оператором if), допустим, нам надо выполнить один из пяти или более операторов, получиться довольно громоздкая и неудобочитаемая запись.

Для этих целей удобней использовать оператор выбора Case. С его помощью, возможно, делать выбор из необходимого количества вариантов. Структура в языке Lazarus оператора Case имеет следующий вид:

case переменная of

значение 1 : оператор\_1;

значение 2 : оператор\_2;

----

значение N : оператор\_N;

else оператор;

end;

В этой записи зарезервированные слова также выделены жирным шрифтом.

Переменная - может быть одним из стандартных типов: символьным типом, целым числом, перечисляемым типом. Вместо значения может стоять простой оператор.

значение N - произвольное значение, того же типа, что и переменная.

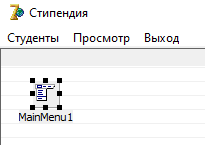
оператор N - любые операторы.

Работа оператора Case осуществляется следующим образом:

Вначале если необходимо вычисляется значение переменной. Затем ищется соответствие переменной значению N. При равенстве происходит выполнение оператора N и оператор case завершает свою работу. Если совпадений не найдено, то выполняется оператор расположенный после слова else. При отсутствии оператора else, когда нет соответствия, оператор case завершит свою работу и будет выполнен следующий оператор.

### Компоненты Delphi используемые в программе.

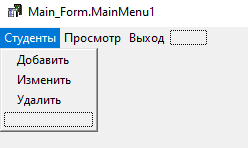
**Компонент TMainMenu**

TMainMenu – позволяет поместить главное меню в программу. При помещении TMainMenu на форму это выглядит, как просто иконка (Рис.1). Иконки данного типа называют невидимым (невизуальным) компонентом, поскольку они невидимы во время выполнения программы.

(рис 1. Невидимая иконка компонента TMainMenu)

Создание меню включает три шага:

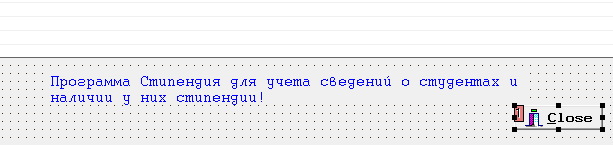
1. помещение TMainMenu на форму;
2. вызов Menu Designer (Рис.2);
3. определение пунктов меню в Menu Designer.



(рис 2. Menu Designer)

**Компонент TLabel**

TLabel – служит для отображения текста на экране (Рис.3). Можно изменить шрифт и цвет метки, если дважды щелкнуть на свойство Font в Инспекторе Объектов. Видно, что это легко сделать и во время выполнения программы, написав всего одну строчку кода.



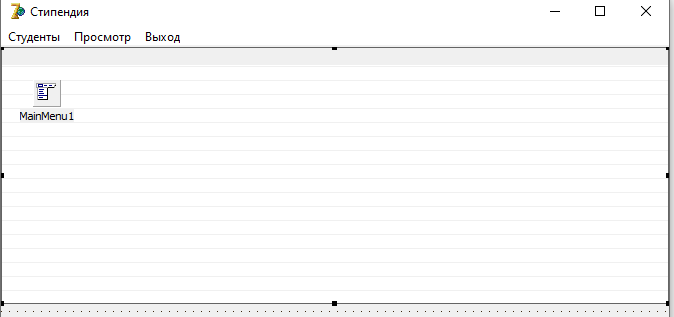
(рис 3. TLabel на форме)

Компонент представляет собой статический текст. С помощью этого компо­нента на рабочей поверхности формы можно отобразить информацию, сделать пояснения и показать названия других компонентов.

**Компонент TList**

Класс TList очень полезный универсальный контейнер списков. Он отличается от массивов, в которых он обеспечивает более богатые функциональные возможности.

В частности объекты TList могут быть отсортированы. Эта сортировка может быть с использованием любых выбранных критериев. Например, список может содержать набор объектов, которые имеют строку и численные поля. Вы можете отсортировать список по строке, по числу, по обоим, с возрастанием или убыванием, как Вы желаете. И пересортировать позже по другим критериям.



(рис 4. Tlist находится почти на всю форму программы)

# Типы данных, используемые для написания программы

Для решения задач в любой программе выполняется обработка каких-либо данных. Данные хранятся в памяти компьютера и могут быть самых различных типов: целые вещественные числа, символы, строки, массивы.

Типы данных определяют способ хранения чисел или символов в памяти компьютера. Они задают размеры ячейки, в которую будет записано то или иное значение, определяя тем самым его максимальную величину или точность задания.

Участок памяти (ячейка), в которой  хранится значение определенного типа, называется переменной. У переменной есть имя (идентификатор), тип и значение. Имя служит для обращения к области памяти, в которой хранится значение.

Во время выполнения программы значение переменной можно изменить.

В Lazarus каждая переменная перед использованием должна быть описана (объявлена). При объявлении переменной задается ее имя и тип. В оперативной памяти выделяется место для хранения переменной.

Для описания переменных  используется служебное слово **var**.

Общий вид:

**Var** имя\_переменной: тип\_переменной;

В программе использовались типы данных **REAL** и **STRING**.

**REAL** — стандартный типа данных. Применяется для представления вещественных чисел, которые могут состоять из десятичной точки и экспоненты, в отличие от типа Integer, который используется для представления целых чисел.

Внутреннее представление типа REAL:

Диапазон: 2.9e-39 ..1.7e+38

Размер: 8 байт

**STRING** — тип данных, который может содержать символы. В Lazarus является основным строковым типом данных.

Поскольку строка в Lazarus имеет формат UTF8, то никаких особых ухищрений для работы со строками тут не нужно, в переменную типа **String** можно записать любой, в том числе и русский текст. Размер строки **String** неограничен, однако имеется возможность жестко задать размер.

# Операции и выражения, используемые для написания программы

Выражение задает порядок выполнения действий над данными и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций.

|  |  |
| --- | --- |
| Оператор | Действие |
| + | Сложение |
| **-** | Вычитание |
| \* | Умножение |
| / | Деление |

# Функции, используемые для написания программы

В языке определены **стандартные функции**. Некоторые арифметические функции представлены в таблице ниже.

|  |  |
| --- | --- |
| **Обозначение** | **Действие** |
| Sqrt(n) | Квадратный корень из n |
| Sqr(n) | Квадрат n |

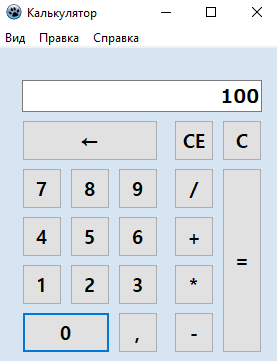
**Функции преобразования** типов часто используются при  вводе и выводе информации

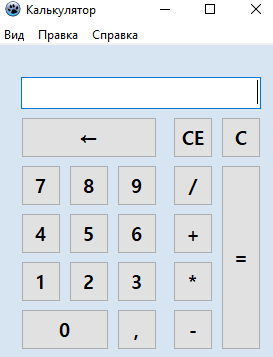
|  |  |
| --- | --- |
| **Основные функции преобразования типов** | |
| **Обозначение** | **Действие** |
| Chr(n) | Символ, код которого равен n |
| FloatToStr(n) | Строка, являющаяся изображением вещественного n |
| StrToFloat(s) | Вещественное, изображением, которого является строка s |

# Описание программы

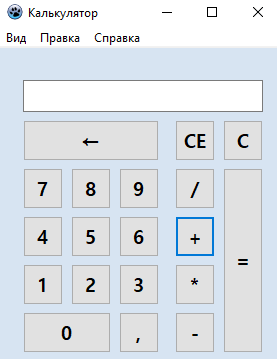
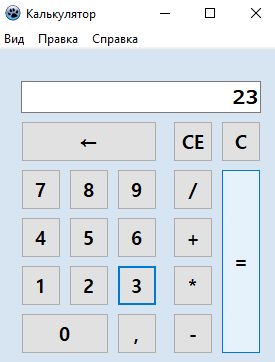
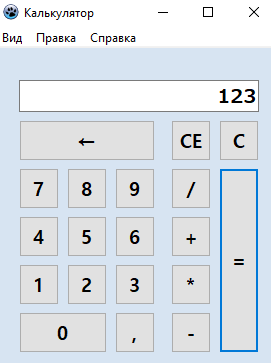
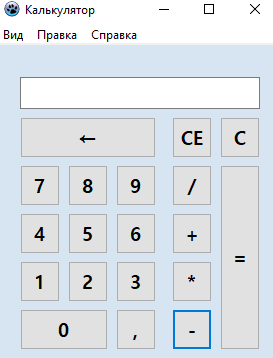
Данная программа «Калькулятор» предназначена для выполнения несложных вычислительных операций: сложения, вычитания, умножения и деления и т.д. В калькуляторе также предусмотрены возможности для выполнения более сложных операций.

Интерфейс программы спроектирован таким образом, чтобы имитировать внешний вид реального калькулятора. Его «кнопки» можно нажимать мышкой.

Для начала работы необходимо запустить исполняемый файл

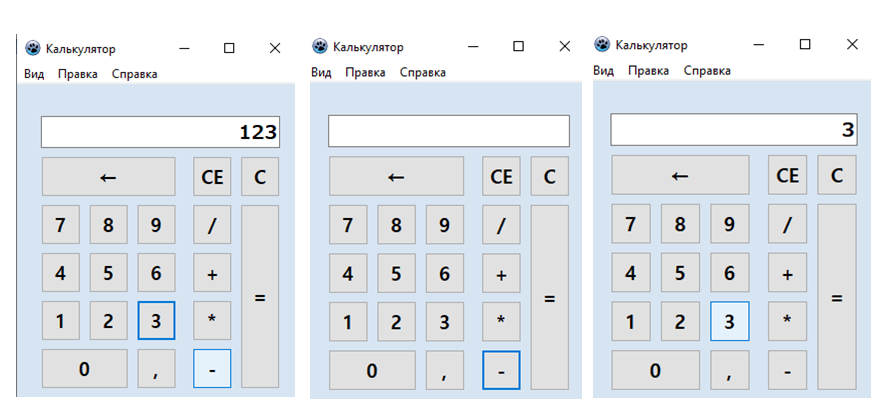


Клавиатура калькулятора содержит клавиши, нажатие которых обеспечивает ввод чисел и выполнение операций.

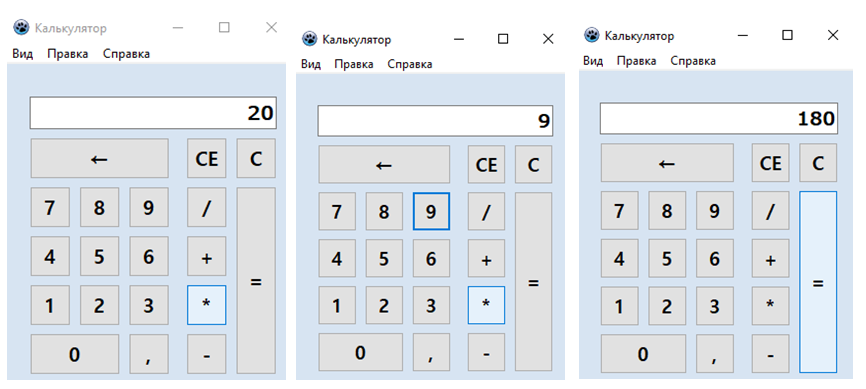


|  |
| --- |
| Описание операций для пользователя |

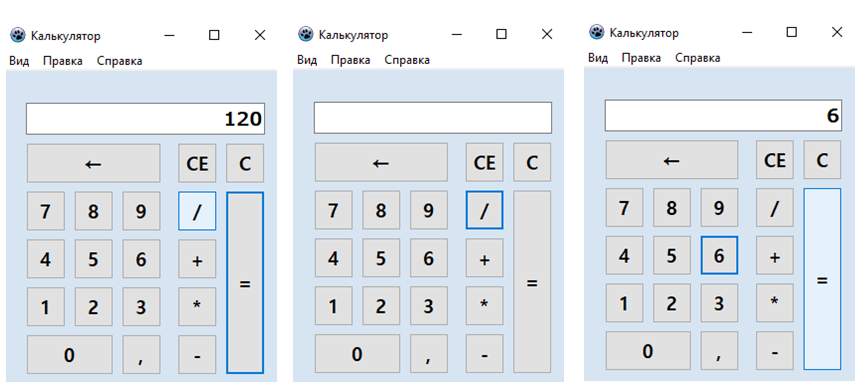
**Операция сложение** — арифметическое действие, выполняемое двумя слагаемыми аргументами: увеличение значения первого аргумента на значение второго, результат - сумма этих двух аргументов.

**Операция вычитание**  обратна сложению. Результат уменьшаемого и вычитаемого является новое число – разность.

|  |
| --- |
| Операция вычитание |

**Операция Умножение** — математическая операция над двумя аргументами, называемыми множитель и сомножитель. Заменяет неоднократное повторенное умножение, результат называется их произведением.

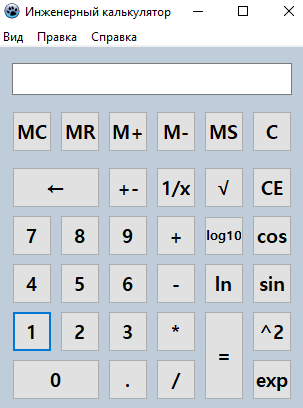
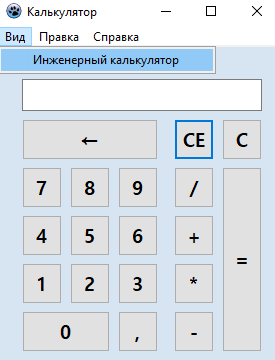
|  |
| --- |
| Операция умножение |

**Операция деление** — действие обратное умножению. Заменяет неоднократное повторенное вычитание. Аргументы имеют названия делимого и делителя, результат называется частным (может иметь остаток).

|  |
| --- |
| Операция деление |

**Кнопки [←], [CE], [C]**

|  |  |
| --- | --- |
| Кнопка **[←]** стирает последнее число | Кнопка **[CE]** удаляет только последнее введённое число, кнопка **[C]** - полная отмена операции. |

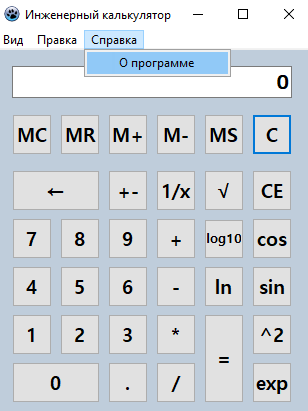
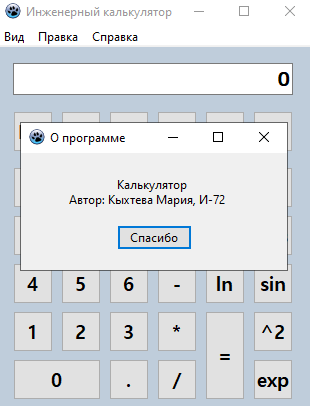
**При клике по меню «Вид», у пользователя есть возможность перейти к инженерному калькулятору, который позволяет выполнять большее количество операций.**

# Описание операций инженерного калькулятора для пользователя

|  |  |
| --- | --- |
| **Возведение в степень** —арифметическая операция, оперделяемая как резаультат множенственного умножения числа на себя. |  |
|  |  |
| **Извлечение корня —**одна из операций, обратных возведению в степень. |  |

|  |  |
| --- | --- |
| **Синус —**одна из тригонометрических функций, означающее отношение противолежащего катета к гипотенузе.  \*sin(100) |  |
|  |  |
| **Косинус —**одна из тригонометрических функций, означающее отношение прилежащего катета к гипотенузе.  \*cos(100) |  |

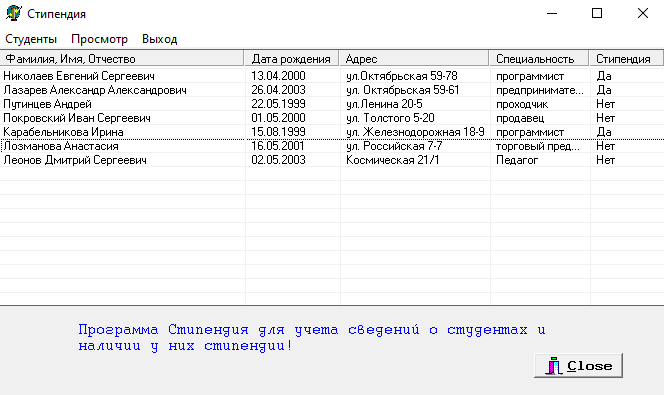
|  |  |
| --- | --- |
| **Десятичный логарифм —**логарифм по основанию 10.  \*log10 (100) |  |
|  |  |
| **Натуральный логарифм —**логарифм по основание e.  \*ln(100) |  |

**В разделе «Справка» указан автор программы.**

# Алгоритм реализации программы «Стипендия»

Данная программа может использоваться для внесения исходных данных, лицами, владеющими необходимой информацией о студентах (преподаватели, методисты), результатной информацией могут пользоваться лица, которым необходима эти данные (круг может быть ограничен). Автоматизированный способ решения рассматриваемой задачи позволяет организовать работу программы, то есть создание файла и необходимых данных в нем. Задача курсовой работы предусматривает просмотр имеющейся информации о студентах и наличие признака получения ими стипендии.

После запуска программы на экран выводится основная форма Main\_Form (рис.5), на которой расположены: главное меню и список студентов.



(Рис 5. Программа при запуске)

В первую очередь программа берет данные из определенного файла (в дальнейшем эти данные можно менять) и выводит на главный экран.

procedure TMain\_Form.FormCreate(Sender: TObject);

var

F: File of TRecord; //Файл с записями о студентах

X: TRecord; //Переменная для хранения записи

begin

AssignFile(F, 'students.dat');

if false = FileExists('students.dat') then

begin

Rewrite(F);//Создаем файл

end; //И открыть для чтения

Reset(F);

//Теперь файл с данными успешно открыт, читаем из него все записи

while not Eof(F) do

begin

Read(F, X); //Читаем запись

//Выводим информацию на экран

with Persons.Items.Add do

begin

Caption := X.Name;

SubItems.Add(DateToStr(X.DOB));

SubItems.Add(X.Address);

SubItems.Add(X.Speciality);

SubItems.Add(NoYes[X.Salary]);

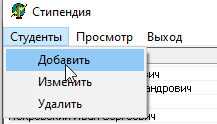
end;

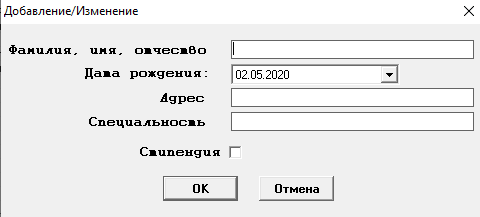
end;

CloseFile(F);

end;

Затем, при нажатии на раздел меню “студенты → добавить” срабатывает функция которая дает пользователю добавить студента в список, записывая необходимые данные.





(Рис 6 и 7. Добавление новых данных)

Процедура выглядит так:

procedure TMain\_Form.N2Click(Sender: TObject);

var

X: TRecord;

F: File of TRecord;

begin

// Добавление записи

X.Name := '';

X.DOB := Now;

X.Address := '';

X.Speciality := '';

X.Salary := False;

Change\_Form.setAll(x);

if Change\_Form.ShowModal = mrOk then

begin

//Если пользователь нажал OK,

//получаем данные с формы

Change\_Form.getAll(X);

AssignFile(F, 'students.dat');

Reset(F); //И пытаемся дописать их в файл

Seek(F, FileSize(F)); //Перемещаемся в конец файла

Write(F, X); //Записываем данные

CloseFile(F);

То же самое происходит при нажатии на подраздела меню “изменить”. Но для изменения данных необходимо выбрать из списка студента которого хотите изменить, если не выбрано на экран высветится соответстующее сообщение.

Item := Persons.Selected; //Проверяем, что элемент выбран

If Item = nil then

begin

MessageDlg('Выберите пожалуйста,запись!', mtWarning, [mbOk], 0);

Exit;

end;

Также можно удалить студента, нажав соответствующий подраздел в меню.

procedure TMain\_Form.N4Click(Sender: TObject);

//процедура удаления записи из файла

var

Item: TListItem;

i: Integer;

X: TRecord;

F: File of TRecord;

begin

Item := Persons.Selected; //Должна быть выбрана запись

if Item = nil then

begin

MessageDlg('Выберите запись, пожалуйста!', mtWarning, [mbOk], 0);

Exit;

end;

if MessageDlg('Вы уверены, что хотите удалить эту запись?',

mtConfirmation, [mbYes, mbNo], 0) = mrYes then

begin

AssignFile(F, 'students.dat');

Rewrite(F);

i := 0;

//Проходимся по всем записям

while i < Persons.Items.Count do

begin

Item := Persons.Items[i];

//Если запись не выбрана, то записываем ее в файл

if not Item.Selected then

begin

X.Name := Item.Caption;

X.DOB := StrToDate(Item.SubItems[0]);

X.Address := Item.SubItems[1];

X.Speciality := Item.SubItems[2];

X.Salary := NoYes[True] = Item.SubItems[3];

Write(F, X);

i := i + 1;

end

else

begin

//Удаляем запись с экрана, и в файл ее не записываем

Persons.Items[i].Delete;

end;

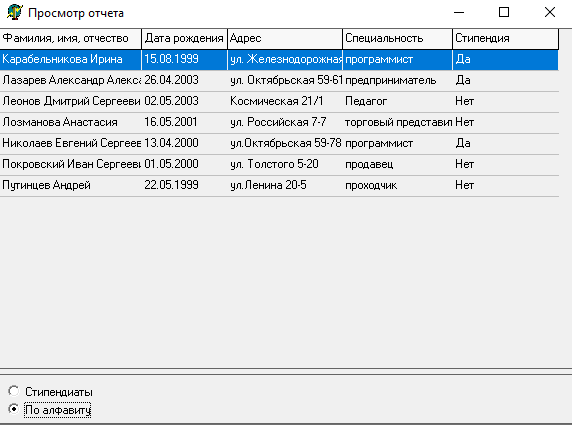
end;

CloseFile(F);

end;

end;

Просмотр отчета. Данные про студентов и отсортировать их по алфавиту либо по присутствии у них стипендии можно в разделе “просмотр”.



(Рис 8. Просмотр.)

begin

Form := TRep\_Form.Create(nil);

Form.ShowModal; //Показываем форму

Form.Free; //И уничтожаем форму

end;

## Заключение

Данная курсовая работа была выполнена в среде разработке программного обеспечения на языке Object Pascal для компилятора Free Pascal Lazarus. 2.0.8 и Borland Delphi 7.0. Были разработаны приложения «Калькулятор» и «Инженерный калькулятор» и их базовая функциональность. А так же программа «Стипендия», для добавления, изменения и удаления данных, с помощью контрастного меню.

Данная программа осуществляет различные вычислительные процессы. При исопльзовании компонентов Classes, SysUtils, Forms, Controls, Graphics, StdCtrls, ExtCtrls, Crt, Dialogs, Menus, Math. Благодаря их сочетанию мы получаем желаемый результат, без них работа программы бы не осуществлялась. Сама программа составлена из различных процедур, где каждая процедура осуществляет свою функцию. Процедура начинает работать по клику мышкой на кнопку, с описанием ее функции.

В качестве дальнейшего развития возможна доработка приложения для повышения функционала.

Программа «Стипендия» несет развлекательный характер и не подходит для использования в конкретных целях, для этого существуют более крупные проекты. Но для обучения и практики с реде программирование, ее написание было полезно.

По итогу выполнения данного проекта, каждый из нас повысил свои знания языка программирования Object Pascal, и были убеждены в его возможностях. Работа была познавательной и интересной, были использованы процедуры , функции и различные компоненты.

## Листинг программы 1

|  |
| --- |
| **Unit1** |
| unit Unit1;  {$mode objfpc}{$H+}  interface  uses  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, Menus, AboutWin, Unit2;  type  { TForm1 }  TForm1 = class(TForm)  But1: TButton;  But0: TButton;  But10: TButton;  CE: TButton;  C: TButton;  Delet: TButton;  Del: TButton;  But2: TButton;  But3: TButton;  But4: TButton;  But5: TButton;  But6: TButton;  But7: TButton;  But8: TButton;  But9: TButton;  Plus: TButton;  Umnozh: TButton;  Min: TButton;  MenuItem5: TMenuItem;  TMainMenu1: TMainMenu;  MenuItem1: TMenuItem;  MenuItem2: TMenuItem;  MenuItem3: TMenuItem;  MenuItem4: TMenuItem;  Ravno: TButton;  Edit1: TEdit;  procedure CClick(Sender: TObject);  procedure CEClick(Sender: TObject);  procedure ClickBut(Sender: TObject);  procedure DeletClick(Sender: TObject);  procedure MenuItem4Click(Sender: TObject);  procedure MenuItem5Click(Sender: TObject);  procedure ClickZnak(Sender: TObject);  procedure RavnoClick(Sender: TObject);  private  public  end;  var  Form1: TForm1;  a, b, w : Real;  znak : String;  implementation  {$R \*.lfm}  { TForm1 }  procedure TForm1.ClickBut(Sender: TObject);  begin  Edit1.Text:=Edit1.Text + (Sender as TButton).Caption;  end;  procedure TForm1.ClickZnak(Sender: TObject);  begin  a := StrToFloat(Edit1.Text);  Edit1.Clear;  znak := (Sender as TButton).Caption;  end;  procedure TForm1.RavnoClick(Sender: TObject);  begin  b := StrToFloat(Edit1.Text);  Edit1.Clear;  case znak of  '+' : w := a+b;  '-' : w := a-b;  '\*' : w := a\*b;  '/' : w := a/b;  end;  Edit1.Text := FloatToStr(w);  Edit1.Text:=Edit1.Text + (Sender as TButton).Caption;  end;  procedure TForm1.CEClick(Sender: TObject);  begin  Edit1.Clear;  end;  procedure TForm1.CClick(Sender: TObject);  begin  Edit1.Clear;  a:=0;  b:=0;  w:=0;  end;  procedure TForm1.DeletClick(Sender: TObject);  var  str: String;  begin  str:=Edit1.Text;  if str <> '' then  Delete(str, Length(str),1);  Edit1.Text := str;  end;  procedure TForm1.MenuItem4Click(Sender: TObject);  begin  wAboutWin:=TwAboutWin.Create(Application);  wAboutWin.Show;  end;  procedure TForm1.MenuItem5Click(Sender: TObject);  begin  Form2:=TForm2.Create(Application);  Form2.Show;  end;  end. |

|  |
| --- |
| **Unit2** |
| unit Unit2;  {$mode objfpc}{$H+}  interface  uses  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, Menus, Math,  StdCtrls, ExtCtrls, Crt, AboutWin;  type  { TForm2 }  TForm2 = class(TForm)  But1: TButton;  MC: TButton;  MM: TButton;  MP: TButton;  But0: TButton;  Plus: TButton;  MR: TButton;  MS: TButton;  Min: TButton;  Koren: TButton;  Expanent: TButton;  Umnozh: TButton;  PlusMinus: TButton;  But2: TButton;  But20: TButton;  Del: TButton;  Pavno: TButton;  Stepen: TButton;  OneDelX: TButton;  Sin1: TButton;  Cos1: TButton;  CE: TButton;  C: TButton;  Delet: TButton;  But3: TButton;  Log1: TButton;  Ln1: TButton;  But4: TButton;  But5: TButton;  But6: TButton;  But7: TButton;  But8: TButton;  But9: TButton;  Edit1: TEdit;  MenuItem1: TMenuItem;  MenuItem2: TMenuItem;  MenuItem3: TMenuItem;  MenuItem4: TMenuItem;  MenuItem5: TMenuItem;  MenuItem6: TMenuItem;  TMainMenu1: TMainMenu;  procedure But0Click(Sender: TObject);  procedure But1Click(Sender: TObject);  procedure But20Click(Sender: TObject);  procedure But2Click(Sender: TObject);  procedure But3Click(Sender: TObject);  procedure But4Click(Sender: TObject);  procedure But5Click(Sender: TObject);  procedure But6Click(Sender: TObject);  procedure But7Click(Sender: TObject);  procedure But8Click(Sender: TObject);  procedure But9Click(Sender: TObject);  procedure CClick(Sender: TObject);  procedure CEClick(Sender: TObject);  procedure Cos1Click(Sender: TObject);  procedure DelClick(Sender: TObject);  procedure DeletClick(Sender: TObject);  procedure KorenClick(Sender: TObject);  procedure Ln1Click(Sender: TObject);  procedure Log1Click(Sender: TObject);  procedure MCClick(Sender: TObject);  procedure MenuItem4Click(Sender: TObject);  procedure MinClick(Sender: TObject);  procedure MMClick(Sender: TObject);  procedure MPClick(Sender: TObject);  procedure MRClick(Sender: TObject);  procedure MSClick(Sender: TObject);  procedure OneDelXClick(Sender: TObject);  procedure PavnoClick(Sender: TObject);  procedure PlusClick(Sender: TObject);  procedure PlusMinusClick(Sender: TObject);  procedure Sin1Click(Sender: TObject);  procedure ExpanentClick(Sender: TObject);  procedure StepenClick(Sender: TObject);  procedure UmnozhClick(Sender: TObject);  private  public  end;  var  Form2: TForm2;  x, y,otv, mem :real;  ard, ard2, bf :integer;  implementation  {$R \*.lfm}  { TForm2 }  procedure TForm2.MenuItem4Click(Sender: TObject);  begin  wAboutWin:=TwAboutWin.Create(Application);  wAboutWin.Show;  end;  procedure TForm2.But20Click(Sender: TObject);  var i,zi:integer;  begin  if bf=1 then  Edit1.Text:=('0,') else begin  for i:=1 to length(Edit1.Text) do  if Edit1.Text[i]=(',') then zi:=1;  if zi<>1 then  Edit1.Text:=Edit1.Text+(',');end;bf:=0;  end;  procedure TForm2.But0Click(Sender: TObject);  begin  if bf=1 then  Edit1.Text:=IntToStr(0) else  if Edit1.Text=IntToStr(0) then Edit1.Text:=IntToStr(0) else  Edit1.Text:=Edit1.Text+IntToStr(0);bf:=0;  end;  procedure TForm2.But1Click(Sender: TObject);  begin  if bf=1 then  Edit1.Text:=IntToStr(1) else  if Edit1.Text=IntToStr(0) then Edit1.Text:=IntToStr(1) else  Edit1.Text:=Edit1.Text+IntToStr(1);bf:=0;  end;  procedure TForm2.But2Click(Sender: TObject);  begin  if bf=1 then  Edit1.Text:=IntToStr(2) else  if Edit1.Text=IntToStr(0) then Edit1.Text:=IntToStr(2) else  Edit1.Text:=Edit1.Text+IntToStr(2);bf:=0;  end;  procedure TForm2.But3Click(Sender: TObject);  begin  if bf=1 then  Edit1.Text:=IntToStr(3) else  if Edit1.Text=IntToStr(0) then Edit1.Text:=IntToStr(3) else  Edit1.Text:=Edit1.Text+IntToStr(3);bf:=0;  end;  procedure TForm2.But4Click(Sender: TObject);  begin  if bf=1 then  Edit1.Text:=IntToStr(4) else  if Edit1.Text=IntToStr(0) then Edit1.Text:=IntToStr(4) else  Edit1.Text:=Edit1.Text+IntToStr(4);bf:=0;  end;  procedure TForm2.But5Click(Sender: TObject);  begin  if bf=1 then  Edit1.Text:=IntToStr(5) else  if Edit1.Text=IntToStr(0) then Edit1.Text:=IntToStr(5)  else Edit1.Text:=Edit1.Text+IntToStr(5);bf:=0;  end;  procedure TForm2.But6Click(Sender: TObject);  begin  if bf=1 then  Edit1.Text:=IntToStr(6) else  if Edit1.Text=IntToStr(0) then Edit1.Text:=IntToStr(6) else  Edit1.Text:=Edit1.Text+IntToStr(6);bf:=0;  end;  procedure TForm2.But7Click(Sender: TObject);  begin  if bf=1 then  Edit1.Text:=IntToStr(7) else  if Edit1.Text=IntToStr(0) then Edit1.Text:=IntToStr(7) else  Edit1.Text:=Edit1.Text+IntToStr(7);bf:=0;  end;  procedure TForm2.But8Click(Sender: TObject);  begin  if bf=1 then  Edit1.Text:=IntToStr(8) else  if Edit1.Text=IntToStr(0) then Edit1.Text:=IntToStr(8) else  Edit1.Text:=Edit1.Text+IntToStr(8);bf:=0;  end;  procedure TForm2.But9Click(Sender: TObject);  begin  if bf=1 then  Edit1.Text:=IntToStr(9) else  if Edit1.Text=IntToStr(0) then Edit1.Text:=IntToStr(9) else  Edit1.Text:=Edit1.Text+IntToStr(9);bf:=0;  end;  procedure TForm2.MCClick(Sender: TObject);  begin  begin  mem:=0;MC.Visible:=False;  end;  end;  procedure TForm2.MRClick(Sender: TObject);  begin  begin  Edit1.Text:=FloatToStr(mem);bf:=1;  end;  end;  procedure TForm2.MPClick(Sender: TObject);  begin  mem:=mem+StrToFloat(Edit1.Text);  if mem<>0 then MC.Visible:=True else  MC.Visible:=False;  end;  procedure TForm2.MMClick(Sender: TObject);  begin  mem:=mem-StrToFloat(Edit1.Text);  if mem<>0 then MC.Visible:=True else  MC.Visible:=False;  end;  procedure TForm2.MSClick(Sender: TObject);  begin  if StrToFloat(Edit1.Text)<>0 then begin  mem:=StrToFloat(Edit1.Text);MC.Visible:=True;end else begin  mem:=0;MC.Visible:=False;end;bf:=1;  end;  procedure TForm2.DeletClick(Sender: TObject);  var  str :String;  begin  str:=Edit1.Text;  if str <> '' then  Delete(str, Length(str),1);  Edit1.Text := str;  end;  procedure TForm2.CClick(Sender: TObject);  begin  x:=0;y:=0;ard:=0;ard2:=0;bf:=0;Edit1.Text:='0';  end;  procedure TForm2.CEClick(Sender: TObject);  begin  Edit1.Text:='0';  end;  procedure TForm2.PlusMinusClick(Sender: TObject);  var ds:string;  begin  ds:=Edit1.Text;  if ds<>'0' then begin  if Edit1.Text[1]='-' then begin  delete(ds,1,1);Edit1.Text:=ds;end else begin  Edit1.Text:='-'+ds;bf:=1;end;  end;  end;  procedure TForm2.PavnoClick(Sender: TObject);  begin  otv:=StrToFloat(Edit1.Text);  if ard=0 then begin  y:=StrToFloat(Edit1.Text);  if ard2=1 then  otv:=y+x;  if ard2=2 then  otv:=y-x;  if ard2=3 then  otv:=y\*x;  if ard2=4 then  otv:=y/x;end  else  begin  x:=StrToFloat(Edit1.Text);  if ard=1 then  otv:=y+x;  if ard=2 then  otv:=y-x;  if ard=3 then  otv:=y\*x;  if ard=4 then  otv:=y/x;  ard2:=ard;ard:=0;  end;  Edit1.Text:=FloatToStr(otv);bf:=1;  end;  procedure TForm2.PlusClick(Sender: TObject);  begin  if ard>0 then begin  x:=StrToFloat(Edit1.Text);  if ard=1 then  otv:=y+x;  if ard=2 then  otv:=y-x;  if ard=3 then  otv:=y\*x;  if ard=4 then  otv:=y/x;  ard:=1;y:=otv;Edit1.Text:=FloatToStr(otv);bf:=1;end  else  y:=StrToFloat(Edit1.Text);  ard:=1;bf:=1;  end;  procedure TForm2.MinClick(Sender: TObject);  begin  if ard>0 then begin  x:=StrToFloat(Edit1.Text);  if ard=1 then  otv:=y+x;  if ard=2 then  otv:=y-x;  if ard=3 then  otv:=y\*x;  if ard=4 then  otv:=y/x;  ard:=2;y:=otv;Edit1.Text:=FloatToStr(otv);bf:=1;end  else  y:=StrToFloat(Edit1.Text);  ard:=2;bf:=1;  end;  procedure TForm2.DelClick(Sender: TObject);  begin  if ard>0 then begin  x:=StrToFloat(Edit1.Text);  if ard=1 then  otv:=y+x;  if ard=2 then  otv:=y-x;  if ard=3 then  otv:=y\*x;  if ard=4 then  otv:=y/x;  ard:=4;y:=otv;Edit1.Text:=FloatToStr(otv);bf:=1;end else  y:=StrToFloat(Edit1.Text);  ard:=4;bf:=1;  end;  procedure TForm2.UmnozhClick(Sender: TObject);  begin  if ard>0 then begin  x:=StrToFloat(Edit1.Text);  if ard=1 then  otv:=y+x;  if ard=2 then  otv:=y-x;  if ard=3 then  otv:=y\*x;  if ard=4 then  otv:=y/x;  ard:=3;y:=otv;Edit1.Text:=FloatToStr(otv);bf:=1;end else  y:=StrToFloat(Edit1.Text);  ard:=3;bf:=1;  end;  procedure TForm2.KorenClick(Sender: TObject);  begin  if StrToFloat(Edit1.Text)<0 then  Edit1.Text:='Недопустимый ввод' else  Edit1.Text:=FloatToStr(sqrt(StrToFloat(Edit1.Text)));bf:=1;  end;  procedure TForm2.OneDelXClick(Sender: TObject);  begin  if StrToFloat(Edit1.Text)=0 then  Edit1.Text:='Деление на ноль невозможно' else  Edit1.Text:=FloatToStr(1/StrToFloat(Edit1.Text));bf:=1;  end;  procedure TForm2.StepenClick(Sender: TObject);  var a:real;  begin  a:= StrToFloat(Edit1.Text);  a:=sqr(a);  Edit1.Text:=FloatToStr(a);  end;  procedure TForm2.Log1Click(Sender: TObject);  var a, b:real;  begin  a:=strtofloat(edit1.Text);  b:=ln(a)/ln(10);  edit1.Text:=floattostr(b)  end;  procedure TForm2.Ln1Click(Sender: TObject);  var a, b:real;  begin  a:=strtofloat(edit1.Text);  b:=ln(a);  edit1.Text:=floattostr(b)  end;  procedure TForm2.Sin1Click(Sender: TObject);  var a, b:real;  begin  a:=strtofloat(edit1.Text);  b:=sin(a);  edit1.Text:=floattostr(b)  end;  procedure TForm2.ExpanentClick(Sender: TObject);  var a, b:real;  begin  a:=strtofloat(edit1.Text);  b:=exp(a);  edit1.Text:=floattostr(b)  end;  procedure TForm2.Cos1Click(Sender: TObject);  var a, b:real;  begin  a:= StrToFloat(Edit1.Text);  b:=cos(a);  Edit1.Text:=FloatToStr(b);  a:=0;  end;  end. |

|  |
| --- |
| **AboutWin** |
| unit AboutWin;  {$mode objfpc}{$H+}  interface  uses  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls;  type  { TwAboutWin }  TwAboutWin = class(TForm)  Button1: TButton;  Label1: TLabel;  procedure Button1Click(Sender: TObject);  procedure FormCreate(Sender: TObject);  procedure Label1Click(Sender: TObject);  private  public  end;  var  wAboutWin:TwAboutWin;  implementation  {$R \*.lfm}  procedure TwAboutWin.FormCreate(Sender: TObject);  begin  Label1.Caption:=('Калькулятор'+chr(13)  +'Автор: Кыхтева Мария'+chr(13)  +'И-72');  end;  procedure TwAboutWin.Button1Click(Sender: TObject);  begin  wAboutWin.Close();  end;  end. |

## Листинг программы 2

|  |
| --- |
| **unit MainForm;** |
| interface  uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,  Dialogs, StdCtrls, Buttons, ComCtrls, ExtCtrls, Menus;  type  TMain\_Form = class(TForm)  Persons: TListView;  MainMenu1: TMainMenu;  N1: TMenuItem;  N2: TMenuItem;  N3: TMenuItem;  N4: TMenuItem;  N5: TMenuItem;  N6: TMenuItem;  Label1: TLabel;  BitBtn1: TBitBtn;  procedure FormCreate(Sender: TObject);  procedure N2Click(Sender: TObject);  procedure N3Click(Sender: TObject);  procedure N4Click(Sender: TObject);  procedure N5Click(Sender: TObject);  procedure N6Click(Sender: TObject);  private  public  end;  TRecord = record  Name: ShortString; //Фамилия, Имя, Отчество  DOB: TDate; //Дата рождения  Address: ShortString; // Адрес  Speciality: ShortString;// Специальность  Salary: Boolean; //Стипендия  end;  const  NoYes: array[False..True] of String = ('Нет', 'Да');  var  Main\_Form: TMain\_Form;  implementation  uses AddEditForm, ReportForm;  {$R \*.dfm}  //Создание формы - происходит при запуске приложения  procedure TMain\_Form.FormCreate(Sender: TObject);  var  F: File of TRecord; //Файл с записями о студентах  X: TRecord; //Переменная для хранения записи  begin  AssignFile(F, 'students.dat');  if false = FileExists('students.dat') then  begin  Rewrite(F);//Создаем файл  end; //И открыть для чтения  Reset(F);  //Теперь файл с данными успешно открыт, читаем из него все записи  while not Eof(F) do  begin  Read(F, X); //Читаем запись  //Выводим информацию на экран  with Persons.Items.Add do  begin  Caption := X.Name;  SubItems.Add(DateToStr(X.DOB));  SubItems.Add(X.Address);  SubItems.Add(X.Speciality);  SubItems.Add(NoYes[X.Salary]);  end;  end;  CloseFile(F);  end;  procedure TMain\_Form.N2Click(Sender: TObject);  var  X: TRecord;  F: File of TRecord;  begin  // Добавление записи  X.Name := '';  X.DOB := Now;  X.Address := '';  X.Speciality := '';  X.Salary := False;  Change\_Form.setAll(x);  if Change\_Form.ShowModal = mrOk then  begin  //Если пользователь нажал OK,  //получаем данные с формы  Change\_Form.getAll(X);  AssignFile(F, 'students.dat');  Reset(F); //И пытаемся дописать их в файл  Seek(F, FileSize(F)); //Перемещаемся в конец файла  Write(F, X); //Записываем данные  CloseFile(F);  with Persons.Items.Add do //Затем добавляем запись на экран  begin  Caption := X.Name;  SubItems.Add(DateToStr(X.DOB));  SubItems.Add(X.Address);  SubItems.Add(X.Speciality);  SubItems.Add(NoYes[X.Salary]);  end;  end;  end;  procedure TMain\_Form.N3Click(Sender: TObject);  var  Item: TListItem;  X: TRecord;  F: File of TRecord;  begin  // Изменение записи  Item := Persons.Selected; //Проверяем, что элемент выбран  If Item = nil then  begin  MessageDlg('Выберите пожалуйста,запись!', mtWarning, [mbOk], 0);  Exit;  end;  X.Name := Item.Caption;  X.DOB := StrToDate(Item.SubItems[0]);  X.Address := Item.SubItems[1];  X.Speciality := Item.SubItems[2];  X.Salary := NoYes[True] = Item.SubItems[3];  Change\_Form.setAll(X);  if Change\_Form.ShowModal = mrOk then  begin  Change\_Form.getAll(X);  AssignFile(F, 'students.dat');  Reset(F); //Окрываем файл  Seek(F, Item.Index); //Перемещаемся к отредактированной записи  Write(F, X); //Переписываем старое содержимое новым  CloseFile(F);  with Item do //Обновляем запись на экране  begin  Caption := X.Name;  SubItems[0] := DateToStr(X.DOB);  SubItems[1] := X.Address;  SubItems[2] := X.Speciality;  SubItems[3] := NoYes[X.Salary];  end;  end;  end;  procedure TMain\_Form.N4Click(Sender: TObject);  //процедура удаления записи из файла  var  Item: TListItem;  i: Integer;  X: TRecord;  F: File of TRecord;  begin  Item := Persons.Selected; //Должна быть выбрана запись  if Item = nil then  begin  MessageDlg('Выберите запись, пожалуйста!', mtWarning, [mbOk], 0);  Exit;  end;  if MessageDlg('Вы уверены, что хотите удалить эту запись?',  mtConfirmation, [mbYes, mbNo], 0) = mrYes then  begin  AssignFile(F, 'students.dat');  Rewrite(F);  i := 0;  //Проходимся по всем записям  while i < Persons.Items.Count do  begin  Item := Persons.Items[i];  //Если запись не выбрана, то записываем ее в файл  if not Item.Selected then  begin  X.Name := Item.Caption;  X.DOB := StrToDate(Item.SubItems[0]);  X.Address := Item.SubItems[1];  X.Speciality := Item.SubItems[2];  X.Salary := NoYes[True] = Item.SubItems[3];  Write(F, X);  i := i + 1;  end  else  begin  //Удаляем запись с экрана, и в файл ее не записываем  Persons.Items[i].Delete;  end;  end;  CloseFile(F);  end;  end;  procedure TMain\_Form.N5Click(Sender: TObject);  var  Form: TRep\_Form;  begin  Form := TRep\_Form.Create(nil);  Form.ShowModal; //Показываем форму  Form.Free; //И уничтожаем форму  end;  procedure TMain\_Form.N6Click(Sender: TObject);  begin  Close;  end;  end. |