МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА
И ПРОДОВОЛЬСТВИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ОСНОВЫ ПРОГРАММИРОВАНИЯ

(в визуальной среде Delphi)

Учебно-методический комплекс
по дисциплине «Основы программирования»
для студентов специальностей
1-25 01 07 Экономика и управление на предприятии,
1-26 02 02 Менеджмент

В двух частях

Часть 1

Минск
БГАТУ
2010

УДК 004.4(07)
ББК 22.18я7
О-75

Рекомендовано научно-методическим советом факультета предпринимательства и управления БГАТУ.
Протокол № 7 от 28 мая 2009 г.

Составители:
профессор Р. И. Фурунжиев,
старший преподаватель Е. М. Исаченко,
старший преподаватель Т. В. Ероховец

Рецензенты:
доцент кафедры программного обеспечения САПР и АСУ Учреждения образования «Белорусский национальный технический университет», кандидат технических наук, доцент *Н. Н. Гурский*;
доцент кафедры вычислительной техники БГАТУ,
кандидат технических наук, доцент *А. И. Шакирин*

Основы программирования (в визуальной среде
О-75 **Delphi).** В 2 ч. Ч. 1 : учебно-методический комплекс / сост. : Р. И. Фурунжиев, Е. М. Исаченко, Т. В. Ероховец. – Минск: БГАТУ, 2010. – 132 с.
ISBN 978-985-519-266-5.

Первая часть учебно-методического комплекса содержит два модуля, в которых рассматриваются основы алгоритмизации и программирование в среде визуального программирования Delphi на алгоритимическом языке Object Pascal, среда программирования и некоторые компоненты, которые обеспечивают создание простых программных приложений.

Учебно-методический комплекс ориентирован на теоретическое изучение и приобретение навыков разработки различных программных приложений, реализующих элементы задач экономики и управления.

Предназначен для студентов высших учебных заведений экономического профиля.

УДК 004.4(07)
ББК 22.18я7

ISBN 978-985-519-266-5 (ч. 1)
ISBN 978-985-519-265-8

ВВЕДЕНИЕ

Развитие компьютерных информационных технологий, потребность в эффективных средствах разработки программного обеспечения привели к появлению систем программирования, ориентированных на так называемую «быструю разработку», в основе которых лежит технология визуального проектирования и событийного программирования. Среда быстрой разработки Delphi, принятая в настоящей учебной дисциплине для изучения основ программирования в качестве языка программирования, использует алгоритмический язык Object Pascal. В основе идеологии Delphi лежит современная технология визуального проектирования и методология объектно-ориентированного событийного программирования.

Цель дисциплины – формирование необходимых знаний и навыков использования современных базовых компьютерных технологий разработки алгоритмов и создания Windows-приложений в визуальной среде программирования Delphi, обеспечивающих эффективное решение задач экономики и управления на предприятии; развитие логико-алгоритмического, пооперационного и системного мышления студентов.

Задачи дисциплины

Изучение дисциплины способствует формированию у студентов следующих компетенций:

- **академических**, включающих знания и умения по анализу, формализации, обобщению, алгоритмизации и программированию процессов окружающего мира, способность и умение учиться;
- **социально-личностных,** включающих культурно-ценностные ориентации, знание идеологических, нравственных ценностей общества и государства и умение следовать им;
- **профессиональных,** включающих знание основных особенностей информационных процессов, протекающих в обществе, и их влияние на темпы экономического развития страны; умение формулировать проблемы, составлять алгоритмы и решать задачи, разрабатывать планы и обеспечивать их выполнение в избранной сфере профессиональной деятельности.

В результате освоения учебного материала дисциплины

студент должен знать:

- особенности современных технологий программирования;

- методику постановки, алгоритмизации и программирования на объектно-ориентированном алгоритмическом языке Object Pascal в визуальной среде программирования Delphi задач экономики и управления на предприятии;

уметь:

- конструировать интерфейс и формировать сценарий применения Windows-приложений;
- математически формулировать, алгоритмизировать и программировать задачи экономики и управления на предприятии в визуальной среде программирования Delphi.

Изучение дисциплины «Основы программирования» базируется на знании «Математики» и «Информатики» в объеме учебной программы средней школы.

Примерный тематический план по дисциплине для специальности 1-25 01 07 Экономика и управление на предприятии

Номер и название модуля	Общее количество часов на модуль	В том числе		
		Теоретические занятия (лекции)	Лаборные занятия	УСРС, в т. ч. контр. занятия
М-1. Основы программирования линейных, разветвляющихся и циклических алгоритмов	30	10	16	4
М-2. Программирование с использованием массивов и переменных строкового типа	24	8	12	4
М-3. Программирование с использованием записей и файлов. Программирование с использованием подпрограмм и модулей. Программирование с использованием средств для построения и отображения графиков	28	10	14	4
М-R. Анализ и обобщение результатов обучения. Знакомство со структурой тестов	2	-	2	-
Всего	84	28	44	12

Примерный тематический план по дисциплине для специальности 1- 26 02 02 Менеджмент

Номер и название модуля	Общее количество часов на модуль	В том числе		
		Теоретические занятия (лекции)	Лаборные занятия	УСРС, в т. ч. контр. занятия
М-1. Основы программирования линейных, разветвляющихся и циклических алгоритмов	22	8	10	4
М-2. Программирование с использованием массивов и переменных строкового типа	20	8	8	4
М-3. Программирование с использованием записей и файлов. Программирование с использованием подпрограмм и модулей. Программирование с использованием средств для построения и отображения графиков	22	6	10	6
М-R. Анализ и обобщение результатов обучения. Знакомство со структурой тестов	2	2		-
Всего	66	24	28	14

МОДУЛЬ 1

ОСНОВЫ ПРОГРАММИРОВАНИЯ ЛИНЕЙНЫХ, РАЗВЕТВЛЯЮЩИХСЯ И ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

В результате изучения модуля студент должен:

знать понятия и определения объектно-ориентированного языка программирования Object Pascal; встроенные функции и процедуры языка; свойства и характерные методы компонентов Label, Memo, Edit, Button; определения операторов сравнения, булевых операторов, условных операторов if и case; операторов организации циклов for, repeat, while; особенности применения компонентов CheckBox, RadioButton, RadioGroup;

уметь формулировать, алгоритмизировать и программировать задачи, модели которых описываются линейными, разветвляющимися и циклическими алгоритмами в визуальной среде Delphi; применять полученные теоретические и практические знания к решению задач экономики и управления на предприятии, в которых предусматривается принятие решений и организация повторяющихся вычислений.

Теоретические сведения

В основе среды программирования Delphi, разработанной фирмой Borland, лежит алгоритмический язык Pascal. Автором этого языка является швейцарский профессор Никлаус Вирт, опубликовавший его в 1971 году.

Компьютерная ***программа*** – это последовательность элементарных команд, представленных в файле в виде ***машинного кода*****,** выполняемого компьютером.

Создание языков программирования позволило записывать программы как обычный текст на алгоритмическом языке. Этот текст называется ***исходным текстом*** (или *исходным кодом*). ***Программа-компилятор*** преобразует текстовую программу, записанную на алгоритмическом языке, в выполняемую программу в машинном коде: Windows-приложение или далее для краткости ***приложение***.

1. Этапы создания приложений в среде визуального программирования Delphi

1. **Анализ и формулировка задачи.** Формулировка целей программы. Определение набора входных и выходных данных.

2. **Разработка схемы алгоритма.** Определение зависимости между входными и выходными данными. Написание алгоритма решения задачи и, если возможно, проверка на простом примере. Разработка схемы алгоритма.

3. **Разработка пользовательского интерфейса и сценария работы пользователя**. Определение того, что пользователь должен видеть на экране, в какой поледовательности и какие данные он будет вводить, где и в каком формате будет представлять выходные данные.

4. **Написание кода.** Преобразование алгоритма в компьютерную программу на языке высокого уровня, например, на алгоритмическом языке Object Pascal в визуальной среде программирования Delphi.

5. **Тестирование и отладка программы.** Термин «тестирование» означает поиск ошибок в программе, а «отладка» – процесс их устранения.

6. **Составление документации**. В документацию программы входят материалы, описывающие назначение, принципы ее работы и инструкция пользователя.

Алгоритмом называется точное, формальное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату.

По структуре алгоритмы условно разделяются на ***линейные, разветвляющиеся*** и ***циклические.***

На основе алгоритмов разрабатываются ***схемы алгоритмов***. При выполнении схем алгоритмов отдельные функции алгоритмов отображаются условными графическими обозначениями – геометрическими символами (блоками), определенными ГОСТом, внутри которых записываются выполняемые действия. Наименования, обозначения и назначение наиболее употребляемых символов приведены в таблице 1.1.

Таблица 1.1

Наиболее употребляемые в схемах алгоритмов блок-символы

Наименование	Обозначение	Назначение
Пуск, останов	Начало	Начало или конец алгоритма
Ввод, вывод	Ввод x, a	Ввод данных и вывод результатов
Процесс		Выполнение операции или группы операций
Решение	x > y	Выбор направления в зависимости от некоторых условий
Модификация	i:=1, 100	Начало цикла
Предопределенный процесс	Инициализация компонентов	Использование ранее созданных алгоритмов
Линия потока		Связь между блоками

Другой, не менее эффективный способ описания алгоритмов – ***словестный***. Например, следующий условный текст представляет описание алгоритма небольшой программы умножения двух чисел:

1. **принять** значения, введенные пользователем в поля *k1* и *k2*;
2. **вычислить** произведение этих значений *s*;
3. **передать** результат s в поле *k3*.

Полужирным шрифтом выделены названия команд.

2. Интегрированная среда визуального программирования Delphi

Визуальная среда программирования Delphi

Запуск системы Delphi осуществляется двойным щелчком кнопкой мыши на ее пиктограмме на рабочем столе Windows, либо выполнением команды ***Пуск | Программы | Borland Delphi 7 | Delphi 7***. Система инициирует начало создания нового ***проекта***. По умолчанию новый проект является ***программным приложением***, т.е. в результате создается выполняемый файл с расширением «**.exe**». Создание проектов этого типа рассматривается в настоящей работе.

В среде Delphi предусмотрено создание различных типов проектов. Программист может легко изменить тип создаваемого проекта, принятый по умолчанию. Другие типы проектов – это приложения *Web*, библиотеки *DLL* (*Dynamic Link Library* – динамически подключаемая библиотека), элементы управления *ActiveX*, активные формы, консольные приложения и др. Здесь и далее рассматривается создание программных приложений, поэтому изменять тип проекта, создаваемый системой по умолчанию, не требуется.

Интегрированная среда разработки (*Integrated Development Environment – IDE*) Delphi – это среда, в которой есть все необходимое для проектирования, запуска, отладки и тестирования приложений: конструктор форм, редактор кодов, отладчик, инструментальные панели, редактор изображений, инструментарий баз данных, Web-приложений и др. Предоставляется возможность расширять возможности этой среды.

На рисунке 1.1 показан вариант раскладки экрана визуальной среды разработки Delphi, включающий пять основных окон:

1. главное окно – **Delphi**;
2. окно формы – **Form1**;
3. окно редактора кода – **Unitl.pas**;
4. окно редактора свойств и реакций на события объектов – **Object Inspector** (Инспектор объектов);
5. окно просмотра списка объектов – **Object TreeView** (Просмотр дерева объектов).

Главное окно включает все основные элементы, предназначенные для управления процессом создания программы: основное меню команд, панели инструментов и палитру компонентов.

Основное меню содержит все необходимые средства для управления проектом (вторая строка сверху на рис. 1.1). При щелчке кнопкой мыши на любом из его пунктов на экране появляется раскрывающееся меню со списком доступных пунктов. Для выбора пункта основного меню следует поместить указатель мыши на нужном пункте и один раз щелкнуть кнопкой мыши, при этом появляется ассоциированное с этим пунктом раскрывающееся меню. Для выбора пункта раскрывшегося меню нужно сделать то же самое – поместить на нем указатель и щелкнуть кнопкой мыши. В дальнейшем будем придерживаться следующего соглашения: если нужно активизировать какой-либо пункт раскрывающегося меню, например ***Browser***, принадлежащий меню ***View***, то в тексте это будет обозначаться как ***View | Browser.***

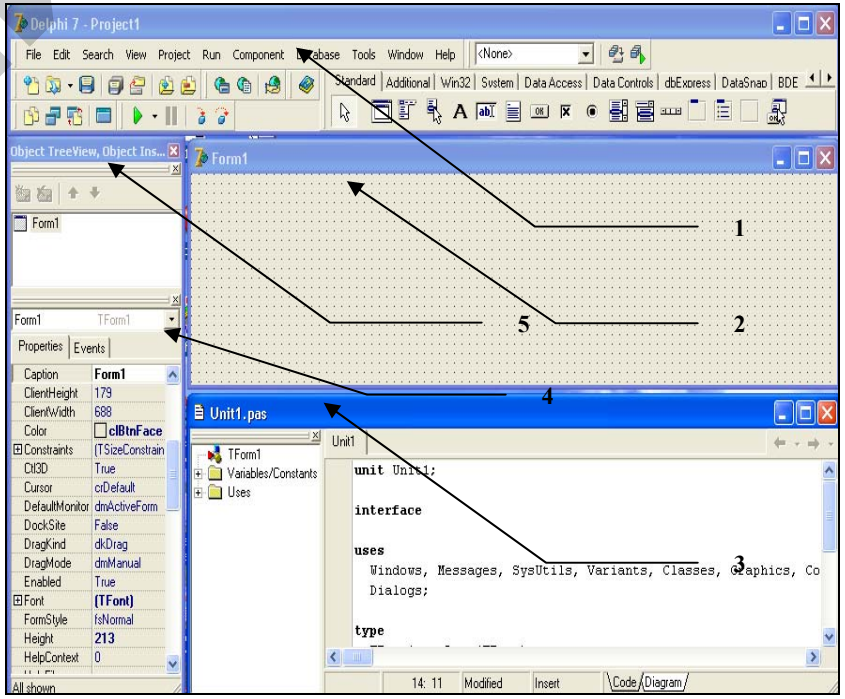

Рис. 1.1. Раскладка экрана визуальной среды разработки Delphi

Ниже строки основного меню в левой части главного окна размещаются ***панели инструментов***. Пиктограммы этой панели облегчают доступ к наиболее часто применяемым командам основного меню. Панели инструментов включают стандартную панель (***Standard***), панель просмотра (***View***), панель отладки (***Debug***), пользовательскую панель (***Custom***). Чтобы узнать назначение любой пиктограммы, нужно установить указатель мыши (***зависнуть***) над ней. Через секунду рядом с указателем появится подсказка, напоминающая программисту о назначении пиктограммы.

Например, при однократном щелчке кнопкой мыши на третьей слева пиктограмме ***Стандартной панели инструментов*** (***Standard***)  компьютер выполняет команду ***Save*** (Сохранить). Эту команду, очевидно, можно выполнить также командой ***File | Save*** (*Файл | Сохранить*).

Окно формы представляет собой заготовку начального проекта Windows-окна программы. В это окно в процессе конструирования приложения помещаются необходимые компоненты. Причем при выполнении приложения помещенные и отредактированные визуальные компоненты будут иметь тот же вид, что и на этапе проектирования.

Окно редактора кода программы предназначено для написания, просмотра и редактирования текста программы. При первоначальной загрузке в окне текста программы находится текст, содержащий минимальный набор операторов для нормального функционирования пустой формы в качестве Windows-окна. Обычно при запуске системы окно редактора кода полностью закрывается окном формы. Для быстрого перехода от одного окна к другому можно воспользоваться клавишами *<F12>* или *Toggle Form/unit*.

Окно инспектора объектов (Object Inspector) предназначено для изменения свойств выбранных компонентов и реакции на то или иное событие. Страница ***Properties*** (Свойства) этого окна предназначена для изменения свойств выбранных компонентов, страница ***Events*** (События) – для определения реакции программы на то или иное событие, связанное с этим компонентом. В Delphi каждому событию присвоено имя. Например, щелчок кнопкой мыши – это событие ***OnClick***, двойной щелчек мышью – событие ***OnDblClick***.

Если окна **Object Inspector** на экране не видно, то для его вывода нужно выбрать команду ***View | Object Inspector*** или нажать клавишу *<F11>*.

Палитра компонентов (третья строка главного окна с правой стороны) представляет собой панель с вкладками, обеспечивающими быстрый доступ к библиотеке визуальных компонентов (VCL – *Visual Component Library),* организованной в виде иерархической структуры. Компоненты служат визуальными элементами при создании графического пользовательского интерфейса приложения Delphi. Во время выполнения приложения компоненты *VCL* появляются на экране как ***элементы управления*** – поля ввода, кнопки, флажки, списки и т.д. Элементы управления составляют подмножество компонентов *VCL*: каждый элемент управления является компонентом, но не каждый компонент является элементом управления.

Через палитру компонентов осуществляется доступ к набору стандартных сервисных программ среды Delphi, с помощью которых программист конструирует интерфейс разрабатываемого приложения в окне формы. Каждый компонент имеет определенный набор ***свойств*** (параметров), которые программист может задавать. Например, заголовок окна, надпись на кнопке, цвет, размер и тип шрифта и др.

При помещении некоторого компонента в окно формы текст программы автоматически дополняется описанием необходимых для его работы библиотек стандартных программ (раздел **uses**) и типов переменных (раздел **type**).

Программа в среде Delphi составляется как текстовое описание на языке программирования предварительно составленных алгоритмов или их фрагментов, которые будут выполняться при возникновении того или иного события (например, щелчка мышью по кнопке – событие ***OnClick***). Для каждого обрабатываемого в форме события с помощью страницы ***Events*** инспектора объектов в тексте программы организуется процедура (procedure), между ключевыми словами **begin** и **end** в которой программист записывает на алгоритмическом языке текст программы, реализующей требуемый алгоритм.

По умолчанию слева внутри окна редактора кода пристыковано окно ***Исследователя кода***. С помощью исследователя кода про-

граммист может легко выбирать и просматривать файлы модулей. На древовидной диаграмме исследователя кода показаны все типы, классы, свойства, методы, глобальные переменные и глобальные процедуры, определенные в модуле кода, загруженного в данный момент в редактор кода. В окне исследователя кода перечислены также все модули, используемые текущим модулем редактора кода.

Справочная система. Система Delphi содержит гипертекстовую справочную систему, с помощью которой программист может легко и быстро получить необходимую информацию о среде разработки Delphi и объектно-ориентированном языке Object Pascal. Для активизации справочной системы выберите команду ***Help | Delphi Help*** (*Справка | Справка Delphi*) или щелкните на пиктограмме *Help Contents* (Содержание справочной системы) на пользовательской панели инструментов. При этом появляется окно справочной системы Delphi. Кроме того, в справочной системе Delphi есть информация об инструментах разработки Delphi (***Help | Delphi Tools***) и о программировании в среде Windows (***Help | Windows SDK***).

Обработка событий

Обо всех происходящих в системе событиях (например, создание формы, нажатие кнопки мыши или клавиатуры и т. д.) ядро операционной системы Windows информирует компоненты путем посылки соответствующих сообщений. Среда Delphi позволяет принимать и обрабатывать большинство таких сообщений. Каждый компонент содержит имена процедур обработчиков сообщений, которые отображаются на странице ***Events*** инспектора объектов.

Для создания обработчика события необходимо раскрыть список компонентов в верхней части окна инспектора объектов и выбрать необходимый компонент. Затем на странице ***Events*** нажатием левой клавиши мыши выбрать название обработчика и дважды щелкнуть по его правой (белой) части. В ответ Delphi активизирует окно редактора кода программы и покажет заготовку процедуры обработки выбранного события. Для каждого обрабатываемого события в тексте модуля организуется процедура (**procedure**), которая первоначально отображается на экране ее названием и ключевыми словами **begin** и **end**. Между этими словами прогрммист на языке Object Pascal записывает требуемый алгоритм обработки события.

Каждый компонент имеет свой набор обработчиков событий, однако некоторые из них присущи большинству компонентов. Наиболее часто применяемые события представлены в таблице 1.2.

Таблица 1.2

Описание событий

Событие	Описание события
OnActivate	Форма получает это событие при активации
OnCreate	Возникает при создании формы (компонент Form). В обработчике данного события следует задавать действия, которые должны происходить в момент создания формы, например установка начальных значений
OnKey Press	Возникает при нажатии кнопки на клавиатуре. Параметр Key имеет тип Char и содержит ASCII-код нажатой клавишей (клавиша Enter клавиатуры имеет код #13, клавиша Esc - #27 и т.д.). Обычно это событие используется в том случае, когда необходима реакция на нажатие одной из клавиш
OnClick	Возникает при нажатии кнопки мыши в области компонента
OnKeyDown	Возникает при нажатии клавиши на клавиатуре. Обработчик этого события получает информацию о нажатой клавише и состоянии клавиш Shift, Alt и Ctrl, а также о нажатой кнопке мыши. Информация о клавише передается параметром Key, который имеет тип Word
OnKeyUp	Является парным событием для OnKeyDown и возникает при отпускании ранее нажатой клавиши
OnDblClick	Возникает при двойном нажатии кнопки мыши в области компонента

Структура программ, создаваемых в Delphi

Программа в Delphi состоит из головного файла проекта (файл с расширением **.dpr**), одного или нескольких файлов модулей (с расширением **.pas**), файлов описания формы (с расширением **.dfm**) и других файлов.

В головном файле проекта находится информация о формах и модулях, составляющих данный проект. Файл проекта автоматически создается и редактируется средой Delphi.

В файлах модулей (**Unit**) размещаются программы в текстовой форме.

В файле описания форм обычно системой размещаются начальные значения свойств компонентов.

Программный файл модуля в обобщенной форме имеет следующую структуру:

```
Unit Unit1;
  Interface
    {Раздел объявлений}
  implementation
    {Раздел реализации}
begin
    {Раздел инициализации}
end.
```

В ***разделе объявлений*** описываются константы, типы переменных, переменные, заголовки процедур и функции, которые могут быть использованы другими модулями через операторы подключения библиотек (**Uses**).

В ***разделе реализации*** располагаются тела процедур и функций, описанных в разделе объявлений, а также типы переменных, процедуры и функции, которые будут функционировать только в пределах данного модуля.

Раздел инициализации используется редко и его можно пропустить.

При ***компиляции*** программы компилятор Delphi создает файл с расширением **.dcu**, содержащий в себе результат перевода в машинные коды содержимого файлов с расширением **.pas** и **.dfm**. ***Компоновщик*** преобразует файлы с расширением **.dcu** в единый выполняемый файл с расширением **.exe**. В файлах, имеющих расширение **.~df, .~dp, .~pa,** хранятся резервные копии файлов с образом формы, проекта и исходного текста соответственно.

Сохранение проекта

В процессе создания приложения Delphi формируются, как указывалось выше, в несколько файлов, которые в совокупности образуют ***проект.*** Каждый проект целесообразно хранить в отдельной папке. Если она не создана предварительно, то ее можно создать в процессе сохранения в окне ***Сохранить как*** нажатием кнопки для создания папки. Для сохранения проекта первый раз следует щелкнуть «мышью» на опции ***Save Project As…*** (*Сохранить проект как...*) пункта ***File*** главного меню. Сначала Delphi откроет панель диалога ***Save Unit As…*** (*Сохранить модуль как...*) для сохранения модуля проекта. В этой панели в соответствующей папке сохраняется модуль под определенным именем. Delphi по умолчанию присвоит этому файлу тип ***Delphi Unit*** с расширением ***.pas**. Затем откроется панель диалога ***Save Project As…*** Присвоив имя проекту, сохраним его в этой же папке. Здесь Delphi присвоит файлу тип *Delphi Project* и расширение ***.dpr**. Следует убедиться в правильности названий в окнах проекта и модуля проекта.

По умолчанию система присваивает модулям и проектам имена соответственно **Unit1, Project1, Unit2, Project2** и т.д. Разработчик при сохранении вводит свои названия с учетом назначения файла в программе, оставляя без изменения расширения имен файлов.

При повторном сохранении проекта использовать команду ***File | Save All,*** которая позволяет автоматически сохранять все файлы проекта без изменения имен и мест расположения.

Вопросы для самоконтроля

1. Какой алгоритмический язык был положен в основу среды визуального программирования Delphi?
2. Перечислите способы запуска на компиляцию и выполнение в Delphi?
3. К каким функциям главного меню интегрированной среды программирования обеспечивает доступ *Панель инструментов Standard*?
4. Какие страницы содержит окно *Инспектора объектов*?
5. Каково предназначение окна *Инспектора объектов*?

3. Компоненты среды визуального программирования Delphi

Компоненты Form, Label, Edit, Memo, Button и их применение

Delphi имеет в своем арсенале сотни эффективных компонент. Рассмотрим простейшие компоненты ***Form, Label, Edit, Memo, Button***, пиктограммы которых находятся во вкладке ***Standard*** (Стандартная) Палитры компонентов. Они обеспечивают создание довольно сложных Windows-приложений. Более того, с применением только этих компонент можно создать минимальную достаточно универсальную среду для программирования простейших задач.

Компонент Form

Из перечисленных компонентов только *Form* (Форма) является объектом, пиктограмма которого отсутствует на палитре компонентов. При запуске системы автоматически создается форма. Добавить новую форму в проект можно командой ***File | New | Form*** (*Файл* | *Новая* | *Форма*) или щелкнуть кнопкой мыши на пиктограмме *New Form* на панели просмотра. Удалить форму из проекта можно командой ***Projects | Remove from Project...*** (*Проект* | *Удалить из проекта...*) или щелкнуть кнопкой мыши на пиктограмме *Remove File from Project* (*Удалить файл из проекта*). При этом появляется окно *Remove From Project* (*Удалить из проекта*). В нем нужно выделить файлы модулей и форм, которые требуется удалить.

Создание интерфейса приложения осуществляется путем размещения на форме нужных компонентов. Затем, изменяя свойства компонентов, формируется их внешний вид. После этого разрабатываются средства взаимодействия компонентов с системой и потенциальным пользователем. Взаимодействие компонентов формы с пользователем означает некоторую реакцию компонентов на действия пользователя.

Компонент Label

Компонент ***Label*** (Метка) используется для вывода на форму текста, который пользователь не может изменить непосредственно (естественно, в программе может быть предусмотрено изменение надписи «изнутри» программы). Для ознакомления с методикой применения компонента *Label* рассмотрим следующее задание.

Задание 1. Создать Windows-приложение, которое после запуска обеспечивает вывод непосредственно в окне формы надписи «Изучаем язык визуального программирования Delphi!» (здесь и далее в тексте работы вводимый текст записывается в кавычках). Выполните следующие операции.

1. Выбрав команду ***File | New | Application*** (*Файл* | *Новое* | *Приложение*), создайте новый проект.

2. Поместите компонент ***Label*** на форму. Для этого дважды щелкните кнопкой мыши на пиктограмме ***Label*** на палитре компонентов. Чтобы удалить компонент с формы, нужно выделить его (щелкнуть на нем кнопкой мыши, при этом компонент выделяется черными квадратиками) и нажать клавишу *Delete*. Чтобы отменить выделение, нужно щелкнуть кнопкой мыши в любом месте за пределами надписи.

3. Переместите компонент в нужное место на форме методом перетаскивания и установите необходимые геометрические размеры компонента.

4. Можно изменить свойство компонента ***Name*** (*Имя*) на другое. Для этого в инспекторе объектов следует щелкнуть на свойстве *Name* и ввести новое имя. Имя компонентов в приложениях, применяемых на практике, как правило, изменяется: присваивается имя, в котором отражается тип компонента и его назначение в программе. В лабораторных работах имя компонента можно не изменять. В этом случае сохранится имя, присвоенное системой.

5. Измените заголовок. Для этого выберите в инспекторе объектов свойство ***Caption*** (Заголовок). Введите новый заголовок «Изучаем язык визуального программирования Delphi!» и нажмите клавишу *Enter*.

6. Измените цвет фона надписи. Для этого выберите свойство *Color* (Цвет фона), щелкните на стрелке, выберите в раскрывшемся списке зеленый цвет и щелкните на нем.

7. Измените ***шрифт*** и цвет текста надписи. Для этого выберите свойство ***Font*** (Шрифт) и щелкните на трех точках. В окне ***Font*** измените шрифт на *Arial*, а размер – на 18. В раскрывающемся списке выберите красный цвет и щелкните на кнопке *OK*.

8. Теперь выделите форму. Это можно сделать двумя способами: щелкнуть в любом свободном месте формы или выбрать форму

в раскрывающемся списке в верхней части инспектора объектов. Если форма видна, то первый способ, конечно, удобнее, но, если в проекте есть много форм, то более удобен второй способ.

9. Измените свойства формы: свойство ***Caption*** на – «Лаб. работа №1а. Принципы разработки программ в среде визуального программирования Delphi». Этот текст отобразится в верхней строке формы.

10. Запустите разработанную программу на выполнение. Это можно сделать одним из трех способов:

- щелкнув на пиктограмме ***Run*** (Выполнить) на панели отладки;
- выбрав в главном меню команду ***Run*** | ***Run***;
- нажав клавишу <*F9*>.

11. Можно убедиться, что пользователь не может изменить эту надпись. Щелкнув на крестике в верхнем правом углу формы, завершите приложение. Это же можно сделать и в среде Delphi. Для этого запустите приложение еще раз, активизируйте любое окно Delphi (однократно щелкнув на нем кнопкой мыши) и выберите ***Run*** | ***Program Reset*** (*Выполнить* | *Переустановка программы*).

Приложение следует внимательно сохранить в соответствующей папке, как указано ниже, например, с именем «Лаб_работа_1а_Иванов_И.».

Компонент Edit

В компоненте ***Edit*** (Поле ввода) хранится текст, который можно помещать в данный компонент как во время разработки, так и во время выполнения приложения. В сущности, это однострочный редактор. Текст, видимый в поле ввода, находится в свойстве компонента ***Text***. С помощью свойства ***Font*** можно устанавливать шрифт текста. Если свойство ***ReadOnly*** (Только чтение) установить в *False* (ложь), то во время выполнения программы пользователь может изменять текст поля ввода.

Задание 2. Создать Windows-приложение, которое после запуска обеспечивает вывод в поле ввода надпись «2009». Выполните следующие операции.

1. Выбрав команду ***File*** | ***New*** | ***Application*** (*Файл* | *Новое* | *Приложение*), создайте новый проект.

2. Разместите компонент ***Edit*** на форме.

3. Переместите поле ввода в другое место и установите необходимые размеры компонента.

4. Выберите в инспекторе объектов свойство ***Text*** и введите его новое значение: «2009». Нажав клавишу *Enter*, зафиксируйте введенный текст. Обратите внимание: во время ввода изменяется текст в поле ввода на форме.

5. Выделите форму. Измените свойство ***Caption*** – на «Пример применения компонента Edit».

6. Нажав клавишу <*F9*>, запустите разработанную программу на выполнение. Введите в Поле ввода другой текст, в том числе буквенный.

7. Для завершения программы щелкните на крестике в правом верхнем углу формы.

8. Проверьте значение свойства ***ReadOnly*** компонента.

9. Нажав клавишу <*F9*>, запустите программу на выполнение еще раз. Попытайтесь изменить содержимое поля ввода: как видите, теперь изменить его содержимое во время выполнения нельзя.

10. Внимательно сохраните проект в соответствующей папке.

Компонент Memo

Компонент ***Memo*** (Область просмотра) предназначен для вывода на экран многих строк текста. По существу это многострочный редактор. Свойство *Lines* содержит отдельные строки текста области просмотра. Оно доступно как во время разработки, так и во время выполнения.

Задание 3. Создать Windows-приложение, которое после запуска обеспечивает вывод в Области просмотра, созданной посредством компонента *Memo*, предложение из известной песни. Выполните следующие операции.

1. Выполнив команду ***File*** | ***New*** | ***Application*** (*Файл* | *Новое* | *Приложение*), создайте новый проект.

2. Разместите компонент ***Memo*** на форме.

3. Измените размер Области просмотра.

4. Выберите свойство ***Lines*** и щелкните на трех точках. При этом появляется окно редактора строк *String List Editor*. Введите текст, например предложение из известной песни. Закончив ввод текста, щелкните на кнопке *OK*.

5. Выделите форму. Измените свойство ***Caption*** – на «Пример применения компонента Memo».

6. Нажав клавишу *<F9>*, запустите программу на выполнение. На экране должно появиться изображение. Введите в области просмотра какой-либо текст, пока он не выйдет за правую границу.

7. Щелкнув на крестике в правом верхнем углу формы, завершите работу программы.

8. В инспекторе объектов измените значение свойства ***WordWarp*** области просмотра *memSample* на *False* (ложь), а значение свойства ***ScrollBars*** – на *ssBoth* (это свойство определяет наличие или отсутствие полос прокрутки).

9. Запустите программу еще раз. Вводите второй куплет песни в Области просмотра до тех пор, пока строка не выйдет за правую границу. Попробуйте также добавлять новые строки, пока они не выйдут за нижнюю границу.

10. Завершите работу программы и сохраните проект в отдельной папке.

Компонент Button

Обычно с помощью компонента ***Button*** (Кнопка) пользователь инициирует выполнение какого-либо фрагмента кода или целой программы. Другими словами, если щелкнуть на элементе управления ***Button***, то программа выполняет определенное действие. При этом кнопка принимает такой вид, будто она вдавлена.

Задание 4. Создать приложение, которое после запуска и нажатия Кнопки обеспечивает вывод надписи «Мы познакомились с некоторыми компонентами Delphi». Выполните следующие операции.

1. Выбрав команду ***File | New | Application*** (*Файл | Новое | Приложение*), создайте новый проект.

2. В инспекторе объектов измените значение свойства ***Caption*** – на «Пример применения компонента Button».

3. Поместите на левой части формы компонент ***Label***. Удалите содержание свойства ***Caption***.

4. Поместите компонент ***Button*** на форму.

5. Измените значение свойства ***Caption*** кнопки на *«&Щелкните здесь»*. Обратите внимание: в заголовке кнопки буква, перед которой стоит символ *&*, оказалась подчеркнутой. Это означает, что теперь кнопке присвоена комбинация клавиш быстрого вызова *<Alt+Щ>*.

6. Измените размер и положение кнопки.

7. Сделайте двойной щелчок указателем мыши на Кнопке. На экране отобразится окно Редактора кода с шаблоном-заготовкой для процедуры-обработчика события нажатия кнопки. Между операторными скобками **begin** и **end** впишите текст первой и пока единственной строки программы:

Button1.Caption:= 'Мы познакомились с некоторыми компонентами Delphi';

Процедура-обработчик события нажатия кнопки представляется в виде:

```
procedure TFormButton1Click(Sender: TObject);
begin
Button1.Caption := 'Мы познакомились с некоторыми компонентами
Delphi';
end;
```

8. Нажав клавишу *<F9>*, запустите программу на выполнение.

9. Щелкните на Кнопке: она принимает такой вид, будто «вдавлена». На экране отобразится надпись.

10. Рассмотрите другой вариант выполнения приложения. Активизируйте кнопку, нажав комбинацию клавиш *<Alt+Щ>*. При активизации клавишами быстрого вызова кнопка не принимает вид вдавленной.

11. Завершите работу программы и сохраните проект в отдельной папке.

Вопросы для самоконтроля

1. Что такое алгоритм?
2. Назовите и опишите назначение пяти различных геометрических фигур, используемых в схемах алгоритмов.
3. Какие преимущества имеет словестное описание алгоритмов по сравнению с графическими схемами?
4. Назовите и опишите шесть этапов полного цикла разработки программы.
5. На каком этапе создания программы разрабатывается сценарий ее применения?

4. Основные понятия и элементы алгоритмического языка Object Pascal

Алфавит языка

Объектно-ориентированный алгоритмический язык Object Pascal, лежащий в основе языка Delphi, оперирует следующим набором символов:

- заглавные и строчные латинские буквы;
- арабские цифры (от 0 до 9);
- шестнадцатеричные цифры (от 0 до F);
- специальные символы; +, -, *, /, (,), $, #, &, _ и др.

Синтаксис языка

Основные синтаксические правила записи программ сводятся к следующему.

Все используемые типы, константы, переменные, функции, процедуры должны быть объявлены или описаны до их первого использования.

Прописные и строчные буквы идентичны.

Каждое предложение языка кончается символом точка с запятой («;»).

В строке может размещаться несколько операторов.

Операторные скобки **begin…end** выделяют составной оператор. Все операторы, размещенные между ключевыми словами **begin** и **end,** воспринимаются синтаксически как один оператор. Нельзя извне составного оператора передавать управление внутрь него.

Программа или отдельный модуль завершаются оператором «**end.**» (ключевое слово **end** с символом точка).

В символьных комментариях могут использоваться любые другие знаки.

Вспомогательный текст программы оформляется в форме комментариев. В тексте программы они выделяются фигурными скобками «{» и «}» или двойными символами «(*» и «*)» в любом месте программы. К комментариям также относятся строки, начинающиеся символами «//».

Данные в зависимости от способа их хранения и обработки можно разбить на две группы: ***константы*** **и** ***переменные***.

Константы

Константы – это данные, значения которых не изменяются в процессе выполнения программы. Константы (именованные константы) могут объявляться после ключевого слова **const**. Используются три вида констант: ***числовые*** (*целые* или *дробные*); ***логические*** (или булевские); ***символьные*** **и** ***строковые***.

Целые константы – это целые числа. Знак + можно опускать. Можно использовать также шестнадцатеричные целые значения. При использовании шестнадцатеричной константы перед ней указывается знак доллара $. Например, $27 определяет число 39 (в десятичном исчислении).

Вещественные константы могут быть представлены в двух видах: с фиксированной и плавающей точками. Константы с фиксированной точкой – это числа, содержащие точку, разделяющую целую и дробную части. Константы с плавающей точкой – это числа, представленные с десятичным порядком: *mEp* (без пробелов), где *m* – мантисса (как целые, так и дробные числа с фиксированной точкой); *E* или *e* – признак записи числа с десятичным порядком; *p* – порядок числа (только целые числа).

Пример

Целые константы: –564; 23.

Вещественные константы с фиксированной точкой – 564.012.

Вещественные константы с плавающей точкой – 5.64E-9.

Логические константы могут принимать два значения: *True* (истина) и *False* (ложь).

Символьные константы – это какой-либо один символ, заключенный в апострофы: *'F', '7'* и т.п.

Строковые константы – это ряд символов, заключенных в апострофы: *'Итог', 'a&b'* и т.п. При этом строчный символ (*a*) не то же самое, что заглавный (*A*), так как они имеют различные значения в коде ASCII. В этой связи в Object Pascal есть еще один способ представления символьных констант: использование знака номера (#), за которым следует код символа ASCII. Так, *#70* – это то же самое, что и '*F*', а *#100* – то же самое, что и '*d*'. Соответственно строковые константы могут быть записаны как последовательность кодов символов: *#70#100* эквивалентна строковой константе *'Fd'***.**

Переменные

Переменные – это некоторые именованные объекты, которые в процессе выполнения программы могут принимать различные значения. Переменным присваиваются имена ***(идентификаторы)***, которые по возможности отражают их предназначение. Кроме буквенно-цифровых символов идентификаторы могут содержать знак подчеркивания «_». Пробелы в идентификаторах не допускаются. Запрещается использовать в качестве идентификаторов ключевые слова. Строчные буквы (*a..z)* в идентификаторах тождественны прописным буквам (A..Z). Поэтому, например, идентификаторы SAP и Sap являются тождественными.

Описание переменных

Переменные, применяемые в программе, должны быть предварительно объявлены (описаны) после ключевого слова **var:** должны быть указаны имена переменных и их типы. Стандартные переменные ***целого, дробного, логического и символьного*** типов описываются соответственно следующими ключевыми словами: **integer, real**, **boolean, char**. Переменные типа *integer* занимают 2 байта памяти, переменные типа *real* – 6 байт, переменные типа *extended* – 10 байт, переменные типа *boolean* и *char* – по одному байту.

Кроме перечисленных, применяются переменные целого типа **shortint**, **byte**, **word**, **longint,** которые занимают память ЭВМ соответственно 1, 1, 2, 4 байта. В зависимости от предполагаемых наибольших и наименьших значений данных, применяются те или иные из них. Например, диапазон допускаемых значений для переменных целого типа *byte* – 0..255, *word* – 0..65535. Диапазон значений переменных вещественного типа *real* – $10^{38}..10^{+38}$, типа *extended* – $10^{-4931}..10^{4931}$. Число значащих цифр для переменных типа *real* – 12, а для переменных типа *extended* – 18.

Структура программ

Программа начинается ключевым словом **program,** после которого следует имя программы. Имя программы назначается по тем же правилам, как и имя идентификатора.

Затем может следовать ключевое слово **uses,** за которым записывается список имен используемых программных модулей.

Далее может следовать группа ***описаний программы***, состоящая в общем случае из пяти ***разделов описаний*** и ***раздела операторов.*** Разделы описаний следуют в произвольном порядке и могут повторяться. Раздел операторов должен присутствовать всегда, остальные разделы могут отсутствовать. В конце блока всегда должно стоять ключевое слово **end** с точкой. После последнего выполняемого оператора (перед **end**) точку с запятой можно не ставить. Рассмотрим каждый из разделов.

Раздел описания меток **label**. Любой выполняемый оператор может иметь метку: некоторый идентификатор, с помощью которого может осуществляться переход к этому оператору из произвольной области программы. Все метки должны быть описаны в разделе **label**:

label *список меток*;

Пример

label ml, m2, 10;

Раздел описания констант **const.** Если в программе используются константы, имеющие достаточно громоздкую запись или многократно использующиеся, то их целесообразно описать в разделе **const**:

const имя_1 = значение_1**;**
имя_2 = значение_2;

Пример

const F = 2.75; t = 'Dinamo';

Имеется ряд заранее определенных констант, которые можно непосредственно использовать в программе (без их описания в разделе const): например, *maxint* = 32767 – наибольшее положительное целое число типа integer; *pi* = 3.1415926.. – число "пи".

Раздел описания типов **type.** В этом разделе описываются имена типов переменных, отличных от стандартных, т.е. переменные типа ***записи, массивы, классы*** и т.д.:

type
имя_1 = вид_типа_1**;**
имя_2 = вид_типа_2;

здесь имя_1, имя_2 – идентификаторы вводимых разработчиком типов.

Пример

```
type
   vuz = (Bntu, Bgatu, Bgpu);
   x = 5..20;
```

Раздел описания переменных **var**. В этом разделе должны быть описаны все переменные, встречающиеся в программе:

```
var
   список_1 : тип_1;
   список_2 : тип_2;
…
```

Тип переменной в программе можно задать двумя способами:

- указать имя типа (из раздела type);
- описать непосредственно сам тип после ключевого слова var.

Пример

```
var
   egu    : vuz;
   a, b, c : extended;
```

Раздел описания процедур и функций включает тексты подпрограмм и функций.

Раздел операторов. Эта часть программы начинается с ключевого слова **begin** и заканчивается словом **end**, после которого должна стоять точка. Раздел операторов – выполняемая часть программы, которая состоит из совокупности исполняемых операторов. Исполняемые операторы отделяются друг от друга точкой с запятой.

Операторы Object Pascal не привязаны к определенной позиции строки. В одной строке можно указывать несколько описаний или операторов. Допускается перенос с одной строки на другую описаний или операторов (но без разделения ключевых слов и идентификаторов).

Арифметические выражния

Арифметические выражения строятся из числовых констант, переменных, стандартных функций и операций над ними. Для обозначения операций используются символы +, -, *, /, **div** (целочисленное деление, например, 5 **div** 4 = 1), **mod** (остаток от целочисленного деления, например, 5 **mod** 3 = 2).

Если один или более операндов в выражении имеют вещественный тип, то результат будет так же вещественного типа. В операциях **div** и **mod** оба операнда должны быть целого типа.

В арифметическом выражении принят следующий приоритет операций: вычисление значений стандартных функций; умножение и деление; сложение и вычитание. Порядок выполнения операций можно регулировать с помощью скобок.

Логические выражения

Логические выражения строятся из логических констант и переменных, логических операций и операций отношения.

В ***операциях отношения*** могут участвовать арифметические и логические выражения, а также символьные данные. Результатом логического выражения является значение *True* или *False*.

Для записи ***логических выражений*** используются следующие ***логические операции***: **not** (логическое «НЕ»), **and** (логическое «И»), **or** (логическое «ИЛИ»), **xor** (исключающее «ИЛИ»).

Применяются следующие ***операции отношения***: =, <>, .>, <, >=, <=. При выполнении операций отношения оба операнда (a, b) должны быть одного и того же типа.

При вычислении логических выражений принят следующий приоритет операций:

- **not,**
- ***, /, div, not, and;**
- **+, -, or. xor;**
- операции отношения.

Стандартные функции

В выражениях могут быть использованы следующие сандартные функции:

sin(x), **cos**(x), **arctan**(x) – соответственно синус, косинус, главное значение арктангенса аргумента x;

ln(x) – натуральный логарифм x;

exp(x) – показательная функция x, т.е. e^x;

abs(x) – абсолютная величина (модуль) x, т.е. $|x|$;

sqr(x) – квадрат (вторая степень) x, т.е. x^2;

sqrt(x) – квадратный корень из x, т.е. $\sqrt{x}$.

trunc(x) – вычисление целой части x**;**

round(x) – округление x;

odd(x) – true, если x – нечетное, false, если x – четное.

Выражение, задающее аргумент, всегда заключается в скобки.

В тригонометрических функциях аргумент x должен быть задан в радианах. Для перевода в радианы угла x, заданного в градусах, используется формула $y = x$*__pi__/180.

Для вычисления других функций применяются математические формулы, некоторые из которых приведены в Приложении 1. Например, для вычисления логарифма с основанием a используется соотношение $\log_a(x) = \ln(x) / \ln(a)$. Следует иметь в виду, что в функциях **arcsin** и **arccos** аргумент должен лежать в пределах от –1 до 1. Функции **arcsin** и **arctan** возвращают результат в пределах [–**pi**/2..**pi**/2], **arccos** – в пределах [0..**pi**].

Для возведения положительного числа x в действительную степень y используется соотношение $x^y = e^{y \ln(x)}$.

Вопросы для самоконтроля

1. Что такое переменная? Чем она отличается от константы?
2. С какой целью переменным назначаются типы?
3. Какие операторы используются для объявления переменных и констант?
4. Какие типы данных предназначены для хранения чисел, а какие – для хранения символов?
5. Использование арифметических операций *div* и *mod*.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Материалы к лабораторной работе № 1

1. Программирование линейных алгоритмов

Цель работы: научиться разрабатывать приложения в среде Delphi, обеспечивающие ввод исходных данных, выполнение последовательности заданных операций, которые реализуют алгоритм линейного типа, и вывод результатов на экран. ***Линейными*** называются алгоритмы, в которых операции выполняются последовательно одна за другой в единственном порядке.

Оператор присваивания

Сочетание символов «:» и «=» образует *оператор присваивания*. Общий вид оператора следующий:

<имя_переменной> ***:= <выражение>;***

где имя_переменной – *идентификатор* переменной, текущее значение которой заменяется значением, определяемым в правой части оператора присваивания.

:= – символ присваивания**.**

Пример

y := sqrt(x) + l; $n := n + 1$.

Составной оператор

Составной оператор – это объединение нескольких операторов в одну группу. Общий вид этого оператора следующий:

begin

<оператор_1>;

...

<оператор_n>

end;

В этой конструкции ключевые слова **begin** и **end** выполняют роль операторных скобок: открывающей и закрывающей. Составной оператор можно вставлять в любое место программы, где допускается использование одного оператора. В свою очередь любой из операторов составного оператора также может быть составным. Нельзя извне составного оператора передавать управление внутрь него.

1.1. Пример создания приложения

Задание. Создать приложение для вычисления по заданным значениям исходных данных величин *x, y, z* значения *r* из арифметического выражения

$$r = \frac{y^{x+1}}{\sqrt[3]{|y-2|}+3} + \frac{x+\frac{y}{2}}{2|x+y|}(x+1)^{-1/\sin z} + \cos(\pi/2)$$

при значениях $x = 13.8e\text{-}2$, $y = 1.54$, $z = 0.2e\text{-}3$.

1.1.1. Настройка формы

Для создания нового проекта выберите в основном меню пункт ***File | New | Application***. Пустая форма, отобразившаяся на экране, в правом верхнем углу имеет кнопки управления, которые предназначены: для свертывания формы в пиктограмму , для разворачивания формы на весь экран и возвращения к исходному размеру  и закрытия формы  . С помощью мыши, «захватывая» одну из кромок формы или выделенную строку заголовка, отрегулируйте нужные размеры формы и ее положение на экране.

1.1.2. Размещение компонентов на Форме

Вариант панели интерфейса создаваемого приложения показан на рисунке 1.2.

Как видно, интерфейсом программного приложения предусматриваются три отдельных окна для ввода исходных данных *x*, *y*, *z* посредством компонентов ***Edit,*** а для вывода результатов – одно многострочное окно (компонент ***Memo***). Надписи на форме организуются компонетами ***Label***. Для запуска приложения предусматривается кнопка управления, расположенная в нижней части интерфейса (компонент ***Button***).

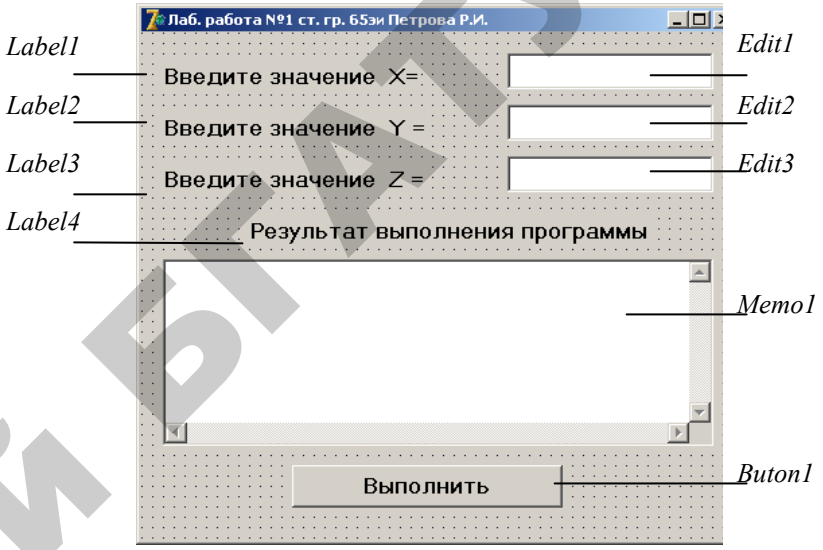

Рис. 1.2. Интерфейс приложения

1.1.3. Сохранение проекта

Для нового проекта создайте новую папку, например ***X:\65эи\ФИО_студента\Mod1\Lab1.***

Сохраните проект ***File | Save Project As….*** Сначала сохраните модуль с именем **Unit1.pas**, затем файл проекта под именем **Project1.dpr.**

Последующие сохранения выполнять командами ***File | Save All.***

1.1.4. Изменение заголовка формы

Первоначально система присваивает свойствам определенные начальные значения. Новая форма имеет одинаковое имя (свойство ***Name***) и заголовок (свойство ***Caption***) – Form1. Для изменения заголовка щелкните кнопкой мыши на форме и вызовите окно Инспектора объектов. На странице ***Properties*** Инспектора объектов щелкните мышью на свойстве ***Caption*** и в правой ячейке наберите, например, «Лаб. работа №1 ст. гр. 65эи Иванова Р.И.».

При создании достаточно простых программных приложений с небольшим числом форм свойство компанента ***Name*** можно не изменять.

1.1.5. Размещение строки ввода (Edit)

Для ввода, а также вывода информации, которая вмещается в одну строку, используется окно однострочного редактирования текста, представляемого компонентом ***Edit***. Доступ к отображаемой в окне информации в виде строки из символов (тип ***String)*** осуществляется с помощь свойства ***Text*** этого компонента.

Выберите в палитре компонентов *Standard* пиктограмму  , щелкните мышью в том месте формы, где ее нужно разместить. Вставьте три компонента ***TEdit*** в форму. Захватывая их «мышью» отрегулируйте размеры окон и их положение.

Обратите внимание на то, что в тексте программы появились три новых однотипных компонента ***Edit1***, ***Edit2***, ***Edit3***. В свойствах ***Text*** соответствующих каждой из этих компонент будет содержаться строка символов (тип *String*) и отображаться в соответствующих окнах ***Edit***. Переменные *x*, *y*, *z* имеют действительный тип, для преобразования строковой записи числа в тип вещественной переменной, используется стандартная функция StrToFloat. Перечень некоторых других процедур и функций для преобразования строкового представления чисел приведен в приложении 1.

В программе считывание строковых данных, их преобразование в данные вещественного типа и присвоение (сочетание двух символов: «:» и «=») полученного значения переменной вещественного типа *x* осуществляется строкой

```
x := StrToFloat(Edit1.Text);
```

Если исходные данные имеют целочисленный тип, например, **integer**, то используется стандартная функция **StrToInt**(Editl.Text).

При этом в записи числа не должно быть пробелов, а действительное число пишется с десятичной запятой. При необходимости правило назначения символа, разделяющего целую и дробную части, можно изменить на точку в разделе «Языки и региональные стандарты» панели управления Windows. Установите нужный шрифт и размер символов, отражаемых в строке ***Edit*** (свойство ***Font***), с помощью Инспектора объектов.

1.1.6. Размещение надписей (Label)

Для размещения надписей на форме используется компонент ***Label***. Выберите на странице *Standard* Палитры компонентов пиктограмму  , щелкните на ней мышью. Затем щелкните мышью в нужном месте формы: появится изображение компонента с надписью ***Label1***. Проделайте это для четырех других надписей. Для каждой надписи, щелкнув на ней мышью, отрегулируйте размер и положение на форме. Измените свойство ***Caption*** компонента в Инспекторе объектов: например, на «Введите значение *x*».

1.1.7. Размещение многострочного окна вывода Memo

Для вывода результатов работы программы в форме многострочного текста обычно используется текстовое окно, которое представлено компонентом (***Memo***). Выберите в Палитре компонентов пиктограмму  и поместите компонент ***Memo*** на форму. С помощью мыши отрегулируйте размеры и местоположение ***Memo1***. Для отображения вертикальной и горизонтальной полос прокрутки на странице *Properties* Инспектора объектов установите свойство ***ScrollBars*** в положение *ssBoth*. Можно убедиться, что в тексте программы появилась переменная ***Memo1*** типа *TMemo*.

Информация, которая отображается построчно в окно типа *TMemo*, находится в массиве строк *Memo1.Lines*. При выводе новых данных в ***Memo1*** используется метод *Add* свойства *Lines*, причем для преобразования данных из действительного значения в строковое и управления формой представления выводимого результата используются соответствующие функции (см. приложение 1). Например, если нужно вывести число *d*, представляющее собой в переменную целого типа, то его надо предварительно преобразовать к типу **string**:

```
Memo1.Lines.Add('d = '+IntToStr(d));
```

и в окне появится строка «d = 2009».

Если, например, переменная x – действительного типа и имеет значение -256,38666, то при использовании метода

Memo1.Lines.Add ('Значение x =' +FloatToStrF(x,ffFixed,8,2));

будет выведена строка «Значение x = -256.39». При этом под все число отводится восемь позиций, из которых две позиции занимает его дробная часть. Для очистки окна используется метод *Memo1.Clear*.

Если число строк в массиве ***Memo1*** превышает размер окна, то для просмотра всех строк используется вертикальная полоса прокрутки. Если длина строки ***Memo1*** превосходит количество символов в строке окна, то в окне отображается только начало строки. Для просмотра всей строки используется горизонтальная полоса прокрутки.

1.1.8. Написание программы обработки события создания формы (FormCreate)

При запуске программы на некотором этапе ее выполнения возникает событие «создание формы» (событие *OnCreate*). Составим подпрограмму – обработчик этого события, который заносит начальные значения переменных *x*, *y*, *z* в соответствующие окна ***Edit***, а в окне ***Memo*** – номер группы и фамилию студента. Для этого дважды щелкнем мышью на любом свободном месте формы. На экране отобразится окно редактора кода, в котором автоматически внесена заготовка процедуры с заголовком процедуры-обработчика события создания формы:

```
Procedure TForm1.FormCreate(Sender:TObject);
  begin
  end;
```

Между **begin** ... **end** программист записывает текст программы.

Последнюю операцию также можно инициировать двойным щелчком в правой части события *onActivate* на странице *Properties* инспектора объектов рассматриваемого компонета ***Form***.

1.1.9. Написание программы обработки события нажатия кнопки (ButtonClick)

Поместите на форму кнопку посредством компонента Button, который находится на странице *Standard* панели компонентов. С помощью инспектора объектов измените заголовок (*Caption*)

компонента ***Button1*** на слово «Выполнить» или другое. Отрегулируйте положение и размер кнопки. После этого два раза щелкните мышью на кнопке: появится заготовка для текста программы с заголовком процедуры-обработчика события – нажатия кнопки:

```
Procedure TForm1.Button1Click(Sender: TObject);
```

Заголовки рассматриваемых процедур формируются средой Delphi автоматически (если набрать их вручную – программа работать не будет). Так, имя процедуры **TForm1.Button1Click** составляется системой из имени формы (*Form1*), имени компонента (*Button*) и имени соответствующего события (*onClick*), но без префикса on. Надо иметь в виду, что при запуске программы на выполнение все функции обработки событий, у которых между **begin** и **end** не было написано текста, удаляются автоматически по соответствующему запросу среды Delphi. Поэтому не следует удалять вручную ошибочно созданные обработчики.

1.1.10. Работа с приложением

После запуска программы (например, щелчком мыши по зеленой пиктограмме ) происходит компиляция, если нет ошибок, компоновка программы и создание единого выполняемого файла с расширением **.exe**. На экране появляется активная форма программы (рис. 1.3).

Работа с программой осуществляется следующим образом. Введите исходные значения *x*, *y*, *z*. Нажмите (щелкните мышью) кнопку «Выполнить». В окне Memo1 отображается результат. Измените исходные значения *x*, *y*, *z* в окнах ***Edit*** и снова нажмите кнопку «Выполнить» – отображаются новые результаты. Завершить работу программы можно путем активизации пункта ***Program Reset*** в главном меню ***Run*** или кнопку  на форме.

На рисунке 1.3 показан интерфейс приложения после ввода исходных данных и нажатия кнопки «Выполнить».

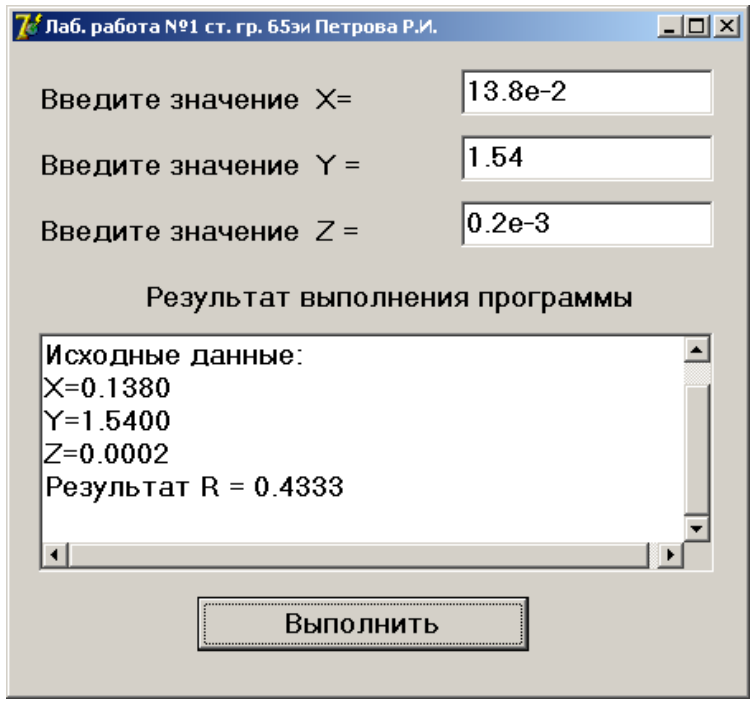

Рис. 1.3. Интерфейс приложения после его выполнения

В листинге 1.1 представлен текст программы. Для наглядности в этой работе операторы, которые набирает программист, выделены полужирным шрифтом. Остальные операторы вставляются средой Delphi автоматически.

Листинг 1.1.

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
   Forms,  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
```



```
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label4: TLabel;
    Memo1: TMemo;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
{$R *.dfm}

{Процедура обработки события создания Формы}
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Text:='13.8e-2';  //начальное значение X
  Edit2.Text:='1.54';     //начальное значение Y
  Edit3.Text:='0.2e-3';   //начальное значение Z
  Memo1.Clear;            //очистка окна редактора Memo1
end;

{Процедура обработки события нажатия кнопки "Выполнить"}
procedure TForm1.Button1Click(Sender: TObject);
 var                      //объявление локальных переменных
   x,y,z,a,b,c,r : extended;
begin
   Memo1.Clear;                  //очистка окна редактора Memo1
   {Ввод, преобразование и присвоение исходных данных }
```

```
    x:=StrToFloat(Edit1.Text);          //считывается значение x
    y:=StrToFloat(Edit2.Text);          //считывается значение y
    z:=StrToFloat(Edit3.Text);          //считывается значение z
    {Вывод заголовка и исходных данных в окно Memo1}
    Memo1.Lines.Add('Программирование линейных
алгоритмов');
    Memo1.Lines.Add('Исходные данные:'); //вывод строки
в Memo1
    Memo1.Lines.Add('x='+FloatToStrF(x,ffFixed,8,4));
    Memo1.Lines.Add('y='+FloatToStrF(y,ffFixed,8,4));
    Memo1.Lines.Add('z='+FloatToStrF(z,ffFixed,8,4));
    {Вычисления}
    a:=exp((x+1)*ln(y))/(exp((1/3)*ln(abs(y-2)))+3);
    b:=(x+y/2)*exp((-1/sin(z))*ln(x+1));
    c:=2*abs(x+y);
    r:=a+b/c+cos(pi/2);
    {Вывод результата r в окно Memo1}
    Memo1.Lines.Add('Результат r = '+FloatToStrF(r,ffFixed,8,4));
  end;
  end.
```

1.2. Индивидуальные задания

Выберете свой вариант индивидуального задания по указанию преподавателя. Уточните условие задания, количество, наименование, типы исходных данных и форму вывода результатов. В соответствии с этим установите необходимое количество окон, тексты заголовков на форме, размеры шрифтов, а также типы переменных и функции преобразования при вводе и выводе результатов. Целесообразно сложные выражения рассматривать как совокупность простых выражений, вычисляемых отдельно.

Индивидуальные задания 1-го уровня

Номер	Арифметическое выражение	Исходные данные
1	$r = y^{\sqrt[3]{x}} + \cos(y)\dfrac{1}{e^{x-y} + \dfrac{x}{2}} + \sin(\pi/2)$	$x = 1.234$, $y = 0.567$
2	$g = y^{x+1} + \dfrac{x + \dfrac{y}{2}}{2\lvert x + y\rvert}(x+1)^3 + \cos(\pi/2)$	$x = 12.3\cdot10^{-1}$, $y = 15.4$
3	$s = \left\lvert x + \dfrac{2y}{1 + x^2 y^2}\right\rvert * \lvert x\rvert^y + \sin^2\left(arctg\ \dfrac{1}{z}\right)$	$x = -3.999\cdot10^{-2}$, $y = -0.601$, $z = 0.245\cdot10^3$
4	$u = e^{\lvert x-y\rvert}\left(tg^2 z + 1\right)^x + \sqrt[3]{x/3 + (x-y)^2 + 1}$	$x = -4.5$, $y = 0.75\cdot10^{-4}$, $z = 0.5\cdot10^{-2}$
5	$w = \left(1 + z + \dfrac{z^2}{2} + \dfrac{z^3}{3}\right)\lvert \sin x - \cos y\rvert^{0.5}$	$x = 0.4\cdot10^4$, $y = -0.85$, $z = -0.475\cdot10^{-5}$
6	$s = x^3 + \sin^2\left(arctg\ \dfrac{1}{z}\right) + \left\lvert y - \dfrac{3x}{1 + xy^2}\right\rvert$	$x = 3.99^{-2}$, $y = -6.11$, $z = 0.245\cdot10^2$
7	$f = \sqrt[4]{y + \sqrt[3]{x-1}} + \lvert x - y\rvert\left(\sin^2(x) + tg(z)\right)$	$x = 0.421$, $y = 10.35\cdot10^{-3}$, $z = 0.8$
8	$t = \ln\left(y^3\right)\left(x - \dfrac{y}{2}\right) + \lvert \sin(x) - \cos(y)\rvert^{0.5}$	$x = -0.24$, $y = 9.642\cdot10^{-2}$
9	$f = y^{\sqrt[3]{x}} + \cos^3(y)\lvert x - y\rvert \times \left(1 + \dfrac{\sin^2 z}{\sqrt{x + y}}\right)$	$x = 6.251$, $y = 0.827$, $z = 25.001$
10	$h = \dfrac{\lvert y - x\rvert^2}{2} - \dfrac{\lvert y - z\rvert^3}{3} + \lvert \sin(x) - \cos(y)\rvert^{0.5}$	$x = -2.44$, $y = 0.869$, $z = -0.13\cdot10^3$

Индивидуальные задания 2-го уровня

Номер	Арифметическое выражение	Исходные данные
11	$b = y^{\sqrt[3]{x}} + \cos^3(y)\dfrac{\lvert x-y\rvert\left(1+\dfrac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{x-y}+\dfrac{x}{2}} + \sin(\pi/2)$	$x = 6.251$, $y = 0.827$, $z = 25.001$
12	$g = \dfrac{y^{x+1}}{\sqrt[3]{\lvert y-2\rvert}+3} + \dfrac{x+\dfrac{y}{2}}{2\lvert x+y\rvert}(x+1)^{-1/\sin z} + \cos(\pi/2)$	$x = 12.3\cdot10^{-1}$, $y = 15.4$, $z = 0.252\cdot10^{3}$
13	$s = \dfrac{1+\cos^2(x+y)}{\left\lvert x+\dfrac{2y}{1+x^2y^2}\right\rvert} * \lvert x\rvert^{y} + \sin^2\left(arctg\ \dfrac{1}{z}\right)$	$x = 3.999\cdot10^{-2}$, $y = -0.601$, $z = 0.245\cdot10^{3}$
14	$t = \dfrac{1+2.71^{\lvert x-y\rvert}}{0.5+\sin^2 y}\left(4+\dfrac{z^2}{3-z^2/4}\right).$	$x = 14.26$, $y = -1.22$, $z = 3.5\cdot10^{-2}$
15	$u = \dfrac{x^2+y^2+2}{\sqrt[3]{x+(x-y)^2+1}} - e^{\lvert x-y\rvert}\left(tg^2 z+1\right)^{x}$	$x = -4.5$, $y = 0.75\cdot10^{-4}$, $z = 0.005\cdot10^{2}$
16	$w = \left(1+z+\dfrac{z^2}{2}+\dfrac{z^3}{3}\right)\lvert\sin x-\cos y\rvert^{\left(1+2\sin^2 y\right)}$	$x = 0.4\cdot10^{4}$, $y = -0.875$, $z = -0.475\cdot10^{-3}$
17	$s = \dfrac{\left\lvert y-\dfrac{x}{1+xy^2}\right\rvert}{1+\cos^2(x+y)}x^3 + \sin^2\left(arctg\ \dfrac{1}{z}\right)$	$x = 3.999\cdot10^{-2}$, $y = -6.011$, $z = 0.245\cdot10^{2}$
18	$f = \dfrac{\sqrt[4]{y+x^3}}{\lvert x-y\rvert\left(\sin^2 z+tgz\right)}\sqrt[4]{y+\sqrt[3]{x-1}}$	$x = 17.421$, $y = 10.365\cdot10^{-3}$, $z = 0.828\cdot10^{5}$
19	$t = \sin^2(arctg\,(z)x^3) + \ln\left(y^{-\sqrt{\lvert x\rvert}}\right)\left(x-\dfrac{y}{2}\right)$	$x = -15.246$, $y = 4.642\cdot10^{-2}$, $z = 20\ ,001\cdot10^{2}$

20	$c = (3x)^{y} - \dfrac{y\left(arctgz-\dfrac{\pi}{6}\right)}{\sin\lvert x\rvert+\dfrac{1}{y^2+1}} + 3\left(x+z^{2.5}\right)$	$x = 3.251$, $y = 0.325$, $z = 0.466\cdot10^{-4}$
21	$f = y^{\sqrt[3]{x}} + \cos^3(y)\dfrac{\lvert x-y\rvert\left(1+\dfrac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{x-y}+\dfrac{x}{2}}$	$x = 6.251$, $y = 0.827$, $z = 25.001$
22	$d = \left\lvert x^{\frac{y}{x}} - \sqrt[3]{\dfrac{y}{x}}\right\rvert + (y-x)\dfrac{\cos y-\dfrac{z}{(y-x)}}{1+\left(y-x\right)^2}$	$x = 1.825\cdot10^{2}$, $y = 18.225$, $z = -3.298\cdot10^{-2}$
23	$h = \dfrac{x^{y+1}+e^{y-1}}{1+x\lvert y-tgz\rvert}\left(1+\lvert y-x\rvert\right) + \dfrac{(y-x)^2}{2} - \dfrac{\lvert y-x}{3}$	$x = 2.444$, $y = 0.869\cdot10^{-2}$, $z = -0.13\cdot10^{3}$
24	$b = \sqrt{x+\sqrt[4]{\lvert y\rvert}} * \sqrt[3]{e^{\left(x-1/\sin z\right)}\cdot\ln(x+z)}$	$x = 3.981\cdot10^{-2}$, $y = -1.625\cdot10^{3}$, $z = 0.59$
25	$f = \dfrac{e^{\lvert x-y\rvert}\lvert x-y\rvert^{x+y}}{arctg\,(x)} + \sqrt[3]{x^2+\ln^2 y}$	$x = -2.235\cdot10^{-2}$, $y = 2.23$, $z = 15.221$

Индивидуальные задания 3-го уровня

26. В совхозе построили цех по приготовлению кормов. В первые k месяцев работы цеха ежедневно вводилась в строй часть оборудования, проектная производительность которой составляет p % от проектной производительности всего цеха. Какая часть оборудования (равная) должна вводиться в каждый из последующих месяцев, чтобы все оборудование цеха ввелось в течение года?

27. Участок луга имеет форму равнобедренной трапеции с большим основанием a, высотой h, и углом k между боковой стороной и большим основанием. Сколько травы должно быть скошено с данного участка, если норма скошенной травы с квадратного метра составляет p кг?

28. Суточная норма кормления одной коровы составляет *a* кг, одной лошади *b* кг сена. Определить, сколько коров можно прокормить в течение *n* дней, располагая массой сена *p* кг, если при этом придется одновременно кормить *r* лошадей.

29. Внутреннее помещение коровника имеет форму прямоугольного параллелепипеда, длиной *a*, шириной *b*, высотой *h*. Для очистки воздуха в коровнике установили вытяжной вентилятор, который должен перекачивать за время *t* через круглый вентиляционный канал диаметром *d* объем воздуха, равный объему помещения коровника. Определить, какова должна быть скорость воздуха в вентиляционном канале?

30. Сложный краситель по массе состоит из 10 % синего компонента, 10 % белого компонента, 18 % желтого компонента, остальное – добавки. Сколько сложного красителя можно приготовить, имея *p* кг синего красителя и сколько для этого необходимо белого и желтого красителей?

Вопросы для самоконтроля

1. Каково должно быть свойство *ReadOnly* компонента *Edit*, чтобы во время выполнения программы пользователь не мог изменять текст поля ввода?
2. Укажите, в левой или правой колонке вкладки *Events* Инспектора объектов перечислены имена событий, которые может воспринимать выбранный компонент (объект).
3. Какие операторы выполняют операции целочисленное деление и деление по модулю?
4. Приведите преимущества и недостатки глобальных переменных.
5. Каким образом в Object Pascal создаются комментарии?

Материалы к лабораторной работе № 2

2. Программирование разветвляющихся алгоритмов

Цель работы: освоить средства организации разветвляющихся алгоритмов в языке Object Pascal в визуальной среде Delphi.

Условный оператор if

Разветвляющимися называются алгоритмы, в которых в зависимости от выполнения некоторых логических условий происходит разветвление вычислительного процесса по одному из нескольких возможных направлений.

Для реализации разветвляющихсяся алгоритмов в языке Object Pascal служат операторы **if** и **case**.

Оператор **if**, называемый условным оператором. Он выбирает один из двух возможных путей выполнения алгоритма в зависимости от условного логического выражения. В общем виде оператор записывается следующим образом:

if *<логическое выражение>* ***then*** *<оператор_1>* ***else*** *<оператор_2>;*

Вначале вычисляется значение *<логического выражения>*. Если оно равно ***true*** (истина), то выполняется *<оператор_1>*, а *<оператор_2>* пропускается, если же значение *<логического выражения>* равно ***false*** (ложь), то выполняется *<оператор_2>*, а *<оператор_1>* пропускается. Далее и в том, и в другом случаях выполняется следующий оператор.

Например, присвоить переменной *g* значение ноль в случае равенства нулю значения выражения $x + y$ и значение один в противном случае можно с помощью оператора:

```
if  x+y = 0  then  g : = 0 else g : = 1;
```

Как *<оператор_1>*, так и *<оператор_2>* могут быть составными, состоящими из группы операторов, заключенной в операторные скобки **begin … end**. Например, в результате выполнения оператора:

```
if x>=0 then
    begin
      y := 1/(x*x+1);
```

```
    z := y*x-1;
  end
else
  begin
    y := 0;
    z := 0;
  end;
```

Переменные *y* и *z* вычисляются по указанным формулам в случае *x*>=0 и примут нулевые значения в противном случае.

Если какое-либо действие должно быть выполнено только при выполнении определенного условия и пропущено в случае невыполнения этого условия, то оператор **if** может быть записан в сокращенной форме:

if *<логическое выражение>* ***then*** *<оператор>;*

Например, если нужно увеличить значение переменной *n* на 1 в случае положительного значения переменной *x* и оставить значение *n* прежним в противном случае, то можно выполнить следующий оператор:

if x > 0 **then** n := n + 1;

В общем виде оператора ***if*** *<оператор_1>* и *<оператор_2>* могут сами являться операторами **if.** Тогда говорят о вложенных операторах **if**. В этом случае каждое **else** относится к ближайшему слева **if**. Для удобства рекомендуется слово **else** писать под соответствующим словом **then**.

Например, фрагмент программы вычисления значения *y* по формуле

$$y = \begin{cases} 2 * \sin(r) + 4, & \text{если } r > 0 \\ 9, & \text{если } r = 0 \\ 4 * \cos(r) - 2, & \text{если } r < 0 \end{cases}$$

будет иметь следующий вид:

```
if  r > 0 then y := 2*sin(r)+4
        else
          if  r = 0 then y :=9
                else  y := 4*cos(r)-2;
```


Фрагмент программы определения наибольшего из трех неравных между собой чисел *a*, *b*, *c* и присвоение найденного значения переменной *p* будет иметь вид:

if a>b **then** p:=a **else** p:=b;
if c>p **then** p:=c;

Второй вариант вычисления

if a>b **then if** a>c **then** p:=a **else** p:=c
else if b>c **then** p:=b **else** p:=c;

В обоих вариантах значение наибольшего числа заносится в переменную *p*.

Оператор выбора case

В отличие от оператора **if** оператор **case** позволяет выбрать один из нескольких операторов, а не из двух. Общий вид оператора:

case *<выражение>* ***of***
  <список меток 1> : <оператор_1>;
  <список меток 2> : <оператор_2>;
…
  <список меток N> : <оператор_N>;
else
  <оператор>
end*;*

Здесь *<выражение>* – это выражение порядкового типа (не может возвращать значение действительного типа или строки). Значение выражения определяет дальнейший ход выполнения программы. Список меток представляет список констант, разделенных запятыми. Если константы представляют диапазон чисел, то указывается первое и последнее число, разделенное двумя точками, например: 1..4.

Оператор выполняется следующим образом.

Если значение *<выражения>* совпадает с одной из констант из какого-либо списка, то выполняется соответствующий этому списку оператор и выполнение оператора **case** завершается. Если значение *<выражения>* не совпадает ни с одной из констант, то выполняется оператор, следующий за **else**, или ни один из приведенных операторов не выполняется, если отсутствует конструкция **else**. Все операторы могут быть составными.

Например, оператор, заносящий в переменную *s* название времени года в зависимости от номера месяца *n* , будет иметь вид.

```
case n of
  1,2,12    : s:= 'Зима';
  3,4,5     : s:= 'Весна';
  6,7,8     : s:= 'Лето';
  9,10,11   : s:= 'Осень'
end;
```

Так, если *n* = 5, то *s* = 'Весна'.

Оператор безусловного перехода

Оператор безусловного перехода имеет вид:

goto *метка***;**

Метка должна быть описана в разделе описаний **label** и записывается перед помечаемым оператором, от которого отделяется двоеточием.

Пример

```
goto 32;
10: a:=2;
32: y:=x/a;
```

Оператор, следующий после **goto,** всегда должен иметь метку (иначе он никогда не будет выполняться).

2.1. Пример создания приложения

Задание. Создать Windows-приложение в визуальной среде Delphi для решения следующей задачи.

Для некоторого значения аргумента *x*, принадлежащего одному из заданных интервалов, вычислить значение соответствующей функции. Графики функций представлены на рисунке 1.4. Для интервала *x* > 2 предусмотреть возможность замены функции *y* = 4 на одну из функций *y* = 3*x* – 2 или *y* = 8/*x*.

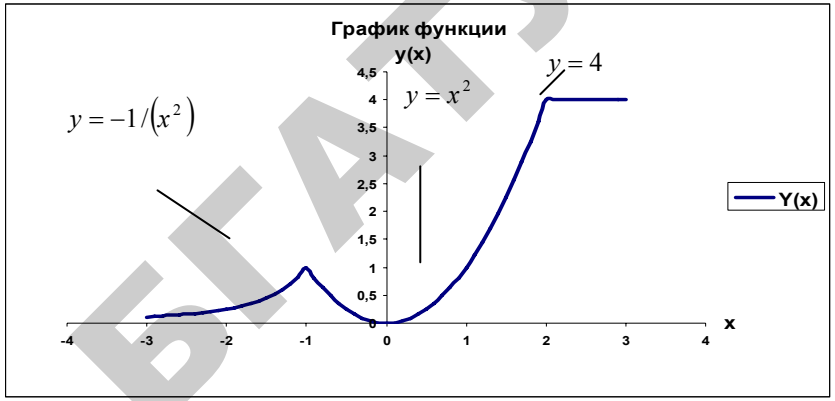

Рис. 1.4. Исходные функции

Математическая модель задачи имеет вид:

$$y = \begin{cases} -1/(x^2), & \text{если } x <= -1 \\ x^2, & \text{если } -1 < x <= 2 \\ 4, & \text{если } x > 2 \end{cases}$$

2.1.1. Размещение компонентов на Форме

При создании приложений в Delphi часто используются компоненты в виде кнопок-переключателей. Состояние такой кнопки (включено-выключено) визуально отражается на форме. На форме (рисунок 1.5) представлены кнопки-переключатели двух типов: ***CheckBox*** и ***RadioGroup***.

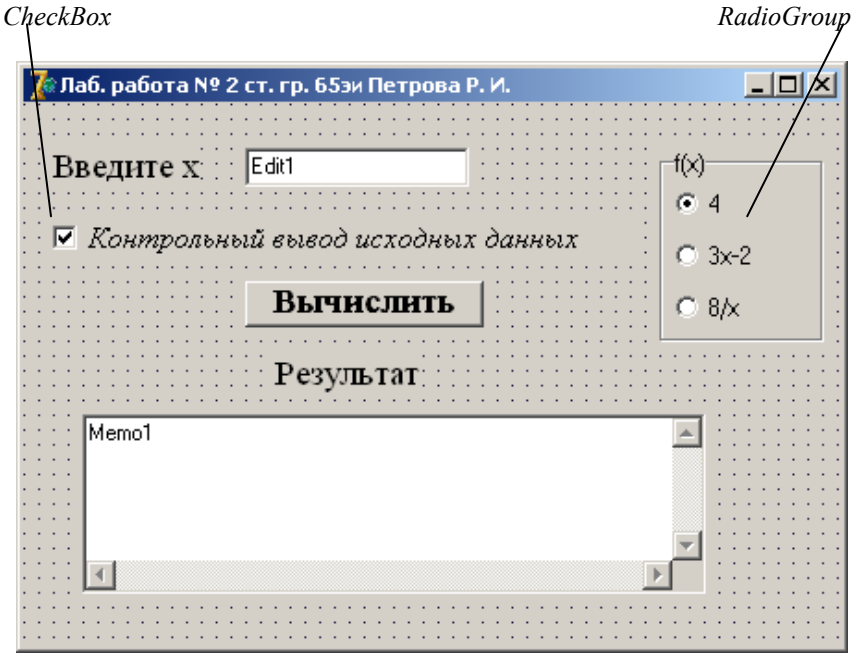

Рис. 1.5. Размещение компонентов на Форме

Компонент ***CheckBox*** организует ***кнопку*** независимого переключателя, с помощью которой пользователь может указать свое решение типа «да/нет».

В нашем примере с помощью этого компонента организована возможность контрольного вывода исходных данных перед выводом результата. При отключении кнопки будет выводиться только результат.

Компонент ***RadioGroup*** организует ***группу кнопок*** – зависимых переключателей. При нажатии одной из кнопок группы все остальные кнопки выключаются. В нашем примере при помощи данного компонента пользователю дается право выбора одной из приведенных функций.

Компоненты на форме размещаются в соответствии с предлагаемым вариантом интерфейса (рис. 1.5).

Разместите на Форме следующие компоненты: ***Label*** для нанесениянадписей, ***Edit*** для ввода исходных данных, ***Memo*** для вывода результата. Далее размещается компонент ***Button (***с именем ***Button1***) для создания кнопки, при нажатии которой будет рассчитываться значение функции. Для организации указанных расчетов в программный модуль помещается процедура обработки события нажатия кнопки ***Button1***, а именно ***Procedure TForm1.Button1Click***. Курсор помещается в начало этой процедуры при двойном щелчке левой кнопкой мыши по кнопке ***Button1*** на форме.

Для размещения на форме компонента ***CheckBox*** выбирается в Палитре компонентов на странице *Standard* пиктограмма  и размещается в нужном месте формы. В свойстве ***Caption*** Инспектора объектов надпись ***CheckBox1*** следует заменить «Контрольный ввод исх. данных». Для того чтобы кнопка ***CheckBox*** оказалась включена при запуске приложения, свойство *Checked* установить в состояние *True*.

Для размещения на форме компонента ***RadioGroup*** выбирается в Палитре компонентов на странице *Standard* пиктограмма  и помещается в нужное место формы. В свойстве ***Caption*** компонента ***RadioGroup1*** следует ввести заголовок – f(*x*). Для размещения кнопок-переключателей в один столбец свойство *Columns* нужно установить равным 1. Далее дважды щелкнув «мышью» по правой части свойства ***Items***, вызовем строчный редактор списка наименований переключателей. Наберем 3 строки с наименованиями: в первой строке – 4; во второй – 3*x*-2, в третьей – 8/*x* и нажмем *OK*. После этого на форме появится группа из трех кнопок-переключателей с соответствующими надписями. Свойство ***ItemIndex*** содержит номер выделенного переключателя. Исходное его значение, равное 1, показывает, что ни один переключатель не выбран. Чтобы при запуске приложения первый переключатель оказался выбранным, свойство ***ItemIndex*** установим равным 0.

2.1.2. Сохранение проекта

Для нового проекта создайте новую папку, например, ***X:\65эи\ФИО_студента\Mod1\Lab2.***

Сохраните проект ***File | Save Project As….*** Сначала сохраните модуль с именем **Unit1.pas**, затем файл проекта под именем **Project1.dpr.**

Последующие сохранения выполнять командами ***File | Save All.***

2.1.3. Создание процедур обработки событий FormCreate и Button1Click

Технология создания процедуры обработки события ***FormCreate*** та же, что и в предыдущей лабораторной работе. Вывод на экран этой процедуры модуля происходит при двойном щелчке кнопкой «мыши» в свободном месте формы.

Текст процедуры ***Button1Click*** появляется на экране при двойном щелчке «мышью» по созданной кнопке ***Button1***. В процедуру следует включить программу вычисления функции для введенного значения аргумента x. По математической модели видно, что данный вычислительный процесс является разветвляющимся, следовательно, в программе должны содержаться операторы организации разветвлений (**if…then…else**). Также в этой процедуре должны содержаться операторы обработки компонентов ***RadioGroup*** и ***CheckBox*** (комментарии в листинге 1.2)**.**

Если активизирована первая кнопка ***RadioGroup1***, в переменную целого типа ***RadioGroup1.ItemIndex*** заносится ноль, если вторая – единица, если третья – два. И в соответствующем операторе **case** определены действия, выполняемые в каждом из этих случаев.

Внимательно изучите и наберите текст приведенного модуля.

2.1.4. Работа с приложением

Запустите созданное приложение. Выполните расчет значения функции для первоначального значения аргумента. Проанализируйте, по какой из ветвей вычислительного процесса выполнились вычисления. Произведя контрольные расчеты на микрокалькуляторе, убедитесь, что расчет выполнился правильно. Введя соответствующие исходные данные, проверьте вычисления по двум другим ветвям вычислительного процесса, изменяя функцию *f(x)*.

На рисунке 1.6 показан интерфейс приложения после ввода исходных данных и нажатия кнопки «Вычислить».

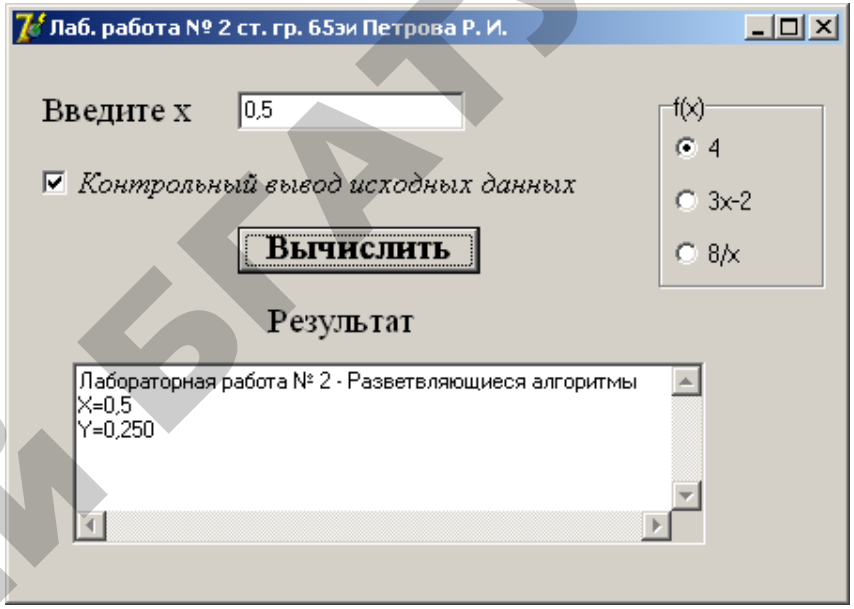

Рис. 1.6. Интерфейс приложения после его выполнения

Текст программы приведен в листинге 1.2.

Листинг 1.2

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    CheckBox1: TCheckBox;
    RadioGroup1: TRadioGroup;
    Button1: TButton;
```

```
    Memo1: TMemo;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
 begin
   Edit1.Text:='0,5';
   Memo1.Clear;
   Memo1.Lines.Add('Лабораторная работа № 2 - Разветвляющиеся
   алгоритмы');
 end;

procedure TForm1.Button1Click(Sender: TObject);
var
   x, y, fx: extended;
 begin
  x:=StrToFloat(edit1.Text);
  fx:=4;
 {Выбор функции, соответствующей нажатой кнопке}
   case RadioGroup1.ItemIndex of
    0: fx:=4;
    1: fx:=3*x-2;
    2: fx:=8/x;
   end;
{Вычисление значения функции}
```



```
      if x<= – 1 then y:= – 1/(x*x)
                 else
                    if x<=2 then y:=x*x
                            else y:=fx;
    {Проверка состояния кнопки CheckBox1}
      if CheckBox1.Checked then
       Memo1.Lines.Add('X='+Edit1.Text); //контрольный вывод ис-
ходных. данных
    {Вывод результата в Memo1}
       Memo1.Lines.Add('Y='+FloatToStrF(y,ffFixed,8,3));
     end;
     end.
```

2.2. Индивидуальные задания

Выберите по указанию преподавателя вариант индивидуального задания. Разработайте приложение, решающее задачу, поставленную в индивидуальном задании, и протестируйте работу созданного приложения.

Индивидуальные задания 1-го уровня

Для вариантов 1–10 вместо функции $f(x)$ использовать константу любого значения.

Индивидуальные задания 2-го уровня

Для вариантов 1–15 на панели интерфейса предусмотрите возможность выбора одной из трех функций: ***f(x): sin(x), cos(x), tg(x).***

1. $C = \begin{cases} |ax-3| + f(x), & \text{если } x \geq 2 \\ \min(a,c), & \text{если } x < 2 \end{cases}$

2. $Z=\begin{cases}\dfrac{a+b}{4+\cos^2 x}, & при\ \ x<2,3\\ (a+b)/(x+1), & при\ \ 2,3\le x<5\\ e^x+f(x), & при\ \ x\ge 5\end{cases}$

3. $C=\begin{cases}ax^2+b-4, & если\ \ x<-2\\ x^4-1-f(x), & если\ \ -2\le x<0\\ x^3+bx, & если\ \ 0\le x<2\\ 4f(x), & если\ \ x>2\end{cases}$

4. $D=\begin{cases}|x|+a^2+4, & если\ \ a>2\\ \dfrac{x^2+2ax}{3}, & если\ \ a=2\\ 8x+f(x), & если\ \ a<2\end{cases}$

5. $D=\begin{cases}f(x), & при\ \ x\le 0\\ x, & при\ \ 0<x\le 1\\ |a|+x^2, & при\ \ x>1\end{cases}$

6. $Z=\begin{cases}\sqrt{3x^2+7\sin^2 x+5}, & при\ \ x<0,1\\ ax+b\sqrt{f^2(x)+1}, & при\ \ x=0,1\\ \sqrt{x^2+7\cos^2 x+x}, & при\ \ x>0,1\end{cases}$

7. $C=\begin{cases}a^3-b^2+2f(x), & если\ \ x>d\\ a^2+b^2-tgx, & если\ \ x=d\\ a-2b^4+\cos x, & если\ \ x<d\end{cases}$

8. $Y=\begin{cases}e^{-bx}*\sin bx, & при\ \ x<a\\ \cos ax, & при\ \ a\le x\le b\\ e^{-ax}*f(x), & при\ \ x>b\end{cases}$

9. $F=\begin{cases}x^3-3x+8, & при\ \ x^2-x<1\\ 4f(x), & при\ \ x^2-x=1\\ 1/(x^2+3x+8), & при\ \ x^2-x>1\end{cases}$

10. $F=\begin{cases}x^3+3x+8, & при\ \ x<0\\ 4-f(x), & при\ \ 0\le x<1\\ a/(x^3+3x+8), & при\ x\ge 1\end{cases}$

11. $Q=\begin{cases}\dfrac{1}{z^2+1}-1, & если\ z<0\\ 4/(f(x)+2)+4, & если\ z=0\\ z^4-x^3+1, & если\ z>0\end{cases}$

12. $K=\begin{cases}f(x), & если\ a^2+b-x>0\\ 2f(x), & если\ a^2+b-x=0\\ 3f(x), & если\ \ a^2+b-x<0\end{cases}$

13. $Y=f^3(x)-z-5,\ где\ z=\begin{cases}x^2-2x-2, если\ x>2\\ 2, \ если -2\le x\le 2\\ -x^2-x-4, если\ x<-2\end{cases}$

14. $F=\begin{cases}f(x)-a, & при\ \ a\ >0,3\\ f^2(x)+a\ , при\ \ a=0,3\\ \cos(x-a), & при\ \ a<0,3\end{cases}$

15. $S=\begin{cases}\min(x^2,y^2)+a, & если\ \ a>0\\ f(x)+y, & если\ \ a=0\\ |x-y|, & если\ \ a<0\end{cases}$

Индивидуальные задания 3-го уровня

16. Вычислить $q = \min(x, y^2, z^2) + 1$.

17. Даны три неравных между собой числа z, r, q. Наибольшее из них разделить на сумму двух других.

18. Дана точка b с координатами (x, y). Определить в какой четверти лежит эта точка.

19. Определить, какая из точек плоскости $a(a_1, b_1)$, $b(a_2, b_2)$, $c(a_3, b_3)$ находится ближе всех к началу координат.

20. Даны три неравные между собой числа p, q, r. Вывести их в порядке возрастания.

Вопросы для самоконтроля

1. Какие операторы организуют разветвления в языке Delphi?
2. Какие формы оператора **if** вы знаете? Для каждой формы привести формат и примеры.
3. Опишите порядок выполнения оператора **case**.
4. Каково назначение компонента ***CheckBox***?
5. На какой странице Палитры компонентов размещены компоненты ***CheckBox*** и ***RadioGroup***?

3. Программирование циклических алгоритмов

Цель работы: изучить инструментальные средства организации циклических вычислительных процессов; овладеть практическими навыками работы с циклами.

Операторы организации циклов

Циклическими называются алгоритмы, которые предусматривают многократное повторение действий в одной и той же последовательности по одним и тем же математическим зависимостям, но при различных значениях некоторых специально изменяемых величин.

Различают два вида циклов – цикл с заданным числом повторений (детерминированный) и цикл, число повторений которого заведомо неизвестно (итерационный).

Для организации циклических алгоритмов в языке Object Pascal используются 3 вида операторов:

for – оператор цикла с управляющим параметром;

while – оператор цикла с предварительным условием;

repeat – оператор цикла с последующим условием.

Оператор **for** используют для организации детерминированных циклов, если шаг изменения параметра цикла равен 1 или –1. Операторы **while, repeat** используются для организации итерационных циклов и в случае детерминированных циклов с шагом изменения параметра отличным от 1 и –1.

Оператор for

Оператор **for** состоит из заголовка и тела цикла. Он может быть представлен в двух форматах:

for *<счетчик>:=n1* ***to*** *n2* ***do*** *<оператор>;*

for *<счетчик>:=n2* ***downto*** *n1* ***do*** *<оператор>;*

где *<счетчик>* – параметр цикла (переменная порядкового типа); *n1, n2* – начальное и конечное значения параметра цикла; *<оператор>* – простой или составной оператор, образующий тело цикла.

Первый формат организует увеличение <*счетчика*> на 1, второй – уменьшение <*счетчика*> на 1.

Ниже в качестве примера приведен фрагмент программы вычисления суммы квадратов натуральных чисел от 1 до 10.

```
s:=0;
for i:=1 to 10 do s:=s + i*i;
```

Оператор while

Общий вид оператора **while**:

while <*условие*> ***do*** **<***оператор***>;**

Выполняется оператор следующим образом. Вначале проверяется <*условие*> и пока оно ***истинно*** повторяется <*оператор*>, стоящий после зарезервированного слова ***do***. Выход из цикла происходит, когда <*условие*> становится ***ложным***. Если при первом вычислении <*условие*> окажется ложным, то оператор цикла не выполнится ни разу.

Если в цикле нужно выполнить не один оператор, а несколько, то их следует заключить в операторные скобки **begin ... end**, то есть использовать составной оператор.

Пример вычисления суммы квадратов натуральных чисел от 1 до 10 с использованием оператора **while** будет выглядеть следующим образом.

```
s:=0;
i:=1;
while i<=10 do
    begin
        s:=s+i*i;
        i:=i+1;
    end;
```

Оператор repeat

Общий вид оператора **repeat.**

```
repeat
  <оператор_1>;
  <оператор_2>;
```



```
  . . .
  <оператор_N>;
until < условие>;
```

Выполняется оператор **repeat** следующим образом. В начале выполняется группа операторов – <*оператор_1*>, <*оператор_2*>, ... <*оператор_N*>. Затем проверяется условие, если оно ***ложно***, то снова выполняется «тело» цикла. Если же условие ***истинно***, то происходит выход из цикла. Так как условие прекращения цикла проверяется в конце цикла, то операторы, образующие тело цикла, выполняются, по крайней мере, один раз.

Рассмотренный выше пример вычисления суммы квадратов натуральных чисел от 1 до 10 с использованием оператора **repeat** будет выглядеть следующим образом.

```
s:=0;
i:=1;
repeat
    s:=s+i*i;
    i:=i+1;
until i>10;
```

3.1. Пример создания приложения

Задание. Создать Windows-приложение, реализующее следующую задачу: даны натуральное число *n* и действительное *x*. Вычислить сумму членов ряда $s = \frac{x}{n+1} + \frac{x}{n+2} + \frac{x}{n+3} + ... + \frac{x}{n+n}$.

3.1.1. Размещение компонентов на Форме

Один из возможных вариантов панели интерфейса создаваемого приложения показан на рисунке 1.7.

Компонент ***SpinEdit*** находится на странице *Samples* Палитры компонентов.

Для задания переменной n удобно использовать компонент ***SpinEdit.*** Компонент ***SpinEdit*** предназначен для отображения и редактирования целого числа. Установите для компонента ***SpinEdit1*** значения свойств: ***MinValue****=1,* ***MaxValue****=15*.

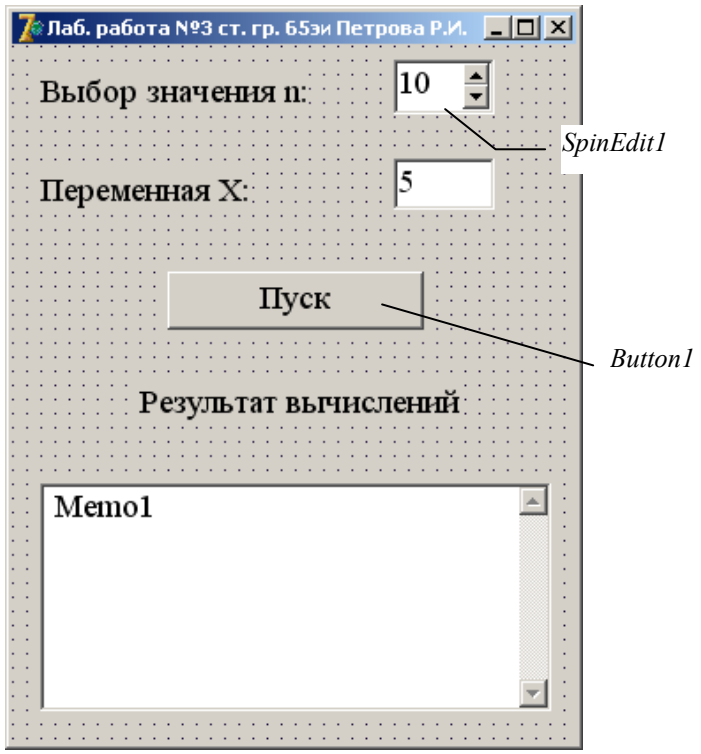

Рис. 1.7. Размещение компонентов на Форме

3.1.2. Сохранение проекта

Для нового проекта создайте новую папку, например, ***X:\65эи\ФИО_студента\Mod1\Lab3.***

Сохраните проект ***File | Save Project As….*** Сначала сохраните модуль с именем **Unit1.pas**, затем файл проекта под именем **Project1.dpr**.

Последующие сохранения выполнять командами ***File | Save All.***

3.1.3. Создание процедуры обработки события FormCreate

Создайте процедуру ***FormCreate***, наберите текст процедуры ***procedure TForm1.FormCreate(Sender: TObject).***

3.1.4. Создание процедуры обработки события нажатия кнопки Button1 (Button1Click)

Создайте процедуру обработки события нажатия кнопки «Пуск» – ***procedure TForm1.Button1Click(Sender: TObject)*** и наберите текст этой процедуры.

3.1.5. Работа с приложением

Запустите созданное приложение, занесите числовые значения и убедитесь, что приложение функционирует в соответствии с заданием.

На рисунке 1.8 показан интерфейс приложения после ввода исходных данных и нажатия кнопки «Пуск».

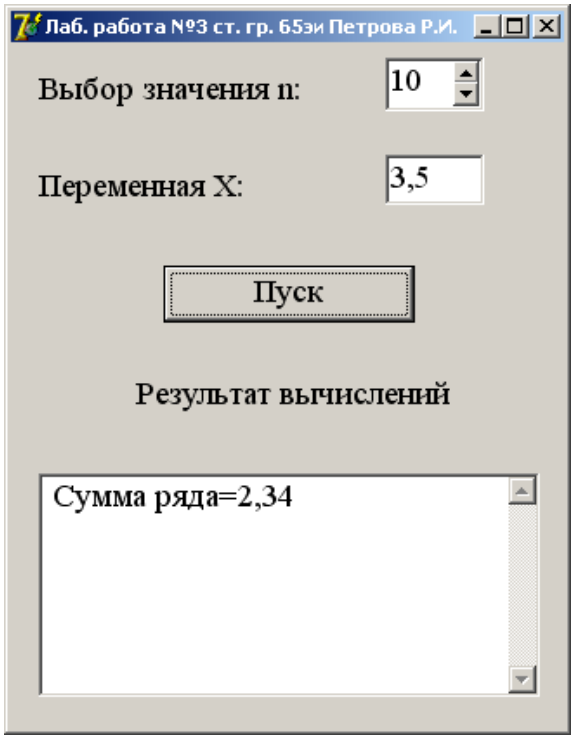

Рис. 1.8. Интерфейс приложения после его выполнения

Текст программы приведен в листинге 1.3.

Листинг 1.3

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,  Dialogs, StdCtrls, Spin;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    SpinEdit1: TSpinEdit;
    Label2: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Memo1: TMemo;
    Label3: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
begin
   SpinEdit1.Value:=10;
   Edit1.Text:='3,5'
end;
procedure TForm1.Button1Click(Sender: TObject);
var
   n, i: integer;
   x, s: real;
```



```
begin
n:=SpinEdit1.Value;
x:=StrToFloat(Edit1.Text);
s:=0;
for i:=1 to n do
 s:=s+(x/(n+i));
Memo1.Clear;
Memo1.Lines.Add('Сумма ряда='+FloatToStrF(s,ffFixed,7,2))
end;
end.
```

3.2. Индивидуальные задания

По указанию преподавателя выберите свое индивидуальное задание. Создайте приложение и протестируйте его работу.

Индивидуальные задания 1-го уровня

1. Дано натуральное *n*. Найти сумму и произведение четных чисел от 1 до *n*.

2. Дано натуральное *n*. Вычислить сумму квадратов первых *n* натуральных чисел.

3. Дано натуральное *n*. Найти сумму членов ряда:

$$s = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + ... + \frac{1}{n}.$$

4. Дано натуральное *n*. Найти сумму членов ряда:

$$S = \frac{1}{1+2} + \frac{1}{2+3} + \frac{1}{3+4} + ... + \frac{1}{n+(n+1)}.$$

5. Дано натуральное *n*. Найти сумму членов ряда:

$$S = 1 + \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + ... + \frac{n}{n+1}.$$

7. Дано натуральное *n*. Найдите сумму s и произведение *p* нечетных чисел от 1 до *n*.

8. Дано натуральное *n*. Найдите сумму *s* и произведение *p* чисел кратных 3 от 1 до *n*.

10. Дано натуральное *n*. Найдите сумму кубов первых *n* натуральных чисел.

Индивидуальные задания 2-го уровня

11. Дано натуральное n, действительное a. Найти сумму членов ряда: $s = \frac{1}{a} + \frac{1}{a \cdot (a+1)} + ... + \frac{1}{a \cdot (a+1) \cdot ... \cdot (a+n)}$.

12. Дано натуральное n. Найти сумму членов ряда:
$s = \frac{1}{3^2} + \frac{1}{5^5} + ... \frac{1}{(2n+1)^2}$.

13. Дано натуральное n. Найти сумму членов ряда:
$y = \frac{1}{1 \cdot 2} - \frac{1}{2 \cdot 3} + ... + \frac{(-1)^{n+1}}{n \cdot (n+1)}$.

14. Дано натуральное n. Найти сумму членов ряда:
$s = 1 \cdot 2 \cdot 3 + 2 \cdot 3 \cdot 4 + ... + (n-1) \cdot n \cdot (n+1)$.

15. Дано натуральное n, действительное a. Найти произведение членов ряда: $p = a \cdot (a-n) \cdot (a-2n) \cdot ... \cdot (a-n \cdot n)$.

16. Дано натуральное n, действительное x. Найти сумму членов ряда: $s = x - \frac{x^2}{2} + \frac{x^2}{3} - \frac{x^2}{4} + ... + (-1)^{n-1} \frac{x^2}{n}$.

17. Дано натуральное число n. Найти сумму членов ряда:
$s = -\frac{1}{3} + \frac{1}{5} - ... + \frac{(-1)^n}{2n+1}$.

18. Дано действительное x. Найти сумму членов ряда:
$y = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \frac{x^6}{6} + \frac{x^7}{7} - \frac{x^8}{8} + \frac{x^9}{9} - \frac{x^{10}}{10}$.

19. Дано действительное x. Найти сумму членов ряда:
$y = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \frac{x^{11}}{11}$.

20. Найти сумму членов ряда:

$z = \frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} + ... + \frac{1}{10 \cdot 11}$.

21. Дано действительное x. Найти сумму членов ряда:
$z = \frac{\cos(2 \cdot x)}{1 \cdot 3} + \frac{\cos(4 \cdot x)}{3 \cdot 5} + ... + \frac{\cos(20 \cdot x)}{19 \cdot 21}$.

22. Дано натуральное n. Найти сумму членов ряда:
$s = \frac{1}{1 \cdot 2} - \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} - \frac{1}{4 \cdot 5} + ... + \frac{(-1)^{n+1}}{n \cdot (n+1)}$.

23. Дано натуральное n. Найти сумму членов ряда:
$s = \frac{1}{2} - \frac{1}{4} + \frac{1}{6} - \frac{1}{8} + ... + \frac{(-1)^{n+1}}{2 \cdot n}$.

24. Дано натуральное n. Найти сумму членов ряда:
$s = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + ... + \frac{(-1)^{n+1}}{n}$.

25. Дано натуральное n. Найти сумму членов ряда:
$s = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + ... + \frac{(-1)^{n+1}}{2 \cdot n - 1}$.

Индивидуальные задания 3-го уровня

26. Дано натуральное n. Найти сумму членов ряда:
$s = n + n^2 + n^3 + n^4 + ... + n^n$.

27. Дано натуральное n. Найти произведение членов ряда:
$p = \frac{\cos 1}{\sin 1} \cdot \frac{\cos 1 + \cos 2}{\sin 1 + \sin 2} \cdot ... \cdot \frac{\cos 1 + ... + \cos n}{\sin 1 + ... + \sin n}$.

28. Дано натуральное n. Найти сумму членов ряда:
$s = \frac{1}{\sin 1} + \frac{1}{\sin 1 + \sin 2} + ... + \frac{1}{\sin 1 + ... + \sin n}$.

29. Дано натуральное n, действительное x. Найти произведение членов ряда:

$$p = \left(1 + \frac{\sin x}{1}\right) \cdot \left(1 + \frac{\sin(2 \cdot x)}{1 \cdot 2}\right) \cdot \ldots \cdot \left(1 + \frac{\sin(n \cdot x)}{1 \cdot 2 \cdot \ldots \cdot n}\right).$$

31. Дано натуральное n, действительное a. Найти сумму членов ряда: $s = \frac{1}{a} + \frac{1}{a^2} + \frac{1}{a^4} + \frac{1}{a^6} + \ldots + \frac{1}{a^{2n}}$.

Вопросы для самоконтроля

1. Какие операторы позволяют создавать циклические алгоритмы в языке Delphi?
2. Какие форматы оператора цикла **for** вы знете, в чем их отличие?
3. Опишите порядок выполнения операторов **for, while, repeat**. Для каждого оператора приведите формат и примеры.
4. На какой странице Палитры компонентов размещен компонент ***SpinEdit***?
5. Каково назначение компонента ***SpinEdit***?

ПРИМЕРЫ РАЗНОУРОВНЕВЫХ ЗАДАНИЙ ДЛЯ КОНТРОЛЯ ЗНАНИЙ

Задания 1-го уровня

1. В массиве имеется информация о стоимости 20 видов товара. Определить, сколько видов товара имеют максимальную стоимость.
2. Дано натуральное n, целые a_1, a_2, ..., a_n. Определить, сколько в этой последовательности нечетных отрицательных чисел.
3. Дан массив a_1, a_2, ..., a_n. Определить среднее арифметическое положительных элементов этого массива.
4. Дано натуральное n и таблица действительных чисел a_1, ..., a_n. Вычислить сумму : $s = a_1 + \frac{a_2}{2^3} + \frac{a_3}{3^3} + \ldots + \frac{a_n}{n^3}$.
5. Дано натуральное n, целые a_1, a_2, ..., a_n. Определить, сколько в этой последовательности положительных чисел.

Задания 2-го уровня

6. Дано натуральное n и таблица действительных чисел a_1, ..., a_n. Определить минимальное значение из этих чисел с нечетными номерами.
7. Дан массив a_1, a_2, ..., a_n. Подсчитать количество пар, состоящих из двух рядом стоящих элементов, имеющих одинаковые значения, но разные знаки.
8. Дано натуральное n и таблица действительных чисел $b(n)$, элементы таблицы не повторяются. Найти минимальный элемент и его порядковый номер среди отрицательных элементов.
9. Дано натуральное n и таблица действительных чисел a_1, a_2, ..., a_n. В этой таблице найти значения наименьшего и наибольшего элементов и их индексов.
10. Дана таблица действительных чисел d_1, d_2, …, d_{10}, числа таблицы не повторяются. Найти максимальное число и его порядковый номер среди положительных чисел таблицы.
11. Дано натуральное n и таблица целых чисел a_1, a_2, ..., a_n. Подсчитать количество пар, состоящих из двух рядом стоящих элементов, имеющих одинаковые значения, но разные знаки.

12. Дано натуральное n и таблица действительных чисел a_1, a_2, ..., a_n. Определить номер наименьшего элемента. Если таких элементов несколько, то найти номер первого из них.

13. В области 10 районов. Заданы площади, засеваемые пшеницей, и средняя урожайность в каждом районе. Определить количество пшеницы, собранное в области.

14. Дан массив $a(40)$, элементы в массиве могут повторяться. Найти первый минимальный и первый максимальный элементы массива.

15. Даны натуральное n и действительные числа a_1, ..., a_n. Вычислить сумму $s = a_1 - 2 \cdot a_2 + 3 \cdot a_3 - 4 \cdot a_4 + ... + n \cdot a_n$.

Задания 3-го уровня

16. Найти сумму ряда: $\sum_{i=1}^{100} \sum_{j=1}^{60} \sin(i^3 + j^4)$.

17. Дано натуральное n. Найти сумму ряда:

$$y = \sqrt{7 + \sqrt{14 + \sqrt{21 + ... + \sqrt{7 * n}}}} .$$

18. Вычислить сумму ряда: $\sum_{k=1}^{10} k^3 \sum_{l=1}^{15} (k - l)^2$.

19. Дано натуральное n. Вычислить

$$y = \sqrt{1 \cdot 2 + \sqrt{2 \cdot 3 + ... + \sqrt{(n-1) \cdot n}}} .$$

20. Найти сумму ряда: $\sum_{k=1}^{\infty} \frac{(-1)^k}{k!(k+1)!} \left(\frac{x}{2}\right)^{2k+1}$.

ВАРИАНТЫ ТЕСТОВЫХ ЗАДАНИЙ ПО МОДУЛЮ 1

1. Для чего предназначено окно Инспектора объектов?

a) для отображения на нем и изменения свойств выбранных компонентов и событий, на которые будет реагировать эта форма или ее компоненты;

b) для изменения свойств выбранных компонентов и событий, на которые будет реагировать эта форма или ее компоненты, создания обработчиков событий;

c) для декоративных целей;

d) проектируемого Windows-приложения.

2. Какое расширение имеет файл проекта в Delphi?

a) dpr;

b) pas;

c) dfm;

d) exe.

3. Выполнение какой операции обеспечивает функция *StrToFloat*?

a) обеспечивает преобразование строкового изображения числа в его значение действительного типа;

b) обеспечивает преобразование действительного числа в его символьное изображение;

c) обеспечивает преобразование действительного числа в его двоичное представление.

4. Первоначальный код головной программы и отдельных модулей создается:

a) автоматически системой Delphi;

b) разработчиком программы;

c) автоматически системой Delphi, далее он недоступен для редактирования разработчиком.

5. Совпадает ли имя проекта с именем файла, в котором он сохранен?

a) имя проекта совпадает с именем файла, в котором он сохранен;

b) нет;
c) не обязательно.

6. Может ли вводиться комментарий в тексте программы с помощью сочетаний символов «(*» и «*)»?

a) да;
b) нет;
c) с разрешения преподавателя.

7. Какую информацию содержит файл проекта в среде Delphi?

a) содержит исходный код главной подпрограммы;
b) информацию об объектах формы и их свойствах;
c) процедуры обработки событий, связанных с нажатием кнопок.

8. Какую информацию содержит файл формы в среде Delphi?

a) в нем перечислены объекты формы и значения свойств этих объектов;
b) информацию о формах компонент и их свойствах;
c) перечислены все параметры компонентов и их свойства.

9. Какую информацию содержит файл модуля в среде Delphi?

a) исходный код модуля, ассоциированного с одноименной формой;
b) процедуры обработки событий, связанных с нажатием кнопок, и описания глобальных переменных;
c) информацию о формах объектов, компонент и их свойствах.

10. Для создания несложных пользовательских интерфейсов чаще всего используются следующие простые компоненты Delphi: метки, поля ввода, области просмотра, кнопки. На какой странице Палитры компонентов они находятся?

a) во вкладке Samples Палитры компонентов;
b) во вкладке Standard Палитры компонентов;
c) во вкладке Additional Палитры компонентов.

МОДУЛЬ 2

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ И ПЕРЕМЕННЫХ СТРОКОВОГО ТИПА

В результате изучения модуля студент должен:

знать описания операторов обработки массивов и переменных строкового типа, их свойства и методы обработки информации; свойства и методы компонентов ComboBox, ListBox и др.;

уметь применять полученные теоретические и практические знания для программирования с применением многомерных массивов данных и переменных строкового типа в задачах экономики и управления на предприятии, в которых предусматривается обработка цифровых и символьных данных, представленных, например, в табличной форме.

Теоретические сведения

1. Основы объектно-ориентированного программирования

Любое приложение, написанное на языке Object Pascal, может иметь в своем составе ***объекты*** и ***классы***.

Класс – это тип данных, который включает в себя данные и операции над ними. ***Объект*** – это экземпляр какого-либо класса. Таким образом, класс – это описание (тип), а объект – это то, что создано в соответствии с этим описанием.

До создания объектно-ориентированных языков программирования данные и операции над данными рассматривали как отдельные элементы.

В состав класса входят поля, свойства и методы. Типичное определение нового класса выглядит следующим образом:

```
type
  Имя класса = class(Класс-предок)
 {Список состава класса}
 private
  {Частные описания}
 protected
```

{Защищенные описания}
public
{Общедоступные описания}
published
{Опубликованные описания}
end;

где имя класса – любое корректное имя (выбирается произвольно); класс-предок – название класса, наследником которого является создаваемый класс; список состава класса содержит свойства и методы нового класса.

Для каждого элемента класса можно установить разную видимость. Для этого предназначены четыре ключевых слова: private, protected, public и published. Видимость элемента класса определяет, где в программе и как будет виден данный элемент класса. Минимальная видимость элемента класса задается ключевым словом private, ключевое слово protected определяет средний уровень видимости. Наконец, public и published определяют наивысшую степень доступности.

Если перед описанием элемента класса не ставится ключевое слово, определяющее его степень видимости, то считается, что видимость элемента такая же, как и у предыдущего элемента класса.

Объекты подобно записям являются хранилищами разных типов данных. Данные объекта называются ***полем*** (field) и аналогичны полям записи. Но объекты, в отличие от записей, содержат еще и процедуры и функции, которые применимы к полям данного объекта. Эти процедуры и функции называются ***методами*** (methods).

Изменять поля объекта можно при помощи ***свойств*** **(properties)** объекта. Каждое свойство объекта представляет собой поле и методы (так называемые методы доступа), которые позволяют считывать значение поля и задавать его. Свойства можно изменять в процессе разработки приложения с помощью Инспектора объектов.

К основным принципам объектно-ориентированного программирования относятся:

- инкапсуляция;
- наследование;
- полиморфизм.

Инкапсуляция – это объединение данных и обрабатывающих их методов внутри одного класса.

Наследование обозначает, что объекты могут получать свои свойства и методы от других объектов (которые называют в данном случае предками). Объекты-наследники берут от предков все свойства, методы и поля. Эти свойства, методы и поля в объекте-наследнике могут сохраняться либо в неизменном виде, либо в измененном. Кроме того, объекты-наследники могут иметь в своем составе дополнительно новые поля, методы или свойства.

Полиморфизм подразумевает, что методы различных объектов могут иметь одинаковые имена, но отличаться по своему содержанию. Это получается в результате переопределения метода объекта-предка в объекте-наследнике. При этом обращение к одному и тому же методу у объекта-предка и объекта-потомка может привести к разным результатам.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Материалы к лабораторной работе № 1

1. Программирование циклических алгоритмов. Обработка одномерных массивов

Цель работы: изучить средства организации циклических вычислительных процессов; овладеть практическими навыками работы с массивами; освоить применение компонента StringGrid для создания приложения, в котором используются массивы.

Обработка массивов

Массив – это структурированный тип данных, состоящий из фиксированного числа однотипных элементов. Массив имеет размеры, определяющие, сколько элементов хранится в структуре. Обращение к отдельным элементам массива производится по его индексу (порядковому номеру).

Если при описании массива задан один индекс, массив – одномерный, если два индекса – двухмерный, если n индексов – n-мерный. Одномерные массивы используются для представления векторов, двухмерные – для представления матриц.

Объявление массива

Перед использованием массив должен быть объявлен:

```
type
  имя_типа = array [тип_индексов] of тип_элементов;
var
  имя_массива: имя_типа;
```

Массивы могут быть определены непосредственно в разделе описания переменных следующим образом.

```
var
   имя_массива: array [тип_индексов] of тип_элементов;
```

Ниже показаны примеры определений массивов.

```
type
  klass = array [1..30] of  integer;       //массив из 30 целых чисел
```



```
var
  mas1: klass;
  mas2, mas3: array [1..5] of extended; //массив из 5 вещественных чисел
```

Элементы массива хранятся в памяти последовательно друг за другом. При обращении к конкретному элементу массива его индекс записывается в квадратных скобках непосредственного после имени массива. Например, элементы массива mas1 могут обозначаться так: *mas1*[1], *mas1*[2], …, *mas1*[30].

Элементы массива могут использоваться в выражениях так же, как и переменные других типов.

```
a:=b+mas1[i+1];
mas2[j]:=sqr(mas3[i]);
```

Если массивы имеют идентичный тип, то к ним допускается применять операцию присваивания.

```
mas2:=mas3;
```

Начиная с Delphi 4, введена возможность использовать ***динамические массивы*** – массивы, количество элементов которых задается в процессе выполнения программы. Такие массивы удобно использовать для хранения данных в тех задачах, где заране неизвестно количество элементов в обрабатываемых массивах.

Задание размера массива и выделение для него памяти выполняется с помощью процедуры

SetLength(*имя_динамического_массива, длина****).***

Аргумент *длина* процедуры должен быть целого типа.

Основные алгоритмы для работы с массивами

К типичным действиям с массивами можно отнести – поиск в массиве заданного элемента; поиск в массиве максимального или минимального элементов; сортировка массива.

Поиск в массиве

При решении многих задач возникает необходимость установить, содержит ли массив определенную информацию или нет. Наиболее простой алгоритм для организации поиска в массиве – это алгоритм простого перебора. Поиск осуществляется последовательным сравнением элементов массива с образцом до тех пор, по-

ка не будет найден элемент, равный образцу, или не будут проверены все элементы. Алгоритм простого перебора применяется, если элементы массива не упорядочены.

Часто такие задачи дополняются требованием определить количество найденных совпадений с образцом, либо определить сумму, произведение или среднее арифметическое элементов, совпадающих с образцом.

Пример 1

Найти сумму положительных и количество отрицательных элементов массива *a*[1..15].

```
procedure TForm1.Button1Click(Sender: TObject);
var  a:array[1..15] of integer;          //объявление массива a
     s, i, k: integer;                   //k-количество нулевых элементов
begin
 {Ввод исходных данных в массив a}
 for  i:=1 to 15 do
   a[i]:=StrToInt(StringGrid1.Cells[i-1,0]);
 {Определение суммы положительных и количества отрицательных элементов массива}
  k:=0;  s:=0;
 for  i:=1 to 15 do
   if  a[i]>=0 then s:=s+a[i] else k:=k+1;
 {Вывод суммы( s) и количества(k)}
 Memo1.Lines.Add('Сумма положительных элементов s = ' +
 IntToStr(s)+  '  Кол-во  отрицательных  элементов  k  =  '
+IntToStr(k));
end;
```

Поиск в массиве минимального (максимального) элемента

Алгоритм поиска минимального (максимального) элемента массива: делается предположение, что первый элемент массива является минимальным (максимальным), затем остальные элементы массива сравниваются с этим элементом. Если обнаруживается, что проверяемый элемент меньше (больше) принятого за минимальный (максимальный), то этот элемент принимается за минимальный (максимальный) и продолжается проверка оставшихся элементов.

Пример 2

Дан массив *a*[1..5]. Найти максимальный элемент массива и его порядковый номер. Предполагается, что такой элемент единственный.

```
procedure TForm1.Button1Click(Sender: TObject);
var    a: array[1..5] of real;
       i, nmax: integer;
       max: real;
begin
{Ввод исходных данных в массив a}
 for  i:=1 to 5 do
   a[i]:=StrToFloat(StringGrid1.Cells[i-1,0]);
{Поиск макс. элемента и его порядкового номера}
 max:=a[1];
 nmax:=1;
 for  i:=2 to 5 do
  if  a[i]>max then
    begin
      max:=a[i];
      nmax:=i;
    end;
 Memo1.Lines.Add('max='+FloatToStrF(max,ffFixed,8,3)+
                     ' nmax='+IntToStr(nmax));
end;
```

Сортировка массива

Под ***сортировкой массива*** понимается процесс перестановки элементов с целью упорядочивания их в соответствии с каким-либо критерием. Например, если имеется массив целых *a*, то после сортировки по возрастанию должно выполняться условие: $a[1] \le a[2] \le \ldots \le a[n]$.

Рассмотрим сортировку методом прямого обмена или методом «пузырька». В основе данного алгоритма лежит обмен соседних элементов массива. Каждый элемент массива, начиная с первого, сравнивается со следующим. Если он больше следующего, то элементы меняются местами. Таким образом, элементы с меньшим значением продвигаются к началу массива, а элементы с большим

значением – к концу массива, поэтому этот метод называют методом «пузырька». Этот процесс повторяется на единицу меньше раз, чем элементов в массиве.

Пример 3

Составить программу для сортировки элементов массива *b*[1..15] по возрастанию.

```
procedure TForm1.Button1Click(Sender: TObject);
const  n=15;
var  b:array[1..n] of integer;                    //объявление массива b
       i, j:integer;     // i - текущий индекс элемента массива ,
                         // j-счетчик циклов
       buf:integer;
begin
 {Ввод исходных данных в массив b}
 for  i:=1 to n do
   b[i]:=StrToInt(StringGrid1.Cells[i-1,0]);
 {Сортировка методом «пузырька»}
 for  i:=2 to n do
  for  j:=n downto i do
   if  b[j-1]>b[j]
     then begin
        {Обменяем (j-1)-й и j-й элементы}
         buf:=b[j-1];
         b[j-1]:=b[j];
         b[j]:=buf;
       end;
  {Вывод на экран отсортированного массива}
  for  i:=1 to n do
     StringGrid1.Cells[i-1,0]:=IntToStr(b[i]);
end;
```

Применение компонента StringGrid для работы с массивами

При работе с массивами ввод и вывод информации на экран удобно организовывать с помощью компонента ***StringGrid***.

Пиктограмма  компонента ***StringGrid*** находится на странице ***Additional*** Палитры компонентов.

Компонент ***StringGrid*** используется для отображения информации в виде таблицы. Таблица содержит две зоны – фиксированную и рабочую. ***Фиксированная зона*** служит для вывода наименований строк и столбцов рабочей зоны и управления их размерами с помощью «мыши». Фиксированная зона выделена другим цветом, в нее запрещен ввод информации с клавиатуры. Количество строк и столбцов фиксированной зоны устанавливается в свойствах ***FixedRows*** и ***FixedCols*** соответственно. Если фиксированная зона не используется, в Инспекторе объектов значения свойств ***FixedRows*** и ***FixedCols*** устанавливаются равными 0.

Рабочая зона содержит ***RowCount*** строк и ***ColCount*** столбцов информации.

Нумерация строк и столбцов таблицы ***начинается с нуля***. Доступ к информации в таблице осуществляется с помощью свойства ***Cells [ACol, ARow: integer]: string***. Координаты каждой ячейки таблицы задаются парой чисел, первое является номером столбца, второе – номером строки. Например, ячейка с координатами Cells [3,5] расположена в четвертом столбце и шестой строке.

По умолчанию в компонент ***StringGrid*** запрещен ввод информации с клавиатуры, поэтому для компонента ***StringGrid*** необходимо в Инспекторе объектов дважды щелкнуть «мышью» на символе + свойства + ***Option*** и в открывшемся списке опций установить значение ***goEditing*** в ***True***.

1.1. Пример создания приложения

Задание. Создать Windows-приложение, реализующее следующую задачу. В целочисленном массиве *a* найти максимальный и минимальный элементы массива (предполагается, что каждый из них единственный) и поменять их местами. Отредактированный массив вывести на экран.

1.1.1. Размещение компонентов на Форме

Один из возможных вариантов панели интерфейса создаваемого приложения показан на рисунке 2.1.

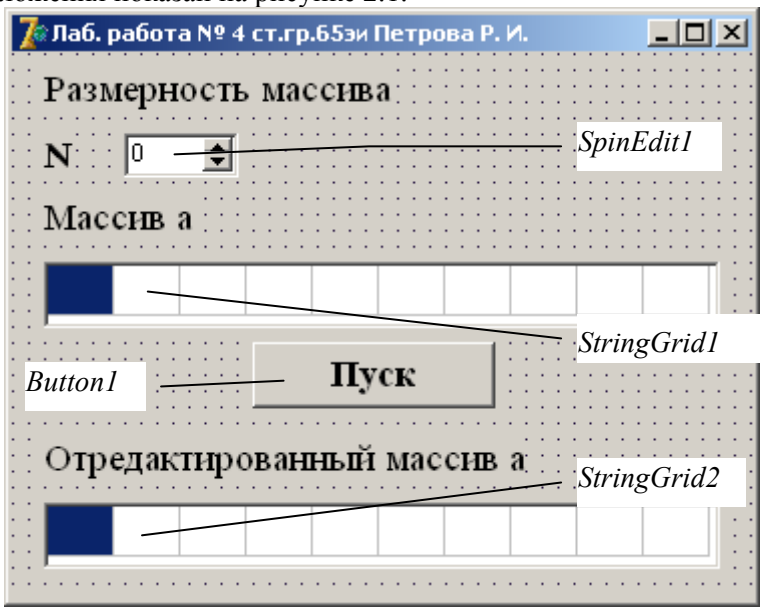

Рис. 2.1. Размещение компонентов на Форме

Компонент ***SpinEdit1*** находится на странице ***Samples*** Палитры компонентов, компоненты ***StringGrid*** – на странице ***Additional.***

Для задания размерности массива удобно использовать компонент ***SpinEdit.*** Компонент ***SpinEdit*** предназначен для отображения и редактирования целого числа. Установите для компонента ***SpinEdit1*** значения свойств: ***MinValue****=1,* ***MaxValue****=10*.

С помощью Инспектора объектов свойствам компонентов ***StringGrid1*** и ***StringGrid2*** задайте следующие значения: ***FixedCols*** – 0, ***FixedRows*** – 0, ***RowCount*** – 1, ***ColCount*** – 10 *(предельное значение количества элементов массива)*, ***DefaultColWidth*** – 32.

Для ввода исходных данных в таблицу ***StringGrid1***, выберете в Инспекторе объектов для компонента ***StringGrid1*** свойство ***Option,*** дважды щелкните «мышью» на символе + и в открывшемся списке опций установить значение ***goEditing*** в ***True.***

1.1.2. Сохранение проекта

Для нового проекта создайте новую папку, например, ***X:\35эи\ \Mod2\Lab1***.

Сохраните проект ***File | Save Project As….*** Сначала сохраните модуль с именем **UnMas.pas**, затем файл проекта под именем **PrMas.dpr.**

Последующие сохранения выполнять командами ***File | Save All.***

1.1.3. Создание процедуры обработки события FormCreate

Создайте процедуру *FormCreate,* наберите текст процедуры ***procedure TForm1.FormCreate(Sender: TObject).***

1.1.4. Создание процедуры обработки события SpinEdit1Change

Событие *SpinEdit1Change* возникает при любом изменении значения в поле редактора ***SpinEdit1***. Создадим процедуру обработки этого события, в котором присвоим значение ***n***, полученное из поля редактора ***SpinEdit1***, свойству ***ColCount*** компонентов ***StringGrid.*** Это позволит управлять размерами таблиц ***StringGrid*** с помощью компонента ***SpinEdit1.*** Изменение значений в поле редактора ***SpinEdit1*** сразу приведет к изменению размера таблиц ***StringGrid.*** Дважды щелкните «мышью» на компоненте ***SpinEdit1*** – курсор установится в тексте процедуры-обработчика события ***SpinEdit1Change.*** Наберите операторы этой процедуры, используя текст модуля ***UnMas.***

1.1.5. Создание процедуры обработки события нажатия кнопки Button1 (Button1Click)

Создайте процедуру обработки события нажатия кнопки «Пуск» – ***procedure TForm1.Button1Click(Sender: TObject)*** и наберите текст этой процедуры.

1.1.6. Работа с приложением

Запустите созданное приложение, выберите размерность массива *a* (например, 10), введите числовые значения элементов массива *a* и убедитесь, что приложение функционирует в соответствии с заданием.

На рисунке 2.2 показан интерфейс приложения после ввода исходных данных и нажатия кнопки «Пуск».

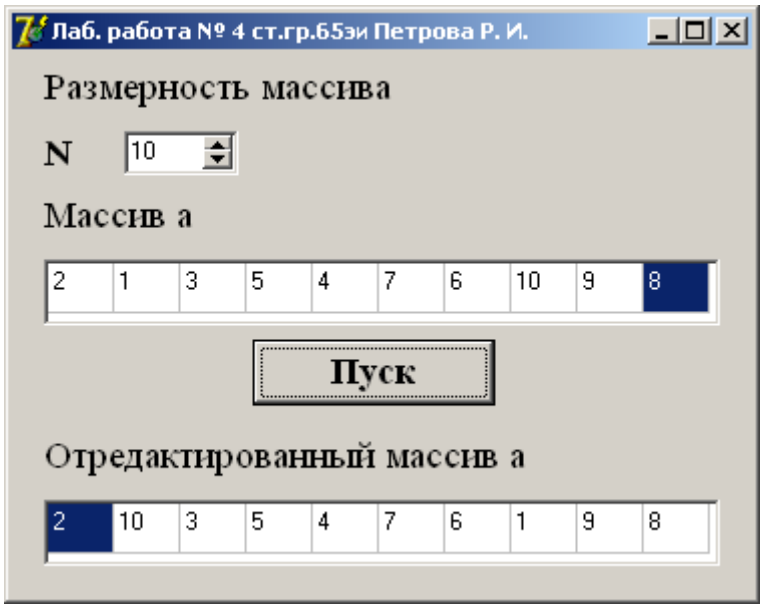

Рис. 2.2. Интерфейс приложения после его выполнения

Текст программы приведен в листинге 2.1.

Листинг 2.1

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, Grids, Spin;
type
  TForm1 = class(TForm)
    SpinEdit1: TSpinEdit;
    Label1: TLabel;
    StringGrid2: TStringGrid;
    Button1: TButton;
    Label2: TLabel;
    StringGrid1: TStringGrid;
    Label3: TLabel;
```



```
    Label4: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
var
  a:array[1..10] of integer;              //объявление массива а
  n:integer;                          //n – кол-во элементов в массиве а
procedure TForm1.FormCreate(Sender: TObject);
  begin
   SpinEdit1.Text:='6';                          //начальное значение n
    StringGrid1.ColCount:=6;      //количество столбцов массива а
    StringGrid2.ColCount:=6;
  end;
procedure TForm1.SpinEdit1Change(Sender: TObject);
  begin
    n:=StrToInt(SpinEdit1.Text);
    StringGrid1.ColCount:=n;
    StringGrid2.ColCount:=n;
  end;
procedure TForm1.Button1Click(Sender: TObject);
var
  i,max,imax,min,imin:integer; //объявление локальных переменных
  begin
    n:=StrToInt(SpinEdit1.Text);
    StringGrid1.ColCount:=n;
    StringGrid2.ColCount:=n;
{Заполнение массива А исходными данными }
    for i:=1 to n do
```

```
      a[i]:=StrToInt(StringGrid1.Cells[i-1,0]);
  {Нахождение мин. и макс. значений  и их порядковых номеров }
      max:=a[1]; imax:=1;
      min:=a[1]; imin:=1;
    for i:=2 to n do
      begin
          if a[i]>max then
   begin
    max:=a[i];
    imax:=i;
   end;
          if a[i]<min then
   begin
    min:=a[i];
    imin:=i;
   end;
  end;
  {Меняем местами минимум и максимум}
   a[imax]:=min;
   a[imin]:=max;
  {Вывод отредактированного массива в таблицу}
  for i:=1 to n do
   StringGrid2.Cells[i-1,0]:=IntToStr(a[i]);
 end;
 end.
```

1.2. Индивидуальные задания

По указанию преподавателя выберите индивидуальное задание желаемого уровня. Создайте приложение и протестируйте его работу.

Индивидуальные задания 1-го уровня

1. Массив $a_1, a_2, \ldots, a_n$ содержит экзаменационные оценки группы студентов по высшей математике. Определить средний балл группы по данной дисциплине.

2. Дан массив $d_1, d_2, \ldots, d_n$ действительных чисел. Определить произведение элементов массива.

3. Массив $c_1, c_2, \ldots, c_n$ – стипендия группы абитуриентов. В текущем месяце каждому студенту положена надбавка к стипендии в размере 50 000 рублей. Рассчитать окончательную сумму стипендии для каждого члена группы в текущем месяце.

4. Каждый элемент массива $c_1, c_2, \ldots, c_n$ увеличить в 3 раза.

5. Определить среднее арифметическое значение элементов массива $r_1, r_2, \ldots, r_n$.

6. Группа комбайнов выкопала соответственно $p_1, p_2, \ldots, p_n$ тонн картофеля. Определить суммарное количество картофеля, убранное всеми комбайнами.

7. Каждый элемент массива действительных чисел $m_1, m_2, \ldots, m_n$ уменьшить на 3 .

8. Определить среднее геометрическое значение элементов массива $p_1, p_2, \ldots, p_n$.

9. Дан массив $d_1, d_2, \ldots, d_n$ действительных чисел. Каждый элемент массива увеличить на величину первого элемента.

10. Имеются сведения о выработке электроэнергии одной из турбин электростанции по месяцам с января по декабрь: $r_1, r_2, \ldots, r_{12}$. Определить среднемесячную выработку электроэнергии турбиной.

Индивидуальные задания 2-го уровня

11. В массиве t содержатся результаты измерений температуры воздуха, которые ежедневно проводились в течение февраля. Определить, сколько раз в течение месяца температура меняла знак. Значения температур ввести с клавиатуры.

12. Имеется n шаров. На одних шарах нанесены отрицательные числа, на других – положительные. Определить количество шаров с отрицательными значениями и сумму положительных чисел на шарах.

13. Пусть дано натуральное число n и вещественные числа $a_1, a_2, \ldots, a_n$. В последовательности $a_1, a_2, \ldots, a_n$ все отрицательные члены возведите в квадрат, а все неотрицательные замените на 1. Полученный массив вывести на экран. Элементы массива и его размерность n задаются пользователем с клавиатуры.

14. Массив $v_1, v_2, \ldots, v_n$ – возраст сотрудников учреждения. Определить средний возраст сотрудников и количество сотрудников старше среднего возраста.

15. Массивы m и f содержат отметки, полученные студентами группы на экзаменах по математике и физике. Определить количество студентов, получивших отличные отметки по обеим дисциплинам.

16. Задан целочисленный массив b. Подсчитать количество элементов этого массива, которые совпадают со своим номером.

17. Имеется k клубней 1-го сорта весом $p_1, p_2, \ldots, p_k$ и n клубней 2-го сорта весом $r_1, r_2, \ldots, r_n$. Определить, клубни какого сорта в среднем тяжелее.

18. Создать приложение, которое осуществляет ввод k значений элементов одномерного массива с клавиатуры, изменяет порядок следования элементов на противоположный и выводит полученный массив.

19. Даны списки команд высшей лиги $a_1, a_2, \ldots, a_k$, количество очков, набранных соответственно каждой из команд $s_1, s_2, \ldots, s_k$. Напечатать список команд, набравших более r очков. Значение r задается пользователем.

20. Массив r содержит сведения о количестве студентов каждой группы I курса. Определить группу с максимальным количеством студентов, считая, что номер группы соответствует порядковому номеру числа в массиве (считая, что такая группа единственная).

21. Дан список сотрудников предприятия с указанием года рождения $r_1, r_2, \ldots, r_k$. Порядковый номер элемента массива соответствует табельному номеру сотрудника. Определить возраст самого молодого сотрудника и указать его табельный номер.

22. Пассажирский самолет может поднять груз общим весом r. Составить программу определения веса почтового груза, который можно поместить в самолет после посадки n пассажиров (условный вес одного человека 100 кг) и загрузки их багажа, составляющего $p_1, p_2, \ldots, p_n$ кг.

23. Даны координаты n точек (x_1, y_1), (x_2, y_2), …, (x_n, y_n). Определить количество точек, расположенных на оси ординат.

24. Верно ли, что отрицательных элементов последовательности $a_1, a_2, \ldots, a_n$ больше, чем положительных?

25. Даны координаты n точек (x_1, y_1), (x_2, y_2), …, (x_n, y_n). Определить количество точек, попадающих в круг радиусом r с центром в начале координат.

Индивидуальные задания 3-го уровня

26. Дана последовательность действительных чисел $a_1, a_2, \ldots, a_n$. Требуется домножить все элементы на квадрат ее наименьшего члена, если $a_1>=0$, и на квадрат ее наибольшего члена в обратном случае.

27. Даны действительные числа $a_1, a_2, \ldots, a_n$. Получить $b_1, b_2, \ldots, b_n$, где b_i равно сумме тех членов исходной последовательности, которые принадлежат интервалу $(i\text{-}1, i)$ $(i = 1, 2, \ldots, n)$. Если интервал не содержит членов последовательности, то соответствующее b_i положить равным нулю.

28. Даны целые числа $a_1, a_2, \ldots, a_n$. Наименьший член последовательности заменить целой частью среднего арифметического всех элементов, остальные элементы оставить без изменения.

29. Даны две последовательности целых чисел $a_1, a_2, \ldots, a_n$ и $b_1, b_2, \ldots, b_n$. В каждой последовательности числа не повторяются. Построить пересечение последовательностей (т. е. получить все числа, входящие в обе последовательности одновременно: $c_1, c_2, \ldots, c_m$).

30. Дана последовательность целых чисел $a_1, a_2, \ldots, a_n$. Найти наибольшее и наименьшее из них (m и n), а так же получить в порядке возрастания все целые числа из интервала (m, n), которые не входят в данную последовательность.

Вопросы для самоконтроля

1. Что такое массив?
2. Как описываются массивы в языке Delphi?
3. На какой странице Палитры компонентов размещен компонент ***StringGrid***?
4. Что обозначают свойства ***FixedRows*** и ***FixedCols*** компонента ***StringGrid***?
5. Каким образом снять блокировку ввода данных в компонент ***StringGrid***, принятую по умолчанию?

Материалы к лабораторной работе № 2

Программирование циклических вычислительных процессов. Обработка двухмерных массивов

Цель работы: овладение практическими навыками работы с многомерными массивами: поиск, сортировка, отбор по значению.

Обработка двухмерных массивов (матриц)

Описание двухмерного массива имеет вид:

var

имя_массива: **array** [тип_индексов1, тип_индексов2] **of** тип_элементов;

Ниже представлен пример описания двухмерного массива:

matr: **array** [1..5,1..10] **of** integer;

Двухмерный массив представляет собой матрицу из 5 строк и 10 столбцов с элементами целого типа.

При использовании элемента массива нужно указать имя массива и индексы элемента. Каждый элемент матрицы имеет два индекса, значения которых позволяют указать местоположение элемента (его координаты).

Первый индекс – это номер строки матрицы, второй – номер столбца. Например, запись *matr*[4, 3] делает доступным для обработки значение элемента, находящегося в четвертой строке третьего столбца матрицы *matr*.

Базовыми алгоритмами при работе с двухмерными массивами являются: вычисление суммы элементов массива (см. пример 1); вычисление суммы элементов каждой строки или каждого столбца (пример 2); поиск максимального (минимального) элемента массива (пример 3).

Пример 1. Вычисление суммы элементов всего двухмерного массива.

Вычислить сумму элементов двухмерного массива *a*(*n*, *m*).


```
procedure TForm1.Button1Click(Sender: TObject);
const n=4;
      m=3;
var   a :array[1..n,1..m] of integer;
      i, j, sum :integer;
begin
{Ввод исходных данных  в массив a}
 for  i:=1 to n do
  for  j:=1 to m do
    a[i,j]:=StrToInt(StringGrid1.Cells[j-1,i-1]);
{Вычисление суммы элементов}
  sum:=0;
 for  i:=1 to n do
  for  j:=1 to m do
    sum:=sum+a[i,j];
{Вывод  результата}
Memo1.Lines.Add('sum=' + IntToStr(sum));
end;
```

Пример 2. Вычисление суммы элементов каждой строки (столбца).

Задана целочисленная матрица *a*(*n*, *m*). Вычислить сумму элементов каждого столбца матрицы.

```
procedure TForm1.Button1Click(Sender: TObject);
const n=4;
      m=3;
var a:array[1..n,1..m] of integer;
    i, j, sum: integer;
begin
 {Ввод исходных данных  в массив a}
 for  i:=1 to n do
  for  j:=1 to m do
   a[i,j]:=StrToInt(StringGrid1.Cells[j-1,i-1]);
{Вычисление суммы элементов каждого столбца}
 for  j:=1 to m do
```

```
    begin
      sum:=0;
       for i:=1 to n do
           sum:=sum+a[i,j];
       Memo1.Lines.Add ('Сумма элементов  ' + IntToStr(j) + ' столбца ='+IntToStr(sum));
    end;
  end;
```

Пример 3. Поиск максимального (минимального) элемента и его индексов.

Задана вещественная матрица *b* (*n*, *m*). Для каждой строки матрицы найти наибольший элемент и его индексы.

```
  procedure TForm1.Button1Click(Sender: TObject);
  const n =4;
        m =3;
  var  b: array[1..n,1..m] of extended;
       i, j, j_min: integer;
       min :extended;
  begin
  {Ввод исходных данных  в массив B}
   for i:=1 to n do
    for j:=1 to m do
      b[i,j]:=StrToFloat(StringGrid1.Cells[j-1,i-1]);
  {Поиск мин.элемента}
   for i:=1 to n do
     begin
        min:=b[i,1];
        j_min:=1;
        for j:=2 to m do
          if  b[i,j] < min  then
            begin
              min:=b[i,j];
              j_min:=j;
```



```
            end;
        Memo1.Lines.Add('min = '+FloatToStrF(Min,ffFixed,5,2);
        Memo1.Lines.Add('i_min='+IntToStr(i)+
                        ' j_min='+IntToStr(j_min));
     end;
  end;
```

Квадратные матрицы

Матрица, в которой количество строк совпадает с количеством столбцов, называется ***квадратной***.

```
  a: array [1..4,1..4] of extended;
```

Для квадратных матриц характерны понятия ***главная и побочная диагональ***. Главная диагональ – элементы a_{11}, a_{22}, a_{33}, …, a_{nn}. Индексы элементов, расположенных на главной диагонали, ***i = j***. Побочная диагональ – элементы a_{41}, a_{32}, a_{23}, a_{14}. Сумма индексов элементов на 1 больше размерности строки (или столбца), т.е. ***I + j = n + 1.***

Для индексов элементов, расположенных над главной диагональю, выполняется отношение ***i < j.***

Для индексов элементов, расположенных под главной диагональю, выполняется отношение ***i > j.***

2.1. Пример создания приложения

Задание. Создать Windows-приложение, реализующее следующую задачу. Задана целочисленная матрица *a* размером *n*×*m*. Получить последовательность $t_1, t_2, \ldots, t_m$ *j*-ый элемент которой получает значение 0, если все элементы *j*-го столбца матрицы *a* имеют значение 0, и значение 1 в противном случае.

2.1.1. Размещение компонентов на Форме

Один из возможных вариантов панели интерфейса создаваемого приложения показан на рисунке 2.3.

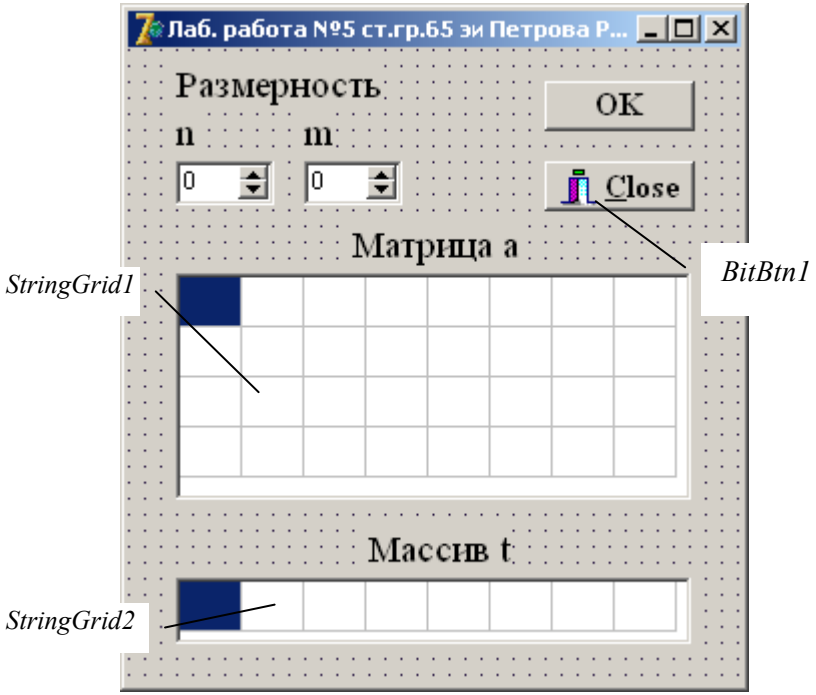

Рис. 2.3. Размещение компонентов на Форме

Компонент ***SpinEdit*** находится на странице ***Samples*** Палитры компонентов, компонент ***StringGrid*** – на странице ***Additional.*** Компонент ***BitBtn*** расположен на странице ***Additiol*** Палитры компонентов и представляет собой разновидность стандартной кнопки ***Button***. Его отличительная особенность – наличие растрового изображения на поверхности кнопки, которое определяется свойством ***Glyph.*** Для кнопки ***BitBtn*** имеется свойство ***Kind***, которое задает одну из 11 стандартных кнопок. Нажатие любой из них, кроме *bkCustom* и *bkHelp*, закрывает модальное окно. Кнопка *bkClose* закрывает главное окно и завершает работу программы.

Для свойств компонентов ***StringGrid1*** и ***StringGrid2*** установите следующие значения ***FixedCols*** – 0, ***FixedRows*** – 0, ***ColCount*** – 8 *(количество столбцов)*, ***DefaultColWidth*** – 30. Свойство ***RowCount*** *(количество строк)* для ***StringGrid1*** установите равным 4, а для ***StringGrid2*** – 1. Для ввода исходных данных в таблицу ***StringGrid1***

– в Инспекторе объектов выберите свойство ***Option*** и установите значение флага *goEditing* в *True.*

Для компонента ***SpinEdit1*** установите значения свойств ***MinValue –1, MaxValue*** – *6,* для компонента ***SpinEdit2***: ***MinValue*** – 1, ***MaxValue*** – 8.

2.1.2. Сохранение проекта

Для нового проекта создайте новую папку, например, ***X:\65эи\ФИО_студента\Mod2\Lab2.*** Сохраните проект ***File | Save Project As….*** Сначала сохраните модуль с именем ***UnMatrix.pas***, затем файл проекта под именем ***PrMatrix.dpr.***

Последующие сохранения выполнять командами ***File | Save All.***

2.1.3. Создание процедур обработки событий

Создайте процедуры обработки событий ***FormCreate, SpinEdit1, SpinEdit2, Button1Click*** и наберите тексты этих процедур, используя листинг 2.2.

Листинг 2.2

```
unit UnMatrix;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, Grids, Spin, Buttons;
type
  TForm1 = class(TForm)
    SpinEdit1: TSpinEdit;
    SpinEdit2: TSpinEdit;
    Label1: TLabel;
    Label2: TLabel;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    Button1: TButton;
    Label5: TLabel;
    Label4: TLabel;
    Label3: TLabel;
```

```
    BitBtn1: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure SpinEdit2Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
var
a: array[1..6,1..8] of integer;                    //объявление массива A
t: array[1..8] of integer;                         //объявление массива T
n, m: integer;                          //объявление глобальных переменных
procedure TForm1.FormCreate(Sender: TObject);
begin
SpinEdit1.Text:='4';                    //начальное значение n
SpinEdit2.Text:='8';                    //начальное значение m
StringGrid1.RowCount:=4;  //количество строк массива a
StringGrid1.ColCount:=8;    //количество столбцов массива a
StringGrid2.ColCount:=8;    //количество столбцов массива t
end;
procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
n:=StrToInt(SpinEdit1.Text);
StringGrid1.RowCount:=n;     //количество строк массива a
end;
procedure TForm1.SpinEdit2Change(Sender: TObject);
begin
m:=StrToInt(SpinEdit2.Text);  //m присваивается содержимое поля редактора
StringGrid1.ColCount:=m;       //количество столбцов массива a
```



```
StringGrid2.ColCount:=m;       //количество столбцов массива t
end;
procedure TForm1.Button1Click(Sender: TObject);
var i, j, fl:integer;                    //объявление локальных переменных
begin
 n:=StrToInt(SpinEdit1.Text);
 StringGrid1.RowCount:=n;
 m:=StrToInt(SpinEdit2.Text);
 StringGrid1.ColCount:=m;
 StringGrid2.ColCount:=m;
 {Ввод значений из таблицы в массив a}
 for i:=1 to n do
  for j:=1 to m do
   a[i,j]:=StrToInt(StringGrid1.Cells[j-1,i-1]);
  {Формирование массива t и вывод его значений в таблицу}
 for j:=1 to m do
  begin
   fl:=0;
   for i:=1 to n do
    if a[i,j]<>0 then fl:=1;
   if fl=0 then t[j]:=0
          else t[j]:=1;
  end;
 for j:=1 to m do
   StringGrid2.Cells[j-1,0]:=IntToStr(t[j]);
end;
end.
```

2.1.4. Работа с приложением

Запустите созданное приложение, выберите размерность массива *a* (например, количество строк – 4, количество столбцов – 8), введите числовые значения элементов массива *a* и убедитесь, что приложение функционирует в соответствии с заданием.

На рисунке 2.4 показан интерфейс приложения после ввода исходных данных и нажатия кнопки «OK».

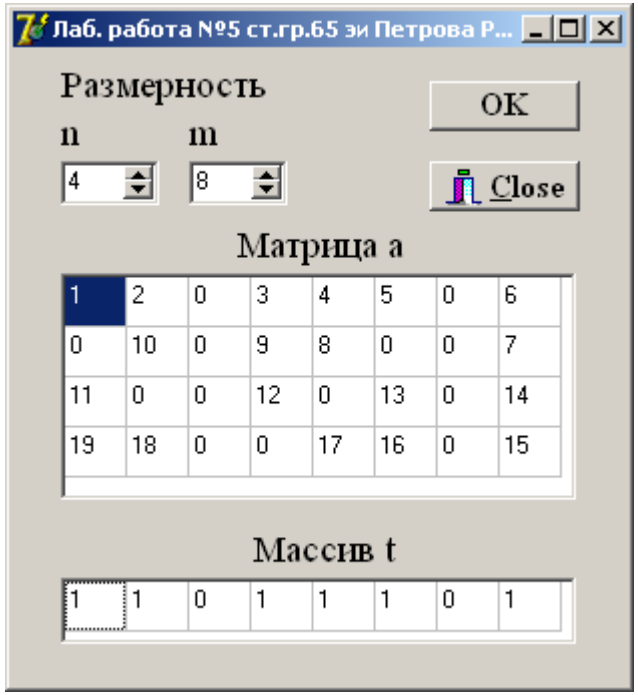

Рис. 2.4. Интерфейс приложения после его выполнения

2.2. Индивидуальные задания

По указанию преподавателя выберите свое индивидуальное задание и создайте приложение. Работа приложения должна завершаться нажатием кнопки Close.

Индивидуальные задания 1-го уровня

1. Определить количество положительных и отрицательных элементов матрицы $a(5, 5)$.
2. Найти наибольший элемент главной диагонали матрицы $a(4, 4)$ и вывести на печать строку, в которой он находится.
3. Найти минимальный элемент матрицы $a(4, 4)$ и вывести на печать столбец, в котором он находится.

4. Даны натуральные m, n и матрица $a(m, n)$. Переписать элементы матрицы в одномерный массив b.
5. Дана матрица $a(10, 14)$. Найти сумму элементов тех столбцов матрицы, первый элемент которых равен 0.
6. Даны натуральные m, n и матрица $a(m, n)$ целых чисел. Получить сумму тех членов последовательности, которые кратны 5.
7. Даны натуральные m, n и матрица $a(m, n)$ целых чисел. Получить сумму и количество тех членов последовательности, которые отрицательны.
8. Дан массив $a(4, 6)$. Перепишите элементы массива a в массив $b(24)$ в том порядке, в котором они были расположены в строках массива a.
9. Дан массив $a(5, 7)$. Перепишите элементы массива a в массив $b(35)$ в том порядке, в каком они были расположены в столбцах массива a.
10. Дан массив $a(3, 4)$. Увеличьте каждый элемент массива на 4 и выведите полученную матрицу на печать.

Индивидуальные задания 2-го уровня

11. Задана матрица $p[n, n]$. Найти в каждой строке наибольший элемент и поменять его местами с элементами главной диагонали.
12. Все элементы матрицы $a(4, 5)$, кратные 3, переписать в массив c. Полученный массив c вывести на экран.
13. Все положительные элементы матрицы $a(3, 5)$ переписать в массив b. Если таких элементов нет, вывести сообщение об этом.
14. Получить вектор $r_1, r_2, \ldots, r_5$, элементами которого являются произведения положительных элементов соответствующих строк матрицы $a(5, 4)$.
15. Задана вещественная матрица $a(n, m)$. Найти строку с наименьшей и наибольшей суммой элементов. Вывести на экран найденные строки и суммы их элементов.
16. Задана целочисленная матрица размером $a(n, m)$. Определить k – количество «особых» элементов матрицы, считая элемент «особым», если он больше суммы остальных элементов своего столбца.
17. Даны натуральное число n, целочисленная квадратная матрица порядка n. Получить $b_1, b_2, \ldots, b_n$, где b_i – это значение первого

по порядку положительного элемента *i*-й строки (если таких элементов нет, то принять $b_i = 0$).

18. Даны натуральные *n*, *m* и действительная матрица размера *a*(*n*, *m*), в которой не все элементы равны нулю. Получить новую матрицу путем деления всех элементов данной матрицы на ее наибольший по модулю элемент.

19. Задана целая квадратная матрица *n*-го порядка. Найти в каждой строке наибольший элемент и поменять его местами с элементом главной диагонали.

20. В заданной действительной матрице *a*(*n*, *m*) поменять местами строку, содержащую элемент с наибольшим значением со строкой, содержащей элемент с наименьшим значением.

21. Задана вещественная квадратная матрица *n*-го порядка. Замените нулями все ее элементы, расположенные на главной диагонали и выше ее.

22. Дана вещественная матрица *b*(*n*, *m*). Упорядочить ее строки по неубыванию их первых элементов.

23. Дана матрица *c*(4, 5). Элементы каждой строки матрицы умножить на наименьший элемент соответствующей строки.

24. Дано натуральное *n* и действительная квадратная матрица *a* порядка *n*. Требуется преобразовать матрицу: поэлементно вычесть последнюю строку из всех строк, кроме последней.

25. Дано натуральное число *n* и действительная квадратная матрица *b* порядка *n*. Сформировать два одномерных массива. В один записать по строкам верхний треугольник двухмерного массива, включая элементы главной диагонали, в другой – нижний треугольник. В каждом массиве найти среднее арифметическое положительных элементов, сравнить их между собой и выдать соответствующее сообщение.

Индивидуальные задания 3-го уровня

26. Из заданной целочисленной матрицы *n*-го порядка получить матрицу (*n* – 1)-го порядка путем удаления из исходной матрицы строки и столбца, на пересечении которых расположен элемент с наибольшим по модулю значением (предполагается, что такой элемент единственный).

27. Для заданной целой матрицы *a*(*n*, *m*) вывести на экран все ее седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или, наоборот, является наибольшим в своей строке и наименьшим в своем столбце.

28. Задана вещественная матрица *c*(*n*, *m*). Переставить ее строки и столбцы таким образом, чтобы наибольший элемент (предполагается, что такой элемент единственный) оказался в верхнем левом углу.

29. Определить, является ли заданная целая квадратная матрица *n*-го порядка магическим квадратом, то есть такой, в которой суммы элементов во всех строках и столбцах одинаковы.

30. Упорядочить по возрастанию элементы каждой строки матрицы *b*(*n*, *m*).

Вопросы для самоконтроля

1. Сколько индексов имеет каждый элемент матрицы?
2. Что обозначают индексы матрицы?
3. Какие матрицы называются квадратными?
4. Что такое главная и побочная диагонали квадратной матрицы?
5. Какое отношение индексов у элементов, расположенных над главной диагональю?

Материалы к лабораторной работе № 3

3. Программирование с использованием строк

Цель работы: овладение практическими навыками работы со строками; освоить применение компонентов ***ListBox*** и ***ComboBox*** для создания приложения, в котором используются строки.

Помимо числовой информации компьютер может обрабатывать символьную информацию. Object Pascal оперирует с символьной информацией, которая может быть представлена отдельными символами или строками символов.

Символьные типы

Данные символьного типа предназначены для хранения одного символа. Для символов используется тип Char. Кроме Char в Delphi 7 имеется еще два символьных типа – ANSIChar и WideChar. Пример объявления переменной типа Char.

var c: char;

Для кодировки символов в *Windows* используется код *ANSI* (*American Natinal Standards Institute – американский национальный институт стандартизации).* В соответствии с этой таблицей каждому символу соответствует целое число в диапазоне 0..255. Это число служит кодом внутреннего представления символа. Символы с номерами от 0 до 31 являются служебными символами, т.е. предназначены для управления отображением информации. Например, символ с кодом 9 вставляет в текст знак табуляции, символ с кодом 13 эквивалентен нажатию клавиши *Enter* (конец абзаца), символ с кодом 27 – ESC.

Строковые типы

Строкой называется последовательность из определенного количества символов, заключенных в апострофы.

Пример

'Текстовая строка',
'abcde',
'S=10,24'.

Delphi 7 поддерживает три физических строковых формата: короткий – ***ShortString***, длинный – ***AnsiString***, широкий – ***WideString*** и один логический строковый тип – ***String.***

Переменные типов AnsiString и WideString – это динамически распределяемые массивы символов, максимальная длина которых ограничивается только наличием памяти.

Тип ShortString имеет максимальную длину, равную 255 символам. Тип String – это, по существу, массив Array [0..255] of char. Тип ShortString предназначен для обеспечения совместимости с ранними версиями Delphi.

Тип String в зависимости от директив компилятора интерпретируется либо как AnsiString , либо как ShortString.

Пример объявления строковых переменных:

var st: String;
st1: String[10];

В Object Pascal имеется простой доступ к отдельным символам строковой переменной: *i*-й символ переменной st записывается как *st*[*i*]. Например, если *st* – это 'Строка', то *st*[1] – это 'С', *st*[2] – это 'т', *st*[3] – 'р' и так далее.

Над строковыми данными определена операция слияния (конкатенации), обозначаемая знаком +.

Пример

a := 'Object';
b := 'Pascal';
c := a + b;

В этом примере переменная *c* приобретет значение 'ObjectPascal'.

Кроме слияния над строками определены операции сравнения **<, >, =, <>, <=, >=**. Две строки сравниваются посимвольно, слева направо, по кодам символов. Если одна строка меньше другой по длине, недостающие символы короткой строки заменяются символом с кодом 0.

Процедуры и функции для работы со строками

Ниже приведены основные стандартные процедуры и функции для работы со строками.

length(s:string): integer;

Функция возвращает число символов в строке *s*.

Пример

```
n := length('Pascal');  //n будет равно 6
```

concat(s1,[s2,...,sn]: string): string;

Возвращает строку, представляющую собой сцепление из строк s_1, s_2, …, s_n. Идентична операции «+» для строк, но работает менее эффективно.

Пример

```
s:=concat('aa', 'xx', 'zz');   //s будет равно 'aaxxzz'
```

copy(s:string; index:integer; count: integer): string;

Функция возвращает подстроку, выделенную из исходной строки *s*, длиной *count* символов, начиная с символа под номером *index*.

Пример

```
s := 'Интегрированная среда Delphi';
s1 := copy(s, 1, 7);       //s1 будет равно 'Интегри'
s2 := copy(s, 17, 5);      //s2 будет равно 'среда'
s3 := copy(s, 23, 6);      //s3 будет равно 'Delphi'
```

delete(var s:string; index,count:integer);

Процедура удаляет из строки *s* подстроку длиной *count* символов, начиная с символа под номером *index*.

Пример

```
s := 'Интегрированная среда Delphi';
delete(s,1,16);                  //s будет равно 'среда Delphi'
```

insert(source:string; var s:string;index:integer);

Процедура предназначена для вставки строки *source* в строку *s*, начиная с символа *index* этой строки.

Пример

```
s := 'Object';
insert('Pascal',s,7);          //s будет равно  'ObjectPascal'
```

pos(substr,s:string):integer;

Функция производит поиск в строке *s* подстроки *substr*. Результатом функции является номер первой позиции подстроки в исходной строке. Если подстрока не найдена, то функция возвращает 0.

Пример

```
s := 'ObjectPascal';
x1 := pos('Pascal', s);    //x1 будет равно 7
x2 := pos('Basic', s);     //x2 будет равно 0
```

Далее приведены функции, связанные с типом ***char***, но которые часто используются при работе со строками.

chr(n: byte): char

Функция возвращает символ по коду, равному значению выражения *n*.

ord(ch: char): byte;

В данном случае функция возвращает код символа *ch*.

upcase(c: char): char;

Если *c* – строчная латинская буква, то функция возвращает соответствующую прописную латинскую букву, в противном случае символ *c* возвращается без изменения.

Применение компонентов ListBox и ComboBox для работы со строками

При работе со строками ввод и вывод информации на экран удобно организовывать с помощью компонентов ***ListBox*** и ***ComboBox***.

Пиктограмма компонента ***ListBox***  находится на странице *Standard* Палитры компонентов. Компонент ***ListBox*** представляет собой список, элементы которого выбираются при помощи клавиатуры или «мыши». Список элементов задается свойством ***Items***. Методы ***Add*** , ***Delete*** и ***Insert*** используются для добавления, удаления и вставки строк соответственно. Для определения номера выделенного элемента используется свойство ***ItemIndex.*** Для добавления строк в ***ListBox*** необходимо воспользоваться компонентом ***Edit***.

Пиктограмма компонента ***ComboBox***  также расположена на странице *Standard* Палитры компонентов. Компонент ***ComboBox*** представляет собой комбинацию списка ***ListBox*** и редактора ***Edit***. По внешнему виду компонент ***ComboBox*** напоминает строку ввода ***Edit***, но дополнительно имеет в правой части кнопку со стрелкой. Если щелкнуть мышью по этой кнопке, появится выпадающий список, подобный списку компонента ***ListBox***. Используя строку ввода, можно вводить в список новые элементы, осуществлять поиск нужного элемента в списке, отображать активный элемент списка.

Свойства компонента ***ComboBox*** заимствованы у компонентов ***Edit*** и ***ListBox***. Для работы с окном редактирования используется свойство ***Text***, как в ***Edit***, а для работы со списком используется свойство ***Items***, как в ***ListBox***. Основные операции для обработки списка в компоненте ***ComboBox*** – добавление, удаление, поиск элементов – осуществляется так же, как и в списке ***ListBox***.

3.1. Пример создания приложения

Задание. Создать Windows-приложение в визуальной среде Delphi для решения следующей задачи. Дана произвольная символьная строка, слова в строке разделяются любым количеством пробелов. Необходимо определить количество слов в данной строке и заменить все строчные символы русского языка прописными.

Полученную строку вывести на экран.

Ввод строки заканчивать нажатием клавиши *Enter*. Работа приложения должна завершаться нажатием кнопки *Close*.

3.1.1. Размещение компонентов на Форме

Разместите на Форме компоненты так, как показано на рисунке 2.5.

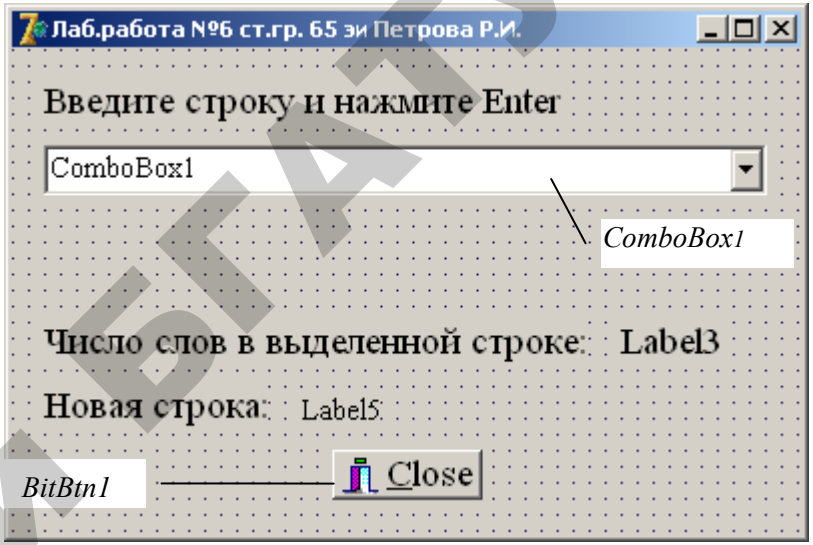

Рис. 2.5. Размещение компонентов на Форме

3.1.2. Сохранение проекта

Для нового проекта создайте новую папку, например ***X:\65эи\ФИО_студента\Mod2\Lab3.*** Сохраните проект ***File | Save Project As….*** Сначала сохраните модуль с именем ***UnStr.pas***, затем файл проекта под именем ***PrStr.dpr***

Последующие сохранения выполнять командами ***File | Save All.***

3.1.3. Создание процедуры-обработчика события активизации Формы FormActivate

В момент запуска приложения, когда панель интерфейса появляется на экране, для пользователя удобно, чтобы курсор уже находился в поле редактора компонента ***ComboBox***. При активизации Формы возникает событие ***OnActivate***, которое можно использовать для передачи фокуса ввода компоненту ***ComboBox***. Для создания процедуры-обработчика этого события необходимо в Инспекторе объектов выбрать компонент ***Form1***, на странице ***Events (События)*** найти событие ***OnActivate*** и дважды щелкнуть «мышью» по его правой (белой) части. Курсор установится в тексте процедуры-

обработчика события активизации Формы **procedure TForm1. FormActivate(Sender: TObject).** Наберите операторы передачи фокуса ввода компоненту ComboBox1.

3.1.4. Создание процедуры-обработчика события ComboBox1KeyPress

В соответствии с заданием необходимо, чтобы при нажатии клавиши *Enter* строка символов, которую пользователь набрал в поле редактирования, переносилась в список выбора компонента ***ComboBox1***.

Для создания процедуры-обработчика этого события необходимо в Инспекторе объектов выбрать компонент ***ComboBox1***, на странице ***Events (События)*** найти событие ***OnKeyPress*** и дважды щелкнуть «мышью» по его правой части. Курсор установится в тексте процедуры **procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key: Char),** которая осуществляет обработку события нажатия клавиши на клавиатуре. Наберите операторы этой процедуры, пользуясь текстом модуля ***UnStr*** (листинг 2.3).

В результате выполнения этой процедуры при нажатии клавиши *Enter* строка из поля редактирования переносится в список выбора и очищается поле редактирования.

3.1.5. Создание процедуры-обработчика события нажатия клавиши «мыши» ComboBox1Click

Создание процедуры **procedure TForm1.ComboBox1Click(Sender: TObject)** выполняется аналогично процессу создания процедуры-обработчика события ***OnKeyPress*** компонента ***ComboBox1*** (см. раздел 3.1.4).

Пользуясь текстом модуля ***UnStr*** (листинг 2.3), наберите операторы, которые осуществляют основной алгоритм обработки символов выбранной строки.

3.1.6. Работа с приложением

Запустите созданное приложение. Занесите с помощью окна редактирования исходные данные в список выбора компонента - ***ComboBox1***. Ввод каждой строки завершайте нажатием клавиши Enter. Далее раскройте список выбора, щелкните «мышью» по

нужной строке – в результате будет определено количество слов в строке и произойдет замена строчных букв прописными.

Преобразование строчных букв в прописные основано на том, что код строчной буквы больше кода прописной. Код прописных букв от «а» до «я» больше соответствующих строчных букв на 32 (см. таблицу кодов Ansi). Например, код символа «а» равен 224, а код символа «А» – 192. Эта закономерность сохраняется и для остальных букв русского алфавита.

На рисунке 2.6 показан интерфейс приложения после ввода исходных данных и выбора строки.

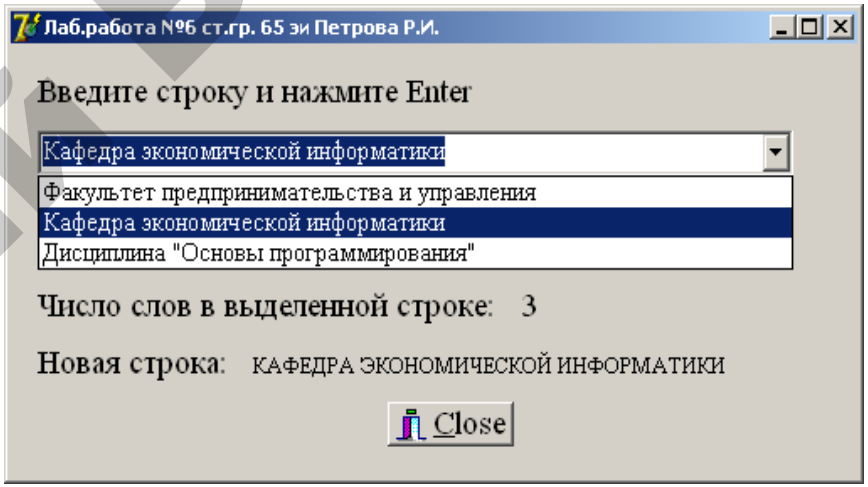

Рис. 2.6. Интерфейс приложения после его выполнения

Текст программы приведен в листинге 2.3.

Листинг 2.3

```
Unit UnStr;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,  StdCtrls, Buttons;
type
  TForm1 = class(TForm)
    Label1: TLabel;
```

```
    ComboBox1: TComboBox;
    Label2: TLabel;
    Label3: TLabel;
    BitBtn1: TBitBtn;
    Label4: TLabel;
    Label5: TLabel;
    procedure FormActivate(Sender: TObject);
    procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);
    procedure ComboBox1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
//Обработка события активизации Формы
procedure TForm1.FormActivate(Sender: TObject);
begin
ComboBox1.SetFocus;
end;
 // Обработка события ввода символа и нажатия клавиши Enter
procedure TForm1.ComboBox1KeyPress(Sender: TObject; var
Key: Char);
begin
{Если нажата клавиша Enter, то строка из поля редактирования
 переносится в список выбора ComboBox1}
if key=#13 then
 begin
 ComboBox1.Items.Add(ComboBox1.Text);
 ComboBox1.Text:='';                    // очистка окна редактирования
 end;
end;
// Обработка события нажатия клавиши «мыши» в списке выбора
procedure TForm1.ComboBox1Click(Sender: TObject);
```



```
var
st: string;
n,i,nst,ind: integer;
begin
  n:=0;                                  //n содержит количество слов
  ind:=0;
  nst:=ComboBox1.ItemIndex;    //определение номера выбранной
строки
  st:=ComboBox1.Items[nst];  //st присваивается выбранная строка
{Определение количества слов в выбранной строке st}
for i:=1 to length(st) do
 case ind of
 0: if st[i]<>' ' then                  //если встретился символ
                begin
                 ind:=1;
                n:=n+1;   //количество слов увеличивается на единицу
                  end;
 1: if st[i]=' ' then                    //если встретился пробел
                ind:=0;
 end;
   Label3.Caption:=IntToStr(n);   //вывод количества слов
// Замена  строчных символов русского языка на прописные
for i:=1 to length(st) do
 if  (st[i]>='a') and (st[i]<='я') then
   st[i]:=chr(ord(st[i])-32);
Label5.Caption:=st;    //полученная строка выводится в Label5
end;
end.
```

3.2. Индивидуальные задания

Во всех заданиях исходные данные вводить с помощью компонента ***Edit*** в компонент ***ListBox***, либо с помощью свойства ***Text*** в свойство ***Items*** компонента ***ComboBox***. Результат выводить с помощью компонента ***Label***. Ввод строки заканчивать нажатием клавиши *Enter*. Работа приложения должна завершаться нажатием кнопки *Close*.

Индивидуальные задания 1-го уровня

1. Дана строка символов, содержащая буквы русского и латинского алфавитов. Верно ли, что в этой строке содержится четное количество строчных латинских букв. Вывести на экран соответствующее сообщение.

2. Дана строка символов, содержащая буквы и цифры. Определить, чего больше – цифр или букв. Вывести на экран соответствующее сообщение.

3. Дана строка символов, состоящая из букв, цифр, запятых, точек. Удалите из данной последовательности все цифры. Полученную строку вывести на экран.

4. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Преобразовать последовательность, заменив пробелы между словами на символ звездочки**.**

5. Строка символов содержит только цифры. Вывести на экран номера позиций цифры, заданной пользователем.

6. Дана строка символов, содержащая заглавные латинские буквы. Определить, упорядочены ли эти буквы по алфавиту.

7. Дана символьная строка. Определить, является ли данная последовательность записью двоичного числа (т.е. содержит только нули и единицы).

8. Дана строка символов. Подсчитайте, сколько среди них латинских заглавных букв.

9. Дано слово, записанное через дефис. Поменяйте части слова до и после дефиса.

10. Дана строка символов. Определить, какой процент слов начинается на букву «К».

Индивидуальные задания 2-го уровня

11. Дана строка, состоящая из букв, цифр, запятых, точек. Заменить каждую точку многоточием и вывести новую строку.

12. Дана строка символов, среди которых есть двоеточия. Выведите в Memo все символы, расположенные до первого двоеточия.

13. Дана символьная строка. Заменить все символы «!» точками, кроме первого. Вывести полученную строку.

14. Дана символьная строка, содержащая два предложения, каждое из которых заканчивается точкой. Поменять их местами, сохранив порядок слов в предложениях.

15. Дана символьная строка. Слово – последовательность символов между пробелами, не содержащая пробелы внутри себя. Определить длину самого короткого слова.

16. Дана символьная строка. Слово – последовательность символов между пробелами, не содержащая пробелы внутри себя. Определить количество слов заданной длины.

17. Дана символьная строка. Слово – последовательность символов между пробелами, не содержащая пробелы внутри себя. Переставить и распечатать слова заданной строки в алфавитном порядке по первой букве.

18. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести на экран числа этой строки в порядке возрастания их значений.

19. Дана строка символов, состоящая из произвольного текста на английском языке, слова отделены пробелами. Вывести на экран порядковый номер слова максимальной длины и номер позиции строки, с которой оно начинается.

20. Дана строка символов, состоящая из букв, цифр, запятых, точек, знаков «+» и «-». Выделить подстроку, которая соответствует записи целого числа (т.е. начинается со знака «+» или «-» и внутри подстроки нет букв, запятых и точек).

21. Дана символьная строка и символ. Слово – последовательность символов между пробелами, не содержащая пробелы внутри себя. Вывести все слова, в которых есть заданный символ.

22. Дана строка символов, содержащая некоторый текст. Разработать программу, которая определяет, является ли данный текст палиндромом, т.е. читается ли он слева направо так же, как и справа налево (например, «А роза упала на лапу Азора»).

23. Дана строка символов, состоящая из произвольного текста на английском языке, слова отделены пробелами. Поменять местами первую и последнюю буквы каждого слова.

24. Составить программу, которая читает построчно текст другой программы (ввести с клавиатуры) на языке Pascal, подсчитыва-

ет количество ключевых слов «begin» и «end» и выводит на экран соответствующее сообщение.

25. Дано предложение, состоящее из слов, отделенных друг от друга «*». В конце предложения стоит точка. Определите все повторяющиеся слова в предложении.

Индивидуальные задания 3-го уровня

26. Дана символьная строка и слово, состоящее из четырех символов. Определить, есть ли в данной строке все буквы данного слова. Вывести соответствующее сообщение.

27. Дана строка символов. Заменить все последовательности символов 'on' на 'online' и вывести новую строку (если искомой последовательности в строке нет, то вывести соответствующее сообщение).

28. Дана строка символов, состоящая из произвольного текста на английском языке. Заданный текст распечатайте по строкам, понимая под строкой часть текста до точки включительно.

29. Дана строка символов. Определить количество букв 'я' между самой левой открывающейся скобкой и самой правой закрывающейся скобкой (если какие-либо скобки отсутствуют, то вывести соответствующее сообщение).

30. Даны две символьные строки. Слово – последовательность символов между пробелами, не содержащая пробелы внутри себя. Вывести слова, которые встречаются в обеих строках.

Вопросы для самоконтроля

1. Что такое строка символов?
2. Какие строковые форматы поддерживает Delphi 7?
3. Как описываются строки в языке Delphi?
4. На какой странице Палитры компонентов размещены компоненты ListBox и ComboBox?
5. Как в Object Pascal осуществляется доступ к отдельным символам строковой переменной?

ПРИМЕРЫ РАЗНОУРОВНЕВЫХ ЗАДАНИЙ ДЛЯ КОНТРОЛЯ ЗНАНИЙ

Задания 1-го уровня

1. Дано натуральное число n, последовательность символов a_1, ..., a_n. Выяснить, каких символов среди последовательности больше – запятых или точек с запятой (не исключается и случай равенства).

2. Дана таблица действительных чисел, состоящая из 4 строк и 5 столбцов. Найти среднее арифметическое наибольшего и наименьшего значений ее элементов.

3. Дана таблица действительных чисел, состоящая из n строк и n столбцов (n – натуральное число). Найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.

4. Дана таблица действительных чисел, состоящая из 15 строк и 10 столбцов. Определить количество элементов, не превышающих среднее арифметическое значение всех элементов таблицы.

5. Дана таблица действительных чисел, состоящая из 14 строк и 18 столбцов. Найти сумму наибольших значений элементов ее строк.

Задания 2-го уровня

6. Даны натуральные m, n и матрица $a(m, n)$. Найти сумму элементов, расположенных в строках с отрицательным элементом на главной диагонали, и произведение элементов, расположенных в строках с положительным элементом в первом столбце.

7. Даны натуральное число n действительные a_1, a_2, ..., a_n. Среди последовательности a_1, a_2, ..., a_n есть как отрицательные, так и неотрицательные числа. Преобразовать последовательность. В новой последовательности вначале должны следовать отрицательные члены последовательности a_1, a_2, ..., a_n в том же порядке, что и в исходной, а затем положительные члены в обратном порядке. Определить количество отрицательных и положительных членов последовательности.

8. Дано натуральное число n, последовательность символов a_1, a_2, ..., a_n. В этой последовательности имеется один восклицатель-

ный знак. Пусть a_1, a_2, ..., a_p – символы данной последовательности, предшествующие восклицательному знаку. Выяснить, встречается ли в последовательности a_1, a_2, ..., a_p слово «БАТУ».

9. Даны натуральные m, n и матрица $a(m, n)$. Числа матрицы не повторяются. Найти максимальное число и его порядковый номер среди положительных чисел таблицы.

10. Даны натуральные числа n, m и два одномерных массива $a(n)$, $b(m)$. Определить, содержится ли наибольший элемент массива a среди элементов массива b, если содержится, то сколько раз?

11. Даны натуральное число n и таблица действительных чисел, состоящая из n строк и n столбцов. В данной таблице найти наибольший по модулю элемент. Получить таблицу, состоящую из n-1 строки и n-1 столбца путем удаления из исходной таблицы столбца и строки, на пересечении которых расположен найденный элемент.

12. Даны натуральные числа m, n и таблица целых чисел, состоящая из n строк и m столбцов, все элементы которых различны. В каждом столбце выбирается элемент с наибольшим значением, затем среди этих чисел выбирается наименьшее. Найти это число.

13. Дано предложение, в котором все слова отделены друг от друга одним пробелом, в конце предложения стоит точка. Определить количество слов в предложении, которые начинаются и заканчиваются одной и той же буквой.

14. Дан одномерный массив $b(15)$. Удалить из массива первый минимальный и последний максимальный элементы массива.

15. Дано предложение, в котором слова отделены друг от друга одним или несколькими пробелами. Удалить из него все лишние пробелы, оставив между словами только по одному.

Задания 3-го уровня

16. Дан текст. Найти максимальное число из имеющихся в нем чисел.

17. Дана таблица целых чисел, состоящая из 20 строк и 30 столбцов. К элементам каждого столбца таблицы прибавить наименьший элемент этого же столбца.

18. Дана целочисленная матрица $x(8, 8)$. Найти наименьшее из значений элементов столбца, который обладает наибольшей суммой модулей элементов. Если таких столбцов несколько, то взять первый из них.

19. Дано предложение, в котором все слова отделены друг от друга одним пробелом, в конце предложения стоит точка. Найти слова, у которых все буквы этого слова различны.

20. Дано предложение, в котором все слова отделены друг от друга одним пробелом, в конце предложения стоит точка. Зашифровать предложение, записав каждое из входящих в него слов в обратном порядке.

ВАРИАНТЫ ТЕСТОВЫХ ЗАДАНИЙ ПО МОДУЛЮ 2

1. В результате выполнения оператора цикла

for i: = 1 to 50 **do If** x[i]>0 **then** x[i]:= x[i]* x[i]; :

a) возведутся в квадрат положительные элементы массива x;
b) перемножатся все элементы массива x;
c) перемножатся отрицательные элементы массива x.

2. Массив – это:

a) совокупность однотипных данных, имеющих общее имя;
b) совокупность строк и столбцов;
c) совокупность разнотипных данных, имеющая общее имя.

3. Выберите правильное обозначение элемента массива в Pascal:

a) a[i] ;
б) a[5] ;
c) a(i) ;
d) a(3) ;
e) a{i} .

4. Выберите правильные варианты оператора объявления массива:

a) var a: array[1..25] of Integer;
b) var a: array(1..25) of Integer;
c) var a: array[1..25] of real;
d) var a: array[1…25] of real.

5. Найти ошибку в фрагменте программы:

```
var i: Integer;
var s: Real;
var a: array[1..25] of Real;
begin
s:=0;
for  i:= 1 to 40 do   s :=s+a[i];
end;
```

a) значения параметра цикла i (от 26 до 40) выходят за пределы размерности объявленного массива [1..25];
b) ошибка в операторе объявления переменной s;
c) нарушение структуры программы.

6. Какие из приведенных констант относятся к строковому типу?

a) ‘CONST’;
b) ‘5467’;
c) 5467;
d) ‘1’.

7. Каким ключевым словом задается в разделе описания переменных строковый тип?

a) char;
b) integer;
c) byte;
d) string.

8. Какое значение примет переменная *k* в результате выполнения фрагмента программы?

```
m := ‘биология’;
k := ‘зоо’ + copy (m, 4, 5);
```

a) зоолог;
b) зоогия;
c) зоология;
d) биология.

9. Какое значение примет переменная **a** в результате выполнения фрагмента программы?

```
a := ‘комар’;
b := ‘ово’;
insert (b, a, 6);
```

a) комар;
b) комарово;
c) корова;
d) ово.

10. Какой результат может дать строковая функция?

a) переменная;
b) число;
c) символ;
d) строка.

ГЛОССАРИЙ

Базовый тип. Тип данных элементов массива.

Бинарный оператор. Оператор, выполняющий некоторую операцию над двумя операндами.

Буквенно-цифровые символы. Символы, являющиеся только буквами или цифрами.

Вещественное число. Число с дробной частью.

Время жизни. Промежуток времени, в течение которого объект существует в памяти компьютера.

Время компиляции. Промежуток времени, в течение которого исходный код программы транслируется в выполнимый файл.

Глобальная переменная. Переменная, доступная во всей программе. Переменная с глобальной областью видимости.

Деление по модулю. Операция, результатом которой является остаток деления двух целых чисел.

Диапазон значений. Интервал допустимых значений переменной.

Длинная строка. В Object Pascal — строка, размер которой ограничен только доступной памятью.

Инициализация. Присвоение переменной начального значения. Применительно к программам и подпрограммам инициализация означает присвоение их переменным и объектам начальных значений.

Ключевое слово. Зарезервированное слово или символ, распознаваемые транслятором как часть синтаксиса языка. Использование ключевых слов в качестве идентификаторов недопустимо.

Конкатенация. Операция объединения строк в предписанном порядке слева направо.

Константа. Неизменное значение, хранящееся в памяти все время выполнения программы. Константы делятся на именные и неименные.

Короткая строка. В Object Pascal это строка, длина которой не превышает 255 символов.

Логическая ошибка. Ошибка в логике программы или в алгоритме. В отличие от синтаксических ошибок, транслятор не распознает логические ошибки.

Локальная переменная. Переменная, доступная только внутри подпрограммы, в которой она объявлена. Переменная с локальной об-

ластью видимости

Библиотека визуальных компонентов (VCL — Visual Component Library). Набор компонентов Delphi, организованных в иерархическую структуру.

Броузер проекта. Окно Delphi, в котором перечислены модули, типы, классы, свойства, методы, переменные и процедуры, объявленные или используемые в текущем проекте.

Бумажная копия. Распечатанное на бумаге содержимое файла.

Быстрая разработка приложений (RAD — Rapid application development). Стратегия разработки программ, на основе которой была создана Delphi. Стратегия RAD предполагает использование многочисленных средств, облегчающих и ускоряющих разработку приложений.

Время выполнения. Время, в течение которого отлаженная и скомпилированная программа выполняется.

Выполнимый модуль. Файл, который может непосредственно выполняться на компьютере. Выполнимым модулем может быть вся программа, хранящаяся в EXE-файле, или ее подпрограмма, хранящаяся в библиотечных файлах с расширением .DLL, .OBJ или .LIB.

Гипертекст. Текст, содержащий ссылки на другие фрагменты текста, или визуальные объекты, хранящиеся в этом же или в других файлах.

Главное меню. Меню, располагаемое в верхней части экрана. При выборе одного из элементов главного меню активизируется ассоциированное с этим элементом раскрывающееся меню или выполняется ассоциированная с ним команда.

Графический интерфейс пользователя (GUI — Graphical user interface). Пользовательский интерфейс, содержащий пиктограммы и управляемый с помощью мыши и клавиатуры.

Группа проектов. Совокупность связанных проектов или проектов, работающих вместе в составе многоуровневого приложения.

Двойной щелчок. Два щелчка кнопкой мыши, быстро следующих один за другим. Если в тексте не указывается, какой кнопкой, – значит, левой.

Динамически подключаемая библиотека (DLL – Dynamic link library). Файл, содержащий библиотеку подпрограмм, которая

может подключаться к программе в процессе ее выполнения.

Дисковая операционная система (DOS). Операционная система без графического интерфейса, в которой программы запускаются из командной строки, например MS DOS компании Microsoft или PC DOS компании IBM.

Диспетчер проектов. Окно Delphi, позволяющее легко перемещаться среди многочисленных проектов и файлов проектов.

Значение по умолчанию. Значение, автоматически присваиваемое свойству объекта системой Delphi.

Инспектор объектов. Окно Delphi, в котором программист может изменять значения свойств компонентов во время разработки.

Интегрированная среда разработки. Рабочая среда Delphi, в которой совмещены (интегрированы) все стадии процесса программирования, включая редактирование, компиляцию и отладку.

Исследователь кода. Окно в среде разработки Delphi, с помощью которого программист может легко перемещаться по файлам модулей. В окне исследователя кода выводится древовидная диаграмма, показывающая все типы, классы, свойства, методы, глобальные переменные и глобальные процедуры, определенные в модуле кода, активного в данный момент в окне редактора кода.

Кнопка. Компонент, с помощью которого пользователь может инициировать выполнение некоторого фрагмента кода или целой программы.

Комбинация клавиш быстрого вызова. Комбинация клавиш, нажатие на которые приводит к выполнению какого-либо действия, ассоциированного с нею.

Компонент. Объект, расположенный на пользовательском интерфейсе, или любой объект палитры компонентов Delphi.

Консольное приложение. Так в Delphi называется 32-битовая программа Windows без графического интерфейса, которая может выполняться в DOS (или в окне DOS операционной системы Windows).

Метод. Подпрограмма, входящая в состав объекта. Чаще всего методы предназначены для обработки свойств объектов.

Модуль. Отдельная часть исходного кода. Разбивка программы на модули предусматривается стратегией структурного программирования.

Мышь. Устройство ввода, используемое для управления графи-

ческим интерфейсом пользователя. Обычно мышь содержит механический шарик, управляющий положением указателя на экране, и две кнопки – левую и правую.

Надпись (метка). Компонент, выводящий на экран фрагмент текста, который пользователь не может изменять непосредственно с помощью мыши или клавиатуры.

Область просмотра. Компонент, выводящий на экран много строк текста.

Обработчик события. Подпрограмма, выполняемая при наступлении некоторого события.

Объект. Совокупность свойств и методов, обозначаемая одним идентификатором.

Объектно-ориентированный язык. Язык программирования, поддерживающий объявление и обработку объектов.

Окно сообщений. Окно, в котором Delphi выводит результаты поиска, предупреждения и сообщения об ошибках.

Окно формы. Окно Delphi, выводящее на экран форму. В среде разработки в этом окне форму можно редактировать, изменять ее размер и размещать на ней компоненты.

Палитра выравнивания. Окно, в котором программисту предоставлены средства быстрого выравнивания компонентов на форме.

Палитра компонентов. Панель инструментов с вкладками, предоставляющая программисту быстрый доступ к компонентам.

Панель инструментов. Группа пиктограмм, расположенных обычно под главным меню. Большинство из них активизируют те же команды, что и элементы меню.

Панель инструментов просмотра. Панель инструментов, содержащая пиктограммы, предназначенные для создания новых форм, просмотра форм и модулей кода и переключения между формами и модулями кода.

Панель управления отладкой. Панель управления, предоставляющая во время разработки доступ к инструментам отладки. Предназначена для интерактивного тестирования и отладки программы.

Панель управления рабочим столом. Панель управления, с помощью которой программист может конфигурировать раскладку окон среды Delphi.

Папка. Синоним термина «каталог».

Перетаскивание. Перемещение визуального объекта в другое место. Чтобы перетащить объект, нужно установить на нем указатель мыши, нажать левую кнопку, передвинуть объект на новое место и отпустить кнопку мыши.

Пиксель. Точка на экране, являющаяся наименьшим элементом изображения. Единица измерения расстояния на экране по горизонтали и вертикали.

Пиктограмма. Прямоугольная область с рисунком, представляющая какой-либо объект или команду. Часто пиктограммы используются в качестве кнопок.

Подсказка завершения кода. Часть системы подсказки кода Delphi, выводящая на экран всплывающее меню со списком допустимых свойств и методов объекта. Такая подсказка помогает программисту быстро написать строку кода.

Поле ввода. Компонент, в который можно вводить строку текста как во время разработки, так и во время выполнения.

Пользовательская панель управления. Панель управления, которая по умолчанию содержит только одну пиктограмму, предоставляющую доступ к справочной системе Delphi. Пользователь (программист) может добавить на нее пиктограммы любых команд.

Приложение. Программа, выполняющая для пользователя какую-либо задачу.

Программа. Последовательность команд, выполняемых компьютером. Поскольку программа хранится в файле, то часто программой называют этот файл.

Проект. Совокупность файлов Delphi, используемых для разработки определенной программы.

Редактор кода. Полнофункциональный текстовый редактор, встроенный в среду разработки Delphi. С его помощью программист может просматривать и редактировать исходные коды программ.

Свойство. Переменная (возможно, объект), входящая в состав объекта.

Система подсказки кода. Набор инструментов Delphi, облегчающий для программиста написание исходного кода.

Список компонентов (окно компонентов). Окно, в котором в алфавитном порядке перечислены все компоненты, доступные в данной версии Delphi и в библиотеке VCL.

Список напоминаний. Встроенная в среду Delphi записная книжка, которая помогает программисту организовать работу над проектом.

Список окон. Окно, позволяющее программисту быстро переключаться между разными окнами среды разработки Delphi.

Стандартная панель инструментов. Панель инструментов Delphi, предназначенных для выполнения стандартных задач, таких, как открытие, сохранение и создание проектов или файлов.

Стыковка. Размещение одного окна внутри другого и закрепление его в определенном месте.

Указатель мыши. Индикатор позиции на экране, обычно имеющий форму стрелки. Указатель мыши используется для работы с графическим интерфейсом пользователя.

Файл модуля. Файл, содержащий исходный код модуля.

Файл проекта. Файл, играющий роль главной подпрограммы проекта Delphi. Из файла проекта вызываются файлы модулей и форм, составляющих проект.

Файл формы. Файл, в котором перечислены объекты, расположенные на форме, а также их свойства.

Форма. Объект, на визуальном изображении которого размещаются компоненты Delphi. При выполнении программы форма выводится на экран в виде окна.

Элемент управления. Компонент, появляющийся на экране во время выполнения. Элементы управления образуют подмножество библиотеки VCL: каждый элемент управления является компонентом, но не каждый компонент является элементом управления.

ЛИТЕРАТУРА

1. Delphi 7: руководство программиста / под ред. И. М. Архангельского. СПб: 2008. – 1070 с.

2. *Культин, Н. В.* Основы программирования в Delphi 2010 / Н. В. Культин. – Санкт-Петербург: БХВ, 2010. – 448 с.

3. *Прищепов, М. А.* Basic, Pascal и Object Pascal в среде Delphi: уч. пособие / М. А. Прищепов [и др.]. – Минск:. «Тетра Системс», 2006. – 320 с.

4. *Севернева, Е. В.* Основы алгоритмизации и программирования: учеб.-метод. комплекс / Е. В. Севернева, Н. М. Жалобкевич. – Мн.: БГАТУ, 2009. – 116 с.

5. *Синицын, А. К.* Основы алгоритмизации и программирования в среде DELPHI. Базовые типы и простейшие алгоритмы: лабораторный практикум по курсу «Основы алгоритмизации и программирования» для студентов 1–2-го курсов для всех. спец. БГУИР / А. К. Синицын, А. А. Навроцкий. – Минск: БГУИР, 2006. – 80 с.

6. *Фаронов, В. В.* Система программирования в Delphi в подлиннике: учебник/ В. В. Фаронов. – Санкт-Петербург: BHV, 2003. – 912 с.

7. *Шакирин, А. И.* Основы программирования на алгоритмическом языке Object Pascal в визуальной среде Delphi: конспект лекций. – Минск: БГАТУ, 2005. – 141 с.

8. *Прищепов, М. А.* Экзамен по информатике. Основы алгоритмизации и программирования: справ. пособие / М. А. Прищепов, В. П. Степанцов, Е. В. Севернева. – Минск: Тетра Системс, 2001. – 192 с.

Перечень методических указаний, методических материалов

9. Программирование в визуальной среде Delphi. В 2-х ч. Ч. 1 : методические указания к лабораторным занятиям / Р. И. Фурунжиев [и др.].– Минск: БГАТУ, 2004. – 63 с.

10. Программирование в визуальной среде Delphi. В 2-х ч. Ч. 2 : методические указания к лабораторным занятиям / Р. И. Фурунжиев [и др.].– Минск: БГАТУ, 2004. – 117 с.

11. Программирование на языке Object Pascal в визуальной среде Delphi: методические указания к выполнению курсовой работы / Р. И. Фурунжиев, Т. В. Ероховец, О. М. Львова. – Минск: БГАТУ, 2004. – 46 с.

12. Программирование на языке Object Pascal в визуальной среде Delphi: методические указания к лабораторным занятиям по дисциплине «Основы информатики и вычислительной техники» в 2-х частях / БГАТУ, сост. А. И. Шакирин. – Минск, 2001. — 118 с.

ПРИЛОЖЕНИЯ

Приложение 1

Процедуры и функции для работы со строковыми переменными и преобразования строкового представления чисел

Для работы со строковыми переменными применяются следующие процедуры и функции (в квадратных скобках указаны необязательные параметры).

1.1. Процедуры и функции для работы со строковыми переменными	
Function Concat(S1 [, S2, ..., SN]: String): String;	Возвращает строку, представляющую собой сцепление строк-параметров S1, S2, ..., SN
Function Copy(St: String; Index, Count: Integer): String;	Копирует из строки St Count символов, начиная с символа с номером Index
Procedure Delete(St: String; Index, Count; Integers;	Удаляет Count символов из строки St, начиная с символа с номером Index
Procedure lnsert(SubSt: String; St, Index: Integer);	Вставляет подстроку SubSt в строку St, начиная с символа с номером Index
Function Length(St: String): Integer;	Возвращает текущую длину строки St
Function Pos(SubSt, St: String): Integer;	Отыскивает в строке St первое вхождение подстроки SubSt и возвращает номер позиции, с которой она начинается. Если подстрока не найдена, возвращается ноль
1.2 Подпрограммы преобразования строк в другие типы	
Function StrToFloat(St: String): Extended;	Преобразует символы строки St в вещественное число. Строка не должна содержать пробелов
Function StrToInt(St: String): Integer;	Преобразует символы строки St в целое число. Строка не должна содержать пробелов
Function StrToDate(St: String): TDateTime;	Преобразует символы строки St в дату. Строка должна содержать два или три числа, разделенных правильным для Windows разделителем даты ; (в русифицированной версии таким разделителем, является “.”). Первое число – день, второе – месяц, если указано третье число, оно задает год

1.2 Подпрограммы преобразования строк в другие типы	
Function StrToDateTime(St: String): TDateTime;	Преобразует символы строки St в формат даты и времени. Строка должна содержать дату и время, разделенные пробелом
Function StrToTime(St: String): TDateTime;	Преобразует символы строки St во время
Procedure Val(St: String; var X; Code: Integer);	Преобразует строку символов St во внутреннее представление целой или вещественной переменной X, которое определяется типом этой переменной. Параметр Code содержит ноль, если преобразование прошло успешно, и тогда в X помещается результат преобразования; в противном случае он содержит номер позиции в строке St, где обнаружен ошибочный символ, и в этом случае содержимое X не меняется. В строке St могут быть ведущие и (или) ведомые пробелы
1.3. Подпрограммы обратного преобразования	
Function FloatToStr(Value: Extended): String:	Преобразует вещественное значение Value в строку символов
Function FloatToStrF(Value: Extended; Format: TFIoatFor-mat; Precision, Digits: Integer) : String;	Преобразует вещественное значение Value в строку символов с учетом параметров Precision и Digits (см. пояснения ниже)
Function DateToStr(Value: TDateTime): String:	Преобразует дату из параметра Value в строку символов
Function DateTimeToStr (Value: TDateTime): String;	Преобразует дату и время из параметра Value в строку символов
Procedure DateTimeToString (var St: String; Format: String; Value:	Преобразует дату и время из параметра Value в строку St
Function FormatDateTime (Format: String; Value: TDateTime):	Преобразует дату и время из параметра Value в строку символов
Function FormatFloat(Format: String; Value: Extended): String;	Преобразует вещественное значение Value в строку
Function IntToStr(Value: Integer): String;	Преобразует целое значение Value в строку символов
Function TimeToStr(Value: TDateTime): String;	Преобразует время из параметра Value в строку символов

Приложение 2

Стандартные функции в Delphi

Математическая запись	Запись в Delphi
sin x	sin (x)
cos x	cos (x)
arctg x	arctan (x)
x^2	sqr (x)
$\|x\|$	abs (x)
e^x	exp (x)
ln(x)	ln (x)
$\sqrt{x}$	sqrt (x)
Выделение целой части x	trunc(x)
Округление x в сторону ближайшего целого	round (x)

Примеры применения функций

Математическая запись	Запись в Delphi
b^a	exp (a*ln(b))
tg x	sin(x)/cos(x)
ctg x	cos(x)/sin(x)
arcsin x	arctan(sqrt(x/(1-sqr(x))))
arccos x	pi/2 – arcsin(x)
arcctg x	pi/2 – arctg(x)
$\log_a(x)$	ln(x)/ln(a)
Целочисленное деление (div)	a div b (3 div 2 равно 1)
Остаток от целочисленного деления (mod)	a mod b (5 mod 3 равно 2)

СОДЕРЖАНИЕ

ДЛЯ ЗАМЕТОК

Учебное издание

ОСНОВЫ ПРОГРАММИРОВАНИЯ
(в визуальной среде Delphi)

Учебно-методический комплекс

В двух частях

Часть 1

Составители:
Фурунжиев Риза Ибраимович,
Исаченко Елена Михайловна,
Ероховец Тамара Викторовна

Ответственный за выпуск *О. Л. Сапун*
Редактор *Н. А. Антипович*
Компьютерная верстка *А. И. Стебуля*

Подписано в печать 2.07.2010 г. Формат $60\times84^{1}/_{16}$.
Бумага офсетная. Печать офсетная.
Усл. печ. л. 7,67. Уч.-изд. л. 6,0. Тираж 150 экз. Заказ 652.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный аграрный технический университет».
ЛИ № 02330/0552841 от 14.04.2010.
ЛП № 02330/0552743 от 02.02.2010.
Пр. Независимости, 99–2, 220023, Минск.