**Раздел 1 ОСНОВНЫЕ ПОНЯТИЯ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ**

**Цель**

УЗНАТЬ:

* два вида интерфейса MDI и SDI;
* структуру интерфейса системы программирования Delphi;
* понятия класс, объект, компонент;
* атрибуты объекта: свойства, методы, события, реакцию на события;
* свойства объекта: Font, Color, Vizible, Name, Top, Width, Heigth, Left, Caption;
* назначения компонент TLabel,  TPanel, TButton;
* команду создания диалогового окна;
* правила задания названия процедуры - реакции на события.

НАУЧИТЬСЯ:

|  |  |
| --- | --- |
| маркированный список | размещать компоненты на форме; |
| маркированный список | задавать свойства и события компонента; |
| маркированный список | описывать реакцию на событие в виде создания процедуры; |
| маркированный список | сохранять и  открывать программу: файлы \*.dpr, \*.pas, \*.dfm, \*.exe; |
| маркированный список | компилировать, запускать программу на исполнение, прекращать выполнение программы. |

**Задача**

Составить программу-светофор, изменяющую цвет текстовой надписи.

**Этапы выполнения задачи**

 **Введение**

Delphi предназначен для профессионалов-разработчиков корпоративных информационных систем, однако часто Delphi используют с чисто прикладной целью, чтобы быстро решить задачи, не привлекая для этого программистов со стороны. Небольшие программы, созданные на Delphi, будут работать на любом компьютере.

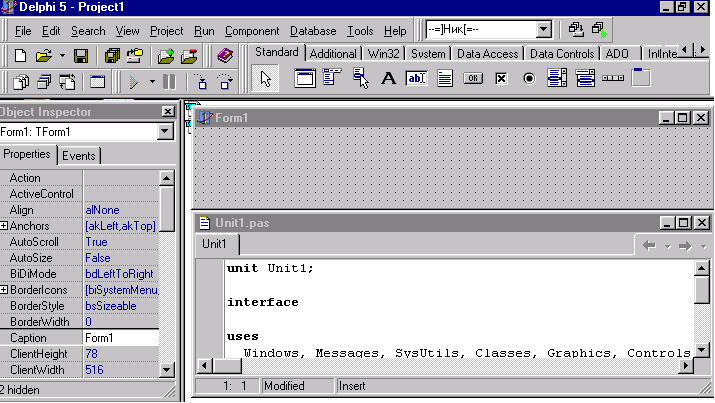
**Пользовательский интерфейс Delphi**

Внешний вид среды программирования Delphi отличается от многих других из тех, что можно увидеть в Windows. К примеру, Borland Pascal for Windows 7.0, Word for Windows, Excel – это все MDI приложения. MDI (Multiple Document Interface) определяет особый способ управления нескольких дочерних окон внутри одного большого окна.

Среда Delphi же следует другой спецификации, называемой Single Document Interface (SDI), и состоит из нескольких отдельно расположенных окон. Это было сделано из-за того, что SDI близок к той модели приложений, что используется в Windows. Окна могут перемещаться по экрану, частично или полностью перекрывая друг друга.

После запуска Delphi экран монитора приобретёт вид, показанный на рис. 4.1 (на рисунке показаны окна Delphi 5, для других версий окна могут иметь незначительные различия):

1. главное окно (заголовок Delphi5 - Project1);
2. окно формы (заголовок Form1);
3. окно Инспектора Объектов (Object Inspector);
4. окно кода программы (заголовок Unit1.pas).



Чтобы упорядочить окна так, как они показаны на рисунке, их размеры и местоположение можно изменить вручную (как вы делаете это с окнами в Windows). Расположение окон не обязательно должно совпадать с рисунком, их расположение не влияет на работу Delphi. Первоначально окно кода перекрыто окном формы, переключение между которыми осуществляется клавишей **F12.**

**Задание1.**

Первоначально создайте свою папку, например, на диске С:, в которой будут сохранены файлы программы.

Запустите систему программирования Delphi. Перечислите окна Delphi. Измените размеры окна формы и окна кода программы как на рисунке. Запомните назначение клавиши F12.

**Delphi - объектно-ориентированная среда**

В основе объектно-ориентированного программирования лежат понятия объект, класс, компонент.

**Объект** - нечто существенное и различимое. Объект является экземпляром класса.

**Класс** - расширенное понятие объекта. Классу присущи все характеристики объекта.

Например, рассмотрим класс ПОКУПАТЕЛЬ. Объектами данного класса (экземплярами класса) являются ПОКУПАТЕЛЬ ИВАНОВ, ПОКУПАТЕЛЬ ПЕТРОВ и т.д.

**Каждый класс и его объекты имеют:**

|  |
| --- |
|  |

* *Свойства* - характеристики объекта (ПОКУПАТЕЛЬ характеризуется возрастом, полом, местом работы и т.д.).
* *Методы* - действия объекта, что объект может делать (ПОКУПАТЕЛЬ может войти в магазин, спросить о товаре, купить товар и др.)
* *События* - изменения в окружающей объект обстановке (события, которые влияют на объект ПОКУПАТЕЛЬ: вопрос менеджера магазина, наличие нужного товара, нехватка времени и др.)
* *Реакция на события* - описания действий, которые необходимо совершить при данном событии.

**Компонент** - визуальный класс.

Примерами компонентов (визуальных классов) в Windows являются окна (в том числе и диалоговые), командные кнопки ("ОК", "ДА","ОТМЕНА" и т.п.), горизонтальное и контекстное меню, таблицы и многое другое. Почти все объекты Windows реализованы в Delphi в виде палитры визуальных компонентов VC. Компоненты сгруппированы в отдельные страницы, каждая из которых снабжена закладкой. Если вы щелкнете мышью на одну из закладок, то сможете перейти на соответствующую ей страницу.



При подведении курсора мыши к компоненте появляется подсказка - название компоненты.

**Задание 2**. Найдите палитру компонент в главном окне. Откройте вкладку Standard на палитре компонент.

**Свойства компонентов**

**Форма** - окно будущей программы. Окно формы представляет собой проект окна Windows: имеет заголовок, кнопку вызова системного меню, кнопку максимизации, минимизации и закрытия окна, рамку окна. На форме размещаются компоненты, тем самым, формируя окно будущей программы.

Рассмотрим первые несколько свойств компонент.

* Свойство Name - имя переменной, определяет как этот элемент будет называться в создаваемой программе. Имена создаются средой Delphi по такому принципу: сначала идет название компонента (Form), а за ним порядковый номер размещенного на форме компонента (1). То есть если добавить еще одну форму, то она получит имя Form2, следующее – Form3 и т.д. Имя, заданное по умолчанию можно изменить, но при этом желательно использовать только английские буквы и цифры. **Запомните**! Свойство Name задается первоначально, это имя переменной с которой вы будете работать при написании программы.
* Свойство Font - шрифт, позволяет выбрать параметры шрифта.
* Свойство Caption - надпись, создает надпись, в соответствии со свойством Font.
* Свойство COLOR - цвет, определяет цвет объекта.
* Свойство Width - определяет ширину объекта.
* Свойство Height - определяет высоту объекта.
* Свойство Left - указывает расстояние от объекта до левой границы формы.
* Свойство Тор - указывает расстояние от объекта до верхней границы формы.

**Окно Инспектора Объектов** содержит две страницы: *Properties (Свойства)* и *Events (События).* Страница *Свойства* служит для указания свойств объекта. На этой странице указаны все свойства активного (выделенного) объекта.

**Задание 3**. Для формы задайте следующие свойства: name - F1, размер 450\*400, цвет - зеленый (clgreen), шрифт - жирный курсив, заголовок (caption) - "Моя первая программа". Перейдите в окно кода программы, и вы увидите описание переменной F1 типа TForm.

**Компоненты TLabel, TButton, TPanel**

Сейчас, когда вы научились задавать свойства компонент, попробуем написать простую программу. Для этого на форме нужно разместить необходимые компоненты, задать их свойства, определить события и реакцию на событие.

Для размещения компоненты на форме нужно щелкнуть мышью на компоненте, после щелкнуть на форме.

**Компонент TLabel -** метка,служит для отображения текста на экране. У компонента TLabel есть свойство AutoSize, по умолчанию установленное в True. Эта установка позволяет метке увеличиваться и уменьшаться при изменении надписи, размера шрифта. Если цвет метки совпадает с цветом фона, то при сокращении надписи до нулевой длины, она может исчезнуть с экрана. В этом случае разыскать ее можно, перейдя в окно Инспектора Объектов, развернув список компонент и выбрав ее из этого списка.

**Компонент TButton** - командная кнопка. TButton позволяет выполнить какие-либо действия при нажатии кнопки во время выполнения программы.

**Компонент TPanel -** контейнер общего назначения. В контейнере отображается  текстовая строка часто по центру, поэтому компонент удобно использовать для вывода сообщения. У компоненте TPanel имеются свойства BevelOuter (внешняя рамка) и  BevelInner (внутренняя рамка). Они могут принимать значения bvRaised (выпуклая), bvLowered (вдавленная), bvNone (отсутствует).

**Запомните!** Определяя свойства любой компоненты, ее необходимо выделить.

**Задание 4**.

4.1.  Разместите в центре формы компоненту Tlabel. Если у Вас не получилось с первого раза установить компоненту по центру, не отчаивайтесь, вы можете переместить компоненту, удерживая левую кнопку мыши, или задав свойства Left и Top. По умолчанию свойство name = Label1. Именно с этим именем мы будем обращаться к метке. Установите свойство AutoSize= True. Задайте надпись метки = "Я учусь программировать на Delphi". Измените свойство Font. Задайте жирный шрифт 14 размера. Обратите внимание, что размер метки автоматически увеличился с увеличением размера шрифта.

4.2. Разместите ниже компонент TPanel. Задайте внешнюю рамку - вдавленной, внутреннюю - выпуклой, укажите надпись "Delphi - система программирования".

4.3. Расположите на панели три командные кнопки с именами Button, Button2, Button3. Задайте надписи на этих кнопках "Желтый цвет", "Красный цвет", "Голубой цвет".

4.4. Перейдите в окно кода программы и убедитесь, что в окне автоматически появилось описание всех компонент. Вы должны увидеть описание класса TF1 типа TForm, содержащей объекты: Label1, Button1, Button2, Button3, Panel1.

4.5. Вернитесь окно формы и еще раз уточните имя каждой компоненты формы. Дальше мы будем писать программу с указанием имен компонент, не запутайтесь в названии каждого.

**События программы**

**События** задают в окне Инспектора Объектов на странице Events (События):

* *OnClick* - событие щелчок мыши.
* *OnDbClick* - событие двойной щелчок мыши.
* *OnClose* - закрытие окна формы.

В нашей программе при нажатии мыши (событие OnClick) на командные кнопки будет соответственно изменяться цвет метки.

Зададим событие для первой командной кнопки Button1. Для этого выделите данный компонент, перейдите в Инспекторе Объектов на страницу Events и щелкните два раза мышью на нужном событии (в нашем случае на событии OnClick). После выбора события автоматически открывается окно кода программы.

**Окно кода программы** предназначено для создания и редактирования текста программы. Этот текст составляется по определенным правилам. Совокупность правил записи текста называется языком программирования. В системе программирования Delphi используется язык программирования Object Pascal.

После задания события в коде программы автоматически создается процедура обработки события, внутри которой описывается реакция на событие. Заголовок процедуры формируется следующим образом:

***Procedure*** *TF1. Button1Click (Sender:TObject);* {событие нажатие мыши на компоненте Button1, расположенной на форме F1 }

*Begin*

*End;*

Между Begin - End указывается **реакция на события** - **перечень совершаемых действий**. При нажатии на командную кнопку Button1 с надписью "желтый цвет" цвет метки должен измениться на желтый. В этом случае в процедуре нужно описать следующие действия:

***Procedure*** *TF1. Button1Click (Sender:TObject);*

*Begin*

*Label1.color:=clyellow;*

*End;*

**Задание 5.** Создайте поочередно события и реакцию на события для командных кнопок. Если вы не знаете, как записать название цвета, посмотрите возможные цвета свойства Color в Инспекторе Объектов.

**Модальные диалоговые окна**

Команда открытия окна с сообщением - модального диалогового окна:

*ShowMessage ('текст сообщения');*

Например, мы хотим вывести сообщение о том, кто разрабатывал данную программу, при завершении работы программы.

***Procedure*** *TF1. F1Close (Sender:TObject);*

*Begin*

*ShowMessage ( ' программа разработана ,,,,,,,,');*

*End;*

**Задание 6**. Для формы создайте событие закрытие программы. Выделите форму (щелкните на заголовке окна) и выберите событие OnClose. В окне кода программы опишите реакцию на событие в виде сообщения ФИО разработчика.

**Отладка, запуск, сохранение программы**

Сейчас можно сказать, что вы закончили написание программы. Остается только исправить ошибки и запустить на исполнение.

Перед этим сохраните программу.

**File / Save as …**- сохраняет программу с расширением .pas. Имя файла задается английскими буквами. Например, sdr.pas.

**File / Save project as …** - сохраняет проект, работающим с программой, с расширением .dpr. Имя проекта желательно формировать следующим образом: римя файла.dpr. Напрмер, psdr.dpr.

Выберите команду **Project/Compile** - для компиляции программы и обнаружение ошибок. В случае если отладчик обнаружит ошибку, он выделить эту строку красным цветом. В этом случае исправьте ошибку и заново запустите команду. Вы можете открыть команду Project/Information - итоги компиляции, надпись Succestfully будет свидетельствовать о том, что ошибки не обнаружены.

Теперь можно запустить программу на исполнение **Run/Run**. Ваша программа должна изменять цвета надписи, выводить сообщение о ФИО разработчика.

Возможно, ваша программа даст сбои при работе - все - таки это ваша первая программа. Тогда выполните команду **Run/ Reset** - прекращение работы программы.

 При сохранении программы создаются файлы:

**\*.pas** - файл модуля (текст в окне кода программы);

**\*.dpr** - файл проекта (основная программа);

**\*.dfm** - файл формы;

**\*.exe** - выполняемый файл, создается после запуска программы на исполнение.

**Запомните!** Для открытия программы в Delphi выбирается файл проекта.

**Задание 7.**

7.1. Сохраните программу в папке, которую Вы создали на 1 этапе. Сохраните файл проекта.

7.2. Откомпилируйте вашу программу. Исправьте ошибки, которые возникли. Запустите программу на исполнение.

7.3. Откройте Вашу папку. Перечислите созданные файлы и их назначение.

**Вопросы контроля**

**Cоставьте ответы в виде конспекта по следующим вопросам:**

1. Основные окна Delphi.
2. Понятие объекта, класса.
3. Назначение окна формы.
4. Две страницы окна Инспектора Объектов.
5. Назначение окна кода программы.
6. Понятия компоненты. Палитра компонент. Порядок размещения компоненты на форме.
7. Укажите окна, в которых задаются свойства, методы, события, реакция на события объектов.
8. Свойства: Name, Caption, Color, Left, Top, Width, Height, Icon, Autosize, Font.
9. Методы OnClick, OnDbClick, OnClose.
10. Расскажите, как формируется заголовок процедуры в окне редактора кода. Приведите примеры.
11. Команда задания модальных диалоговых окон.
12. Как осуществляется отладка и запуск программы на исполнение?
13. Назначение команд Run, Compile, Information, Reset.
14. Назовите расширения, с которыми сохраняется программа, проект. Как формируется имя файла программы, проекта? Какие еще файлы создаются при сохранении?

**Упражнения**

**1. Составьте программу, которая при выборе соответствующей командной кнопки, изменяет:**

*уровень А*

* а) содержание надписи на определение понятий "Язык программирования", "Система программирования", "Алгоритм";
* б) цвет формы на голубой, фиолетовый, серый;

*уровень В*

* в) содержание надписи на определение понятий "Язык программирования", "Система программирования", "Алгоритм";
* г) размер формы на "крупный" (600\*800), "средний" (400\*600), "мелкий" (200\*400);

*уровень С*

* д)вид отображения рамки панели "выпуклая", "вдавленная", "отсутствует".

**2. В программе (желательно созданной в п.1 или вновь созданной) добавьте вывод окна сообщения:**

*уровень А*

* а) об условии задачи при запуске программы;
* б) "Спасибо за работу" при завершении работы программы;

*уровень В*

* в)  Ф.И.О. автора при нажатии на соответствующую командную кнопку;

*уровень С*

* г) о назначении компонентов при щелчке мыши на данной компоненте.

**Раздел 2 ЛИНЕЙНЫЕ СТРУКТУРЫ**

**Цель:**

УЗНАТЬ:

* этапы решения вычислительной задачи;
* свойства компонента TEdit;
* этапы выполнения вычислительных задач.

НАУЧИТЬСЯ:

|  |  |
| --- | --- |
| маркированный список | рисовать линейные блок-схемы; |
| маркированный список | составлять простейшие программы вычисления; |
| маркированный список | формировать структуры проекта. |

**Теоретические сведения**

**Введение**

Из курса «Экономическая теория» и «Математика» Вам известно широкое использование в современном обществе экономических моделей. Экономические модели  позволяют выявить особенности функционирования экономического объекта и на его основе предсказывать будущее поведение объекта.

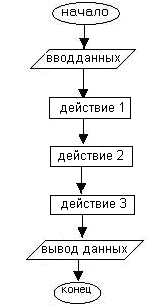
Экономическая модель – условный образ объекта, представленный в виде математического уравнения или системы уравнений.

В модели все переменные могут быть оценены количественно, что позволяет получать числовой прогноз. Например, *зависимость спроса y (ед.) от цены x (руб.) выражается степенной функцией: y=3,4\*x1,4*. Модель позволяет оценивать спрос на товар.

Программирование задач вычисление значения выражений, в том экономических моделей, не является трудоемким и, как правило, реализуется посредством линейных структур.

**Линейная алгоритмическая структура**

В линейной алгоритмической структуре действия выполняются последовательно. В вычислительных задачах алгоритм (блок-схема) начинается с ввода данных и заканчивается выводом результата.



**Основные понятия языка Object Pascal**

**Алфавит** языка Object Pascal включает буквы, цифры, специальные символы, пробелы, зарезервированные (ключевые слова).

**Цифры** – арабские цифры от 0 до 9. При записи числа целая часть от дробной отделяется точкой: 5.4.

**Ключевые (зарезервированные) слова** – это последовательность символов, имеющая в данном языке особый установленный смысл. Ключевые слова нельзя использовать в качестве идентификаторов. Ключевые слова выделяются жирным шрифтом.

Например, в Object Pascal имеются следующие зарезервированные слова:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **And**  **Array**  **Begin**  **Class**  **Const**  **Construction**  **Destruction**  **Div** | **Do**  **Downto**  **Else**  **End**  **for**  **File**  **Function**  **If** | **Implementation**  **In**  **Initialization**  **Interface**  **Mod**  **Nil**  **Not**  **Object** | **Of**  **Or**  **Out**  **Procedure**  **Program**  **Record**  **Repeat**  **Set** | **String**  **Then**  **To**  **Type**  **Until**  **Unit**  **Uses**  **Var** | **While**  **With**  **xor** |

**Идентификаторы** - это имена констант, переменных, типов, модулей, объектов и т.п. Идентификаторы могут иметь различную длину, но значащими являются только первые 63 символа. Идентификаторы всегда начинаются буквой, за которой могут следовать буквы и цифры, не одержат пробелов и специальных символов.

**Оператор** – минимальная структура в программе, производящая законченное действие. Оператор содержит ключевое слово, которое определяет его смысл.

**Данные. Структура типов данных. Простые типы данных**

Данные в Object Pascal бывают:

* константы (сохраняют свое значение в течение работы всей программы);
* переменные (могут изменять значение в результате работы программы);
* значение функций или выражений.

Любые данные характеризуются своими типами. Тип определяет:

|  |  |
| --- | --- |
| маркированный список | множество допустимых значений, которые может иметь то или иное данное; |
| маркированный список | множество допустимых операций, которые применимы к данному; |
| маркированный список | объем памяти, выделяемый для хранения данного. |

Object Pascal характеризуется разветвленной структурой типов данных.

*Данные простого типа хранят в себе только одно значение.* К простым типам относятся целые, вещественные, логические, символьные, перечисляемые, тип-диапазон.

**Целые типы** данных используются для представления целых чисел. Есть несколько различных целых типов, которые могут хранить целые значения. Различные целые типы имеют существенно различные диапазоны хранимых значений, затраты памяти растут с ростом допустимого диапазона значений.

**Таблица 1. Целые типы данных**

|  |  |  |
| --- | --- | --- |
| **Тип** | **Диапазон значений** | **Требования к памяти** |
| Byte | [0; 255] | 1 байт |
| Word | [0; 65535] | 2 байт |
| Shorting | [-128; 127] | 1 байт |
| Smallint | [-32768; 32767] | 2 байт |
| Integer | [-2147483648; 2147483647] | 4 байт |
| Cardinal | [0; 2147483647] | 4 байт |
| Longint | [-2147483648; 2147483647] | 4байт |

**Вещественные типы** данных предназначены для хранения чисел, имеющих дробную часть. В табл. 2 представлены несколько различных вещественных типов данных.

**Таблица 2. Вещественные типы данных**

|  |  |  |
| --- | --- | --- |
| **Тип** | **Диапазон значений** | **Требования к памяти** |
| Real | [2.9\*10-39; 1.7\*1038] | 6 байт |
| Single | [1.5\*10-45; 3.4\*1038] | 4 байт |
| Extended | [3.4\*10-4932; 1.1\*104392] | 10 байт |

**Логический тип данных** – один из простейших и часто используемых. Примером логического типа является ***Boolean*.** Переменные типа Boolean представляют собой логические значения True или False.

**Символьный тип данных**. Значениями символьного типа являются множество всех символов компьютера (буквы, цифры, арифметические знаки, орфографические знаки, специальные символы). Каждому символу приписывается код – целое число в диапазоне [0;255]. Код необходим для внутреннего представления символа.

Для кодировки в Windows используется код ANSI (назван по имени American National Standard Institute – Американского института стандартизации). Примером символьного типа данных является тип ***Char***, использующийся для хранения одного символа и в памяти отводится 1 байт.

**Математические выражения, операции и функции**

**Математические выражения** – это операции с числами, переменными, значениями функций.

Порядок выполнения операций в математических выражениях слева направо, при этом сначала выполняются действия в скобках, после умножение, деление, арифметические функции, а дальше сложение, вычитание. Числа записываются при помощи цифр, причем целая часть от дробной отделяется точкой.

Например: 4.5\*(b-2.2)/(c+5.1).

Числа могут быть представлены в показательной форме в виде:

1.76Е-3 – запись соответствует 1.76\*10-3=0.00176

3.765Е4 – запись соответствует 3.765\*104= 37650

Основные арифметические операции и функции, используемые при записи выражений, представлены в табл. 3.

**Таблица 3**.О**сновные арифметические операции и функции.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Операция** | **Действие** | **Тип операндов** | **Тип результата** | **Пример** |
| + | Сложение | Любой | Соответствует типу операндов | 4+5 = 9 |
| - | Вычитание | 9 - 4 = 5 |
| \* | Умножение | 2 \* 2 = 4 |
| / | Деление | Любой | Вещественный | 10/4 = 2.5 |
| Div | Целочисленное деление | Целый | Целый | 9 div 4 = 2 |
| Mod | Остаток от деления | Целый | Целый | 9 mod 4 = 1 |
| Abs | Модуль числа | Любой | Соответствует типу операнда | Abs(-5.3)=5.3 |
| SQR | Возведение в квадрат | Любой | Соответствует типу операнда | SQR(4)=16 |
| SQRT | Квадратный корень | Любой | Вещественный | SQRT(78)=2.97 |
| Cos | Косинус | Любой | Вещественный | Cos(4)=0.997 |
| Sin | Синус | Любой | Вещественный | Sin(4)=0.069 |
| Exp | Экспонента | Любой | Вещественный | Exp(4) |
| Ln | Натуральный логарифм | Любой | Вещественный | Ln(4)=1.38 |
| Round | Округление до ближайшего целого | Вещественный | Целый | Round(4.5)=5 |
| Trunc | Возвращает целое число путем отбрасывания дробной части | Веществен-ный | Целый | Trunc(4.5)=4 |
| exp(b\*ln(a)) | Возведение в степень ab | Любые | Соответствует типу операндов | exp(3\*ln(2))=8 |

**Оператор присваивания**

Это наиболее часто используемый оператор языка. Синтаксис оператора присваивания: *идентификатор:= выражение;*

Символы «:=» всегда пишутся слитно, без разделяющих пробелов, хотя перед двоеточием и после знака равенства можно для лучшей читаемости программы вставлять пробелы. Как и любой другой оператор языка, оператор присваивания завершается точкой с запятой. Переменные и результат выражения должны быть одного типа.

**Операторные скобки**

**Составной оператор** - это последовательность произвольных операторов программы, заключенная в операторные скобки – **begin … end**. Составной оператор может состоять из любого количества операторов:

***begin***

*оператор 1;*

*оператор 2;*

*...*

*оператор N;*

***end****;*

**Функции перевода числа в строку, строку в число**

##### В окне формы ввод-вывод данных осуществляется при помощи компонента класса TEdit. Введенное или выводимое значение представляет собой строку, например, Edit1.text:String.

##### В модуле программы строки переводятся в числа, над которыми выполняются математические операции.

Преобразования числа в строку

**STR(x, st: string) –**  преобразует числовое значение х в строку символов st.

**IntToStr (х: integer)** –  преобразует целое число х в строку.

**FloatToStr (y: real)** – преобразует вещественное число в строку.

Преобразования строки в число

**StrToInt (st:string)** –  преобразует строку в целое число.

## StrToFloat (st:string) – преобразует строку в вещественное число.

**Val(st:string, x, code:integer) –** преобразует величину st в числовую величину вещественного или целочисленного типа x. Если преобразование успешно, то переменной cod присваивается значение 0; если обнаруживается ошибочный символ, то значение х не определяется, а переменной cod присваивается значение номера ошибочной позиции.

**Задача**

Зависимость спроса y (ед.) в сутки на 1000 жителей от цены x (руб.) выражается степенной функцией[\*](file:///E:\\Мои%20документы\\Delphi\\Programmirovanie\\baza\\l3.htm" \l "_ftn1" \o "): y=3,4\*x2,6. Составить программу оценки спроса на товар.

**Этапы выполнения задачи**

**Анализ требований программы**

В задаче вычисления значения выражения используются данные:

* *входная переменная: x*
* *выходная переменная: y*

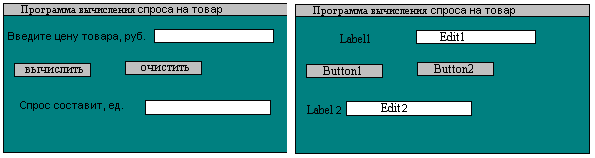
Пусть переменная х будет целочисленная. Результат y=3,4\*x1,4 будет  вещественным числом. Описание переменных примет вид:

* *x: целое число;*
* *y: вещественное число;*

**Задание 1**. Запишите задание и переменные программы в тетрадь.

**Окно будущей программы**

***Для визуального проектирования*** зарисовываются два окна. В первом окне определяется внешний облик окна, во втором окне указываются используемые элементы управления (в частности, компоненты).



В окне исполняемой программы «ответ» первоначально должен быть невидим. Поэтому для компонентов вывода Label2 и Edit2 устанавливается свойство видимости visible=false. Напомним, что для всех компонентов по умолчанию свойство visible=true (компоненты видны на форме).

Свойства объектов формы:

**Форма**: name=S1, caption=Вычисление спроса на товар, color  ....

**Метка**: name=Label1, caption= Введите цену товара, руб., font (шрифт) = 12 пт

name=Label2, caption=Спрос составит, ед., font (шрифт) = 12 пт, visible=false

**Командная кнопка** name Button1, caption=вычислить

name Button2, caption=очистить

**Однострочное окно** name=Edit1, text � (пустая строка)

name=Edit2, text � (пустая строка), visible=false

**Задание 2**. Создайте в Delphi окно будущей программы.

**Описание событий**

После визуального проектирования окна формы определяются события программы. В программе будут происходить события:

     щелчок мыши на командной кнопке "Вычислить".

     ДОПОЛНИТЕЛЬНО: щелчок мыши на "Очистить".

Для каждого события в Delphi создаются процедуры:

**Procedure** TS1.Button1Click(); **Procedure** TS1.Button2Click();

**Задание 3**. Создайте события программы.

**Алгоритмизация и программирование**

На этапе алгоритмизации составляется алгоритм каждого события. Алгоритмы реализуются на языке программирования.

Процедура «**Вычислить**» примет вид:

|  |  |
| --- | --- |
| **Procedure** TS1.Button1Click();  **Var**  x:Integer; y: real;  **begin**  x:=StrToInt(edit1.text);  y:=3.4\*exp(1.4\*ln(x));  label2.visible:=true;  edit2.visible:=true;  edit2.text:=FloatToStr(S);  **end;** | **Заголовок** подпрограммы  **Переменные**  х:целое; у:вещественное; |

Алгоритм и процедура «**Очистить**» следующие:

|  |  |
| --- | --- |
| **Procedure** TS1.Button2Click();  **begin**  edit1.clear;  edit2.clear;  label2.visible:=false;  edit2.visible:=false;  edit1.SetFocus;  **end;** | **Заголовок** подпрограммы |

При программировании события "Очистить" используются методы компонента TEdit:

* **Clear** -удаляет весь текст.
* **SetFocus** - устанавливает фокус ввода (курсор).

**Задание 4**. Зарисуйте алгоритмы событий программы. Реализуйте процедуры.

**Тестирование и отладка программы**

Напомним, тестирование – обнаружение ошибок, отладка – их исправление. Ошибки бывают синтаксические и структурные. Как уже говорилось, при запуске программы осуществляется компиляция, в процессе которой обнаруживаются синтаксические ошибки. После исправления синтаксических ошибок работающую программу проверяют на наличие структурных ошибок. Например, в нашем задании необходимо на бумаге задать значение х. При помощи калькулятора посчитать значение выражения. Ввести эти данные в программу и проверить вычисленный результат с рассчитанным Вами.

Пусть цена товара x=50 руб., тогда спрос у=3,4\*501,4 = 812,90 (руб.).

**Задание 5**. Приведите свой пример проверки результатов программы.

**Исполняемая программа**

Конечный программный продукт должен быть подготовлен к реализации – в нашем случае к демонстрации студенческой группе и преподавателю. Для этого:

**a)** **Необходимо сохранить программу**. *Например, сохраним модуль с именем  Sproc.pas.Файл проекта удобно формировать путем добавления начальной буквы P – Pspros.dpr*

***b)* Убедиться, что файл модуля прикреплен к проекту**. *Команды Project/View source, Project/Add to project, Project/Remove from  project.*

***c)*Желательно подготовить сопроводительную информацию**: ФИО разработчика, формулировка задачи и др. *Например, сопровождение занятия "Моя первая программа".*

**Структура модуля** в окне кода программы примет вид:

**unit** Sproc; {*имя модуля совпадает с именем файла Sproc.pas*}

**interface**

{список модулей, с которыми устанавливается связь}

**uses** Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

**type**  Ts1 = class(TForm)  {объявления типа TS1 класса формы}

Label1: TLabel;  {перечень компонентов формы типа TS1}

Label2: TLabel;

Edit1: TEdit;

Edit2: TEdit;

Button1: TButton;

Button2: TButton;

**procedure** Button1Click(Sender: TObject);   {объявления заглавия процедур}

**private**   {переменные раздела **private** доступны в пределах данного модуля }

**public**    {переменные раздела **public** доступны в пределах модуля и за его пределами}

**end;** {завершение описания типа}

**var** s1: Ts1;   x:Integer; y: real;   {**глобальные переменные модуля**}

**implementation***{$R \*.DFM}*

**procedure** Ts1.Button1Click(Sender: TObject);

**var**  x:Integer; y: real;   {переменные доступны внутри процедуры – **локальные переменные**

**begin                                                                                                                      подпрограммы**}

x:=StrToInt(edit1.text);

y:=3.4\*exp(0,4\*ln(x));

label2.visible:=true; edit2.visible:=true;

edit2.text:=FloatToStr(S);

**end; end.**

Обратите внимание, что переменные можно описывать в разных  разделах, в зависимости от требований программы.

Текст проекта (*Project/View source)* принимает вид:

**program** Psproc;

**uses**Forms,

 Sproc in 'Sproc.pas' {s1}; *{подключили модуль Sproc и форму S1  файла ‘Sproc.pas’ }*

{$R \*.RES}

**begin**

  Application.Initialize;

  Application.CreateForm(Ts1, s1);

  Application.Run;

**end.**

[**Упражнения**](file:///C:\документы\информатика\inform\metodic\2%20tema2\Progr2\ypragn.htm)

1. Производственная функция Кобба-Дугласа имеет вид: P=5\*L0,4\*K0,6, где P-объем продукции, L-затраты труда, K-величина капитала. Составить программу вычисления объема продукции.
2. Объем V выработки молока фермерского хозяйства описывается моделью: Vt0 (1997) = 5,6\*e3(t-t0) . Составить программу вычисления V в году t.
3. Спрос на получения кредита S (руб.) семьи в год для семьи со среднемесячный доход семьи на одного человека D (руб.) и  количество детей в семье m  описывается моделью: S = 3,4+5m +4,3LnD. Составить программу вычисления спроса.
4. Для вклада S руб. под n% годовых рассчитать сумму через год.
5. Программа вычисления стоимости телефонного разговора. Запрашивает стоимость минуты  и длительность разговора, а затем вычисляет стоимость разговора.
6. Для суммы S (руб.) делается скидка n%. Рассчитать скидку в руб. и сумму оплаты.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Раздел 3 ВЕТВЛЕНИЕ**  **Цель:**  УЗНАТЬ:   * оператор ветвления, его синтаксис, блок-схема; * правила записи сложных условий неравенств типа 0<x<10 и др.; * возможные состояния флажка TcheckBox, его свойства; * понятие независимый переключатель.     НАУЧИТЬСЯ:   |  |  | | --- | --- | | маркированный список | рисовать линейные блок-схемы; | | маркированный список | составлять простейшие программы вычисления; | | маркированный список | формировать структуры проекта. |   **Теоретические сведения**  **Введение**  Ветвление  - основная алгоритмическая структура, в которой в зависимости от выполнения или невыполнения условия выполняется одна или другая цепочка действий.  Существует два типа ветвления: *полное* и *неполное*.  **Оператор ветвления**   Структура полного ветвления имеет следующий вид:  **If <**условие> **then <**оператор1> **else <**оператор 2>;  l4  Условный оператор работает по следующему алгоритму. Сначала вычисляется условие. Если результат условия True (истина), то выполняется оператор1, а оператор2 пропускается. Если результат условия False (ложь), то выполняется оператор2, а оператор1 пропускается.  Структура неполного ветвления имеет следующий вид:  **if** условие **then** оператор;  l4                  Неполный условный оператор работает по следующему алгоритму. Сначала вычисляется условие. Если условие принимает значение TRUE, то выполняется оператор, иначе никаких действий не происходит.  **Запись условия**  Сложные логические выражения составляются с использованием логических связок **and, or, not, xor**.  Выражения можно сравнивать при помощи операции отношения:  *= равно;*    *<> неравно;*    *< меньше;*    *> больше;*    *<= меньше или равно;*    *>= больше или равно.*   Результатом применения операций отношения является логическое данное True или False. Например, результатом выражения 3>1 является True, а результатом отношения 3<>1 - False.  Сложные логические выражения составляются с использованием логических связок:  ***Not*** - *логическое НЕ;*  ***And*** *- логическое И;*  ***Or*** - *логическое ИЛИ;*  ***Xor*** - *исключительное ИЛИ.*    **Таблица 1**. Логическая операция **And** над данными логического типа.   |  |  |  | | --- | --- | --- | | **Операнд1** | **Операнд2** | **Результат выражения о*перанд1 and операнд2*** | | True | True | True | | False | True | False | | True | False | False | | False | False | False |     **Таблица 2**. Логическая операция **Or** над данными логического типа.   |  |  |  | | --- | --- | --- | | **Операнд1** | **Операнд2** | **Результат выражения** *операнд1* ***or****операнд2* | | True | True | True | | False | True | True | | True | False | True | | False | False | False |     **Таблица 3**. Логическая операция **Xor** над данными логического типа.   |  |  |  | | --- | --- | --- | | **Операнд1** | **Операнд2** | **Результат выражения о*перанд1 xor операнд2*** | | True | True | False | | False | True | True | | True | False | True | | False | False | False |     **Таблица 4**. Логическая операция **Not** над данным логического типа.   |  |  | | --- | --- | | **Операнд1** | **Результат выражения  not (о*перанд)*** | | False | True | | True | False |     **Флажок . Математические выражения, операции и функции**  **Задача 1**  **Программа вычисления значения функции на промежутках**      E:\Мои документы\Delphi\Programmirovanie\images\l4.htm1.gifE:\Мои документы\Delphi\Programmirovanie\images\l4.htm2.gif  **Этапы выполнения задачи**  **Анализ требований программы**  В задаче вычисления значения выражения используются данные:   * *входная переменная: x* * *выходная переменная: y*   Переменные х и у вещественные.  **Задание 1**. В тетради запишите условие задачи 1. На оси ОХ отметьте промежутки, на которых функция описывается различными уравнениями.  **Окно будущей программы**  Для решения задачи форма формируется согласно рисунку задачи1. Для компонентов TForm, TEdit, TLabel задаются уже известные свойства.  Форма: name = Form1,  Caption = Вычисление значения функции.  Метка: name = Label1, Caption = Введите Х,  Autosize=true.  Аналогично Label2.  Однострочное окно: name = Edit1, Text   .  Аналогично Edit2.  **Задание 2**. В Delphi создайте окно будущей программы (окно формы).  **Алгоритмизация**  Алгоритм вычисления значения функции определяет промежуток, к которому принадлежит введенное значение х.  В алгоритме проверяются условия:  1 условие: x<0  2 условие: х>1  3 условие: x э [0;1]  - сложное, состоит из двух частей (x<=0) and (x>=1).  Блок-схема задачи содержит три неполных ветвления поверки каждого условия.  **Задание 3.** Зарисуйте блок-схему в тетрадь. Выделите три блока ветвления.  **Программирование**  В программе будут происходить два события:  1.  Щелчок мыши  по «Вычислить»: **Procedure** TForm1.Button1Click();  2.  Дополнительно «Очистить»: **Procedure** TForm1. Button2Click();  Программирование алгоритма «Вычислить» записывается:  **procedure** TForm1.Button1Click(Sender: TObject);  **var** x,y:real;  **begin**     x:=strtofloat(edit1.text);     if (x>=0) and (x<=1) then y:= x\*x\*x+3\*x +4;     if (x<0) then y:=sqr(x\*x\*x +3\*x+4);     if (x>1) then y:= 8;     edit2.text:=floattostr(y);  **end;**  **Задание 4.**  4.1. Создайте событие "Вычислить" и запишите реакцию на него.  4.2. Дополнительно: Напишите реакцию на событие "Очистить" самостоятельно (см. лаб./работу №2).  **Тестирование и отладка программы**  Наличие структурных ошибок проверяется тремя частными вариантами.  1)Ввести значение х ' [0;1]. Например, при х=0,5 y=x3 + 3\*x +4=5,625.  2) Ввести значение х<0. Например, при х=-2 y=(x3 + 3\*x +4)2=100.  3) Ввести значение х>1. Например, при х=2 y=8.  Расчетные результаты  должны совпадать с результатами работающей программы. Несовпадение результатов показывает наличие структурных ошибок алгоритмизации или программирования.  **Задание 5.** Сохраните модуль и проект. Запустите программу на исполнение. Осуществите тестирование и отладку программы.    **Задача 2.**  **Программа нахождения максимального значения**  Max (5x2 -4y; ey; ln (x/2))    **Этапы выполнения задачи 2**  **Определение требований программы**   Для нахождения max (5x2-4y; ey; ln(x/2)) рассчитываются три значения:  ü  а1 = (5x2 -4y)  - первое значение;  ü  а2 = ey  - второе значение;  ü  а3 = ln (x/2) – третье значение,  Входными  данными в задаче будут переменные х, у.  Промежуточными данными являются переменные a1, a2, a3.  Выходом будет переменная max среди переменных a1, a2, a3.  **Задание 1.** Опишите в тетради входящие и выходящие данные.  **Окно формы**  На рисунке задачи 2 компоненты имеют известные Вам свойства.  OKNO2213  *Рис.*  Проектирование формы задачи 2.  Усложним форму указанием видимости компонентов. Компоненты  входных данных (Edit1, Edit2) сделаем видимыми при открытии формы: установим свойство visible=true. Компоненты отображения результатов (Edit3 – Edit6) первоначально невидимы: установим свойство visible=false.  **Задание 2**. В Delphi на оформите окно формы. Сохраните модуль и проект.  **События и реакция на события программы**  В программе событие «Вычислить» реализуется процедурой:   |  |  | | --- | --- | | **procedure** TForm1.Button1Click(Sender: TObject);  **var** x,y,a1,a2,a3, max: real;  **begin**x:=strtofloat(edit1.text);  y:= strtofloat(edit2.text);  a1:=5\*x\*x-4\*y\*y; Edit3.text:=floattostr(a1);  **Edit3.visible:=true;**  a2:=exp(y); Edit4.text:=floattostr(a2);  **Edit4.visible:=true;**  a3:=ln(x/2); Edit5.text:=floattostr(a3);  **Edit5.visible:=true;**  if (a1>a2) and (a1>a3) then max:=a1;  If (a2>a1) and (a2>a3) Then max:=a2;  IF( a3>a1) and (a3> a2) then max:=a3;  Edit6.text:=floattostr(max);  **Edit6.visible:=true;**  **end;** | *Обратите внимание, что максимальное значение присваивается переменной max.* |   Типичная ошибка a1:=max  *изменяет значение переменной a1* и нигде не записывает максимальное значение (max - неопределенно).  **Задание 3**. Создайте событие «Вычислить» и реакцию на него.  **Тестирование**  Найдем max (5x2-4y; ey; ln x/2) при  x=2, y=3.   Для этого значения a1, a2, a3 рассчитываются в Пуск/Программы/ Стандартные/ Калькулятор.Среди a1, a2, a3 находится максимальное значение.  **Задание 4**. Сравните полученные Вами значения с результатом программы.  **Задание 5.** Алгоритмизация: Зарисуйте в тетради блок-схему решения задачи.  **Дополнительно.** Создайте кнопку «Условие задачи» с выводом окна (рис. 3.3).  㿷ᛷΨ  *Рис.* Окно сообщение задачи 2.    **Задача 3. Программа "Флажок"**      *Рис.* Окно задачи 3.  **Этапы выполнения задачи 3**  **Определение требований программы**  В рассматриваемой задаче:  ü вход: х (целое число);  ü выход: y1 (модуль числа х) и y2 (квадрат числа х).  **Задание 1. В** тетради опишите тип переменных x, y1, y2. Запишите функции нахождения модуля и квадрата числа.  **Окно формы**  В окне формы компоненты располагаются, как показано на рис. 3.5.    *Рис.* Проектирование окна формы задачи 3.  Для компонентов TForm, TEdit, TLabel задаются уже известные Вам свойства.  Форма: name = Form1, Caption = Программа вычисления модуля и квадрата.  Метка: name = Label1, Caption = Введите Х, Autosize = true.  Флажки: name = CheckBox1, Caption =Модуль                name = CheckBox2, Caption =Квадрат  Однострочное окно: name = Edit1, Text ?.  Аналогично Edit2, Edit3. Установите окна результата невидимыми.      **Флажок.** Кроме известных компонентов на форму помещается новый компонент флажок. Флажок может принимать три состояния, которые задаются свойством **State**:  ДА – cbChecked;  НЕТ – cbUnchecked;  НЕ ЗНАЮ – cbGrayed.  **State** = [cbChecked,  cbUnchecked, cbGrayed].  Чаще всего флажок принимает только два состояния ДА-НЕТ. Свойство **AllowGrayed**=[true, false] включает/выключает третье состояние - НЕ ЗНАЮ.  **Задание 2**.  2.1. Запишите в тетради свойства компонента TCheckBox: AllowGrayed и State.  2.2. Оформите окно формы Delphi.  2.3. Изучите состояния флажков. Поэкспериментируйте: для CheckBox1 задайте включен (ДА), для CheckBox2 - НЕ ЗНАЮ.  2.4. Установите для флажков свойства:   Флажок: name = CheckBox1 (CheckBox2)                Caption = модуль (квадрат)               State - флажки выключены.  **События изменения состояния флажков**  Событие щелчок мыши по  кнопке «модуль» - CheckBox1Click, находит абсолютное значение числа Х, записанное в Edit1, и выводит его в Edit2.  **Procedure** TForm1.CheckBox1Click (Sender: TObject);  **var** x,y1: integer;  **begin**  x:=StrToInt(edit1.text);  y1:=abs(x);  edit2.text:= IntToStr(y1);  **if** checkbox1.state=cbChecked  **then** edit2.visible :=true  **else**  edit2.visible :=false;  **end;**  Обратите внимание на условие для флажка: *checkbox1.state=cbChecked.* Условие истинно при включенном флажке. В этом случае пользователю выводится результат. При ложном условие флажок выключен, следовательно, результат не отображается на экране.  **Задание 3**. Создайте события для двух флажков, реакции на эти события.  **Тестирование и отладка**  Введите отрицательное значение х, например, х=-2. Выберите флажок "модуль" и должны получить результат. Выключите флажок - результат исчезнет. Аналогично протестируйте флажок "квадрат".  Обратите внимание, Вы можете включить два флажка вместе, можете отключить все флажки, включать только один. Говорят, что состояние одного флажка не зависит от состояния других, поэтому **флажок - независимый переключатель**.  **Задание 4**. Выпишите в тетрадь понятия независимый переключатель. Создайте исполняемую программу. Не забудьте сохранить программу. |
| **Упражнения**  1.     Вычислить Y – размер ВВП страны в зависимости от:  а – индекса расходов на сельское хозяйство к бюджету страны (от 0 до 1)  b – индекса расходов на промышленность к бюджету страны (от 0 до 1)  z – индекс расходов на образование к общему бюджету страны (от 0 до 1)  x =a-b  – разница индексов расходов на сельское хозяйство и промышленность     |  |  |  |  | | --- | --- | --- | --- | |  | *Уровень начальный* |  | *Уровень средний* | | А |  | E | C:\Users\User\Application Data\Microsoft\Word\информатика\inform\metodic\2 tema3\Progr\ypragn9.gif | | B |  | F | C:\Users\User\Application Data\Microsoft\Word\информатика\inform\metodic\2 tema3\Progr\ypragn11.gif | | C |  | G | C:\Users\User\Application Data\Microsoft\Word\информатика\inform\metodic\2 tema3\Progr\ypragn10.gif | | D | C:\Users\User\Application Data\Microsoft\Word\информатика\inform\metodic\2 tema3\Progr\ypragn8.gif | I | ~~ypragn11~~ |   *Уровень высокий.* Дополнить программу флажками. Задать три флажка, соответствующие каждому промежутку. При выводе результата включать соответствующий флажок.  2. Даны x, y, z. Составить программу нахождения максимального элемента   |  |  |  |  | | --- | --- | --- | --- | |  | *Уровень начальный* |  | *Уровень средний* | | A | max (x+y+z, 3x, y-4z) | E | min (sin(x2+y2), tg(z), x/y) | | B | min (x2+y2, |z|, xyz). | F | max (5x2-4y2), ey , ln|x/2|) | | C | max (x2+y2-z2, xyz, (x-y)4) | G | max (sin(x+y), tg(10-2z), 6y3) | | D | min (|x+y|, (z/x)2, 45/x3) | I | max (ln|x-y|, tg(z), x/y). | |  |  |
| **Творческие задания**  1.    Напишите программу расчета скидки на покупку.  Если сумма покупки составляет более 500 р., то покупателю предоставляется скидка n%.  2.    Известна величина прожиточного минимума на человека в месяц. Напишите программу, которая для семьи n человек с доходом S включает флажок выше или ниже прожиточного минимума. Дополнительно: предусмотреть возможность равенства значений.  3.    Некоторый товар можно приобрести в трех торговых точках. В каждом магазине известна цена товара S1, S2, S3 и предоставляемые скидки r1, r2, r3. Найти минимальную цену с указанием магазина. Вывести реальную цену покупки в каждом магазине.  4.     Стоимость печати стандартной визитки S рублей. При печати визитки на глянцевой бумаге стоимость увеличивается на 10%. Нанесение логотипа на визитку увеличивает стоимость печати на 5%. Печать более 100 экземпляров уменьшает стоимость на 8%. Написать программу расчета стоимости печати одной визитки. В программе выбор параметров осуществляется флажками «Глянцевая бумага», «Логотип», «Объем более 100 экз.». |  |  |
| **Работа с блок-схемам**  1.  Укажите значение следующего выражения: (x>y) **и** ( p **или** (y>z)) при:  **а)** x = 5, y = 3, z=10, p=**нет**;  **б)** x = -10, y = -3, z=-5, p=**да**;  в) x = 0, y = 1, z=-1, p=**да**;  г) x = 1, y = 0, z = 5, p = **да**;  **д)** x = 5, y = 0.5, z = -4, p = **нет**;  е) x = 8, y = - 4.5, z = 7, p = **нет**;  ж) x = -4, y = 5, z = 8, p = **нет**;  з) x = -15, y = -25, z = 10, p = **да**;  2.  Фрагмент алгоритма изображен в виде блок-схемы. Определите значение переменной S. Запишите процедуру.    А) а = 3, b = 4         Б) а = -3, b = 10  В) а = 6, b = -3        Г) а = 8, b = 0  Д) а = -3, b = -4       Е) а = 2, b = 15  Ж) а = 3, b = -8       З) а = 1, b = -5 |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Раздел 4 ОПЕРАТОРЫ ВЫБОРА**  **Цель:**  УЗНАТЬ:   * правила записи констант выбора оператора CASE; * представление набора строк в TListBox и TRadioGroup; * выбор радиокнопок, понятие зависимый переключатель; * различия в использование флажка и радиокнопки; * отличия компонентов TMemo и TListBox, TMemo и TEdit, TRadioGroup и TCheckBox,   TListBox и TRadioGroup.   НАУЧИТЬСЯ:   |  |  | | --- | --- | | маркированный список | составлять программы с множественным выбором решения; | | маркированный список | использовать компоненты TRadioGroup и TListBox для выбора данных; | | маркированный список | выделять результат в списке через компоненты TRadioGroup и TlistBox; | | маркированный список | выводить результат в TMemo. |   **Теоретические сведения**  **Введение**  Структура выбора  - структура множественного ветвления, не является классической алгоритмической структурой.  **Оператор выбора**  Оператор выбора позволяет выбрать одно из нескольких возможных продолжений программы. Параметром, по которому осуществляется выбор, служит **ключ выбора** - выражение любого целого, символьного и логического типа.   Структура оператора выбора такова:  **case** <ключ выбора**> of**   <константа выбора>: оператор 1;   <константа выбора>: оператор \*;   .............   <константа выбора>: оператор N;  **else** оператор;  **end;**  Оператор выбора работает следующим образом. Вычисляется значение выражения <ключ выбора>, затем полученное значение последовательно сравнивается с константами выбора из списка констант. Если значение выражения совпадает с константой из списка, то выполняется соответствующая этому списку последовательность операторов, и на этом выполнение инструкции **case** завершается. Если значение выражения не совпадает ни с одной константой из всех списков, то выполняется последовательность операторов, идущих после **else.**  Константа выбора может содержать не одно, а несколько значений, разделенных запятой, или представлять диапазон значений, в котором начальное и конечное значения разделены двумя точками.  Например:  **case**  N **of**  1 .. 4: s := a;  5 .. 9 : s := a-5;  10,11: s:= 2\*a;  **else** s:= 0;  **end ;**  **TRadioGroup - Радиокнопки**    **TListBox - Список строк**    **TMemo - многострочное окно ввода - вывода**  **Задача 1.**  **Программа выбора прессы в TListBox по дню недели**      **Этапы выполнения задачи**  **Анализ требований программы**  В задаче 1:  входная переменная: n: целого типа (вводится в Edit1 - номер дня недели).  результат: действие - выделение строки в списке изданий TListbox.  **Задание 1.** В тетради опишите этап 1.  **Окно будущей программы**      В окне формы располагаются известные TEdit, TLabel, TButton.  Зададим свойства этих компонентов.  Форма: name = F1, Caption = Программа 1 выбора прессы по дню недели.  Метки: name = Label1, Caption = Введите номер дня недели.               name = Label2, Caption = Издания в этот день.  Окно ввода: name = Edit1, Text ð.  Командная кнопка: name = Button1, Caption = OK.  Разместим на форме новый компонент с именем  ListBox1 в виде прямоугольного окна. Размер компонента можно изменять при помощи мыши.  TListBox представляет список строк, для обращения к которому используется свойство C:\Users\User\Application Data\Microsoft\Word\информатика\inform\metodic\2 tema4\Progr\OKNO122.gif.    Items (список строк) относится к классу TStrings и ему присущи все свойства и методы данного класса. Свойство Items имеет кнопку […] для открытия окна списка строк.  C:\Users\User\Application Data\Microsoft\Word\информатика\inform\metodic\2 tema4\Progr\OKNO123.gif  *Рис.* Окно списка строк компонента TlistBox.  В окне списка набирают строки, которые **индексируется с нуля.**  В нашей программе у компонента ListBox1 свойство *Items* равно:   |  | | --- | | «Эксперт», «КП»;  «Власть», «КП»;  «КП»;  «КП», «АиФ», «АмК», «Коммерсант»;  «КП», «МК», «Деньги», «Тема»;– общий выпуск», «Персональпьютер»;т издан |            Для выделения строки в списке служит свойство *iteminde*x:  = -1 при отсутствии выделения;  = 0, 1, 2, … -  номер выделенной строки.  Первоначально окно строк невидимо, свойство visible = false.  **Задание 2.**  2.1.  В Delphi на форме расположите указанные выше компоненты.  2.2. Для ListBox1 выделите первую строку (какой номер она будет иметь?).  2.3. Сохраните модуль и проект.  2.4. Запишите в тетради свойства TlistBox: Items, ItemIndex.  **Алгоритмизация и программирование**  Оператор выбора не является классической алгоритмической структурой, поэтому не имеет установленной блок-схемы. При реализации алгоритма  будем использовать словесной способ его представления.   |  |  | | --- | --- | | *Алгоритм  события "ОК"*  **переменная: n: целого типа;**  **начало**  ввод данных в Edit1 (номер недели) - переменная n;  **выбор** значения переменной  n из списка:       1 (понедельник): в ListBox1 выделить строку [0];       2 (вторник): в ListBox1 выделить строку [1];       3 (среда): в ListBox1 выделить строку [2];       4 (четверг): в ListBox1 выделить строку [3];       5 (пятница): в ListBox1 выделить строку [4];       6 (суббота): в ListBox1 выделить строку [5];      7 (воскресенье): в ListBox1 выделить строку [6];  **иначе** снять выделение (неверный день недели);  **конец выбора;**  сделать видимым ListBox;  **конец** | **Procedure** TF1.Button1Click ();  **var** n: integer;  **begin**  n:=StrToInt(edit1.text);  **case** n **of**        1:ListBox1.itemindex:=0;        2:ListBox1.itemindex:=1;        3:ListBox1.itemindex:=2;        4:ListBox1.itemindex:=3;        5:ListBox1.itemindex:=4;        6:ListBox1.itemindex:=5;        7:ListBox1.itemindex:=6;  **else** ListBox1.itemindex:= -1;  **end;**  ListBox1.visible:=true;  **end;** | | **Дополнительное событие** |  | | *Алгоритм  события "Очистить"*  **начало**  Очистить Edit1;  Поместить курсор в Edit1;  Снять выделение строк в ListBox1;  Сделать невидимым ListBox1;  **Конец** | **Procedure** TF1. Button2Click(…);  **begin**  Edit1.clear;  Edit1.setfocus;  ListBox1.itemindex:=-1;  ListBox1.visible:=false;  **end;** |   **Задание 3.** В тетради запишите алгоритм и программу события «ОК». Создайте событие «ОК». По-возможности, создайте событие «Очистить».  Зарисуйте блок-схему в тетрадь. Выделите три блока ветвления.  **Тестирование и отладка программы**  После запуска программы на исполнение программа проверяется на наличие структурных ошибок. При вводе дня недели понедельник (1) выделяется первая строка, воскресенья (7) - седьмая строка списка изданий и т.д. При вводе неверного дня недели (8 и др.) -  выделение снимается.  **Особенность работы TListbox** изучается при работе с выполняемой программой. TListbox – набор строк, поэтому щелчок мыши на строке приводит к ее выделению. Строки нельзя редактировать и удалять.  **Задание 4.** Завершить построение исполняемой программы. Изучите особенности компонента TlistBox.    **Задача 2.**  **Программа вывода дня выхода выбранного издания.**  **Компоненты TRadioGroup, TMemo.**      **Этапы выполнения задачи 2**  **Определение требований программы**   Для нахождения max (5x2-4y; ey; ln(x/2)) рассчитываются три значения:  ü  а1 = (5x2 -4y)  - первое значение;  ü  а2 = ey  - второе значение;  ü  а3 = ln (x/2) – третье значение,  Входными  данными в задаче будут переменные х, у.  Промежуточными данными являются переменные a1, a2, a3.  Выходом будет переменная max среди переменных a1, a2, a3.  **Задание 1.** Опишите в тетради входящие и выходящие данные.  **Окно формы**  На рисунке задачи 2 компоненты имеют известные Вам свойства.  OKNO2213  *Рис.*  Проектирование формы задачи 2.  Усложним форму указанием видимости компонентов. Компоненты  входных данных (Edit1, Edit2) сделаем видимыми при открытии формы: установим свойство visible=true. Компоненты отображения результатов (Edit3 – Edit6) первоначально невидимы: установим свойство visible=false.  **Задание 2**. В Delphi на оформите окно формы. Сохраните модуль и проект.  **События и реакция на события программы**  В программе событие «Вычислить» реализуется процедурой:   |  |  | | --- | --- | | **procedure** TForm1.Button1Click(Sender: TObject);  **var** x,y,a1,a2,a3, max: real;  **begin**x:=strtofloat(edit1.text);  y:= strtofloat(edit2.text);  a1:=5\*x\*x-4\*y\*y; Edit3.text:=floattostr(a1);  **Edit3.visible:=true;**  a2:=exp(y); Edit4.text:=floattostr(a2);  **Edit4.visible:=true;**  a3:=ln(x/2); Edit5.text:=floattostr(a3);  **Edit5.visible:=true;**  if (a1>a2) and (a1>a3) then max:=a1;  If (a2>a1) and (a2>a3) Then max:=a2;  IF( a3>a1) and (a3> a2) then max:=a3;  Edit6.text:=floattostr(max);  **Edit6.visible:=true;**  **end;** | *Обратите внимание, что максимальное значение присваивается переменной max.* |   Типичная ошибка a1:=max  *изменяет значение переменной a1* и нигде не записывает максимальное значение (max - неопределенно).  **Задание 3**. Создайте событие «Вычислить» и реакцию на него.  **Тестирование**  Найдем max (5x2-4y; ey; ln x/2) при  x=2, y=3.   Для этого значения a1, a2, a3 рассчитываются в Пуск/Программы/ Стандартные/ Калькулятор.Среди a1, a2, a3 находится максимальное значение.  **Задание 4**. Сравните полученные Вами значения с результатом программы.  **Задание 5.** Алгоритмизация: Зарисуйте в тетради блок-схему решения задачи.  **Дополнительно.** Создайте кнопку «Условие задачи» с выводом окна (рис. 3.3).  㿷ᛷΨ  *Рис.* Окно сообщение задачи 2.    **Задача 3. Программа "Флажок"**      *Рис.* Окно задачи 3.  **Этапы выполнения задачи 3**  **Определение требований программы**  В рассматриваемой задаче:  ü вход: х (целое число);  ü выход: y1 (модуль числа х) и y2 (квадрат числа х).  **Задание 1. В** тетради опишите тип переменных x, y1, y2. Запишите функции нахождения модуля и квадрата числа.  **Окно формы**  В окне формы компоненты располагаются, как показано на рис. 3.5.    *Рис.* Проектирование окна формы задачи 3.  Для компонентов TForm, TEdit, TLabel задаются уже известные Вам свойства.  Форма: name = Form1, Caption = Программа вычисления модуля и квадрата.  Метка: name = Label1, Caption = Введите Х, Autosize = true.  Флажки: name = CheckBox1, Caption =Модуль                name = CheckBox2, Caption =Квадрат  Однострочное окно: name = Edit1, Text ?.  Аналогично Edit2, Edit3. Установите окна результата невидимыми.      **Флажок.** Кроме известных компонентов на форму помещается новый компонент флажок. Флажок может принимать три состояния, которые задаются свойством **State**:  ДА – cbChecked;  НЕТ – cbUnchecked;  НЕ ЗНАЮ – cbGrayed.  **State** = [cbChecked,  cbUnchecked, cbGrayed].  Чаще всего флажок принимает только два состояния ДА-НЕТ. Свойство **AllowGrayed**=[true, false] включает/выключает третье состояние - НЕ ЗНАЮ.  **Задание 2**.  2.1. Запишите в тетради свойства компонента TCheckBox: AllowGrayed и State.  2.2. Оформите окно формы Delphi.  2.3. Изучите состояния флажков. Поэкспериментируйте: для CheckBox1 задайте включен (ДА), для CheckBox2 - НЕ ЗНАЮ.  2.4. Установите для флажков свойства:   Флажок: name = CheckBox1 (CheckBox2)                Caption = модуль (квадрат)               State - флажки выключены.  **События изменения состояния флажков**  Событие щелчок мыши по  кнопке «модуль» - CheckBox1Click, находит абсолютное значение числа Х, записанное в Edit1, и выводит его в Edit2.  **Procedure** TForm1.CheckBox1Click (Sender: TObject);  **var** x,y1: integer;  **begin**  x:=StrToInt(edit1.text);  y1:=abs(x);  edit2.text:= IntToStr(y1);  **if** checkbox1.state=cbChecked  **then** edit2.visible :=true  **else**  edit2.visible :=false;  **end;**  Обратите внимание на условие для флажка: *checkbox1.state=cbChecked.* Условие истинно при включенном флажке. В этом случае пользователю выводится результат. При ложном условие флажок выключен, следовательно, результат не отображается на экране.  **Задание 3**. Создайте события для двух флажков, реакции на эти события.  **Тестирование и отладка**  Введите отрицательное значение х, например, х=-2. Выберите флажок "модуль" и должны получить результат. Выключите флажок - результат исчезнет. Аналогично протестируйте флажок "квадрат".  Обратите внимание, Вы можете включить два флажка вместе, можете отключить все флажки, включать только один. Говорят, что состояние одного флажка не зависит от состояния других, поэтому **флажок - независимый переключатель**.  **Задание 4**. Выпишите в тетрадь понятия независимый переключатель. Создайте исполняемую программу. Не забудьте сохранить программу. |
| **Упражнения**  1.     Вычислить Y – размер ВВП страны в зависимости от:  а – индекса расходов на сельское хозяйство к бюджету страны (от 0 до 1)  b – индекса расходов на промышленность к бюджету страны (от 0 до 1)  z – индекс расходов на образование к общему бюджету страны (от 0 до 1)  x =a-b  – разница индексов расходов на сельское хозяйство и промышленность     |  |  |  |  | | --- | --- | --- | --- | |  | *Уровень начальный* |  | *Уровень средний* | | А |  | E | C:\Users\User\Application Data\Microsoft\Word\информатика\inform\metodic\2 tema3\Progr\ypragn9.gif | | B |  | F | C:\Users\User\Application Data\Microsoft\Word\информатика\inform\metodic\2 tema3\Progr\ypragn11.gif | | C |  | G | C:\Users\User\Application Data\Microsoft\Word\информатика\inform\metodic\2 tema3\Progr\ypragn10.gif | | D | C:\Users\User\Application Data\Microsoft\Word\информатика\inform\metodic\2 tema3\Progr\ypragn8.gif | I | ~~ypragn11~~ |   *Уровень высокий.* Дополнить программу флажками. Задать три флажка, соответствующие каждому промежутку.  При выводе результата включать соответствующий флажок.  2. Даны x, y, z. Составить программу нахождения максимального элемента   |  |  |  |  | | --- | --- | --- | --- | |  | *Уровень начальный* |  | *Уровень средний* | | A | max (x+y+z, 3x, y-4z) | E | min (sin(x2+y2), tg(z), x/y) | | B | min (x2+y2, |z|, xyz). | F | max (5x2-4y2), ey , ln|x/2|) | | C | max (x2+y2-z2, xyz, (x-y)4) | G | max (sin(x+y), tg(10-2z), 6y3) | | D | min (|x+y|, (z/x)2, 45/x3) | I | max (ln|x-y|, tg(z), x/y). | | |
| **Творческие задания**  1.    Напишите программу расчета скидки на покупку.  Если сумма покупки составляет более 500 р., то покупателю предоставляется скидка n%.  2.    Известна величина прожиточного минимума на человека в месяц. Напишите программу, которая для семьи n человек с доходом S включает флажок выше или ниже прожиточного минимума. Дополнительно: предусмотреть возможность равенства значений.  3.    Некоторый товар можно приобрести в трех торговых точках. В каждом магазине известна цена товара S1, S2, S3 и предоставляемые скидки r1, r2, r3. Найти минимальную цену с указанием магазина. Вывести реальную цену покупки в каждом магазине.  4.     Стоимость печати стандартной визитки S рублей. При печати визитки на глянцевой бумаге стоимость увеличивается на 10%. Нанесение логотипа на визитку увеличивает стоимость печати на 5%. Печать более 100 экземпляров уменьшает стоимость на 8%. Написать программу расчета стоимости печати одной визитки. В программе выбор параметров осуществляется флажками «Глянцевая бумага», «Логотип», «Объем более 100 экз.». | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Раздел 5 ЦИКЛЫ**  **Цель:**  Теоретический контроль:  1.      Понятие цикла.  2.      Виды циклических операторов.  3.      Цикл с предусловием: принцип работы, блок-схема, оператор.  4.      Цикл с постусловием: принцип работы, блок-схема, оператор.  5.      Цикл с параметром: принцип работы, блок-схема, оператор, два способа записи.  6.      Два различия использования циклов с пост - и предусловиями.  УЗНАТЬ:   |  |  | | --- | --- | | маркированный список | переменные, используемые в вычислительных циклических задачах; | | маркированный список | начальные значения суммы и произведения ряда; | | маркированный список | различия между циклами с пост- и предусловиями; | | маркированный список | особенность цикла с параметром. |   НАУЧИТЬСЯ:   |  |  | | --- | --- | | маркированный список | вычислять произведение, сумму,  среднее значение числовых рядов. |   **Теоретические сведения**  **Введение**  Циклические операторы содержат повторяющиеся команды. Циклические операторы бывают трех видов: счетный цикл (цикл с параметром), цикл с предусловием, цикл с постусловием.  **Оператор цикла с предусловием**  Структура оператора цикла с предусловием следующая:  **while** <условие> **do** <тело цикла>;  Здесь **while, do** - зарезервированные слова (пока [выполняется условие], делать).  Блок-схема цикла с предусловием следующая:    E:\Мои документы\Delphi\Programmirovanie\images\l6.htm1.gif   Циклический оператор с предусловием работает по следующему алгоритму. Первоначально проверяется условие. Если оно истинно, то выполняется тело цикла (оператор). Затем снова проверяется условие и т.д. Если условие ложно, то цикл завершается.  Если тело цикла состоит из нескольких операторов, то используется составной оператор.  **while** <условие> **do** **begin**                                          <тело цикла>;  **end**;  **Оператор цикла с постусловием**  Структура оператора цикла с постусловием следующая:  **repeat** <тело цикла> **until** <условие>;  Блок-схема цикла с постусловием следующая:    E:\Мои документы\Delphi\Programmirovanie\images\l6.htm2.gif  Циклическая структура цикла с постусловием работает по следующему алгоритму: сначала выполняется тело цикла (операторы), потом проверяется условие. Если условие ложно, то снова выполняются операторы. И так до тех пор, пока условие не примет значение TRUE.  **Оператор цикла с параметром**  Блок-схема счетного цикла (с параметром) следующая:  **for** <параметр цикла>:=<нач.знач.> **to** <кон. знач.> **do** <тело цикла>;  Блок-схема цикла с параметром следующая:  E:\Мои документы\Delphi\Programmirovanie\images\l6.htm3.gif  Циклическая структура счетного цикла работает по следующему алгоритму: параметру цикла присваивается начальное значение и выполняется тело цикла. Затем параметру цикла присваивается следующее значение, и снова выполняется тело цикла. И так до тех пор, пока не будут перебраны все значения параметра цикла. Каждый раз параметр цикла увеличивается на единицу.  Другая структура счетного оператора:  **for <**параметр цикла>:=<старш. знач.> **downto <**млад. знач.> **do** < тело цикла>;  Циклическая структура данного счетного цикла работает по следующему алгоритму: параметру цикла присваивается начальное (старшее) значение и выполняется тело цикла. Затем значение параметра уменьшается на единицу, и снова выполняется тело цикла. И так до тех пор, пока не будут перебраны все значения параметра цикла до конечного (младшего) значения.    **Задача**  **Программа вычисления суммы числового ряда**  E:\Мои документы\Delphi\Programmirovanie\images\l6.htm4.gif    **Этапы выполнения задачи**  **Анализ требований программы**  Входные  данные   a, b: целого типа – границы диапазона  Промежуточные переменные                 i: целого типа - параметр цикла, изменяется [a;b]                 v: вещественного типа – элемент ряда v =sin 2i  Результат   S: вещественного типа - сумма ряда.  **Задание 1.** Опишите в тетради переменные задачи, их тип и назначение.  **Окно будущей программы**  Рис. Пример визуального проектирования окна программы 1.  **Задание 2.**  В Delphi на форме расположите компоненты программы.  **Алгоритмизация и программирование**  Алгоритм вычисления числового ряда содержит  действия:  1) ввод данных;  2) задание первоначальных значений переменных;  3) тело цикла:    ·   вычисление элемента ряда;  ·        прибавление его к сумме;  ·        изменение параметра цикла;  4) вывод результата  Запомните! Первоначальное значение суммы S=0, произведения P=1.   |  |  | | --- | --- | | *Алгоритм цикла с предусловием* | **Procedure TForm1.Button1Click();**  **var**  a,b,i:Integer;  V,S:real;  **begin**  a:=StrToInt (edit1.text);  b:=StrToInt (edit2.text);  S:=0 ;  i:=a;  **while** i<=b **do begin**  V:=sin (2\*i);  S:=S+V;  i:=I+1;  memo1.lines.add(FloatToStr(V));  **end;**  edit3.text:= FloatToStr(S);  **end;** |  |  |  | | --- | --- | | *Алгоритм цикла с постусловием* | **Procedure TForm1.Button2Click();**  **var**  a,b,i:Integer;      V,S:real;  **begin**  a:=StrToInt (edit1.text);  b:=StrToInt (edit2.text);  S:=0 ;  i:=a;  **Repeat**                     V:=sin (2\*i);                     S:=S+v;                      i:=I+1;         memo2.lines.add(FloatToStr(v));  **Until not (i<=b);**  Edit4.text:= FloatToStr(S);  **end;** |   **Особенности циклических структур**  Цикл с предусловием:  ·         условие проверяется в начале цикла;  ·         тело цикла выполняется в случае истинности условия;  ·         возможна ситуация, когда тело цикла не выполнится ни разу.  Цикл с постусловием:  ·         условие проверяется в конце цикла;  ·        тело цикла выполняется в случае ложности условия;  ·         тело цикла выполняется хотя бы один раз.  Цикл с параметром не является классической алгоритмической структурой.  Блок-схема цикла с параметром напоминает блок-схему цикла с предусловием.  Цикл с параметром имеет два отличия:  ·        параметр цикла изменяется строго на 1;  ·        изменение параметра цикла происходит автоматически.   |  |  |  | | --- | --- | --- | |  | **Procedure TForm1.Button3Click();**  **var**  a,b,i:Integer; V,S:real;  **begin**       a:=StrToInt (edit1.text);       b:=StrToInt (edit2.text);       S:=0;  ~~I:=a;~~ | | | *{FOR   TO   DO  }*  **For i:=a to b do begin**                     V:=sin (2\*i);                     S:=S+V;  *~~i:=I+1;~~*  memo3.lines.add(FloatToStr(V));  **end;** | *{FOR  DOWNTO   DO  }*  **For i:=b downto a do begin**                     V:=sin (2\*i);                     S:=S+V;  *~~i:=I+1;~~*  memo3.lines.add(FloatToStr(V));  **end;** | | edit5.text:= FloatToStr(S);  **end** | |   **Задание 3.**  3.1.Зарисуйте блок-схемы и программы трех видов циклов в тетрадь.  3.2. Создайте события программы и реакцию на него.  3.3. Выпишите отличия циклов с пред- и постусловиями, особенность цикла с параметром. 3.4. Для цикла с параметром реализуйте один из двух способов записи.  **Тестирование и отладка программы**  Задача «Вычисление суммы ряда» решается тремя способами с использованием различных видов циклов. При этом результаты всех вариантов решения должны совпадать.  При тестирование задаются входные данные и рассчитывается результат.  **В Lazarus sin рассчитывается в радианах.**  Пусть a=2, b=7.  S=0  I=2,  V=sin 2\*2=\_\_\_\_,  S=S + sin 4 = 0+0,07 =0,07  I=3,  V=sin 2\*3=\_\_\_\_,  S=S + sin 6 = 0,07+\_\_\_=  I=4,  V=sin 2\*4=\_\_\_\_,  S=S+\_\_\_\_  I=5,  V=sin 2\*5=\_\_\_\_,  S=S+\_\_\_\_  I=6,  V=sin 2\*6=\_\_\_\_,  S=S+\_\_\_\_  I=7,  V=sin 2\*7=\_\_\_\_,  S=S+\_\_\_\_  Сумма S =\_\_\_\_\_\_  Задание 4.  4.1. Рассчитайте на бумаге результат задачи при a=2, b=7.  4.2. Запустите программу на исполнение. Все алгоритмы задачи должны выводить одинаковые результаты, согласно Вашим расчетам. |
| **Упражнения**  1)    Составьте программы вычисления:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | *Уровень начальный* |  | *Уровень средний* |  | *Уровень высокий* | | А |  | E |  | I | E:\Мои документы\Delphi\Programmirovanie\images\l6.htm5.gif | | B |  | F |  | H | E:\Мои документы\Delphi\Programmirovanie\images\l6.htm6.gif | | C |  | G |  | L | E:\Мои документы\Delphi\Programmirovanie\images\l6.htm7.gif | | D |  | J |  | M | E:\Мои документы\Delphi\Programmirovanie\images\l6.htm8.gif | |

|  |
| --- |
| **Творческие задания**  1.     Начальный объем выпуска предприятия составляет S ед. Ежегодное увеличение выпуска составляет  k%  к предыдущему году: St = St-1 \*k.  Рассчитать предполагаемый выпуск предприятия на t лет вперед.  2.     Для кредита на сумму S на t лет  под n% годовых рассчитать сумму выплат: погашение основного кредита и выплата процентов. Известно, что сумма основного долга выплачивается равными долями ежемесячно, ежемесячный процент n/12 рассчитывается на оставшуюся сумму долга.  3.     Первоначальная стоимость основного фонда предприятия составляет S млн. руб. Амортизация в год составляет n%: ежегодная стоимость уменьшается на n%. Рассчитать стоимость основного фонда предприятия на t лет вперед. |
| **Работа с блок-схемам**     |  |  |  | | --- | --- | --- | | Задание 1 | Задание 2 | Задание 3 | | **http://www.integro.ru/~trushin/TESTS/IM-2/201.GIF** | 207 | 120 | | **А) при А = -2; С= -3**  **Б) при А = -3; С= -2**  **В) при А = 2; С= 3**  **Г) при А = 10; С= -3**  **Д) при А = 0; С= -5**  **Е) при А = 12; С= 3** | | | |

|  |
| --- |
| **Работа с блок-схемам**  1.  Укажите значение следующего выражения: (x>y) **и** ( p **или** (y>z)) при:  **а)** x = 5, y = 3, z=10, p=**нет**;  **б)** x = -10, y = -3, z=-5, p=**да**;  в) x = 0, y = 1, z=-1, p=**да**;  г) x = 1, y = 0, z = 5, p = **да**;  **д)** x = 5, y = 0.5, z = -4, p = **нет**;  е) x = 8, y = - 4.5, z = 7, p = **нет**;  ж) x = -4, y = 5, z = 8, p = **нет**;  з) x = -15, y = -25, z = 10, p = **да**;  2.  Фрагмент алгоритма изображен в виде блок-схемы. Определите значение переменной S.  Запишите процедуру.    А) а = 3, b = 4         Б) а = -3, b = 10  В) а = 6, b = -3        Г) а = 8, b = 0  Д) а = -3, b = -4       Е) а = 2, b = 15  Ж) а = 3, b = -8       З) а = 1, b = -5 |