СОДЕРЖАНИЕ

УРОК №19. ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ РАБОТЫ С ГРАФИКОЙ

Цель:

Образовательная: Изучить основные компоненты, с помощью которых можно открыть графически оформить программу (рисунком, рамкой, геометрическим изображением). Рассмотреть, каким образом можно программно рисовать на поверхности формы, какие для этого используются методы и свойства.

Воспитательная: воспитание организованности, дисциплинированности; воспитание любви к профессии;

Развивающая: развитие познавательных способностей

Тип занятия: урок изучения нового материала

Форма организации учебного процесса: интерактивная работа с текстом

Наглядные пособия и ТСО: фрагменты учебного материала для интерактивной работы с текстом

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана работы на уроке. Правила работы в домашних и рабочих группах.

2. Группа разбивается на домашние группы по 4 человека (примерно 6 домашних групп). Далее выбирается капитан домашней группы и представляется всем.

   Капитан домашней группы распределяет роли (1,2,3,4) и получает задание на всех.

   Формируются рабочие группы (все единицы собираются в первую рабочую группы, все двойки – во вторую и т.д.).

3. Рабочая группа в течении ***20 минут*** разбирает учебный материал, составляет план объяснения и разрабатывает схему своего материала, по которой можно было бы понять главную идею фрагмента (схема потом будет сдаваться преподавателю, который будет в целом оценивать работу группы).

4. Члены рабочей группы возвращаются в домашние группы. Каждому дается 5 минут на объяснение и конспектирование своего фрагмента. Итого на изучение и обсуждение дается ***25 минут***. Капитан домашней группы выставляет оценку каждому обучающему.

5. Далее каждой домашней группе выдается пример рисования простейшего графического рисунка и задание нарисовать свой рисунок.

6. **Оценка работы групп и вставление оценок**

7. **Домашнее задание:**

8. **Подведение итогов урока.**

Таблица оценок

Фамилия	Оценка за схему в группе	Оценка капитана за работу в дом. группе	Оценка за конспект в тетради	Оценка за выполненное задание

В стандартную библиотеку визуальных компонент Delphi входит несколько объектов, с помощью которых можно придать своей программе совершенно оригинальный вид. Это:

1. **Image** позволяет поместить графическое изображение в любое место на форме. (Свойства: Picture,Center, Stretch)
2. **Shape** - простейшие графические объекты на форме типа круг, квадрат и т.п. Свойства: Shape, Pen, Brush.
3. **Bevel** - объект для украшения программы, может принимать вид рамки или линии.
4. **PaintBox**, является канвой для рисования.

В графическом редакторе можно нарисовать практически что угодно. Но, зная приемы программирования, создаются быстро такие элементы, на которые в графическом редакторе уйдет много времени и не будет возможности многократного повторения и внесения быстрых изменений в рисунок.

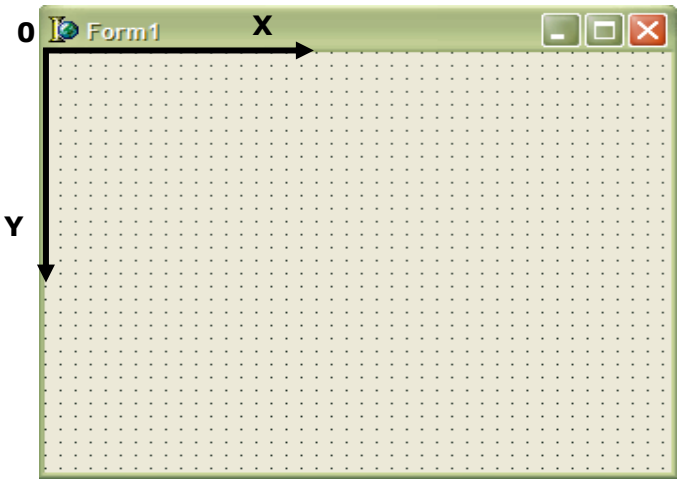
Среда Delphi позволяет достаточно просто создавать программы вывода на экран схем, чертежей, различных иллюстраций.

Графические элементы выводятся на поверхность формы (или некоторых других компонентов), которой соответствует свойство Canvas. Для того чтобы вывести на поверхность объекта графический элемент (прямую линию, окружность и т.д.), необходимо применить к свойству canvas этого объекта соответствующий метод. Например, следующий метод вычерчивает в окне формы прямоугольник.

Form1.Canvas.Rectangle(10,10,100,100);

ХОЛСТ

Методы вывода графических примитивов рассматривают свойство Canvas как некоторый абстрактный холст, на котором можно рисовать. Этот холст состоит из отдельных точек – пикселей, положение каждого из которых характеризуется его горизонтальной (x) и вертикальной (y) координатами.



Форма представляет собой координатную сетку с началом в левом верхнем углу.

КАРАНДАШ И КИСТЬ

Методы, обеспечивающие вычерчивание на поверхности холста графических примитивов, используют:

1. Карандаш (для вычерчивания линий и контуров)
2. Кисть (для закрашивания областей, ограниченных контурами).

Карандашу соответствует свойство **Pen**, кисти – **Brush**. Они представляют собой объекты типов TPen и TBrush соответственно.

Свойства объекта TPen

Свойство	Возможные значения	Пример использования
Color – цвет линии	clBlack, clOlive, clGray, clGreen, clBlue, clMaroon, clNavy, clSilvar, clAqua, clPurple, clRed, clWhite;	form1.Canvas.Pen.Color:=$0000FF00; form1.Canvas.Pen.Color:=clWhite;
width – толщина линии	Целое число. Если число больше единицы, то пунктирная линия вычерчивается сплошной	form1.Canvas.Pen.Width:=5;

style – вид линии	psSolid – сплошная линия psDash – пунктирная линия, длинные штрихи psDot – пунктирная линия, короткие штрихи psClear – линия не отображается	form1.Canvas.Pen.Style:=psDash;

Свойства объекта TBrush

Свойство	Возможные значения	Пример использования
Color – цвет линии	clBlack, clOlive, clGray, clGreen, clBlue, clMaroon, clNavy, clSilvar, clAqua, clPurple, clRed, clWhite;	form1.Canvas.Brush.Color:=$0000FF00; form1.Canvas.Brush.Color:=clWhite;
style – вид линии	bsSolid – сплошная заливка bsClear – область не закрашивается bsHorizontal – горизонтальная штриховка bsVertical – вертикальная штриховка bsFDiagonal – диагональная штриховка	form1.Canvas.Brush.Style:=bsHorizontal;

Методы вычерчивания графических примитивов

LineTo(X, Y);	Проводит линию текущим пером из текущей точки в (X, Y)
MoveTo(X, Y);	Перемещает текущее положение пера (свойство PenPos) в точку (X, Y)
Ellipse (XI, Y1, X2, Y2);	Рисует и закрашивает эллипс, вписанный в прямоугольник (X1, Y1) - (X2, Y2)
Rectangle(X1, Y1,X2, Y2);	Рисует прямоугольник с верхним левым углом в (X1, Y1) и нижним правым в (X2, Y2)
RoundRect(XI, Yl, X2, Y2, X3, Y3);	Рисует прямоугольник с закругленными углами. Координаты вершин — те же, что и в методе Rectangle. Закругления рисуются как сегменты эллипса с размерами осей по горизонтали и вертикали X3 и Y3
Arc (XI, Y1, X2, Y2, Y3, X4, Y4);	Метод рисует сегмент эллипса. Эллипс определяется описывающим прямоугольником (X1, Y1) — (X2, Y2); его размеры должны лежать в диапазоне от 2 до 32767 точек. Начальная точка сегмента лежит на пересечении эллипса и луча, проведенного из его центра через точку (X3, Y3). Конечная точка сегмента лежит на пересечении эллипса и луча, проведенного из его центра через точку (X4, Y4). Сегмент рисуется против часовой стрелки
TextOut(x,y,s);	Выводит текст s, начиная с точки (x,y)

Пример: Составить программу, которая рисует эллипс с желтой заливкой и синей границей.

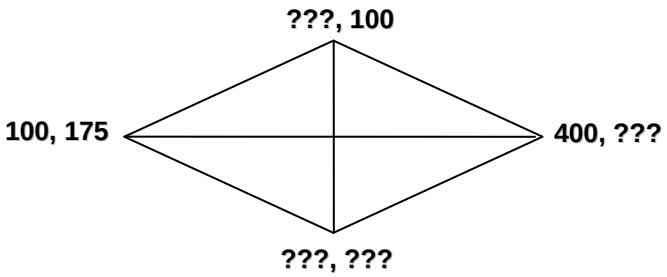
```
procedure TForm1.FormPaint(Sender: TObject);
begin
  Canvas.Pen.Color := clBlue;  // Выбрать цвет контура
  Canvas.Brush.Color := clYellow; // Выбрать цвет заливки
  Canvas.Ellipse(10, 20, 200, 200); // Нарисовать эллипс
end;
```

Задание: Составить программу рисования прямоугольника красного цвета в произвольном месте, произвольного размера.

Домашнее задание

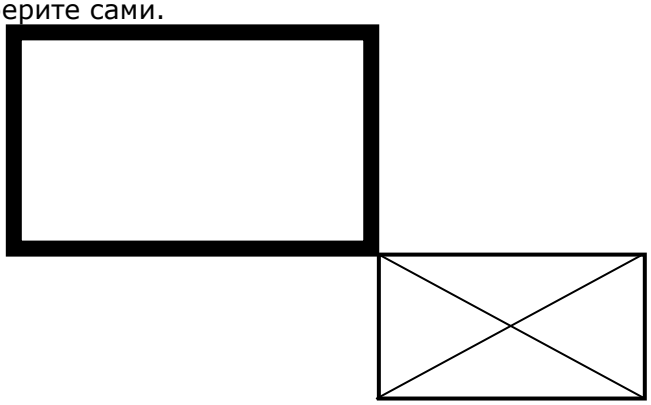
Уровень №1 обязателен для выполнения.
Если выполняется уровень №2, то уровень №1 можно не выполнять.
Творческий уровень (№3) выполняется на дополнительную оценку.

Уровень №1. Напишите программу, изображающую на экране ромб с диагоналями.

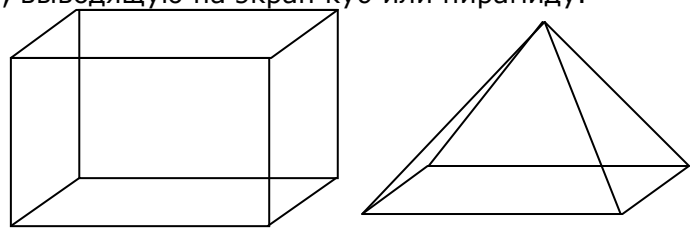


Уровень №2.

1. Напишите программу, выводящую рамку с конвертом. Рамка рисуется с помощью двух прямоугольников (оператор Rectangle) разного цвета, конверт – придумайте сами. Координаты тоже выберите сами.



2. Создайте программу, выводящую на экран куб или пирамиду.



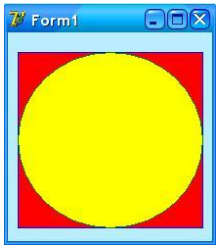
Уровень №3. Творческий. Составить программу, выводящую иллюстрацию к какой-либо сказке.

Рабочая группа в течении ***20 минут***: разбирает учебный материал, составляет план объяснения, разрабатывает схему своего материала, по которой можно было бы понять главную идею фрагмента (схема потом будет сдаваться преподавателю, который будет в целом оценивать работу группы).

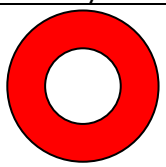
Домашняя группа в течении ***25 минут*:** Каждому дается 5 минут на объяснение и конспектирование своего фрагмента. Капитан домашней группы выставляет оценку каждому члену группы за его объяснение.

Пример: Составить программу, которая рисует эллипс с желтой заливкой и синей границей.

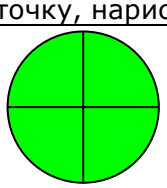
```
Canvas.Pen.Color:=clBlue;  //Граница синяя
Canvas.Brush.Color:=clRed; //Заливка красная
Canvas.Rectangle(10,20,200,200); //прямоугольник
Canvas.Pen.Color:=clGreen;  //Граница зеленая
Canvas.Brush.Color:=clYellow;//Заливка желтая
Canvas.Ellipse(10,20,200,200); //эллипс
```



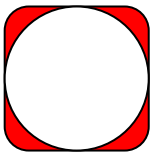
Чтобы быстро и четко нарисовать заданный объект, рекомендуется взять листок бумаги в клеточку, нарисовать экран в масштабе 1 см=50 пикселей.



1. Красное кольцо

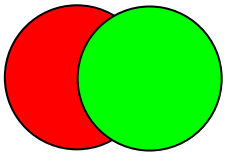


2. Перечеркнутый зеленый круг



3. Красный орнамент

4. Красный крест



5. Окружности

Фрагмент №1. Основные компоненты и холст

В стандартную библиотеку визуальных компонент Delphi входит несколько объектов, с помощью которых можно придать своей программе совершенно оригинальный вид. Это:

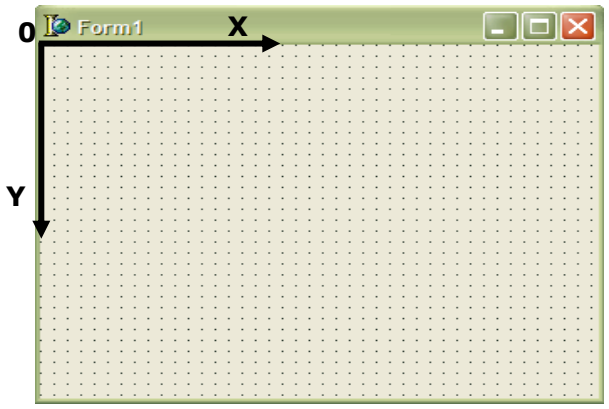
1. **Image** позволяет поместить графическое изображение в любое место на форме. Свойства:
   a. Picture – загружает картинку,
   b. Center = true – устанавливает картинку по центру,
   c. Stretch= true – масштабирует картинку по размеру компонента)
2. **Shape** - простейшие графические объекты на форме типа круг, квадрат и т.п. Свойства:
   a. Shape – указывает форму объекта,
   b. Pen – настройка границы компонента,
   c. Brush – настройка заливки компонента.
3. **Bevel** - объект для украшения программы, может принимать вид рамки или линии.
4. **PaintBox**, является канвой для рисования.

Зная приемы программирования можно создавать быстро такие элементы, на которые в графическом редакторе уйдет много времени и не будет возможности многократного повторения и внесения быстрых изменений в рисунок. Среда Delphi позволяет достаточно просто создавать программы вывода на экран схем, чертежей, различных иллюстраций.

ХОЛСТ. Графические элементы выводятся на поверхность формы, которой соответствует свойство **Canvas**. Для того чтобы вывести на поверхность объекта графический элемент (прямую линию, окружность и т.д.), необходимо применить к свойству **Canvas** этого объекта соответствующий метод. Например, следующий метод вычерчивает в окне формы прямоугольник:

Form1.Canvas.Rectangle(10,10,100,100);

Методы вывода графических примитивов рассматривают свойство Canvas как некоторый холст, на котором можно рисовать. Этот холст состоит из отдельных точек – пикселей, положение каждого из которых характеризуется его горизонтальной (x) и вертикальной (y) координатами.



Форма представляет собой координатную сетку с началом в левом верхнем углу.

Фрагмент №2. Карандаш

Методы, обеспечивающие вычерчивание на поверхности холста графических примитивов, используют: карандаш (для вычерчивания линий и контуров)

Карандашу соответствует свойство **Pen** и он представляет собой объект типа TPen.

Свойства объекта TPen		
Свойство	**Возможные значения**	**Пример использования**
Color – цвет линии	clBlack, clOlive, clGray, clGreen, clBlue, clMaroon, clNavy, clSilvar, clAqua, clPurple, clRed, clWhite;	form1.Canvas.Pen.Color:=$0000FF00; form1.Canvas.Pen.Color:=clWhite;
width – толщина линии	Целое число. Если число больше единицы, то пунктирная линия вычерчивается сплошной	form1.Canvas.Pen.Width:=5;
style – вид линии	psSolid – сплошная линия psDash – пунктирная линия, длинные штрихи psDot – пунктирная линия, короткие штрихи psClear – линия не отображается	form1.Canvas.Pen.Style:=psDash;

Фрагмент №3. Кисть

Методы, обеспечивающие вычерчивание на поверхности холста графических примитивов, используют: кисть (для закрашивания областей, ограниченных контурами).

Кисти соответствует свойство **Brush**, и она представляет собой объект типа TBrush

Свойства объекта TBrush		
Свойство	**Возможные значения**	**Пример использования**
Color – цвет линии	clBlack, clOlive, clGray, clGreen, clBlue, clMaroon, clNavy, clSilvar, clAqua, clPurple, clRed, clWhite;	form1.Canvas.Brush.Color:=$0000FF00; form1.Canvas.Brush.Color:=clWhite;
style – вид линии	bsSolid – сплошная заливка bsClear – область не закрашивается bsHorizontal – горизонтальная штриховка bsVertical – вертикальная штриховка bsFDiagonal – диагональная штриховка	form1.Canvas.Brush.Style:=bsHorizontal;

Фрагмент №4. Графические объекты

Методы вычерчивания графических примитивов	
LineTo(X, Y);	Проводит линию текущим пером из текущей точки в (X, Y)
MoveTo(X, Y);	Перемещает текущее положение пера в точку (X, Y)
Ellipse (XI, Y1, X2, Y2);	Рисует и закрашивает эллипс, вписанный в прямоугольник (X1, Y1) - (X2, Y2). *Например*: Canvas.Ellipse(10, 20, 200, 200);
Rectangle(X1, Y1,X2, Y2);	Рисует прямоугольник с верхним левым углом в (X1, Y1) и нижним правым в (X2, Y2)
RoundRect(XI, Yl, X2, Y2, X3, Y3);	Рисует прямоугольник с закругленными углами. Координаты вершин — те же, что и в методе Rectangle. Закругления рисуются как сегменты эллипса с размерами осей по горизонтали и вертикали X3 и Y3
Arc (XI, Y1, X2, Y2, Y3, X4, Y4);	Метод рисует сегмент эллипса. Эллипс определяется описывающим прямоугольником (X1, Y1) — (X2, Y2); его размеры должны лежать в диапазоне от 2 до 32767 точек. Начальная точка сегмента лежит на пересечении эллипса и луча, проведенного из его центра через точку (X3, Y3). Конечная точка сегмента лежит на пересечении эллипса и луча, проведенного из его центра через точку (X4, Y4). Сегмент рисуется против часовой стрелки
TextOut(x,y,s);	Выводит текст s, начиная с точки (x,y)

УРОК №20. ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИЙ. АНИМАЦИЯ ГЕОМЕТРИЧЕСКИХ ПРИМИТИВОВ

Цель:

Образовательная: Создать условия для усвоения принципов построения графиков функций средствами Delphi и принципов анимации объектов.

Воспитательная: Содействовать воспитанию организованности, дисциплинированности; воспитанию любви к профессии, воспитанию навыков коллективного труда

Развивающая: Содействовать развитию познавательных способностей

Тип занятия: урок изучения нового материала

Форма организации учебного процесса: работа в группах

Наглядные пособия и ТСО: Карточки входного контроля, дидактический материал (учебный материал, разбитый на уровни)

ХОД УРОКА

1. **Организационный момент.** Мотивация на урок. Раскрытие темы урока, целей, задач и плана урока.
2. **Входной контроль (10 мин).** Карточки для проверки знаний. Проверка знаний основных терминов, их свойств и методов.
   Проверка и оценка работы по карточкам происходит в парах. Учитель после учащихся перепроверяет любые три работы (с каждого ряда). И выставляются оценки в журнал.
   В результате проверки выявляются консультанты, успевающие и неуспевающие (по количеству выполненных заданий).
3. **Составляются группы:**
   - 1 уровень
   - 2 уровень
   - 3 уровень

   Каждая группа работает над своим учебным материалом и выполняет задания своего уровня.
4. **Работа в группах по изучению учебного материала** и составлению конспекта.
5. **Работа в группах по выполнению заданий** своего уровня
6. **Подведение итогов урока.**

Таблица оценок

Фамилия	Оценка за входной контроль	Оценка за конспект в тетради	Оценка за выполненное задание

КАРТОЧКИ ДЛЯ ВХОДНОГО КОНТРОЛЯ

Вариант №1. Тема «Графика в Delphi»

1. Какое свойство рассматривают в Delphi в качестве холста и для чего оно предназначено?
2. Найдите ошибку в записи фрагмента:
```
Form1.Canvas.Pen:=clRed;
Form1.Canvas.Rectangle(0,0,10,10);
```
3. Определите, что рисует следующий фрагмент:
```
Form1.Canvas.Rectangle(10,10,100,100);
Form1.Canvas.MoveTo(10,10);
Form1.Canvas.LineTo(100,100);
```

Вариант №2. Тема «Графика в Delphi»

1. Какое свойство рассматривают в Delphi в качестве карандаша и для чего оно предназначено?
2. Найдите ошибку в записи фрагмента:
```
Form1.Canvas.Pen.Color.clRed;
Form1.Canvas.Rectangle(0,0,10,10);
```
3. Определите, что рисует следующий фрагмент:
```
Form1.Canvas.Rectangle(10,10,100,100);
Form1.Canvas.MoveTo(100,10);
Form1.Canvas.LineTo(10,100);
```

Вариант №3. Тема «Графика в Delphi»

1. Какое свойство рассматривают в Delphi в качестве кисти и для чего оно предназначено?
2. Найдите ошибку в записи фрагмента:
```
Form1.Canvas.Brush.Style:=clRed;
Rectangle(0,0,10,10);
```
3. Определите, что рисует следующий фрагмент:
```
Form1.Canvas.Rectangle(0,0,100,100);
Form1.Canvas.Ellipse(0,0,100,100);
```

Вариант №4. Тема «Графика в Delphi»

1. Какое свойство рассматривают в Delphi в качестве холста и для чего оно предназначено?
2. Найдите ошибку в записи фрагмента:
```
Form1.Canvas.Color:=clRed;
Form1.Canvas.Rectangle(0,0,10,10);
```
3. Определите, что рисует следующий фрагмент:
```
Form1.Canvas.Rectangle(0,0,100,100);
Form1.Canvas.Rectangle(100,100,200,200);
```

Вариант №5. Тема «Графика в Delphi»

1. Какое свойство рассматривают в Delphi в качестве карандаша и для чего оно предназначено?
2. Найдите ошибку в записи фрагмента:
```
Form1.Pen.Brush.Color:=clRed;
Form1.Canvas.Rectangle(0,0,10,10);
```
3. Определите, что рисует следующий фрагмент:
```
Form1.Canvas.Ellipse(0,0,100,100);
Form1.Canvas.Ellipse(30,30,40,40);
Form1.Canvas.Ellipse(60,30,70,40);
```

Вариант №6. Тема «Графика в Delphi»

1. Какое свойство рассматривают в Delphi в качестве кисти и для чего оно предназначено?
2. Найдите ошибку в записи фрагмента:
```
Form1.Canvas.Brush.Color:=clRed;
Form1.Rectangle(0,0,10,10);
```
3. Определите, что рисует следующий фрагмент:
```
Form1.Canvas.Ellipse(0,0,100,100);
Form1.Canvas.Ellipse(45,30,55,40);
Form1.Canvas.Ellipse(45,60,55,70);
```

Вариант №7. Тема «Графика в Delphi»

1. Какое свойство рассматривают в Delphi в качестве холста и для чего оно предназначено?
2. Найдите ошибку в записи фрагмента:
```
Form1.Canvas.Pen.Color(clRed);
Form1.Canvas.Rectangle(0,0,10,10);
```
3. Определите, что рисует следующий фрагмент:
```
Form1.Canvas.Rectangle(0,0,100,100);
Form1.Canvas.Rectangle(45,30,55,40);
Form1.Canvas.Rectangle(45,60,55,70);
```

Вариант №8. Тема «Графика в Delphi»

1. Какое свойство рассматривают в Delphi в качестве карандаша и для чего оно предназначено?
2. Найдите ошибку в записи фрагмента:
```
Form1.Pen.Canvas.Color:=clRed;
Form1.Canvas.Rectangle(0,0,10,10);
```
3. Определите, что рисует следующий фрагмент:
```
Form1.Canvas.Rectangle(0,0,100,100);
Form1.Canvas.Rectangle(100,0,200,100);
```

Вариант №9. Тема «Графика в Delphi»

1. Какое свойство рассматривают в Delphi в качестве кисти и для чего оно предназначено?
2. Найдите ошибку в записи фрагмента:
```
Form1.Pen.Color:=clRed;
Form1.Canvas.Rectangle(0,0,10,10);
```
3. Определите, что рисует следующий фрагмент:
```
Form1.Canvas.Ellipse(0,0,100,100);
Form1.Canvas.Ellipse(100,0,200,100);
```

Вариант №10. Тема «Графика в Delphi»

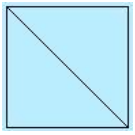
1. Какое свойство рассматривают в Delphi в качестве холста и для чего оно предназначено?
2. Найдите ошибку в записи фрагмента:
```
Form1.Pen.Color:=clRed;
Form1.Canvas.Rectangle(0,0,10,10);
```
3. Определите, что рисует следующий фрагмент:
```
Form1.Canvas.Ellipse(0,0,100,100);
Form1.Canvas.Ellipse(0,100,100,200);
```

ОТВЕТЫ НА КАРТОЧКИ ВХОДНОГО КОНТРОЛЯ

Вариант №1. Тема «Графика в Delphi»

1. В качестве холста в Delphi рассматривают свойство Canvas некоторых компонентов, на холсте рисуют.
2. **Ошибка**: Form1.Canvas.Pen:=clRed;

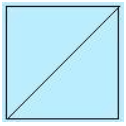
Правильно: Form1.Canvas.Pen.Color:=clRed;

3. 

Вариант №2. Тема «Графика в Delphi»

1. В качестве карандаша рассматривают свойство Pen, оно предназначено для вычерчивания линий и контуров.
2. **Ошибка**: Form1.Canvas.Pen.Color.clRed;

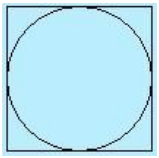
Правильно: Form1.Canvas.Pen.Color:=clRed;

3. 

Вариант №3. Тема «Графика в Delphi»

1. В качестве кисти рассматривают свойство Brush, оно предназначено для закрашивания областей.
2. **Ошибка**: Rectangle(0,0,10,10);

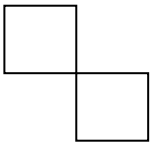
Правильно:form1.Canvas.Rectangle(0,0,10,10);

3. 

Вариант №4. Тема «Графика в Delphi»

1. В качестве холста в Delphi рассматривают свойство Canvas некоторых компонентов, на холсте рисуют.
2. **Ошибка**: Form1.Canvas.Color:=clRed;

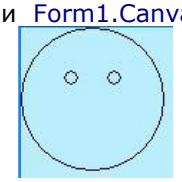
Правильно: Form1.Canvas.Pen.Color:=clRed;
или Form1.Canvas.Brush.Color:=clRed;

3. 

Вариант №5. Тема «Графика в Delphi»

1. В качестве карандаша рассматривают свойство Pen, оно предназначено для вычерчивания линий и контуров.
2. **Ошибка**: Form1.Pen.Brush.Color:=clRed;

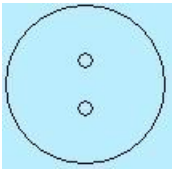
Правильно:Form1.Canvas.Brush.Color:=clRed;
или Form1.Canvas.Pen.Color:=clRed;

3. 

Вариант №6. Тема «Графика в Delphi»

1. В качестве кисти рассматривают свойство Brush, оно предназначено для закрашивания областей.
2. **Ошибка**: Form1.Rectangle(0,0,10,10);

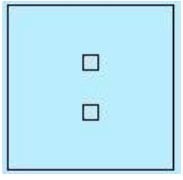
Правильно: Form1.Canvas.Rectangle(0,0,10,10);

3. 

Вариант №7. Тема «Графика в Delphi»

1. В качестве холста в Delphi рассматривают свойство Canvas некоторых компонентов, на холсте рисуют.
2. **Ошибка**: Form1.Canvas.Pen.Color(clRed);

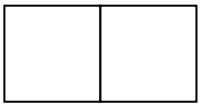
Правильно: Form1.Canvas.Pen.Color:=clRed;

3. 

Вариант №8. Тема «Графика в Delphi»

1. В качестве карандаша рассматривают свойство Pen, оно предназначено для вычерчивания линий и контуров.
2. **Ошибка**: Form1.Pen.Canvas.Color:=clRed;

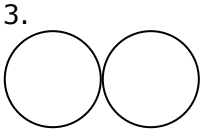
Правильно: Form1.Canvas.Pen.Color:=clRed;

3. 

Вариант №9. Тема «Графика в Delphi»

1. В качестве кисти рассматривают свойство Brush, оно предназначено для закрашивания областей.
2. **Ошибка**: Form1.Pen.Color:=clRed;

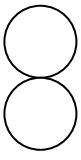
Правильно: Form1.Canvas.Pen.Color:=clRed;

3. 

Вариант №10. Тема «Графика в Delphi»

1. В качестве холста в Delphi рассматривают свойство Canvas некоторых компонентов, на холсте рисуют.
2. **Ошибка**: Form1.Pen.Color:=clRed;

Правильно: Form1.Canvas.Pen.Color:=clRed;

3. 

УЧЕБНЫЙ МАТЕРИАЛ

Свойство Pixels

Используя свойство **Pixels** объекта Canvas можно задать требуемый цвет для любой точки графической поверхности.

Например, инструкция: **Form1.Canvas.Pixels[10,10]:=clRed**; окрашивает точку (10,10) формы в красный цвет.

А с помощью конструкции:

For i:=1 to 100 do form1.Canvas.Pixels[i,10]:=clRed;

можно нарисовать горизонтальную линию, начинающуюся в точке (1;10) и заканчивающуюся в точке (100;10).

Простейший график функции

Используя свойство Pixels можно строить графики функций. График строится на основе вычислений.

С помощью следующего цикла можно построить простейший график функции:

```
X:=0;
While x<100 do
Begin
 Y:=cos(x);
 Form1.Canvas.Pixels[round(x),round(y)]:=clred;
 x:=x+0.1;
End;
```

Однако, график построенный с помощью подобной функции будет очень мелким.

График функции в увеличенном масштабе

Для того, чтобы увеличить масштаб графика, добавим переменную m – масштаб. Установим ее равной 10, и при выводе точки на экран будем умножать координату на это значение.

```
X:=0;
M:=10;
While x<100 do
Begin
 Y:=cos(x);
 Form1.Canvas.Pixels[round(x*m),round(y*m)]:=clred;
 x:=x+0.1;
End;
```

Центр координат

Однако данный график будет не привычен, для математика, привыкшего к центру координат расположенных в центре экрана, а не в левом верхнем углу. Для того, чтобы переместить центр координат в центр экрана (формы), необходимо вычислить его центр.

Если мы разделим ширину формы на 2, то получим центр по горизонтали. Если разделим высоту формы на 2, то получим центр по вертикали. Это и будет новый центр координат:

```
x0:=Form1.width div 2;
y0:=Form1.Height div 2;
```

Оси координат

Ось y начинается в (середина по горизонтали; 0), т.е. в точке (x0;0), а заканчивается в точке (x0; крайняя точка по вертикали), т.е. в точке (x0;form1.Height) или (x0, y0*2)

```
Form1.Canvas.MoveTo(x0, 0);
Form1.Canvas.LineTo(x0, y0*2);
```

Ось x начинается в (0; середина по вертикали), т.е. в точке (0;y0), а заканчивается в точке (крайняя точка по горизонтали; y0), т.е. в точке (x0*2;form1.Width) или (x0*2, y0)

```
Form1.Canvas.MoveTo(0, y0);
Form1.Canvas.LineTo(x0*2, y0);
```

Сокращение записи программы

Каждый раз писать Form1.Canvas достаточно долго.

Поэтому, чтобы упростить запись программы используется конструкция With … do …

Сравните запись без этой конструкции и с конструкцией:

Form1.Canvas.MoveTo(x0, 0); Form1.Canvas.LineTo(x0, y0*2);	With Form1.Canvas do Begin

```
    Form1.Canvas.MoveTo(0, y0);
    Form1.Canvas.LineTo(x0*2, y0);
```

```
    MoveTo(x0, 0);
    LineTo(x0, y0*2);
    MoveTo(0, y0);
    LineTo(x0*2, y0);
End;
```

А теперь изучите программу, строящую график функции y = cos x, с учетом центра координат и масштаба.

```
procedure TForm1.Button1Click(Sender: TObject);
var x,y:real; //координаты точки графика
x0,y0:integer;//координаты центра
m: integer;   //масштаб
begin
//вычисление центра координат и масштаба
x0:=PaintBox1.width div 2;
y0:=PaintBox1.Height div 2;
m:=10;
with PaintBox1.Canvas do
begin
 //рисование осей координат
 MoveTo(x0,0);
 LineTo(x0,y0*2);
 MoveTo(0,y0);
 LineTo(x0*2,y0);
 //вычисление функции и вывод графика
 x:=-x0 div m;
 while x<=x0 div m do
  begin
   y:=cos(x);
   Pixels[x0-round(x*m),y0-round(y*m)]:=clRed;
   x:=x+0.0001;
  end;
end;
end;
```

Анимация геометрических объектов

Движение геометрических объектов осуществляется по следующему принципу**:**

1. Установили цвет границы и заливки объекта
2. Нарисовали объект в заданной позиции
3. Изменили цвет границы на цвет формы
4. Нарисовали этот же объект в этой же позиции.
5. Далее изменили координаты объекта (небольшой сдвиг)
6. Установили цвет границы и заливки объекта
7. Нарисовали объект в заданной позиции
8. Изменили цвет границы на цвет формы
9. Нарисовали этот же объект в этой же позиции.
10. Изменили координаты объекта и т.д.

Например:

```
Form1.Canvas.Brush.Color:=clRed;
Form1.Canvas.Rectangle(x, 100, x+50, 200);
Form1.Canvas.Pen.Color:=clBtnFace;
Form1.Canvas.Rectangle(x, 100, x+50, 200);
X:=x+1;
```

Все эти операции можно поместить в цикл, для того чтобы эта последовательность повторялась столько раз, сколько потребуется. В итоге получится примерно следующая программа:

```
procedure TForm1.Button2Click(Sender: TObject);
const r=50;
var   y0:integer; x:real;
begin
```

```
with PaintBox1.Canvas do
 begin
  x:=0;y0:=100;
  while x<300 do
   begin
    brush.Color:=clred;
    rectangle(round(x-r), y0-r, round(x+r),y0+r);
    Pen.Color:=clBtnFace;
    rectangle(round(x-r), y0-r, round(x+r),y0+r);
    x:=x+0.01;
   end;
 end;
end;
```

Уровень №1

Задание 1. Законспектировать новый материал по построению графика функции и анимации геометрических объектов. Выполнить примеры с помощью Delphi.

График функции

Свойство Pixels

Используя свойство **Pixels** объекта Canvas можно задать требуемый цвет для любой точки графической поверхности.

Например, инструкция: **Form1.Canvas.Pixels[10,10]:=clRed**; окрашивает точку (10,10) формы в красный цвет.

А с помощью конструкции:

For i:=1 to 100 do form1.Canvas.Pixels[i,10]:=clRed;

можно нарисовать горизонтальную линию, начинающуюся в точке (1;10) и заканчивающуюся в точке (100;10).

Простейший график функции

Используя свойство Pixels можно строить графики функций. График строится на основе вычислений.

С помощью следующего цикла можно построить простейший график функции:

```
X:=0;
While x<100 do
Begin
 Y:=cos(x);
 Form1.Canvas.Pixels[round(x),round(y)]:=clred;
 x:=x+0.1;
End;
```

Анимация геометрических объектов

Движение геометрических объектов осуществляется по следующему принципу**:**

1. Установили цвет границы и заливки объекта
2. Нарисовали объект в заданной позиции
3. Изменили цвет границы на цвет формы
4. Нарисовали этот же объект в этой же позиции.
5. Далее изменили координаты объекта (небольшой сдвиг)
6. Установили цвет границы и заливки объекта
7. Нарисовали объект в заданной позиции
8. Изменили цвет границы на цвет формы
9. Нарисовали этот же объект в этой же позиции.
10. Изменили координаты объекта и т.д.

Пример: программы выводит на экран движущийся прямоугольник:

```
procedure TForm1.Button2Click(Sender: TObject);
var   x:real;
begin
      x:=0;
      with PaintBox1.Canvas do
       begin
        while x<300 do
         begin
          brush.Color:=clred;     rectangle(round(x-r), 50, round(x+r),150);
          Pen.Color:=clBtnFace; rectangle(round(x-r), 50, round(x+r),150);
          x:=x+0.01;
         end;
       end;
end;
```

Задание 2. Изменить пример таким образом, чтобы на экране двигался следующий элемент:

Уровень №2

Задание 1. Законспектировать новый материал по построению графика функции и анимации геометрических объектов. Выполнить примеры с помощью Delphi.

График функции

Свойство Pixels

Используя свойство **Pixels** объекта Canvas можно задать требуемый цвет для любой точки графической поверхности.

Например, инструкция: **Form1.Canvas.Pixels[10,10]:=clRed**; окрашивает точку (10,10) формы в красный цвет.

А с помощью конструкции:

For i:=1 to 100 do form1.Canvas.Pixels[i,10]:=clRed;

можно нарисовать горизонтальную линию, начинающуюся в точке (1;10) и заканчивающуюся в точке (100;10).

Простейший график функции

Используя свойство Pixels можно строить графики функций. График строится на основе вычислений.

С помощью следующего цикла можно построить простейший график функции:

```
X:=0;
While x<100 do
Begin
 Y:=cos(x);
 Form1.Canvas.Pixels[round(x),round(y)]:=clred;
 x:=x+0.1;
End;
```

Однако, график построенный с помощью подобной функции будет очень мелким.

График функции в увеличенном масштабе

Для того, чтобы увеличить масштаб графика, добавим переменную m – масштаб. Установим ее равной 10, и при выводе точки на экран будем умножать координату на это значение.

```
X:=0;
M:=10;
While x<100 do
Begin
 Y:=cos(x);
 Form1.Canvas.Pixels[round(x*m),round(y*m)]:=clred;
 x:=x+0.1;
End;
```

Анимация геометрических объектов

Движение геометрических объектов осуществляется по следующему принципу**:**

1. Установили цвет границы и заливки объекта
2. Нарисовали объект в заданной позиции
3. Изменили цвет границы на цвет формы
4. Нарисовали этот же объект в этой же позиции.
5. Далее изменили координаты объекта (небольшой сдвиг)
6. Установили цвет границы и заливки объекта
7. Нарисовали объект в заданной позиции
8. Изменили цвет границы на цвет формы
9. Нарисовали этот же объект в этой же позиции.
10. Изменили координаты объекта и т.д.

Например:

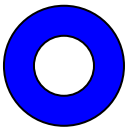
```
Form1.Canvas.Brush.Color:=clRed;
Form1.Canvas.Rectangle(x, 100, x+50, 200);
Form1.Canvas.Pen.Color:=clBtnFace;
Form1.Canvas.Rectangle(x, 100, x+50, 200);
X:=x+1;
```

Все эти операции можно поместить в цикл, для того чтобы эта последовательность повторялась столько раз, сколько потребуется. В итоге получится примерно следующая программа:

```
procedure TForm1.Button2Click(Sender: TObject);
const r=50;
var   y0:integer; x:real;
begin
with PaintBox1.Canvas do
 begin
  x:=0;y0:=100;
  while x<300 do
   begin
    brush.Color:=clred;
    rectangle(round(x-r), y0-r, round(x+r),y0+r);
    Pen.Color:=clBtnFace;
    rectangle(round(x-r), y0-r, round(x+r),y0+r);
    x:=x+0.01;
   end;
 end;
end;
```

Задание 2. Изменить программу таким образом, чтобы окружность двигалась не по горизонтали, а по вертикали.

Задание 3. Составить программу, выводящую на экран движение следующего элемента:



Уровень №3

Задание 1. Законспектировать новый материал по построению графика функции и анимации геометрических объектов. Выполнить примеры с помощью Delphi.

График функции

Свойство Pixels

Используя свойство **Pixels** объекта Canvas можно задать требуемый цвет для любой точки графической поверхности.

Например, инструкция: **Form1.Canvas.Pixels[10,10]:=clRed**; окрашивает точку (10,10) формы в красный цвет.

А с помощью конструкции:

For i:=1 to 100 do form1.Canvas.Pixels[i,10]:=clRed;

можно нарисовать горизонтальную линию, начинающуюся в точке (1;10) и заканчивающуюся в точке (100;10).

Простейший график функции

Используя свойство Pixels можно строить графики функций. График строится на основе вычислений.

С помощью следующего цикла можно построить простейший график функции:

```
X:=0;
While x<100 do
Begin
 Y:=cos(x);
 Form1.Canvas.Pixels[round(x),round(y)]:=clred;
 x:=x+0.1;
End;
```

Однако, график построенный с помощью подобной функции будет очень мелким.

График функции в увеличенном масштабе

Для того, чтобы увеличить масштаб графика, добавим переменную m – масштаб. Установим ее равной 10, и при выводе точки на экран будем умножать координату на это значение.

```
X:=0;
```

```
M:=10;
While x<100 do
Begin
 Y:=cos(x);
 Form1.Canvas.Pixels[round(x*m),round(y*m)]:=clred;
 x:=x+0.1;
End;
```

Центр координат

Однако данный график будет не привычен, для математика, привыкшего к центру координат расположенных в центре экрана, а не в левом верхнем углу. Для того, чтобы переместить центр координат в центр экрана (формы), необходимо вычислить его центр.

Если мы разделим ширину формы на 2, то получим центр по горизонтали. Если разделим высоту формы на 2, то получим центр по вертикали. Это и будет новый центр координат:

```
x0:=Form1.width div 2;
y0:=Form1.Height div 2;
```

Оси координат

Ось y начинается в (середина по горизонтали; 0), т.е. в точке (x0;0), а заканчивается в точке (x0; крайняя точка по вертикали), т.е. в точке (x0;form1.Height) или (x0, y0*2)

```
Form1.Canvas.MoveTo(x0, 0);
 Form1.Canvas.LineTo(x0, y0*2);
```

Ось x начинается в (0; середина по вертикали), т.е. в точке (0;y0), а заканчивается в точке (крайняя точка по горизонтали; y0), т.е. в точке (x0*2;form1.Width) или (x0*2, y0)

```
Form1.Canvas.MoveTo(0, y0);
Form1.Canvas.LineTo(x0*2, y0);
```

Сокращение записи программы

Каждый раз писать Form1.Canvas достаточно долго.

Поэтому, чтобы упростить запись программы используется конструкция With … do …

Сравните запись без этой конструкции и с конструкцией:

```
Form1.Canvas.MoveTo(x0, 0);
Form1.Canvas.LineTo(x0, y0*2);
Form1.Canvas.MoveTo(0, y0);
Form1.Canvas.LineTo(x0*2, y0);
```

```
With Form1.Canvas do
Begin
    MoveTo(x0, 0);
    LineTo(x0, y0*2);
    MoveTo(0, y0);
    LineTo(x0*2, y0);
End;
```

А теперь изучите программу, строящую график функции y = cos x, с учетом центра координат и масштаба.

```
procedure TForm1.Button1Click(Sender: TObject);
var x,y:real; //координаты точки графика
x0,y0:integer;//координаты центра
m: integer;   //масштаб
begin
//вычисление центра координат и масштаба
x0:=PaintBox1.width div 2;
y0:=PaintBox1.Height div 2;
m:=10;
with PaintBox1.Canvas do
begin
 //рисование осей координат
 MoveTo(x0,0);
 LineTo(x0,y0*2);
 MoveTo(0,y0);
 LineTo(x0*2,y0);
 //вычисление функции и вывод графика
 x:=-x0 div m;
 while x<=x0 div m do
  begin
   y:=cos(x);
   Pixels[x0-round(x*m),y0-round(y*m)]:=clRed;
```

```
    x:=x+0.0001;
   end;
  end;
 end;
```

Анимация геометрических объектов

Движение геометрических объектов осуществляется по следующему принципу**:**

11.Установили цвет границы и заливки объекта
12.Нарисовали объект в заданной позиции
13.Изменили цвет границы на цвет формы
14.Нарисовали этот же объект в этой же позиции.
15.Далее изменили координаты объекта (небольшой сдвиг)
16.Установили цвет границы и заливки объекта
17.Нарисовали объект в заданной позиции
18.Изменили цвет границы на цвет формы
19.Нарисовали этот же объект в этой же позиции.
20.Изменили координаты объекта и т.д.

Например:

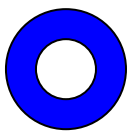
```
Form1.Canvas.Brush.Color:=clRed;
Form1.Canvas.Rectangle(x, 100, x+50, 200);
Form1.Canvas.Pen.Color:=clBtnFace;
Form1.Canvas.Rectangle(x, 100, x+50, 200);
X:=x+1;
```

Все эти операции можно поместить в цикл, для того чтобы эта последовательность повторялась столько раз, сколько потребуется. В итоге получится примерно следующая программа:

```
procedure TForm1.Button2Click(Sender: TObject);
const r=50;
var   y0:integer; x:real;
begin
with PaintBox1.Canvas do
 begin
  x:=0;y0:=100;
  while x<300 do
   begin
    brush.Color:=clred;
    rectangle(round(x-r), y0-r, round(x+r),y0+r);
    Pen.Color:=clBtnFace;
    rectangle(round(x-r), y0-r, round(x+r),y0+r);
    x:=x+0.01;
   end;
 end;
end;
```

Задание 2. Изменить программу таким образом, чтобы окружность двигалась не по горизонтали, а по вертикали и с другой скоростью.

Задание 3. Составить программу, выводящую на экран движение следующего элемента по диагонали формы (или PaintBox):



УРОК №21. ПРОЦЕДУРЫ. ПРОЦЕДУРЫ С ПАРАМЕТРАМИ

Цель:

Образовательная: Создать условия для усвоения понятия подпрограммы, процедуры. Рассмотреть на примерах использование процедур и процедур с параметрами.

Воспитательная: Содействовать воспитанию организованности, дисциплинированности; воспитанию любви к профессии, воспитанию навыков коллективного труда

Развивающая: Содействовать развитию познавательных способностей

Тип занятия: урок изучения нового материала

Форма организации учебного процесса: Самостоятельная работа с текстом

Наглядные пособия и ТСО: Карточки входного контроля, учебный материал и задания, разбитые по уровням в электронном виде

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Входной контроль.** Работа по карточкам. Проверка домашнего задания.
3. **Определение уровней.** Учащимся предлагается самим, по результатам выполнения входного контроля, определить какого уровня задания они будут выполнять.
   1 уровень – задание на воспроизведение учебного материала и по аналогии
   2 уровень – задание по аналогии и задание, требующее разработки своего алгоритма.
4. **Изучение нового материала.**
5. **Выполнение заданий по уровням.**
6. **Домашнее задание:** разрешается выполнить одно из заданий, при этом выполнение 1 задания оценивается на оценку 4, а выполнение второго задания оценивается на оценку 5
7. **Подведение итогов урока. Выставление оценок за урок** (оценка за домашнее задание по 1 уроку, оценка за входной контроль, оценка за работу на уроке).

КАРТОЧКИ ДЛЯ ВХОДНОГО КОНТРОЛЯ

Вариант №1. Тема «Графика в Delphi»

1. С помощью какого оператора можно нарисовать прямоугольник, и какими координатами он задается?
2. Для чего служит свойство Pixels, и для чего его можно применять?
3. Определите, что рисует следующий фрагмент:

```
PaintBox1.Canvas.MoveTo(10,20);
PaintBox1.Canvas.LineTo(20,10);
PaintBox1.Canvas.LineTo(30,20);
PaintBox1.Canvas.Rectangle(10,20,30,40);
```

Вариант №2. Тема «Графика в Delphi»

1. С помощью какого оператора можно нарисовать окружность, и какими координатами она задается?
2. Что выполняет оператор PaintBox1.Canvas.MoveTo(10,20)?
3. Что рисует следующий фрагмент?

```
PaintBox1.Canvas.Rectangle(10,20,30,40);
PaintBox1.Canvas.MoveTo(10,20);
PaintBox1.Canvas.LineTo(30,40);
PaintBox1.Canvas.MoveTo(30,20);
PaintBox1.Canvas.LineTo(10,40);
```

Вариант №3. Тема «Графика в Delphi»

1. Какой оператор рисует линию из текущей точки в точку с координатами (x;y)?
2. Что выполняет оператор PaintBox1.Canvas.Rectangle(0,0,10,10)?
3. Определите, что рисует следующий фрагмент:

```
PaintBox1.Canvas.MoveTo(10,20);
PaintBox1.Canvas.LineTo(30,40);
PaintBox1.Canvas.MoveTo(30,20);
PaintBox1.Canvas.LineTo(10,40);
```

Вариант №4. Тема «Графика в Delphi»

1. Каким оператором можно переместить перо в точку (x;y)?
2. Что выполняет оператор PaintBox1.Canvas.Pixels(10,10):=clRed?
3. Определите, что рисует следующий фрагмент:

```
PaintBox1.Canvas.MoveTo(10,20);
PaintBox1.Canvas.LineTo(30,40);
PaintBox1.Canvas.LineTo(10,40);
PaintBox1.Canvas.LineTo(10,20);
```

Вариант №1. Тема «Графика в Delphi»

1. С помощью какого оператора можно нарисовать прямоугольник, и какими координатами он задается?
2. Для чего служит свойство Pixels, и для чего его можно применять?
3. Определите, что рисует следующий фрагмент:

```
PaintBox1.Canvas.MoveTo(10,20);
PaintBox1.Canvas.LineTo(20,10);
PaintBox1.Canvas.LineTo(30,20);
PaintBox1.Canvas.Rectangle(10,20,30,40);
```

Вариант №2. Тема «Графика в Delphi»

1. С помощью какого оператора можно нарисовать окружность, и какими координатами она задается?
2. Что выполняет оператор PaintBox1.Canvas.MoveTo(10,20)?
3. Что рисует следующий фрагмент?

```
PaintBox1.Canvas.Rectangle(10,20,30,40);
PaintBox1.Canvas.MoveTo(10,20);
PaintBox1.Canvas.LineTo(30,40);
PaintBox1.Canvas.MoveTo(30,20);
PaintBox1.Canvas.LineTo(10,40);
```

Вариант №3. Тема «Графика в Delphi»

1. Какой оператор рисует линию из текущей точки в точку с координатами (x;y)?
2. Что выполняет оператор PaintBox1.Canvas.Rectangle(0,0,10,10)?
3. Определите, что рисует следующий фрагмент:

```
PaintBox1.Canvas.MoveTo(10,20);
PaintBox1.Canvas.LineTo(30,40);
PaintBox1.Canvas.MoveTo(30,20);
PaintBox1.Canvas.LineTo(10,40);
```

Вариант №4. Тема «Графика в Delphi»

1. Каким оператором можно переместить перо в точку (x;y)?
2. Что выполняет оператор PaintBox1.Canvas.Pixels(10,10):=clRed?
3. Определите, что рисует следующий фрагмент:

```
PaintBox1.Canvas.MoveTo(10,20);
PaintBox1.Canvas.LineTo(30,40);
PaintBox1.Canvas.LineTo(10,40);
PaintBox1.Canvas.LineTo(10,20);
```

ОТВЕТЫ НА КАРТОЧКИ ВХОДНОГО КОНТРОЛЯ

Вариант №1. Тема «Графика в Delphi» 1. Rectangle(x1,y1,x2,y2) 2. Pixels рисует точку с координатами (x;y), указанного цвета, его можно применять для построения графиков функций. 3. 	**Вариант №2. Тема «Графика в Delphi»** 1. Ellipse(x1,y1,x2,y2) 2. Перемещает перо в точку (10,20) 3. 
Вариант №3. Тема «Графика в Delphi» 1. LineTo(x;y) 2. Рисует прямоугольник с левым верхним углом (0;0) и правым нижним (10,10) 3. 	**Вариант №4. Тема «Графика в Delphi»** 1. MoveTo(x;y)? 2. Рисует красную точку с координатами (10;10) 3. 

ОСНОВНОЙ УЧЕБНЫЙ МАТЕРИАЛ

Предположим, перед нами стоит задача, составить программу, для рисования рисунка 1.

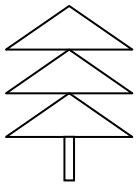


Рисунок 1

Вопрос: Каким образом выполняется этот рисунок с помощью операторов графики в Delphi?
Ответ: с помощью операторов LineTo и MoveTo.
Давайте составим программу, которая рисует такую елочку.

```
with form1.Canvas do
begin
   //первый треугольник
    MoveTo(100,10);
    LineTo(150,50);
    LineTo(50,50);
    LineTo(100,10);
   //второй треугольник
    MoveTo(100,50);
    LineTo(150,100);
    LineTo(50,100);
    LineTo(100,50);
   //третий треугольник
    MoveTo(100,100);
    LineTo(150,150);
    LineTo(50,150);
    LineTo(100,100);
    Rectangle(95,150,105,170);
end;
```

Обратите внимание на то, что три раза повторяется практически одинаковый фрагмент программы (треугольник описывается четырьмя операторами), незначительно меняются только координаты. Каким образом можно оформить программу, чтобы не писать каждый раз одно и то же.

Эту последовательность действий (4 оператора) можно оформить в виде подпрограммы. Подпрограммы сокращают текст программы, существенно уменьшают время их исполнения, облегчают жизнь программистам, которые могут создавать программы модульно, т.е. собирая сложную программу из законченных кусочков более простых составляющих. Это позволяет создавать большие программы группой программистов, разрабатывать и реализовывать глобальные проекты.

Давайте запишем, каким образом описывается подпрограмма, и затем попробуем все вместе составить подпрограмму для рисования треугольника.

Подпрограммы представляют собой самостоятельные фрагменты программы, оформленные особым образом и снабженные именем. Упоминание этого имени в тексте программы называется *вызовом* подпрограммы.

Два типа подпрограмм: процедуры и функции.

Сегодня мы остановимся подробнее на процедурах, а с понятием функции вы познакомитесь на следующем уроке.

Процедуры могут содержаться в любом месте до основного тела программы. Для процедур используется следующий формат:

```
Procedure имя_процедуры (параметры);
begin
  {тело процедуры - операторы}
end;
```

В описании процедуры или функции задается список формальных параметров. Все формальные параметры могут быть использованы только в тех подпрограммах, где они объявлены.

Теперь вернемся к нашему примеру и для начала придумаем имя нашей процедуре (например, Treug или любое другое, записанное латинскими буквами).

Разберемся, что будет делать наша процедура (рисовать треугольник по трем заданным точкам). Таким образом, параметрами, отправляемыми в процедуру, будут координаты трех точек, т.е. 6 переменных.

```
procedure treug(x1,y1,x2,y2,x3,y3:integer);
begin
  with form1.Canvas do
   begin
   MoveTo(x1,y1);
   LineTo(x2,y2);
   LineTo(x3,y3);
   LineTo(x1,y1);
 end;
end;
```

В данной процедуре x1,y1,x2,y2,x3,y3 – формальные параметры. Процедура готова, теперь осталось вызвать ее из основной программы три раза, при этом передавая ей различные фактические параметры.

```
procedure treug(x1,y1,x2,y2,x3,y3:integer);
begin
  with form1.Canvas do
begin
 MoveTo(x1,y1);
 LineTo(x2,y2);
 LineTo(x3,y3);
 LineTo(x1,y1);
 end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
with form1.Canvas do
begin
     treug(100,10,150,50,50,50);  //в скобках фактические параметры процедуры
     treug(100,50,150,100,50,100);
     treug(100,100,150,150,50,150);
     Rectangle(95,150,105,170);
end;
end;
```

Задание: какую процедуру надо создать, чтобы нарисовать несколько елочек в разных частях экрана. Опишите такую процедуру. Что в ней будет являться параметрами?

```
procedure treug(x1,y1,x2,y2,x3,y3:integer);
begin
  with form1.Canvas do
```

```
begin
 MoveTo(x1,y1);
 LineTo(x2,y2);
 LineTo(x3,y3);
 LineTo(x1,y1);
 end;
end;
procedure elka(x0,y0,k,t:integer);
begin
with form1.Canvas do
begin
   treug(x0,y0,      x0+k,   y0+t,    x0-k,  y0+t);
   treug(x0,y0+t,    x0+k,  y0+2*t, x0-k,  y0+2*t);
   treug(x0,y0+2*t, x0+k,  y0+3*t, x0-k,  y0+3*t);
   Rectangle(x0-k div 10,y0+3*t,x0+k div 10,y0+3*t+20);
end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
 elka(100,10,50,50);
 elka(300,10,50,50);
 elka(250,150,50,50);
end;
```

ДОПОЛНИТЕЛЬНЫЙ МАТЕРИАЛ

Пример процедуры: Написать программу для вычисления сочетаний из n по m, оформив вычисление факториала процедурой без параметров, процедурой с параметрами, функцией. Число сочетаний из n по m вычисляется по формуле: $C_n^m = \frac{n!}{m!(n-m)!}$.

Процедура без параметров	Процедура с параметрами

Var n,m,c,fn,fm,fnm,p,q:longint;
Procedure fact;
Var i:longint;
Begin
P:=1;
For i:=1 to q do p:=p*i;
End;
Begin
N:=strtoint(edit1.text);
M:=strtoint(edit2.text);
q:=n;
fact;
fn:=p;
q:=m;
fact;
fm:=p;
q:=n-m;
fact;
fnm:=p;
c:=fn div (fm*fnm);
Memo1.Lines.Add(‘c=’+inttostr(c));
End;
	Var n,m,c,fn,fm,fnm:longint;
	Procedure fact(q:longint;var p:longint);
	Var i:longint;
	Begin
	P:=1;
	For i:=1 to q do p:=p*i;
	End;
	Begin
	N:=strtoint(edit1.text);
	M:=strtoint(edit2.text);
	fact(n,fn);
	fact(m,fm);
	fact(n-m,fnm);
	c:=fn div (fm*fnm);
	Memo1.Lines.Add(‘c=’+inttostr(c));
	End;

УЧЕБНЫЙ МАТЕРИАЛ, РАЗБИТЫЙ НА УРОВНИ

Задание 1 (на оценку 3). Законспектировать учебный материал. Примеры выполнить с помощью Delphi.

Пример №1. Составить программу, для рисования рисунка 1.

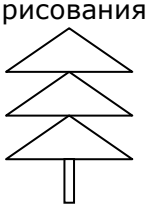


Рисунок 1

Решение:

```
with PaintBox1.Canvas do
begin
   //первый треугольник
    MoveTo(100,10);
    LineTo(150,50);
    LineTo(50,50);
    LineTo(100,10);
   //второй треугольник
    MoveTo(100,50);
    LineTo(150,100);
    LineTo(50,100);
    LineTo(100,50);
   //третий треугольник
    MoveTo(100,100);
    LineTo(150,150);
    LineTo(50,150);
    LineTo(100,100);
    Rectangle(95,150,105,170);
end;
```

Обратите внимание на то, что три раза повторяется практически одинаковый фрагмент программы (треугольник описывается четырьмя операторами), незначительно меняются только координаты. Каким образом можно оформить программу, чтобы не писать каждый раз одно и то же.

Эту последовательность действий (4 оператора) можно оформить в виде подпрограммы. Подпрограммы сокращают текст программы, существенно уменьшают время их исполнения.

Подпрограммы представляют собой самостоятельные фрагменты программы, оформленные особым образом и снабженные именем. Упоминание этого имени в тексте программы называется *вызовом* подпрограммы.

Существует два типа подпрограмм: процедуры и функции.

Сегодня мы остановимся подробнее на процедурах, а с понятием функции вы познакомитесь на следующем уроке.

Процедуры могут содержаться в любом месте до основного тела программы. Для процедур используется следующий формат:

```
Procedure имя_процедуры (параметры);
begin
  {тело процедуры - операторы}
end;
```

В описании процедуры или функции задается список формальных параметров. Все формальные параметры могут быть использованы только в тех подпрограммах, где они объявлены.

Теперь вернемся к нашему примеру и для начала придумаем имя нашей процедуре (например, Treug или любое другое, записанное латинскими буквами).

Разберемся, что будет делать наша процедура (рисовать треугольник по трем заданным точкам). Таким образом, параметрами, отправляемыми в процедуру, будут координаты трех

точек. Т.к. каждая точка описывается двумя координатами x и y, то для описания треугольника потребуется 6 переменных.

```
procedure treug(x1,y1,x2,y2,x3,y3:integer);
begin
  with form1.Canvas do
   begin
   MoveTo(x1,y1);
   LineTo(x2,y2);
   LineTo(x3,y3);
   LineTo(x1,y1);
 end;
end;
```

В данной процедуре x1,y1,x2,y2,x3,y3 – формальные параметры. Процедура готова, теперь осталось вызвать ее из основной программы три раза, при этом передавая ей различные фактические параметры.

```
procedure treug(x1,y1,x2,y2,x3,y3:integer);
begin
  with form1.Canvas do
begin
 MoveTo(x1,y1);
 LineTo(x2,y2);
 LineTo(x3,y3);
 LineTo(x1,y1);
 end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
with form1.Canvas do
begin
     treug(100,10,150,50,50,50);  //в скобках фактические параметры процедуры
     treug(100,50,150,100,50,100);
     treug(100,100,150,150,50,150);
     Rectangle(95,150,105,170);
end;
end;
```

ЗАДАНИЯ ДЛЯ УРОВНЯ №1.

Задание 2 (на оценку 4). Вопросы для устного индивидуального ответа:

1. Для чего используется процедуры в программировании?
2. Как описывается процедура в программировании, и в каком месте программы она располагается?
3. Какие параметры называются формальными, какие - фактическими?

Задание 3 (на оценку 5): Составить процедуру рисующую ромб и с ее помощью нарисовать рисунок 2:

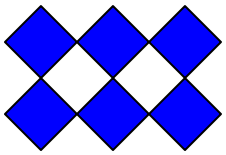


Рисунок 2

ЗАДАНИЯ ДЛЯ УРОВНЯ №2.

Задание 2 (на оценку 4): Создайте процедуру рисования стрелочки и с ее помощью создайте рисунок 2.

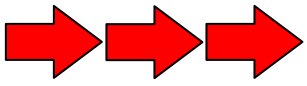


Рисунок 2.

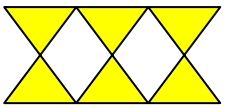
Задание 3 (на оценку 5): Какую процедуру надо создать, чтобы нарисовать несколько елочек в разных частях экрана. Опишите такую процедуру. Что в ней будет являться параметрами?

ДОМАШНЕЕ ЗАДАНИЕ

Уровень 1 (на оценку 4). Составить процедуру рисования прямоугольного треугольника. С помощью этой процедуры составить следующий рисунок:

Уровень 2 (на оценку 5). Составить процедуру рисующую объект 1. Используя эту процедуру составить вторую процедуру рисующую объект 2. Используя вторую процедуру вывести объект 2 в трех различных частях экрана.

Объект 1



Объект 2

УРОК №22. ФУНКЦИИ. РЕКУРСИВНЫЕ ФУНКЦИИ. ЛОКАЛЬНЫЕ И ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ.

Цель:

Образовательная: Создать условия для усвоения понятия функции и рекурсии. Рассмотреть на примерах использование функции. Ознакомится с понятием локальной и глобальной переменной.

Воспитательная: Содействовать воспитанию организованности, дисциплинированности; воспитанию любви к профессии, воспитанию навыков коллективного труда

Развивающая: Содействовать развитию познавательных способностей

Тип занятия: урок изучения нового материала

Форма организации учебного процесса: Разноуровневое обучение

Наглядные пособия и ТСО:

Литература: С.Н. Лукин, Turbo Pascal 7.0

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Входной контроль** (устная проверка знаний) **и проверка домашнего задания.**
3. **Изучение нового материала.** Функции.
4. **Изучение нового материала.** Локальные и глобальные переменные.
5. **Изучение нового материала.** Рекурсия.
6. **Выполнение упражнений и решение задач по уровням**
7. **Домашнее задание:**
8. **Подведение итогов урока.**

ВХОДНОЙ КОНТРОЛЬ

Вопросы для устного ответа:

1. Что такое подпрограмма и для чего она используется в программировании?
2. Какие два типа подпрограмм вы знаете?
3. Каким образом описывается процедура?
4. Что называется формальными параметрами? Где используются формальные параметры?
5. Что называется фактическими параметрами? Где они используются?
6. Как вызывается процедура?

Критерии оценки:

Правильно ответили на 6 вопросов – «5 – отлично»
Правильно ответили на 4-5 вопросов – «4 - хорошо»
Правильно ответили на 2-3 вопроса – «3 - удовлетворительно»

Правила проверки знаний: Те, кто готов отвечать на вопросы без подготовки подсаживаются к преподавателю и отвечают. Каждый учащийся, которого проверил преподаватель, должен проверить еще двух учащихся и оценить их. Результаты предоставить преподавателю. Преподаватель имеет право перепроверить знания учащегося, опрошенного другим учащимся, если сомневается в правильности оценки знаний.

ДОПОЛНИТЕЛЬНЫЙ УЧЕБНЫЙ МАТЕРИАЛ

Пример функции: Составить функцию, для возведения числа a в степень n.
Решение:

```
Function Power(a,n:integer):integer;
Var s, i: integer;
Begin
  S:=1;
  For i:=1 to n do s:=s*a;
  Power:=s;
End;

Procedure TForm1.Button1Click(Sender:TObject);
Var A, N:integer;
Begin
  A:=StrToInt(Edit1.Text);
  N:=StrToInt(Edit2.Text);
  ShowMessage(IntToStr(Power(A ,N)))
End;
```

Пример рекурсивной функции: Составить рекурсивную функцию fib, для вычисления чисел Фибоначчи.

Решение:

```
Function fib(n:integer):integer;
Begin
  If n=1 then fib:=1;
  If n=2 then fib:=1;
  If N>2 then fib:=fib(n-2)+fib(n-1);
End;
Var i : integer;
Begin
  For i:=1 to 10 do memo1.lines.add(inttostr(fib(i)));
End;
```

Пример функции и рекурсии: Написать программу для вычисления сочетаний из n по m, оформив вычисление факториала функцией и рекурсивной функцией. Число сочетаний из n по m вычисляется по формуле: $C_n^m = \frac{n!}{m!(n-m)!}$.

Функция	Рекурсия

Var n,m,c:integer;
Function fact(q:integer):integer;
Var i,p:integer;
Begin
P:=1;
For i:=1 to q do p:=p*i;
Fact:=p;
End;
Begin
n:=strtoint(Edit1.text);
m:=strtoint(Edit2.text);
c:=fact(n)div(fact(m)*fact(n-m));
ShowMessage(inttostr(c));
end;
	Var n,m,c:integer;
	Function fact(n:integer):integer;
	Begin
	If n=1 then f:=1;
	If n>1 then f:=n*f(n-1);
	End;
	Begin
	n:=strtoint(Edit1.text);
	m:=strtoint(Edit2.text);
	c:=fact(n)div(fact(m)*fact(n-m));
	ShowMessage(inttostr(c));
	end;

УПРАЖНЕНИЯ

УПРАЖНЕНИЕ 1. Процедура с параметрами. Определить, что выводится на печать в результате выполнения программы, если переменная R имеет значение:

- 2
- 3
- 4

```
Var r, s: real;
Procedure lang(b:real; var c:real);
```

```
Begin
  c:=pi*sqr(b);
End;

Begin
  Read(r);
  Lang(r, s);
  Memo1.Lines.Add('s='+floattostr(s));
End;
```

Ответ:
1) s=12,5663706143592
2) s=28,2743338823081
3) s=50,2654824574367

УПРАЖНЕНИЕ 2. Функция. Определить чему равно f(3,9)?

```
Function f(x,y:real):real;
Begin
  If x>=y Then f:=(x+y)/2
          Else f:=(x+2)*(y+1);
End;
```

Ответ: результат = 0,5.

УПРАЖНЕНИЕ 3. Локальные и глобальные переменные. Определить, что будет напечатано в результате работы первого и второго варианта программы. Сравнить результаты и сделать выводы:

1 вариант	2 вариант

Var x:real;
Procedure WriteX;
Var x:real;
Begin
ShowMessage(floattostr(x));
end;
 	
begin
x:=pi;
writeX;
end;
	Var x:real;
	Procedure WriteX;
	
	Begin
	ShowMessage(floattostr(x));
	end;
	
	begin
	x:=pi;
	writeX;
	end;

Ответ: в первом случае на экран будет выведено случайное число, т.к. переменная описанная внутри процедуры и не видна снаружи. Во втором случае на экран выводится значение Пи.

УПРАЖНЕНИЕ 4. Локальные и глобальные переменные. Определите результат выполнения фрагмента программы.

```
var  x,z:integer;

procedure B;
var x,y:integer;
begin
 x:=20; y:=30; z:=40;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
x:=1;z:=2;
B;
ShowMessage(IntToStr(x)+' '+IntToStr(z));
end;
```

Ответ: 1 40

УПРАЖНЕНИЕ 5. Рекурсия. Определить чему равно f(1,10)?

```
Function f(x,y:real):real;
Begin
  If x>=y Then f:=(x+y)/2
          Else f:=f(f(x+2,y-1),f(x+1,y-2));
End;
```

Ответ: результат = 5,5

КРИТЕРИИ ОЦЕНКИ РАБОТЫ НА УРОКЕ

Оценка	Критерии оценки
3	Составлен конспект в тетради и примеры выполнены в Delphi
4	Составлен конспект в тетради и примеры выполнены в Delphi. Упражнения выполнены и объяснен принцип решения упражнения.
5	Составлен конспект в тетради и примеры выполнены в Delphi. Упражнения выполнены и объяснен принцип решения упражнения. Решена задача на Delphi.

КОНСПЕКТ ПОД ЗАПИСЬ

Общий вид функции:

```
Function имя_функции (параметры): тип данных;
Begin
    {тело функции}
End;
```

Отличие функции от процедуры:

1. В заголовке функции указывается тип функции.
2. Внутри тела функции ей хотя бы один раз должно быть присвоено какое-нибудь значение (имя_функции:=число).
3. Результатом исполнения функции, всегда является некоторое значение, поэтому обращение к функции можно использовать в соответствующих выражениях наряду с переменными и константами (например X:=Sin(A)).

Пример №1: Составить программу вычисления суммы длин заборов вокруг трех несоприкасающихся прямоугольных дворов с шириной и длиной 5 и 10, 9 и 4, 4 и 5.

Решение: создадим функцию Pr, которая будет вычислять периметр забора.

```
Function Pr(a,b:integer):integer;
Begin
 Pr:=2*(a+b);
End;
Procedure TForm1.Button1Click(Sender:TObject);
Begin
 Memo1.Lines.Add(IntToStr(Pr(5,10))+ IntToStr(Pr(9,4))+ IntToStr(Pr(5,4)));
End;
```

Локальные и глобальные переменные

Локальная переменная **–** это переменная, описанная внутри подпрограммы. Локальная переменная невидима снаружи (в других подпрограммах). Она существует, пока работает подпрограмма, и исчезает при выходе из нее.

Глобальная переменная – это переменная, описанная выше всех подпрограмм. Она видна во всех процедурах и каждая подпрограмма может изменить ее значение. Но, если подпрограмма натыкается на переменную, с таким же именем, как глобальная, описанную в самой подпрограмме, она работает только с ней (с локальной) и не изменяет значение глобальной.

Локальные и глобальные переменные, компилятор считает разными, даже если они имеют одинаковые имена.

Рекурсия

Определение факториала: Факториал единицы равен 1. Факториал любого положительного целого числа N, большего 1, равен числу N, умноженному на факториал N-1.

Обозначим кратко факториал числа n как f(n) и распишем определение так:

Если n=1, то f(n)=1.
Если n>1, то f(n)=n*f(n-1).

Напишем функцию вычисления факториала, используя определение:

```
Function f(n:integer):Integer;
Begin
  If n=1 then f:=1;
  If n>1 then f:=n*f(n-1);
End;
Procedure Tform1.Button1Click(Sender:TObject);
Begin
  ShowMessage(inttostr(f(3)));
End;
```

Рекурсия – вызов подпрограммы из тела самой подпрограммы.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ ДЛЯ ПОДРОБНОГО ИЗУЧЕНИЯ

Функции имеют такой же формат, что и процедуры, за исключением того, что они начинаются с заголовка Function и заканчиваются типом данных для возвращаемого значения функции.

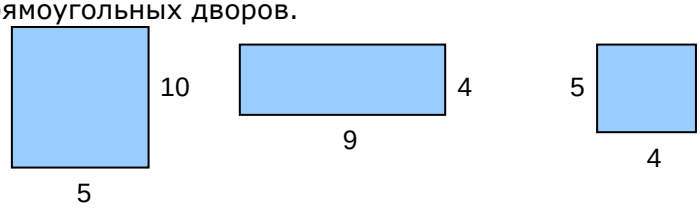
Общий вид функции:

```
Function имя_функции (параметры): тип данных;
Begin
    {тело функции}
End;
```

Отличие функции от процедуры:

1. в заголовке функции указывается тип функции
2. внутри тела функции ей хотя бы один раз должно быть присвоено какое-нибудь значение (имя_функции:=число)
3. результатом исполнения функции, всегда является некоторое значение, поэтому обращение к функции можно использовать в соответствующих выражениях наряду с переменными и константами (например X:=Sin(A))

Пример №1: Составить программу вычисления суммы длин заборов вокруг трех несоприкасающихся прямоугольных дворов.



Решение: создадим функцию Pr, которая будет вычислять периметр забора, по длине и ширине.

```
Function Pr(a,b:integer):integer;
Begin
 Pr:=2*(a+b);
End;

Procedure TForm1.Button1Click(Sender:TObject);
Begin
 Memo1.Lines.Add(IntToStr(Pr(5,10))+ IntToStr(Pr(9,4))+ IntToStr(Pr(5,4)));
End;
```

Локальные и глобальные переменные

Локальная переменная – это переменная, описанная внутри подпрограммы.

Глобальная переменная – это переменная, описанная выше всех подпрограмм.

Сравним два примера:

var x:integer;***//глобальная переменная***
 	
procedure Primer;
begin
x:=10*10;
end;
 	
procedure TForm1.Button1Click(Sender:TObject);
begin
x:=5;
B;
ShowMessage(IntToStr((x)));
end;
	var x:integer; ***//глобальная переменная***
	
	procedure Primer;
	var x:integer; ***//локальная переменная***
	Begin x:=10*10;
	end;
	
	procedure TForm1.Button1Click(Sender:TObject);
	begin
	x:=5;
	B;
	ShowMessage(IntToStr((x)));
	end;

Ответ: X=100	**Ответ:** X=5

Пояснение: локальные и глобальные переменные, компилятор считает разными, даже если они имеют одинаковые имена. Т.е. переменная x, описанная в процедуре *Primer* во втором примере, и x, описанная снаружи, это две разные переменные, которые помещаются в разных местах памяти.

Локальная переменная невидима снаружи. Она локальна в этой подпрограмме. Она существует, пока работает подпрограмма, и исчезает при выходе из нее.

Глобальная переменная видна во всех процедурах. Каждая подпрограмма может изменить ее значение. Но, если подпрограмма натыкается на переменную, с таким же именем, как глобальная, описанную в самой подпрограмме, она работает только с ней (с локальной) и не изменяет значение глобальной.

Рекурсия

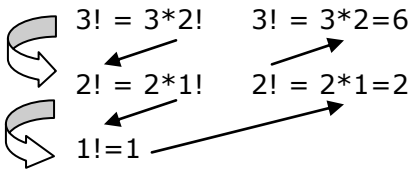
Рассмотрим задачу о нахождении факториала числа.

Определение первое: Факториал N – произведение целых чисел от 1 до N. Например: факториал 5!=1*2*3*4*5.

Существует **второе определение факториала**: Факториал единицы равен 1. факториал любого положительного целого числа N, большего 1, равен числу N, умноженному на факториал N-1.

? Как по второму определению узнать, чему равен факториал, например, трех?

Будем рассуждать так:



Рассуждая таким образом, можно вычислить факториал любого числа. Способ рассуждения называется **рекурсивным**.

Обозначим кратко факториал числа n как f(n) и распишем второе определение так:

Если n=1, то f(n)=1.
Если n>1, то f(n)=n*f(n-1).

Напишем функцию вычисления факториала, используя второе определение:

```
Function f(n:integer):Longint;
Begin
   If n=1 then f:=1;
   If n>1 then f:=n*f(n-1);
End;
Procedure Tform1.Button1Click(Sender:TObject);
Begin
   ShowMessage(inttostr(f(3)));
End;
```

Объяснение примера: обратите внимание, что в программе нигде не употребляется оператор цикла. Вся соль программы в том, что функция f вместо этого включает в себя вызов самой себя.

Что же происходит в компьютере во время выполнения программы? Механизм происходящего в точности соответствует нашему рекурсивному рассуждению.

Сначала в основной программе вызывается функция f. Выполнение подпрограммы начинается с того, что в стеке отводится место для всех формальных параметров и локальных переменных, а значит, и для формального параметра n. Затем фактический параметр 3 подставляется на место формального параметра n, т.е. в стек, в ячейку n посылается 3.

N
3

Затем выполняется тело функции. Т.к. 3>1, то выполняется попытка умножить 3*f(3-1) и сталкивается с необходимостью знать значение функции f(2), для чего вызывает ее,т.е. отправляется ее выполнять, недовыполнив f(3), но предварительно запомнив, куда возвращаться.

N	F
3	3*f(2)=

Поднимаемся на ступеньку выше. В соседнем месте стека отводится место для n. Это уже другое n, путать их нельзя. В эту ячейку n посылается 2.

N	F
2	
3	3*f(2)=

Затем выполняется тело функции. Т.к. 2>1, то выполняется попытка выполнять умножение 2*f(2-1) и сталкивается с необходимостью знать значение функции f(1), для чего вызывает ее.

N	F
2	2*f(1)=
3	3*f(2)=

Поднимаемся еще на одну ступеньку. В соседнем месте стека отводится место еще для одного n. В эту ячейку n посылается 1.

N	F
1	
2	2*f(1)=
3	3*f(2)=

Затем выполняется тело функции. Т.к. 1=1, то вычисляется f=1. Первое конкретное число. Тело функции заканчивает свою работу.

N	F
1	=1
2	2*f(1)=
3	3*f(2)=

Начинаем спускаться вниз. Программа возвращается внутрь тела функции (когда n=2) и успешно выполняет умножение f=2*1=2.

N	F
1	=1
2	2*1=2
3	3*f(2)=

Спускаемся еще на одну ступеньку. Возвращаемся в тело функции (когда n=3) и успешно выполняется умножение f=3*2=6.

N	F
1	=1
2	2*1=2
3	3*2=6

Задача решена. После выхода из подпрограммы место в стеке освобождается.

Рекурсия – вызов подпрограммы из тела самой подпрограммы.

УПРАЖНЕНИЯ И ЗАДАНИЯ

УПРАЖНЕНИЕ 1. Функция. Определить чему равно f(3,9)?

```
Function f(x,y:real):real;
Begin
  If x>=y Then f:=(x+y)/2
          Else f:=(x+2)*(y+1);
End;
```

УПРАЖНЕНИЕ 2. Локальные и глобальные переменные. Определить, что будет напечатано в результате работы первого и второго варианта программы. Сравнить результаты и сделать выводы:

1 вариант	2 вариант

Var x:real;
Procedure WriteX;
Var x:real;
Begin
ShowMessage(floattostr(x));
end;
	Var x:real;
	Procedure WriteX;
	
	Begin
	ShowMessage(floattostr(x));
	end;

begin
x:=pi;
writeX;
end;
	begin
	x:=pi;
	writeX;
	end;

УПРАЖНЕНИЕ 3. **Локальные и глобальные переменные**. Определите результат выполнения фрагмента программы.

```
var  x,z:integer;

procedure B;
var x,y:integer;
begin
 x:=20; y:=30; z:=40;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
x:=1;z:=2;
B;
ShowMessage(IntToStr(x)+' '+IntToStr(z));
end;
```

Ответ: 1 40

УПРАЖНЕНИЕ 4. Рекурсия. Определить чему равно f(1,10)?

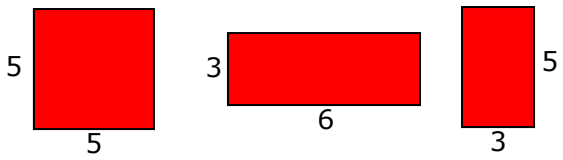
```
Function f(x,y:real):real;
Begin
  If x>=y Then f:=(x+y)/2
          Else f:=f(f(x+2,y-1),f(x+1,y-2));
End;
```

Ответ: результат = 5,5

Задание: Составить функцию max(x,y), возвращающую наибольшее из двух чисел x и y. Даны четыре числа x1, x2, x3, x4. Найти сумму s=max(x1,x2)+max(x1,x3)+ max(x1,x4)+max(x2,x3)+max(x2,x4)+max(x3,x4).

Домашнее задание:

На оценку 4: Составить функцию, вычисляющую площадь прямоугольника по его заданной длине и ширине. С ее помощью вычислить сумму площадей следующих прямоугольников:



На оценку 5: Составить функцию min(x,y), вычисляющую наименьшее из двух чисел x и y. Даны числа a,b,c. С помощью функции вычислить выражение: s=min(a,b)*min(b,c)*min(a,c)

УРОК №23. МНОЖЕСТВЕННЫЙ ТИП ДАННЫХ

Цель:

Образовательная: Создать условия для усвоения понятия множества. Рассмотреть основные операции над множествами. Познакомится с операторами сравнения множества. Рассмотреть примеры, использующие множества.

Воспитательная: Содействовать воспитанию организованности, дисциплинированности; воспитанию любви к профессии, воспитанию навыков коллективного труда

Развивающая: Содействовать развитию познавательных способностей

Тип занятия: урок изучения нового материала

Форма организации учебного процесса: Урок самостоятельной работы

Наглядные пособия и ТСО: Учебный материал в электронном виде

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Изучение нового материала.**
3. **Выполнение упражнений.**
4. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Множество - это ограниченная совокупность различных элементов. Для построения конкретного множественного типа используется перечисляемый или интервальный тип данных. Тип элементов, составляющих множество, называется базовым типом.

Количество элементов не может быть более 256 значений. Из простых типов этому условию удовлетворяют только типы byte, char и перечислимый тип. При использовании значений других типов они должны быть заранее переопределены или указаны явно простым перечислением.

Описание множества: **type <имя типа>=set of <базовый тип>**

Пример:

```
Type num = set of integer;
Var  x: num;
```

Допускается сокращение за счет перенесения определения типа в раздел Var.

Примеры:

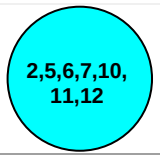
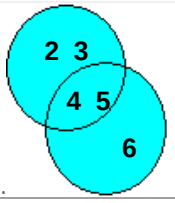
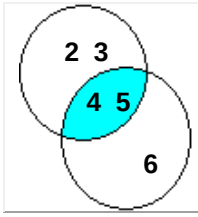
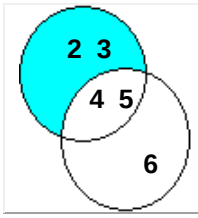
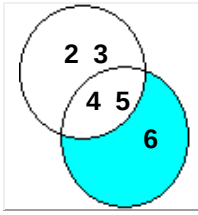
```
Var   Y       : set of char;
      Z       : set of 'A'..'Z';
      D       : set of '0'..'9';
      S, S1   : set of 2..12;
Begin
      S:=[2,6,3];
      S1:=[6,5,9,10];
      Y:=[];
End;
```

Множество представляет собой неповторяющуюся совокупность элементов. Это означает, что все повторяющиеся элементы множества не рассматриваются. Например, приведенные ниже множества эквивалентны:

['a', 'b', 'c'];

['a', 'a', 'b', 'c']
['a', 'b', 'c', 'b', 'a', 'c', 'b']

Операции над множествами

Описание действия	Фрагмент программы	Результат
Множество можно сделать пустым:	S:=[];	
Множеству можно присвоить значение	S:=[2,5..7,10..12];	
Объединение двух множеств обозначается знаком плюс. Результат объединения - множество, содержащее все элементы множеств A и B без повторений.	S:=[2..5]+[4..6];	
Пересечение обозначается звездочкой Результат пересечения - множество, содержащее элементы, входящие в множества A и B.	S:=[2..5]*[4..6];	
Разность двух множеств обозначается знаком минус. Результат разности - множество, состоящее из элементов, которые входят в множество A и не входят в B.	S:=[2..5]-[4..6];	
	S:=[4..6]-[2..5];	

Сравнение множеств:

Принадлежность отдельного элемента множеству может быть выявлена при помощи операции **IN**:	6 in [4..6] – true 6 in [2..5] – false
Для выяснения, содержит ли одно множество другое, используется знак **>=**.	[2..12]>=[3..5] – true [3..5]>=[2..12] – false
Для выяснения, содержится ли одно множество в другом, используется знак **<=**.	[2..12]<=[3..5] – false [3..5]<=[2..12] – true
Для проверки совпадения двух множеств используется **=** или **<>**	[3..5] = [5,4,3] – true [3..5]<>[5,4,3] - false

Операторы:

Include(B, x) – включает во множество B элемент x.
Exclude(B, x) – исключает из множества B элемент x.

Пример №1: Проверить принадлежит ли число N множеству A=[21,4,12,25].

```
Var N:integer;
   A:set of 1..100;          {описываем множество}
Begin
 N:=StrToInt(Edit1.Text);
 A:=[21,4,12,25];            {заполняем множество}
{Проверяем, является ли N элементом множества A}
```

```
 If N in A       Then ShowMessage('Число входит в множество')
         Else ShowMessage('Число не входит в множество')
end;
```

Пример №2. Составить программу, которая вырабатывает и выводит на экран дисплея наборы случайных чисел для игры в "Спортлото 5 из 36". Для заполнения каждой карточки спортлото необходимо получить набор из пяти псевдослучайных чисел. К этим числам предъявляются два требования:

1. числа должны находиться в диапазоне 1..36;
2. числа не должны повторяться.

Решение:

```
var
  nb, k: Set of 1..36;
  kol, l, i, n: Integer;
  s:string;
begin
Randomize;
nb:=[1..36];
k:=[];
for l:=1 to 5 do
 begin
   repeat
     n:=Random(36)
   until (n in nb) and not (n in k);
   k:=k+[n];
   s:=s+', '+inttostr(n);
 end;
 memo2.Lines.Add(s);
end;
```

Пример №3: Дана непустая последовательность символов. Построить и напечатать множества, элементами которого являются встречающиеся в последовательности цифры от 0 до 9 и знаки арифметических выражений.

Решение:

```
Var s,s1 : string;
      I : integer;
    A, B : set of Char;          {объявляем переменные A и B типа множество}
Begin
  s:=Edit1.Text;
  A:=['0'..'9','+','-','*','/'];     {заполняем множество}
  B:=[];                                    {пустое множество}
  For i:=1 to length(s) do
       If (s[i] in A) and (not(s[i] in B))
         then
           Begin
               Include(B,s[i]);
            s1:=s1+s[i];
           end;
 ShowMessage(s1);
End;
```

В цикле проверяем является ли s[i] элементом множества A (т.е. s[i] цифра или знак ’+’,’-‘,’*’,’/’). Если условие выполняется, то определяем содержится ли s[i] во множестве B. Операторы Include(B,s[i]); s1:=s1+s[i] будут выполняться в том случае, когда s[i] не окажется во множестве B.

Пример №4. Заданы два слова. Определить буквы, которые не являются общими для обоих слов.

Решение: образуем множества, содержащие буквы первого и второго слова. Затем найдем разности первого и второго, второго и первого множеств. Их объединение даст ответ.

```
procedure TForm1.Button4Click(Sender: TObject);
type setchar = set of char;
```

```
const alf: string = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'; //строка - русский алфавит
var s1,s2:string; //заданные слова
i:integer;
ms1:setchar;
ms2:setchar;
g1,g2:setchar;

procedure print(mn:setchar);
var i:integer;
begin
 for i:=1 to length(alf) do
  if alf[i] in mn then memo1.Lines.Add(alf[i]);
end;

begin
 s1:=edit2.Text;
 s2:=edit3.text;
 ms1:=[];
 ms2:=[];
 for i:=1 to length(s1) do ms1:=ms1+[s1[i]];
 for i:=1 to length(s2) do ms2:=ms2+[s2[i]];
 g1:=ms1-ms2;
 g2:=ms2-ms1;
 print(g1+g2);
end;
```

УПРАЖНЕНИЯ

Вычислите значения выражений:

1. [2..3,4,6]=[2..4,6];
2. [1,2]<>[2,1];
3. [‘b’..’a’]=[‘c’..’b’];
4. [1..15,13..18]<>[1..18];
5. [1]<>[1,1,1];
6. [1]<[2];
7. [5..7]>=[8,6,7,5];
8. [5..7]<=[8,6,7,5];
9. [1,3]*[2,4];
10.[‘a’,’b’]+[‘c’];
11.[3]-[1];
12.[5,4]*[]+[4,3,2]-[3..5];
13.[]*[5,4]+[4,3,2]-[3..5];
14.[5,4]*[4,3,2]+[]-[3..5];
15.[5,4]*[4,3,2]+[3..5]-[];
16.[5,4]*[3..5]+[]-[4,3,2];
17.[3..5]*[4,5]+[4,3,2]-[3,5];
18.[1] in [[1],[2],[3]];

Вопросы:

1. Что такое множество?
2. Элементы каких типов может содержать множество?
3. Может ли существовать множество, не содержащее элементов?
4. Имеются ли ограничения на количество элементов в множестве?
5. Какие операции определены для множеств?
6. Какие операции сравнения определены для множеств?
7. Каким образом можно задать начальное значение для множества?
8. Каким образом можно включить элемент в множество?
9. Каким образом можно исключить элемент из множества?

УРОК №24. КОМБИНИРОВАННЫЙ ТИП ДАННЫХ - ЗАПИСЬ

Цель:

Образовательная: Создать условия для усвоения типа данных - запись. Рассмотреть примеры, использующие тип данных запись.

Воспитательная: Содействовать воспитанию организованности, дисциплинированности; воспитанию любви к профессии, воспитанию навыков коллективного труда

Развивающая: Содействовать развитию познавательных способностей

Тип занятия: урок изучения нового материала

Форма организации учебного процесса: урок-лекция

Наглядные пособия и ТСО: Учебный материал в электронном виде, карточки входного контроля

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Входной контроль.** Актуализация темы «Множественный тип данных».
3. **Изучение нового материала.**
4. **Рассмотрение примера.**
5. **Выполнение заданий по уровням.**
6. **Домашнее задание:** повторить теоретический материал
7. **Подведение итогов урока.**

КАРТОЧКИ ДЛЯ ВХОДНОГО КОНТРОЛЯ

Вариант №1. Тема «Множество» 1. Что такое множество? 2. Вычислите значение выражений: **a.** [2..3,4,6]=[2..4,6]; **b.** [3..5]*[4,5]+[4,3,2]-[3,5];	**Вариант №2. Тема «Множество»** 1. Элементы каких типов может содержать множество? 2. Вычислите значение выражений: **a.** [‘b’..’a’]=[‘c’..’b’]; **b.** [5,4]*[4,3,2]+[]-[3..5];
Вариант №3. Тема «Множество» 1. Может ли существовать множество, не содержащее элементов? 2. Вычислите значение выражений: **a.** [1]<>[1,1,1]; **b.** [1]<[2];	**Вариант №4. Тема «Множество»** 1. Имеются ли ограничения на количество элементов в множестве? 2. Вычислите значение выражений: **a.** [5..7]>=[8,6,7,5]; **b.** [5,4]*[]+[4,3,2]-[3..5];
Вариант №5. Тема «Множество» 1. Какие операции определены для множеств? 2. Вычислите значение выражений: **a.** [5..7]<=[8,6,7,5]; **b.** [5,4]*[3..5]+[]-[4,3,2];	**Вариант №6. Тема «Множество»** 1. Какие операции сравнения определены для множеств? 2. Вычислите значение выражений: **a.** [1,3]*[2,4]; **b.** []*[5,4]+[4,3,2]-[3..5];
Вариант №7. Тема «Множество» 1. Каким образом можно задать начальное значение для множества? 2. Вычислите значение выражений: **a.** [‘a’,’b’]+[‘c’]; **b.** [1..15,13..18]<>[1..18];	**Вариант №8. Тема «Множество»** 1. Каким образом можно включить элемент в множество? 2. Вычислите значение выражений: **a.** [3]-[1]; **b.** [1] in [[1],[2],[3]];
Вариант №9. Тема «Множество» 1. Каким образом можно исключить элемент из множества? 2. Вычислите значение выражений: **a.** [5,4]*[4,3,2]+[3..5]-[]; **b.** [1,2]<>[2,1];	**Вариант №10. Тема «Множество»** 1. Каким образом можно исключить элемент из множества? 2. Вычислите значение выражений: **a.** [5,4]*[4,3,2]+[3..5]-[]; **b.** [1,2]<>[2,1];
Вариант №11. Тема «Множество» 1. Что такое множество? 2. Вычислите значение выражений: **a.** [2..3,4,6]=[2..4,6]; **b.** [3..5]*[4,5]+[4,3,2]-[3,5];	**Вариант №12. Тема «Множество»** 1. Элементы каких типов может содержать множество? 2. Вычислите значение выражений: **a.** [‘b’..’a’]=[‘c’..’b’]; **b.** [5,4]*[4,3,2]+[]-[3..5];
Вариант №13. Тема «Множество» 1. Может ли существовать множество, не содержащее элементов? 2. Вычислите значение выражений: **a.** [1]<>[1,1,1]; **b.** [1]<[2];	**Вариант №14. Тема «Множество»** 1. Имеются ли ограничения на количество элементов в множестве? 2. Вычислите значение выражений: **a.** [5..7]>=[8,6,7,5]; **b.** [5,4]*[]+[4,3,2]-[3..5];
Вариант №15. Тема «Множество» 1. Какие операции определены для множеств? 2. Вычислите значение выражений: **a.** [5..7]<=[8,6,7,5]; **b.** [5,4]*[3..5]+[]-[4,3,2];	**Вариант №16. Тема «Множество»** 1. Какие операции сравнения определены для множеств? 2. Вычислите значение выражений: **a.** [1,3]*[2,4]; **b.** []*[5,4]+[4,3,2]-[3..5];
Вариант №17. Тема «Множество» 1. Каким образом можно задать начальное значение для множества? 2. Вычислите значение выражений: **a.** [‘a’,’b’]+[‘c’]; **b.** [1..15,13..18]<>[1..18];	**Вариант №18. Тема «Множество»** 1. Каким образом можно включить элемент в множество? 2. Вычислите значение выражений: **a.** [3]-[1]; **b.** [1] in [[1],[2],[3]];

УЧЕБНЫЙ МАТЕРИАЛ

В то время как массив – объединение компонент одинакового типа, запись – объединение компонент различного типа.

Запись – это структура, предназначенная для хранения в оперативной памяти компьютера сложных данных, состоящих из отдельных компонент различных типов.

Общий вид записи: Запись записывают в разделе **type** с помощью такой конструкции:

```
<имя записи>=record
        <имя поля_1>:<тип поля_1>;
        …
        <имя поля_n>:<тип поля_n>;
end;
```

Сравните массивы и записи:

МАССИВ:

1. Описывается: Var a: array [1..10] of integer;
2. Приведенный выше массив состоит из 10 ячеек одинакового типа
3. К элементу массива обращаются так: a[i]

ЗАПИСЬ:

Описывается:

Информацию об анкетных данных студентов можно представить с помощью таких полей: фамилия, имя, дата рождения и средний балл. Данную структуру опишем как тип записи Students.

```
Type Student  =  record
        Surname      : string;
        Name         : string;
        Age          : 18..65;
        Number       : integer;
      End;
Var   Group1: array [1..27] of Student; //массив записей
      Petrov: Student;
```

Приведенный в примере массив Group типа Student состоит из 27*4 ячеек разных типов

Доступ к полю конкретной записи обеспечивает составное имя.

<имя записи>.<имя поля>

Пример №1:

```
Petrov.Surname:='Петров';
Petrov.Name:='Владимир';
Petrov.Age:=19;
Petrov.Number:=152;
```

Составными именами пользоваться неудобно. Они ведут к громоздким выражениям. Для их упрощения служит команда присоединения **with**.

Команда присоединения (with) дает возможность записывать в программе только имена полей. Общий вид команды with:

with <имя переменной типа запись> do <команда>

В команде используют только имена полей соответствующей записи. К переменным из предыдущего примера можно обратиться так:

```
With Petrov do
Begin
        Name:='Владимир';
        Surname:='Петров';
        Age:=19;
        Number:=152;
End;
```

Пример №2: Создать массив автовладельцев. Для каждого автовладельца известны номер, марка автомобиля, фамилия и адрес. Нужно подсчитать количество владельцев автомобилей определенной марки и вывести все сведения на экран.

Решение: Сведения об автовладельцах представим массивом записей. Исходные данные вводятся с клавиатуры. Работа с массивом записей аналогична работе с одномерным массивом.

```
const nn=100;
Type mash = record
          nomer:integer;
          marka : string[20];
          fio:string[40];
          adres:string[60];
        end;
var v:array[1..nn] of mash; k, n, i:integer; s:string;
begin
 n:=StrToInt(Edit1.Text);
 for i:=1 to n do
  begin
   v[i].nomer:=i;
   v[i].fio:=InputBox('ФИО автовладельца' , 'Введите ФИО' , 'Петров');
   v[i].marka:=InputBox('Марка автомобиля' , 'Введите марку' , 'Mersedes' );
   v[i].adres:=InputBox('Адрес автовладельца' , 'Введите адрес' , 'Калинина 35-152');
  end;

  s:=InputBox('???'  ,  'Какая марка вас интересует?' , 'Mersedes');
  k:=0;
  for i:=1 to n do
   if v[i].marka = s  then
     begin
       with v[i] do Memo1.Lines.Add(inttostr(nomer)+': '+marka+' - '+adres);
       k:=k+1;
     end;
  Memo1.Lines.Add(inttostr(k));
end;
```

ПО РЕЗУЛЬТАТАМ ВХОДНОГО КОНТРОЛЯ:

УРОВЕНЬ №1	**УРОВЕНЬ №2**
На оценку 4	**На оценку 5**
Измените пример таким образом, чтобы он выводил, по вашему запросу, информацию об автомобилях указанного вами автовладельца.	Известны ФИО абонента, дата выдачи книги, автор, издательство, название, год издания и цена книги. Составить программу, позволяющую внести данные и выяснить сколько абонентов пользовалось данной книгой.
На оценку 5	**На дополнительную оценку**
Известны ФИО сотрудника, его адрес (улица, номер дома, номер квартиры), телефон, пол и возраст. Составить программу, позволяющую внести данные и определить, сколько сотрудников мужского пола проживают по заданной улице.	Известны ФИО студента и оценки по четырем предметам (матем., информатика, программирование, философия). Определить сколько студентов имеют неудовлетв. оценку хотя бы по одному предмету.

Вопросы:

1. Чем отличается тип данных запись от типа данных массив?
2. Что называется полем записи? Как к нему обратится?
3. Могут ли массивы содержать элементы типа запись? Как можно обращаться к полям в этом случае?
4. Верно ли, что все поля записи должны быть разных типов?

УРОК №25. ФАЙЛОВЫЙ ТИП ДАННЫХ. ТЕКСТОВЫЕ ФАЙЛЫ

Цель:

Образовательная: Создать условия для усвоения файлового типа данных. Рассмотреть принципы работы с текстовыми файлами и основными процедурами и функциями обработки текстовых файлов.

Воспитательная: Содействовать воспитанию организованности, дисциплинированности; воспитанию любви к профессии, воспитанию навыков коллективного труда

Развивающая: Содействовать развитию познавательных способностей

Тип занятия: урок изучения нового материала

Форма организации учебного процесса: урок-лекция

Наглядные пособия и ТСО: Электронный учебный материал

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Изучение нового материала.**
3. **Разбор примеров и выполнение на Delphi**
4. **Выполнение заданий.**
5. **Домашнее задание:** повторить изученный материал
6. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Часто в программе требуется считать какую-либо информацию из файла, например какой-либо текст. Или текст, формируемый программой, сохранить в файл, чтобы в последующем с ним работать.

В Delphi для работы с файлами используется специальный файловый тип и для связи с файлом заводится переменная файлового типа (файловая переменная), через которую происходит запись в файл или чтение из файла.

Любой файл имеет три характерные особенности

1. У файла есть имя, что дает возможность программе работать одновременно с несколькими файлами.
2. Файл содержит компоненты одного типа. Типом компонентов может быть любой тип Pascal, кроме файлов. Иными словами, нельзя создать "файл файлов".
3. Длина вновь создаваемого файла никак не оговаривается при его объявлении и ограничивается только емкостью устройств внешней памяти.

Файловая переменная – структура данных, представляющая собой последовательность элементов одного типа.

Переменную файлового типа можно задать одним из трех способов в разделе **Var**:

<имя>*:* **TextFile**	текстовые файлы (**TextFile**)
<имя>: **File of** <тип>	типизированные файлы (**File of** ...)
<имя>: **File**	не типизированные файлы (**File**)

С файлами работают в следующей последовательности:

1. файловую переменную связывают физическим файлом (AssignFile)
2. Файл открывают для чтения из него или записи в него (Rewrite, Reset, Append)
3. Работают с компонентами файла, например, считывают строку (WriteLn, ReadLn и др.)
4. разрывают связь с физическим файлом (CloseFile)

1. Связь файловой переменной с физическим файлом

Файлы становятся доступны программе только после выполнения процедуры AssignFile. Она связывает файловую переменную с именем существующего или вновь создаваемого файла.

AssignFile (<ф.п.>, <имя файла>);

где <ф.п.> - файловая переменная;
<имя файла> - строка, содержащая имя файла и путь доступа к нему.

Например: AssignFile(f, 'C:\1.txt'); - связываем переменную f с файлом 1.txt на диске C.

Если требуемый файл находится в папке с программой, то для того, чтобы не указывать полный путь можно использовать функцию ExpandFileName.

Функция **ExpandFileName(s:string) -** возвращает полный путь к файлу, искать файл, она начинает с папки, в которой расположена сама программа.

Например: AssignFile(f, ExpandFileName('1.txt')); - связываем переменную f с файлом 1.txt, расположенным в папке с программой.

Проверка существования физического файла

Чтобы проверить, существует ли файл, можно использовать стандартную функцию **FileExists**, которая возвращает True, если указанный при обращении этой функции файл существует, и False - если не существует.

Например

```
begin
if FileExists(FileName)      then ..... // Файл существует
                             else ..... // Файл не существует
end;
```

2. Открытие файла для чтения, записи или добавления

Для записи: Rewrite (<ф.п.>);

Процедура Rewrite открывает файл для записи в него данных. При этом старый файл (если он был) уничтожается и создается новый файл.

Для добавления (только с текстовыми файлами): **Append (<ф.п.>)**

Процедура Append открывает файл для добавления в него данных. Старый файл не уничтожается.

Для чтения: Reset (<ф.п.>);

Процедура Reset открывает файл для чтения из него данных.

3. Работа с компонентами файла

Запись строки в файл:

Write(<ф.п.>, s1 [, s2,..., sn]) - записывает значения строковых переменных s1, s2, .. sn в текстовый файл в виде одной строки

WriteLn (<ф.п.>, [s1 [, s2, ..., sn]) - записывает значения строковых переменных s1, s2, .. sn в текстовый файл в виде одной строки, в конце которой устанавливает знак абзаца (Enter)

При создании текстового файла в конце каждой строки ставится специальный признак **EOLN** (End Of LiNe - конец строки), а в конце всего файла - признак **EOF** (End Of File - конец файла). Эти признаки можно протестировать одноименными логическими функциями.

Чтение строки из файла:

ReadLn(<ф.п.>, s) - читает из текстового файла строку и присваивает ее значение строковой переменной s.

Доступ к каждой строке возможен лишь последовательно, начиная с первой.

EOF(<ф.п.>) - возвращает True когда достигнут конец файла.

4. Закрываем файл

CloseFile(F) - разрывает связь файловой переменной F с физическим файлом.

Пример №1. Программа, которая создает файл 1.txt в папке с программой и заносит в него 3 строки.

```
Var f:textFile; i:integer;
```

```
Begin
        Assignfile(f, ExpandFileName('1.txt'));
        Rewrite(f);
        Writeln(f, 'строка 1');
        Writeln(f, 'строка 2');
        Writeln(f, 'строка 3');
        CloseFile(f);
End;
```

Пример №2. программа, которая считывает из файла 1.txt все строки и выводит их в компонент memo1.

```
Var f:textFile; i:integer;s:string;
Begin
        Assignfile(f, ExpandFileName('1.txt'));
        Reset(f);
        While not EOF(f) do
        begin
                Readln(f,s);
                Memo1.lines.add(s);
        End;
        CloseFile(f);
End;
```

ЗАДАНИЕ

1. **(на оценку 4)** Измените пример №1, таким образом, чтобы в файл выводились следующие строки:
   Begin
   End
   For
   If
   While
   Repeat
   Readln
   Writeln
   Var
2. **(на оценку 5)** Составить программу, которая управляет файлом 1.txt:
   a. Создает файл
   b. Считывает из файла всю информацию в Memo
   c. Добавляет в файл текст, введенный пользователем, в компонент Edit.

ДОПОЛНИТЕЛЬНЫЙ МАТЕРИАЛ

Пример. В сочетании с проверкой конца файла функцией **eof** процедура **Read** позволяет организовать простой ввод массивов данных, например, так:

```
const
N = 1000; // Максимальная длина ввода
Var F : TextFile; M : array [1..N] of Real;
        i : Integer;
begin
        AssignFile(F, 'prog.dat');
        Reset(F);
        i := 1;
        while not EOF(f) and (i <= N) do
        begin
                Read(F, M[i]);
            inc (i)
        end;
        CloseFile(F) ;
End;
```

УРОК №26. РЕШЕНИЕ ЗАДАЧ ПО ТЕМЕ «ТЕКСТОВЫЕ ФАЙЛЫ»

Цель:

Образовательная: Создать условия для усвоения файлового типа данных. Создать условия для успешного составления программ, работающих с текстовыми файлами.

Воспитательная: Содействовать воспитанию организованности, дисциплинированности; воспитанию любви к профессии.

Развивающая: Содействовать развитию познавательных способностей

Тип занятия: урок закрепления новых знаний

Форма организации учебного процесса: Урок практических работ

Наглядные пособия и ТСО: Учебный материал по теме «Файловый тип данных»

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Входной контроль.**
3. **Решение задач**
4. **Подведение итогов урока.**

УЧЕБНЫЕ ЗАДАНИЯ

УРОВЕНЬ 1. (оценка 3)

1. Дан файл F. Переписать все строки файла F в файл G (новый).

Решение:

```
var F,G:TextFile;S:String;
begin
AssignFile(F, ExpandFileName('f.txt'));
Reset(F);
AssignFile(G, ExpandFileName('g.txt'));
Rewrite(G);
While not EOF(F) do
 begin
  Readln(F,S);
  Writeln(G,S);
 end;
CloseFile(F);
CloseFile(G);
end;
```

УРОВЕНЬ 2. (оценка 4)

2. Программа читает свой собственный текст и выводит количество строк в нем.

Решение:

```
var F:TextFile;S:String;i:integer;
begin
AssignFile(F, ExpandFileName('Unit1.pas'));
Reset(F);
i:=0;
While not EOF(F) do
 begin
  Readln(F,S);
  inc(i);
 end;
```

```
CloseFile(F);
memo1.Lines.Add(inttostr(i));
end;
```

3. Программа читает свой собственный текст и выводит количество ключевых слов Begin в ней.
   Решение:

```
var F:TextFile;S:String;i:integer;
begin
AssignFile(F, ExpandFileName('Unit1.pas'));
Reset(F);
i:=0;
While not EOF(F) do
 begin
  Readln(F,S);
  if pos('begin',s)<>0 then inc(i);
 end;
CloseFile(F);
memo1.Lines.Add(inttostr(i));
end;
```

УРОВЕНЬ 3 (оценка 5)

4. Создайте при помощи блокнота два текстовых файла: A.txt и B.txt. Составьте программу, которая меняет местами их содержимое.
5. В файле удалить строки-дубликаты, т.е. те строки, которые повторяются.

КАРТОЧКИ ДЛЯ ВХОДНОГО КОНТРОЛЯ

Вариант №1. Тема «Текстовые файлы» 1. Какая процедура связывает файловую переменную с физическим файлом? Укажите параметры процедуры. 2. Какая процедура открывает файл для чтения? Укажите параметры процедуры. 3. Какая процедура записывает строку s в файл f?	**Вариант №2. Тема «Текстовые файлы»** 1. Каким образом можно проверить физическое существование файла 1.txt? 2. Какая процедура открывает текстовый файл для добавления в него информации? 3. Какая процедура считывает из текстового файла f строку S?
Вариант №3. Тема «Текстовые файлы» 1. Что делает функция ExpandFileName('1.txt')? 2. Какая процедура открывает файл для записи в него данных (каждый раз она создает новый файл)? 3. Какая процедура разрывает связь физического файла с файловой переменной?	**Вариант №4. Тема «Текстовые файлы»** 1. Что делает функция **EOF**(<ф.п.>)? 2. Какая процедура открывает файл для чтения? Укажите параметры процедуры. 3. Каким образом можно проверить физическое существование файла 1.txt?
Вариант №5. Тема «Текстовые файлы» 1. Что делает процедура **ReadLn**(f, s) 2. Какая процедура разрывает связь физического файла с файловой переменной? 3. Какая процедура открывает текстовый файл для добавления в него информации?	**Вариант №6. Тема «Текстовые файлы»** 1. Какая процедура открывает текстовый файл для добавления в него информации? 2. Что делает процедура **WriteLn**(f, s)? 3. Какая процедура связывает файловую переменную с физическим файлом? Укажите параметры процедуры.
Вариант №7. Тема «Текстовые файлы» 1. Какая процедура связывает файловую переменную с физическим файлом? Укажите параметры процедуры. 2. Какая процедура открывает текстовый файл для добавления в него информации? 3. Какая процедура разрывает связь физического файла с файловой переменной?	**Вариант №8. Тема «Текстовые файлы»** 1. Что делает функция **EOF**(<ф.п.>)? 2. Какая процедура разрывает связь физического файла с файловой переменной? 3. Какая процедура связывает файловую переменную с физическим файлом? Укажите параметры процедуры.
Вариант №9. Тема «Текстовые файлы» 1. Какая процедура записывает строку s в файл f? 2. Какая процедура открывает текстовый файл для добавления в него информации? 3. Что делает функция ExpandFileName('1.txt')?	**Вариант №10. Тема «Текстовые файлы»** 1. Каким образом можно проверить физическое существование файла 1.txt? 2. Какая процедура разрывает связь физического файла с файловой переменной? 3. Какая процедура открывает текстовый файл для добавления в него информации?
Вариант №11. Тема «Текстовые файлы» 1. Что делает функция ExpandFileName('1.txt')? 2. Каким образом можно проверить физическое существование файла 1.txt? 3. Какая процедура связывает файловую переменную с физическим файлом? Укажите параметры процедуры.	**Вариант №12. Тема «Текстовые файлы»** 1. Какая процедура открывает текстовый файл для добавления в него информации? 2. Какая процедура разрывает связь физического файла с файловой переменной? 3. Что делает функция **EOF**(<ф.п.>)?
Вариант №13. Тема «Текстовые файлы» 1. Какая процедура открывает файл для записи в него данных (каждый раз она создает новый файл)? 2. Какая процедура считывает из текстового файла f строку S? 3. Какая процедура записывает строку s в файл f?	**Вариант №14. Тема «Текстовые файлы»** 1. Каким образом можно проверить физическое существование файла 1.txt? 2. Какая процедура открывает файл для чтения? Укажите параметры процедуры. 3. Какая процедура открывает файл для чтения? Укажите параметры процедуры.

УРОК №27. ТИПИЗИРОВАННЫЕ И НЕ ТИПИЗИРОВАННЫЕ ФАЙЛЫ

Цель:

Образовательная: Создать условия для усвоения типизированных и не типизированных файлов. Рассмотреть принципы работы с файлами и основными процедурами и функциями обработки типизированных и не типизированных файлов.

Воспитательная: Содействовать воспитанию организованности, дисциплинированности; воспитанию любви к профессии, воспитанию навыков коллективного труда

Развивающая: Содействовать развитию познавательных способностей

Тип занятия: урок изучения нового материала

Форма организации учебного процесса: урок-лекция

Наглядные пособия и ТСО: Учебный материал в электронном виде

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Изучение нового материала.**
3. **Разбор примеров и выполнение их на Delphi**
4. **Домашнее задание:**
5. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Длина любого компонента типизированного файла строго постоянна, что дает возможность организовать прямой доступ к каждому из них (т. е. доступ к компоненту по его порядковому номеру).

Перед первым обращением к процедурам ввода-вывода указатель файла стоит в его начале и указывает на первый компонент с номером 0. После каждого чтения или записи указатель сдвигается к следующему компоненту файла. Переменные в списках ввода-вывода должны иметь тот же тип, что и компоненты файла. Если этих переменных в списке несколько, указатель будет смещаться после каждой операции обмена данными между переменными и дисковым файлом.

Подпрограммы для работы с типизированными файлами

FilePos(F)	Возвращает текущую позицию в файле, т.е. номер компонента, который будет обрабатываться следующей операцией ввода-вывода
FileSize (F)	Возвращает количество компонентов файла. Чтобы переместить указатель в конец типизированного файла, можно написать: seek (FileVar, FileSize(FileVar));
Seek(F; N: Longint);	Смещает указатель файла F к требуемому компоненту: n - номер компонента файла (первый компонент файла имеет номер 0)
Read(F, VI,..., Vn) ;	Читает данные из типизированного файла f. Vi - переменные такого же типа, что и компоненты файла
Write(F,P1, ...,Pn)	Записывает данные в типизированный файл F. Pi - выражения такого же типа, что и компоненты файла

Пример: Составить программу, которая создает файл типа запись, в котором будет храниться информация о студенте: его фамилия и оценки по двум предметам. А также выводит информацию из файла на экран.

Решение:

```
Type Students = record
       fio : string[255];
       otm1, otm2 : integer;
     end;
var
  Form1: TForm1;
  Student : array [1..5] of Students;
  f : file of Students;
  i: integer;
implementation

{$R *.dfm}

//ввод данных
procedure TForm1.Button1Click(Sender: TObject);
begin
  for i:=1 to 5 do
   begin
    with Student[i] do
     begin
      fio:=InputBox('Ввод данных','Введите фамилию','Петров');
      otm1:=StrToInt(InputBox('Ввод данных','Введите отметку по 1 предмету','5'));
      otm2:=StrToInt(InputBox('Ввод данных','Введите отметку по 2 предмету','4'));
     end;
   end;
end;

//запись в файл
procedure TForm1.Button2Click(Sender: TObject);
begin
 AssignFile(f,'1.dat');
 Rewrite(f);
 for i:=1 to 5 do Write(f, student[i]);
end;

//чтение из файла
procedure TForm1.Button3Click(Sender: TObject);
begin
 AssignFile(f,'1.dat');
 Reset(f);
 for i:=1 to 5 do
 begin
  read(f,student[i]);
  StringGrid1.Cells[1,i]:=student[i].fio;
  StringGrid1.Cells[2,i]:=IntToStr(student[i].otm1);
  StringGrid1.Cells[3,i]:=IntToStr(student[i].otm2);
 end;
end;
```

Задание: Сведения о студенте состоят из его имени, фамилии, названия группы и сведений об отметках, полученных в последнем семестре. Сформировать файл, содержащий сведения о студентах колледжа. Выяснить, сколько учеников школы не имеют отметок ниже четырех.

Нетипизированные файлы объявляются как файловые переменные типа File и отличаются тем, что для них не указан тип компонентов. Отсутствие типа делает эти файлы, с одной стороны, совместимыми с любыми другими файлами, а с другой - позволяет организовать высокоскоростной обмен данными между диском и памятью.

При инициации нетипизированного файла процедурами Reset или Rewrite можно указать длину записи нетипизированного файла в байтах. Например, так:

```
Var  F: File;
begin
     AssignFile(F,'myfile.dat');
     Reset(f,512);
End;
```

Длина записи нетипизированного файла указывается вторым параметром при обращении к процедурам Reset или Rewrite, в качестве которого может использоваться выражение типа *Longint.* Если длина записи не указана, она принимается равной 128 байтам.

Object Pascal не накладывает каких-либо ограничений на длину записи нетипизированного файла за исключением требования положительности и ограничения максимальной длины 2 Гбайт. Для обеспечения максимальной скорости обмена данными рекомендуется задавать длину, которая была бы кратна длине физического сектора дискового носителя информации (512 байт). Однако операции обмена данными с дисковыми устройствами в среде Windows кэшируются, т. е. осуществляются через промежуточный буфер памяти, поэтому обычно задают Recsize = 1, что позволяет обмениваться с файлом блоками любой длины, начиная с одного байта.

При работе с нетипизированными файлами могут применяться все процедуры и функции, доступные типизированным файлам, за исключением Read и write, которые заменяются соответственно высокоскоростными Процедурами BlockRead И BlockWrite:

Procedure **BlockRead**(F: File; Buf; Count: Integer [; AmtTransferred: Integer]) ;
Procedure **BlockWrite**(F: File; Buf; Count: Integer [; AmtTransferred: Integer]);

Здесь Buf - буфер: имя переменной, которая будет участвовать в обмене данными с дисками; count - количество записей, которые должны быть прочитаны или записаны за одно обращение к диску;

AmtTransferred - необязательный параметр, содержащий при выходе из процедуры количество фактически обработанных записей.

За одно обращение к процедурам может быть передано до count*RecSize байт, где RecSize - длина записи нетипизированного файла. Передача идет, начиная с первого байта переменной Buf. Программист должен позаботиться о том, чтобы длина внутреннего представления переменной Buf была достаточной для размещения всех count*RecSize байт при чтении информации с диска. Если при чтении указана переменная недостаточной длины или если в процессе записи на диск не окажется нужного свободного пространства, возникнет ошибка ввода-вывода, которую можно заблокировать, указав необязательный параметр AmtTransferred.

После завершения процедуры указатель смещается на count записей. Процедурами Seek, FilePos И FileSize можно обеспечить доступ к любой записи нетипизированного файла.

УРОК №28. РЕШЕНИЕ ЗАДАЧ ПО ТЕМЕ «ТИПИЗИРОВАННЫЕ ФАЙЛЫ»

Цель:

Образовательная: Создать условия для усвоения типизированных файлов. Создать условия для успешного составления программ, работающих с типизированными файлами.

Воспитательная: Содействовать воспитанию организованности, дисциплинированности; воспитанию любви к профессии, воспитанию навыков коллективного труда

Развивающая: Содействовать развитию познавательных способностей

Тип занятия: урок закрепления новых знаний

Форма организации учебного процесса: Урок практических работ

Наглядные пособия и ТСО: Учебный материал в электронном виде по теме «Типизированные файлы»

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Входной контроль.**
3. **Решение задач.**
4. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

УРОК №29. МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ

Цель:

Образовательная:

Воспитательная: воспитание организованности, дисциплинированности; воспитание любви к профессии;

Развивающая: развитие познавательных способностей

Тип занятия: урок изучения нового материала

Форма организации учебного процесса: урок-лекция

Наглядные пособия и ТСО:

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Изучение нового материала.**
3. **Изучение нового материала.**
4. **Домашнее задание:**
5. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Разберем структуру стандартного проекта:

Файлы, составляющие основу любого проекта, можно разбить на три группы:

- формы (.dfm) - двоичные файлы, в которых содержаться визуальные составляющие проекта;
- модули (.pas) - это текстовые файлы, в которых храниться программный код;
- файлы ресурсов (.res) – могут содержать текстовую и графическую информацию, используемую в проекте.

В проекте стандартного приложения главным является файл, начинающийся служебным словом program и имеющий расширение .dpr. Этот файл создается автоматически и содержит команды, инициализирующие и запускающие процесс выполнения программы. Файл проекта очень редко требует пользовательского редактирования и поэтому явно в окне редактора кода не отображается. Чтобы его увидеть, необходимо выполнить команду Project – View Source (Проект – просмотреть ресурсы). После этого откроется окно, содержащее примерно следующий программный код:

```
program Project1;                          {название программы}
uses                                       {подключение необходимых модулей}
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;                  {инициализация программы}
  Application.CreateForm(TForm1, Form1);   {создание формы}
  Application.Run;                         {запуск программы}
end.
```

Проект состоит из набора модулей и форм. Каждая форма обязательно имеет соответствующий модуль, где описываются процедуры обработки событий для самой формы и для всех объектов, которые она содержит. Однако модули могут использоваться и без формы.

Каждый модуль является текстовым файлом, который начинается служебным словом unit и в общем случае имеет следующую жестко заданную структуру:

```
unit Unit1;        {имя модуля – должен совпадать с именем файла модуля}

interface
uses               {список используемых модулей}

{описание интерфейсной части модуля}

implementation
uses               {список используемых модулей}

{реализация подпрограмм, описанных в интерфейсной части}

Initialization     {необязательная часть инициализации модуля, которая выполняется один раз при подключении модуля}
Finalization       {Необязательная часть завершения модуля, которая выполняется при любом завершении работы}
end.               {конец модуля}
```

В интерфейсной части могут размещаться списки подключаемых модулей, объявления типов, констант, переменных, функций и процедур, к которым необходимо осуществлять доступ из других модулей.	**В разделе реализации модуля** размещаются списки подключаемых модулей, объявления типов, констант переменных, к которым не нужен доступ из других модуле, а также выполняется реализация всех объявленных в интерфейсной части функций и процедур. Здесь же выполняется реализация всех дополнительных, не объявленных ранее функций и процедур.
Подобное разделение модуля на части позволяет создавать и распространять модули в виде откомпилированного файла с расширением .dcu, в котором видно только описание интерфейсной части. Внести изменения в такой модуль нельзя, т.к. исходный код, реализующий описанные в интерфейсной части возможности, недоступен. Это позволяет повторно использовать модули, ранее написанные для других программ, и предоставить доступ к данному модулю нескольким программистам.	

Инициирующая и завершающая части

Инициирующая и завершающая части чаще всего отсутствуют, вместе с начинающим их словами initialization и finalization.

В инициирующей части размещаются операторы, которые исполняются *до* передачи управления основной программе и обычно используются для подготовки ее работы. Например, в них могут инициироваться переменные, открываться нужные файлы и т. д.

В завершающей части указываются операторы, выполняющиеся *после* завершения работы основной программы (в них освобождаются выделенные программе ресурсы, закрываются файлы и т. д.).

Если несколько модулей содержат инициирующие части, эти части выполняются последовательно друг за другом в порядке перечисления модулей в предложении uses файла проекта. Если несколько модулей содержат завершающие части, эти части выполняются последовательно друг за другом в порядке, обратном перечислению модулей в предложении uses главной программы.

Связывание с модулем

Проект может состоять из набора моделей, часть которых может быть стандартными, а часть пользовательскими. Чтобы в одном модуле использовать типы, переменные, процедуры и функции, объявленные и реализованные в другом модуле, требуется соответствующее объявление. Для доступа к содержимому другого модуля его имя необходимо указать в текущем модуле после служебного слова uses.

Например: Uses Unit2;

Перечисленные после служебного слова Uses модули должны содержаться в одной из папок, которые определены в настройках Delphi.

Возможность подключения разных модулей делает создание программы более гибким. Имея набор модулей, в которых описаны необходимые особенности, можно избежать непродуктивного использования времени на «изобретение велосипеда».

Со средой разработки Delphi поставляется большое количество различных модулей, в которых охватываются всевозможные стандартные ситуации и задачи (например, модуль Math содержащий различные математические функции).

Возможность использования множества модулей является не только достоинством, но и таит в себе некоторые опасности. Одна из них возникает при определении в разных модулях одинаковых идентификаторов, которые могут иметь разную реализацию.

В этом случае при использовании соответствующих идентификаторов перед ними необходимо явно указать имя модуля. Например: Unit1.Identificator и Unit2.Identificator.

Пример: Расчет оплаты

Задание: Предположим, мы хотим проверить правильность оплаты услуг за использование Интернета. Пусть можно оплачивать либо количество «скачанных» мегабайт (трафик), либо затраченное время в минутах, либо то и другое вместе. Известны цены провайдера в долларах (**у.е**.), и необходимо перевести общую сумму в «родную» валюту.

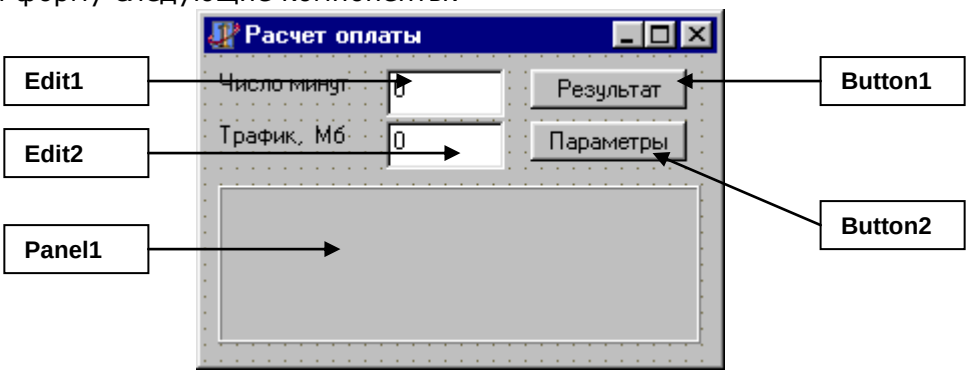
Решение:

Нетрудно сообразить, что общую стоимость в этом случае можно посчитать по следующей формуле:

(Число минут*Стоимость минуты в у.е. +Трафик в мегабайтах*Стоимость 1 мегабайта в у.е.)*Курс у.е.

Составим небольшую программу для подобного расчета.

Поместим на форму следующие компоненты:



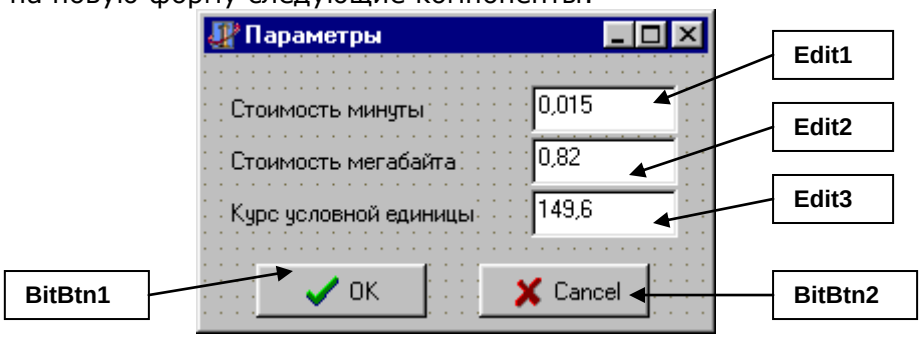
В текст модуля **Unit1** внесите следующий код программы:

```
var
  Form1: TForm1;
  C_time,                  // Цена за 1 минуту
  C_traf,                  // Цена за 1 мегабайт
  C_kurs : double;         // Курс доллара
implementation


{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var Time, Traf, Val : double;
begin
  Time:=StrToFloat(Edit1.Text);
  Traf:=StrToFloat(Edit2.Text);
  Val:=(C_time*Time + C_traf*Traf)*C_kurs;
  Panel1.Caption:=FloatToStr(Val);
end;

initialization
  C_time:=0.015;
  C_traf:=0.82;
  C_Kurs:=149.6;

end.
```

Но что делать, если цены и курс **у.е.** изменятся? Придется снова открывать **Delphi** и изменять текст программы – это неудобно. Лучше, если реализовать возможность изменять цены и курс во время работы программы. Для этого заведем еще одну форму.

Добавление новой формы: File – New - Form

Поместите на новую форму следующие компоненты:



Для того, чтобы вторая форма появлялась на экране, необходимо, вызвать ее следующим образом. В обработчик события нажатия второй кнопки вставьте следующую строку: *Form2.Show;*

Также с помощью раздела описания модулей Uses необходимо связать два модуля: *uses Unit2;*

Теперь по нажатию на кнопку появляется вторая форма, которая пока ничего не делает. Для того, чтобы с ее помощью устанавливать новые параметры, необходимо создать следующие два обработчика кнопок:

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  C_time:=StrToFloat(Edit1.Text);
  C_traf:=StrToFloat(Edit2.Text);
  C_kurs:=StrToFloat(Edit3.Text);
end;

procedure TForm2.Button2Click(Sender: TObject);
begin
form2.Close;
end;
```

Однако, для того, чтобы первая кнопка присваивала значения именно переменным, описанным в первой форме, необходимо связать модуль 2 с модулем 1. Для этого в Uses Unit2 укажите ссылку на Unti1: *Uses Unti1;*

ДОПОЛНИТЕЛЬНЫЙ МАТЕРИАЛ

ТИПЫ МОДУЛЕЙ В DELPHI

Наиболее распространенным типом модуля является **форма** - модуль со связанным с ним окном. Интерфейсная часть такого модуля обычно содержит объявление нового класса и автоматически обновляется в ходе конструирования окна. В интерфейсной части модуля-формы содержится также объявление объекта для соответствующего оконного класса.

Модули данных имеют связанные с ними окна, однако, эти окна никогда не появляются на экране. Необходимость в окнах вызвана тем, что компоненты доступа к данным страницы можно вставить только в форму, хотя все они не имеют видимого воплощения в работающей программе. Невидимое окно модуля данных предназначено для размещения этих компонентов и связанных с ними объектов-полей. Разумеется, для размещения компонентов и полей можно использовать и обычное окно-форму, однако в этом случае пиктограммы компонентов загромождают видимое пространство окна и затрудняют его конструирование.

Модули динамических библиотек предназначены для создания широко используемых в Windows динамически связываемых библиотек DLL (Dynamic-Link Libraries). DLL служат универсальным средством согласования подпрограмм, написанных на разных языках программирования. В Windows содержится множество DLL, написанных на языке Си или на языке ассемблера, что ничуть не мешает Delphi-программам использовать их. Модули динамических библиотек предназначены для разработки DLL с помощью Object Pascal. Такие

DLL затем смогут использовать программы, созданные с помощью других языков программирования.

Пакеты - это особым образом откомпилированные DLL, оптимизированные для совместного использования Delphi-программами, или средой Delphi, или и программами, и средой. В отличие от DLL пакеты могут хранить и передавать программе типы (включая классы) и данные. Они разработаны специально для хранения компонентов, разного рода экспертов, редакторов сложных свойств и т. п. Например, в пакете VCL60.bpl содержатся основные компоненты Delphi.

Модули потоков предназначены для реализации так называемых потоков команд - фрагментов программы, которые исполняются параллельно с другими фрагментами, разделяя с ними время процессора и остальные системные ресурсы. Модуль потока не имеет связанного с ним окна.