УРОК №1. СРЕДА ПРОГРАММИРОВАНИЯ DELPHI.

Цель:

Образовательная: Рассмотреть основные возможности среды программирования Delphi. Изучить главные элементы среды и принципы управления проектом. Рассмотреть основные стандартные компоненты и их свойства. Создание обработчика события

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: элементы среды программирования на отдельных листах (форма, окно кода программы, главное окно, окно Object Inspector)

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Изучение нового материала.** Среда программирования Delphi. Основные характеристики и исторические сведения.
3. **Изучение нового материала.** Среда программирования Delphi. Основные структурные элементы среды. Файлы проекта: создание, сохранение и открытие проекта.
   При объяснении учебного материала по среде программирования, на доску постепенно прикрепляются наглядные пособия в виде элементов среды.
4. **Домашнее задание:** знать ответы на вопросы к лекции №1 и уметь на них отвечать.
   Вопросы к лекции №1:
   1. Назовите и покажите основные элементы среды программирования Delphi.
   2. Расскажите о том, как создаются новые, сохраняются и открываются проекты.
5. **Подведение итогов урока.**

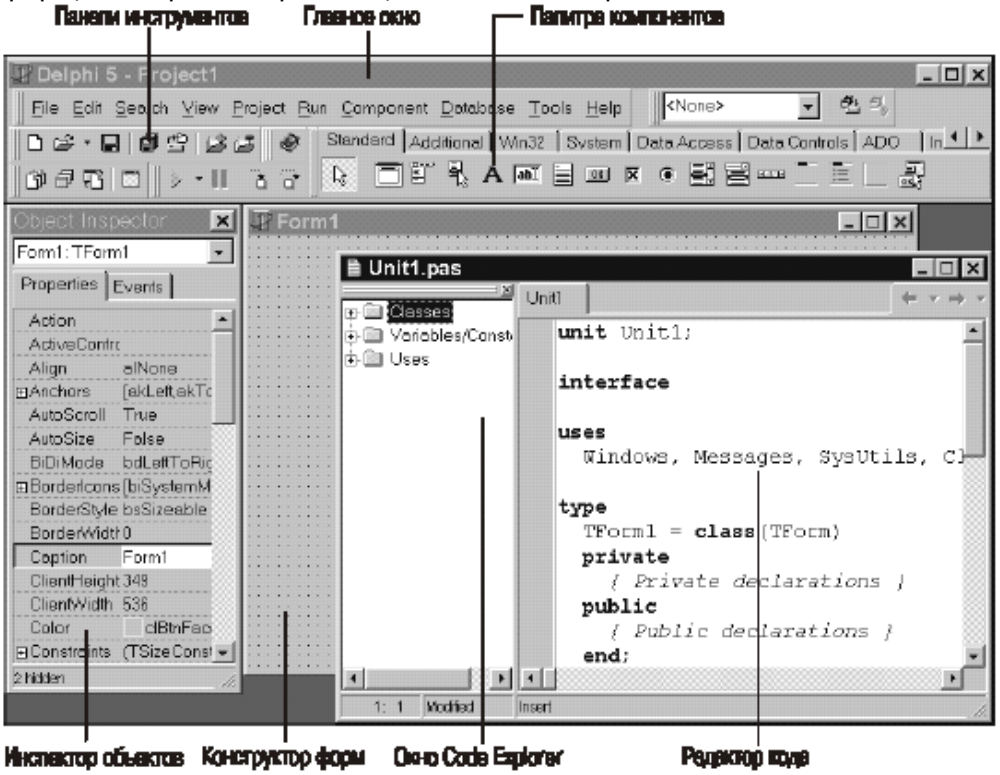
УЧЕБНЫЙ МАТЕРИАЛ

Delphi – один из самых мощных инструментов разработки программных продуктов любой сложности и направленности. Инструментарий этого пакета позволяет создавать полноценные приложения даже тем, кто имеет минимальный опыт работы с Delphi.

Концепция Delphi была реализована в конце 1994 года, когда вышла первая версия среды разработки. В основу этого программного продукта легли концепции объектно-ориентированного программирования на базе языка Object Pascal и визуального подхода к построению приложений.

Delphi представляет собой интегрированную среду разработки, инструменты которой позволяют значительно ускорить процесс разработки, создания и отладки программ – небольшую программу можно «написать», не написав ни одной строки программного кода.

При запуске Delphi на экране появляется четыре ее основных окна: главное окно, окно конструктора форм, окно редактора кода, окно инспектора объектов.



Главное окно состоит из трех частей:

1. *главное меню* осуществляет доступ к любой команде Delphi.
2. *панели инструментов* предоставляют доступ к различным функциям главного меню с помощью щелчка на соответствующей кнопке.
3. *палитра компонентов* представляет собой панель инструментов, содержащую несколько вкладок, в которых находятся все установленные в среде компоненты.

Создание нового приложения производится командой главного меню **File – New Application**.

При создании нового приложения на экране появляются два основных окна, с помощью которых происходит настройка визуального интерфейса программы и обработка данных.

Окно конструктора форм (в дальнейшем ***Форма***) используется для создания интерфейса будущей программы, с его помощью определяется, как будет выглядеть программа с точки зрения пользователя. Форма представляет стандартное окно Windows. Процесс создания интерфейса заключается в установке необходимых компонентов на форму. Delphi располагает большим количеством стандартных компонентов – элементов, необходимых для управления информацией и программой.

Все компоненты собраны в палитре компонентов, расположенной в главном окне программы.

Установка компонентов на форму происходит так:

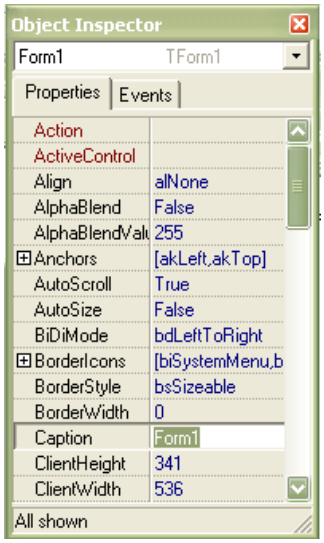
1. необходимо найти требуемый компонент в палитре компонентов (при наведении на пиктограмму компонентов появляется подсказка об его названии);
2. щелкнуть по пиктограмме компонента;

3. щелкнуть в любом месте формы.

Например, на странице Standart располагаются следующие стандартные компоненты:
- **Button** – кнопка, основной элемент управления Windows окна.
- **Label** – метка, предназначена для вывода поясняющего текста к другим компонентам.
- **Edit** – однострочный компонент для ввода или вывода текста.
- **Memo** – многострочные компонент для ввода и вывода текста.

После установки компонента на форму его можно настроить. Настройка свойств компонентов происходит с помощью окна Object Inspector.

Инспектор объектов Object Inspector вызывается клавишей ***F11***



Окно инспектора объектов содержит две страницы:
– **Properties** (свойства) служит для установки нужных свойств компонента;
– **Events** (события) позволяет определить реакцию компонента на то или иное событие.

Совокупность свойств отображает видимую сторону компонента: положение относительно левого верхнего угла рабочей области формы, его размеры и цвет, шрифт и текст надписи на нем и т.д.

Совокупность событий – его поведенческую сторону: будет ли компонент реагировать на щелчок мыши или на нажатие клавиши, как он будет вести себя в момент появления на экране или в момент закрытия окна и т.д.

Каждая страница окна имеет двух колончатую таблицу, левая колонка содержит названия свойства или события, а правая – конкретное значение свойства или имя подпрограммы, обрабатывающей соответствующее событие. Строки таблицы выбираются щелчком мыши и могут отображать простые или сложные свойства.

Простые – это свойства, определяемые одним значением. Например, **caption** (заголовок) - текст, **height** (высота), **width** (ширина)- числа, **Enabled** (доступность) - значение **true** или **false**.

Сложные свойства определяются совокупностью значений. Слева от имени таких свойств указывается значок (+), а двойной щелчок мышью на имени свойства (в левой колонке таблицы) приводит к раскрытию списка составляющих сложного свойства.

Чтобы закрыть раскрытый список, нужно вновь дважды щелкнуть на имени сложного свойства (свойство **Font**).

В верхней части окна **Инспектора объектов** располагается раскрывающийся список всех помещенных на форму компонентов. Поскольку форма сама по себе является компонентом, ее имя также присутствует в этом списке.

Например, рассмотренные стандартные компоненты имеют следующие основные свойства, которые можно настроить с помощью объектного инспектора:
- **Font** – сложное свойство для настройки шрифта отображаемого на компоненте текста.
- **Width** – простое свойство, указывающее на ширину компонента.
- **Height** – простое свойство, указывающее на ширину компонента.

Для отображения текста на компоненте используются следующие свойства:
- **Text** у компонента **Edit**
- **Caption** у компонентов **Button** и **Label**
- **Lines** у компонента **Memo**

С помощью объектного инспектора можно настраивать не только свойства компонентов, но и создавать обработчики событий, возникающих в результате воздействия на компонент пользователя. Например, для того, чтобы определить, как будет реагировать компонент Button на щелчок мышью, необходимо создать обработчик события **onClick**.

Создание обработчика события происходит так:
1. перейти в окне объектного инспектора на вкладку Events;
2. найти требуемое событие в левой колонке;

3. дважды щелкнуть в правой колонке, напротив найденного события;
4. в результате откроется окно редактора кода с заготовкой обработчика события, в котором нужно внести код программы.

Окно редактора кода программы (окно Code Editor) предназначено для ввода команд, определяющих поведение программы. Окно Code Editor использует технологию вкладок, причем отдельная вкладка создается для каждого модуля или файла. При каждом добавлении в приложение новой формы создается новый модуль, а в окно Code Editor добавляется соответствующая вкладка. Контекстное меню окна Code Editor предоставляет широкий диапазон команд редактирования, включая команды работы с файлами, создания закладок и поиска символов.

Первоначально окно кода содержит минимальный исходный текст, обеспечивающий нормальное функционирование пустой формы в качестве полноценного Windows-окна. В ходе работы над проектом программист вносит в него необходимые дополнения, чтобы придать программе нужную функциональность.

Автоматически Delphi генерирует в окне кода для новой формы следующие строки:

Листинг 1.1

```
unit Unit1;
interface
uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs;
type
    TForm1 = class(TForm)
     private
      { Private declarations }
     public
      { Public declarations }
    end;
var
    Form1: TForm1;
Implementation
{$R *.dfm}
end.
```

Этот фрагмент указывает, что объект TForm1 порожден от объекта TForm. Тут же комментариями отмечены места, предназначенные для ввода собственного кода — как доступного извне (public), так и скрытого (private) от внешних программ

Delphi сама вставляет необходимые строки между строками **unit** Unit1 и **implementation.**

Переключение между формой, редактором кода и объектным инспектором происходит с помощью функциональной клавиши **F11**. Переключение между формой и редактором кода происходит с помощью клавиши **F12**.

Над проектом можно производить операции: создания, сохранения, закрытия и открытия. Также проект можно запускать на выполнение, чтобы проверить правильность работы программы.

Сохранение проекта. **File, Save All** (требуется сохранить 2 файла – файл проекта .dpr и файл модуля .pas)**.** Совет: каждый новый проект сохраняйте в отдельной папке.

Закрытие проекта**:** выполните команду **File, Close All**.

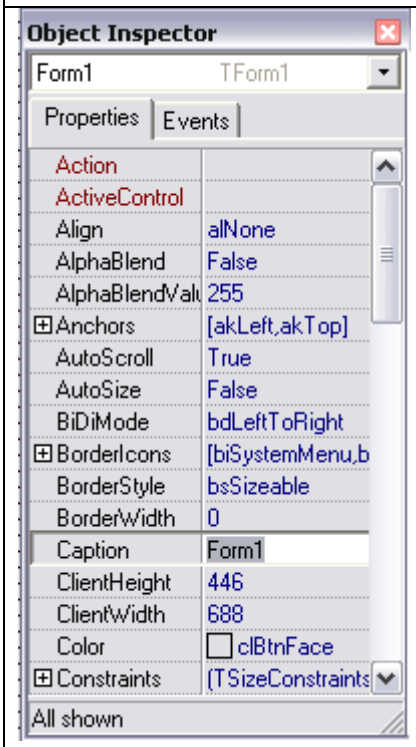
Открытие проекта**:** выполните команду **File, Open Project.** В появившемся окне найти папку с нужным проектом, войти в нее и выделить файл проекта (.dpr). Затем нажать кнопку «**Открыть**».

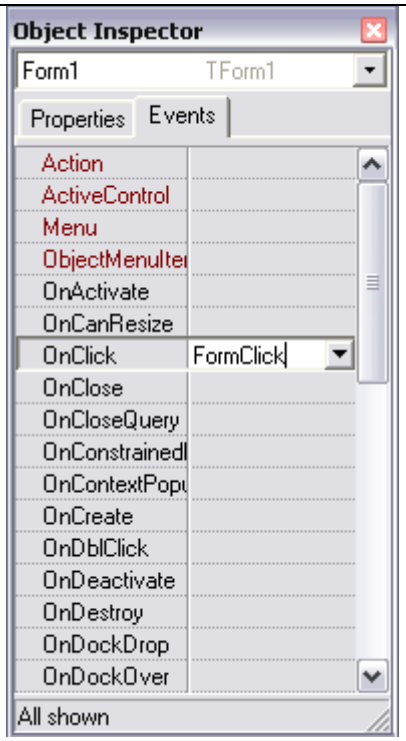
Запуск программы на выполнение осуществляется одним из трех способов:

1. командой главного меню Run – Run;
2. клавишей ***F9***;
3. кнопкой Run на панели инструментов.

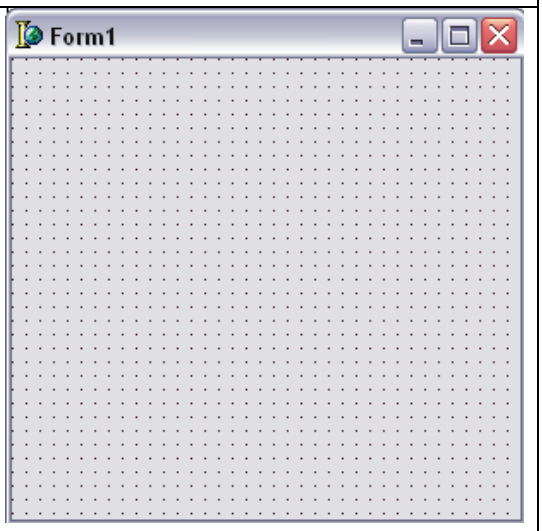
ГРАФИЧЕСКИЙ МАТЕРИАЛ

Инспектор объектов – окно Object Inspector

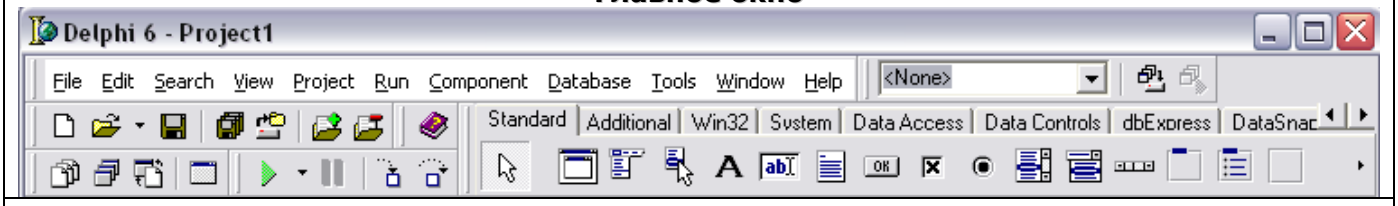




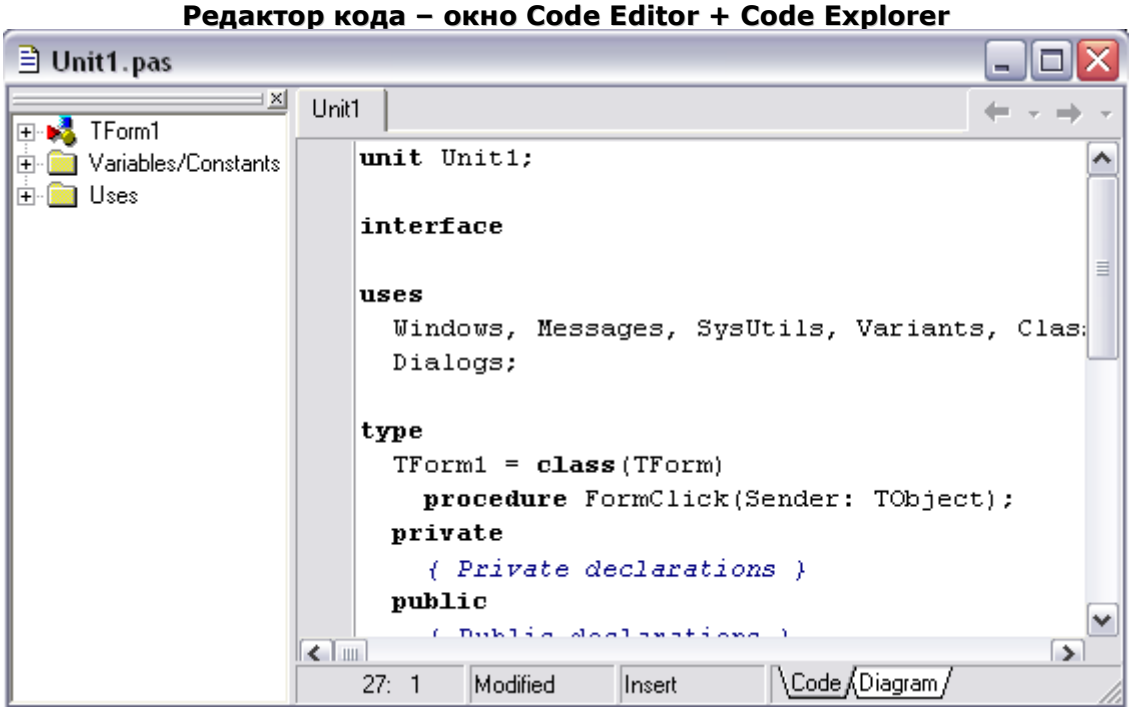
Конструктор форм - форма



Главное окно



Редактор кода – окно Code Editor + Code Explorer



УРОК №2. ПЕРЕМЕННЫЕ. ТИПЫ ДАННЫХ И МАТЕМАТИЧЕСКИЕ ВЫРАЖЕНИЯ

Цель:

Образовательная: изучить основные понятия языка программирования (константы, переменные, стандартные типы данных). Изучить математические функции и правила перевода математических выражений с математического языка на язык программирования.

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

Развивающая: развитие познавательных способностей

Тип занятия: Комбинированный урок

Форма организации учебного процесса: Комбинированный урок

Наглядные пособия и ТСО: Изображения элементов среды программирования

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Актуализация прежних знаний. Повторение материала.**
   1. Пять человек отвечают на тестовые задания по уроку №1.
   2. Три человека (по очереди) у доски отвечают об основных элементах среды программирования, для чего каждый элемент нужен, где расположен (демонстрируют с помощью наглядных пособий).
3. **Изучение нового материала.** Лекция.
4. **Практическая работа:** перевод математических выражений на язык программирования
5. **Домашнее задание:** запишите следующие выражения на языке программирования Pascal:

$\frac{(3{,}27+a)(6{,}98+b)}{3{,}52x}$	$\frac{(x+1)^3}{2x^2-1}$

6. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

ПЕРЕМЕННЫЕ

Важнейшей составляющей любого языка программирования являются переменные. Их основная задача – постоянное или временное сохранение данных, используемых в программе.

Переменная – это именованный фрагмент памяти, выделяемый или резервируемый для сохранения данных. Главная характеристика переменной – ее тип, который определяет содержание переменной и объем выделяемой под нее памяти.

Раздел описания переменной.

Для использования переменной в программе ее необходимо объявить. Объявление переменной выполняется с помощью служебного слова **var** в блоке описания переменных.

Синтаксис: var имя переменной: тип переменной;

Пример: var a, b: integer;

Зачем нужно описание переменных: После того как программа написана, ее запускают на выполнение. Компьютер сначала совершает компиляцию (перевод на машинный язык), во время которой производит подготовительные действия, одно из которых отведение в памяти места под переменные, упомянутые в программе. Итак, описание переменных необходимо. Чтобы перечислить компьютеру переменные, под которые он должен отвести ячейки памяти. *Объявление переменной обязательно*. При появлении незнакомой переменной, компиляция проекта будет прервана и появится сообщение об ошибке.

Имя переменной

Имя (идентификатор) переменной – обозначение переменной. Мы привыкли переменные обозначать буквами, в Delphi их можно обозначать целыми словами. Компьютеру все равно, как вы обозначите переменную. Исключение составляют зарезервированные слова, которые нельзя использовать в качестве имени.

Правило: именем переменной может служить любая последовательность цифр, латинских букв и знака присваивания, не начинающаяся с цифры.

Примеры:

Правильно	Неправильно
A, X, Velichina Zz, s25, k1, _k	Ж – русская буква 2as – начинается с цифры Domby&Son – содержит & Polnay summa – пробел.

Задание №1: определите, правильно или неправильно записано имя переменной.

1. A
2. Ж
3. polnaja summa
4. zzz
5. s25
6. domby&son
7. x_y
8. k1
9. _rk
10. Velichina
11. 2as

В описании переменной используется тип переменной. В языке программирования Pascal существует большое количество типов. Рассмотрим самые простые из них.

ТИПЫ ДАННЫХ:

1. целый тип
2. действительный тип
3. логический (булевый) тип
4. строковый тип

Целые типы данных. Целые типы данных используются для представления целых чисел.

Тип	Диапазон значений	Размер в байтах	Знаковый ?
Byte	0 .. 255	1	нет
Word	0 .. 65535	2	нет
ShortInt	-128 .. 127	1	да
SmallInt	-32768 .. +32767	2	да
Cardinal	0 .. 4 294 967 295	4	нет
Integer	-2 147 483 648 .. + 2 147 483 647	4	да
LongInt	-2 147 483 648 .. +2 147 483 647	4	да
Int64	–2^63..2^63–1	8	да

В настоящий момент тип **Integer** эквивалентен типу **LongInt**, но в последующих версиях это может быть изменено. Приведенные в таблице затраты памяти могут изменяться от версии к версии и от системы к системе. Поэтому, если требуется достоверно знать затраты памяти для того или иного типа, следует пользоваться функцией **SizeOf**.

Действительные типы данных

Действительные типы данных предназначены для хранения чисел, имеющих дробную часть.

Тип	Диапазон значений	Число значащих разрядов	Размер в байтах
Real48	-2.9 x 10^−39 .. 1.7 x 10^38	11–12	6
Single	-1.5 x 10^−45 .. 3.4 x 10^38	7–8	4
Real	-5.0 x 10^−324 .. 1.7 x 10^308	15–16	8
Double	-5.0 x 10^−324 .. 1.7 x 10^308	15–16	8
Extended	-3.6 x 10^−4951 .. 1.1 x 10^4932	19–20	10
Comp	−2^63+1 .. 2^63 −1	19–20	8
Currency	−922337203685477.5808.. 922337203685477.5807	19–20	8

Тип **Extended** обладает максимальной точностью, но могут возникать проблемы с его переносимостью на другие платформы. Тип **Currency** используется для представления денежных величин. В памяти он хранится как масштабированное в **10000** раз 8-байтовое целое. Благодаря этому при операциях с величинами типа **Currency** минимизируются ошибки округления, что очень важно для денежных расчетов. В выражениях, в которых смешаны величины типа **Currency** с величинами других действительных типов, значения **Currency** автоматически умножаются или делятся на **10000**.

Приведенные в таблице затраты памяти могут изменяться от версии к версии и от системы к системе. Поэтому, если требуется достоверно знать затраты памяти для того или иного типа, следует пользоваться функцией **SizeOf**.

Тип данных Boolean –логический (булевый)

Использование этих операций расширяет возможности по формированию сложных условий в ряде операторов.

Переменная булевого типа может принимать только два значения: true (истина) и false (ложь).

Переменные логического типа получают значение в результате выполнения операций сравнения (отношения).

Операции сравнения: <, >, <=, >=, =, <>

Результат операции отношения равен True, если отношение удовлетворяется для значений, входящих в него операндов, и False – в противном случае.

Символьный тип – String. К символам относятся все буквы алфавита, знаки и числа 0-9. Они могут использоваться по отдельности (a, Z,!, 3) или соединяться друг с другом в строки (например, 'Это только проверка').

КОНСТАНТЫ

Помимо переменных, которые могут менять свое значение в процессе работы программы, в языке программирования Pascal предусмотрены константы. Константа, в отличии от переменной не может менять своего значения. Константы описываются в разделе описания констант, который начинается со служебного слова **Const**.

Например:

```
Const
        A = 207;
        Count = 355;
```

МАТЕМАТИЧЕСКИЕ ВЫРАЖЕНИЯ

Операции с целыми числами

Над целыми числами можно выполнять следующие арифметические операции: сложение, вычитание, умножение, целочисленное деление, получение остатка от деления. Знаки этих операций: **+**, **-** , *****, **div**, **mod**.

Примеры**:** 17 div 2 = 8, 3 div 5 = 0. 17 mod 2 = 1, 3 mod 5 = 3.

Стандартные функции для аргументов целого типа

На Pascal	Математическое выражение	Тип результата
Abs(X)	|X|	целый
Sqr(X)	X^2	целый
Sin(X)	Синус X	действительный
Cos(X)	Косинус X	действительный
ArcTan(X)	Арктангенс X	действительный
Ln(X)	LnX	действительный
Exp(X)	e^X	действительный
Sqrt(X)	$\sqrt{x}$	действительный
Odd(X)	Проверка числа X на четность	логический, (x=5 - odd(x)=true, при x=4 - odd(x)=false).
inc(X)	Увеличивает X на 1	X:=X+1
inc(X,N)	Увеличивает X на N	X:=X+N
dec(X)	Уменьшает X на 1	X:=X-1
dec(X,N)	Уменьшает X на N	X:=X-N

Операции и стандартные функции для действительных типов

Frac(X)	дробная часть X
Int(X)	Целая часть X
Pi	Число Пи
Trunc(X)	Целая часть путем отсечения дробной части
Round(X)	округляет аргумент до ближайшего целого

Задание №2: Вычислите:

1. Sqr(2+1)
2. 10+sqr(2+1)
3. 1+abs(5-8)
4. Sqr(2)+sqrt(35+1)
5. Sqrt(8+int(41.5))
6. 21 div (Round(pi+1))

Задание №3: запишите на математический язык.

1. (sin(2)+sqr(35.8))/(2.6+sqr(1.3))
2. (a+3*b-5*a)/(2*a +tan(25*pi/180))

УРОК №3. ОПЕРАТОР ПРИСВАИВАНИЯ. ВВОД И ВЫВОД ДАННЫХ

Цель:

Образовательная: Изучить оператор присваивания и принцип его работы. Рассмотреть основные функции преобразования типов. Получить практические навыки по составлению программ на вычисление математических выражений, параллельно изучив способы ввода и вывода информации в программе.

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

Развивающая: развитие познавательных способностей

Тип занятия: Комбинированный урок

Форма организации учебного процесса: Комбинированный урок

Наглядные пособия и ТСО: карточки для организации самостоятельной работы по вариантам.

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Самостоятельная работа.** Учащимся выдаются карточки с заданиями по пройденной теме (12 вариантов).
3. **Изучение нового материала.** Оператор присваивания и принцип его работы
4. **Выполнение устных упражнений.** 4 задания
5. **Рассмотрение примера** и параллельное изучение нового теоретического материала по вводу и выводу данных. Рассмотрение основных функций преобразования типов.
6. **Домашнее задание.** Составить программы для вычисления математических выражений (A-E).
7. **Подведение итогов урока.**

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

ОПЕРАТОР ПРИСВАИВАНИЯ

Что мы можем делать с переменными? Прежде всего, мы можем задавать значение той или иной переменной при помощи оператора присваивания. Если мы хотим сказать, что переменная a имеет значение 6, то должны записать a:=7 (переменной a присвоить значение 7). После выполнения данного оператора, компьютер будет помнить, что a=7. Знак присваивания – «:=».

Справа от знака присваивания можно писать числа, переменные, арифметические выражения. Слева только переменные.

переменная	:=	число
		Переменная
		Арифметическое выражение
		функция

Можно	Нельзя
C:=34 Z:=f-4	34:=c f-4:=z

Оператор присваивания устроен так, что сначала он вычисляет правую часть, а затем присваивает значение переменной стоящей в левой части.

Например, после выполнения фрагмента программы:

```
a:=2*3+4;
b:=a;
y:=a+b+1;
```

компьютер будет знать, что a=10, b=10, y=21.

Несколько примеров:

Фрагмент программы	Что запомнит компьютер
V:=-2+10; h:=10*v; s:=v+h;	V=8, h=80,s=88
T:=0; n:=2*t+40; z:=z-n;	T=0, n=40, z=-40

Что делает оператор присваивания с памятью?

Рассмотрим пример:

```
Var a,b,y:integer;
Begin
  A:=10;
  B:=6;
```

```
    Y:=a+b+1;
  End;
```

В программе между begin и end встречаются три переменные, поэтому все они перечислены в описании var a,b,y:integer. Компьютер отведет под них три 2-байтные ячейки.

Работает оператор присваивания:

Выполняя оператор y:=a+b+1, компьютер сначала смотрит на его правую часть a+b+1. Если в ней встречается переменные a и b, то компьютер перед вычислением ищет их значения в соответствующих ячейках (и находит там 10 и 6). Подставляет эти значения в правую часть и вычисляет ее. Затем вычисленное значение (17) компьютер заносит в ячейку памяти, отведенную под переменную, находящуюся в левой части выражения (y).

Задание: поменяйте местами 4 и 5 строки. Что произойдет?

Оператор присваивания (:=) заменяет текущее значение переменной новым значением, которое определяется выражением, или определяет выражение, значение которого должно возвращаться функцией.

Пример: Объяснить, чему последовательно равны значения переменных на каждом этапе выполнения программы.

```
var  X, Y: Integer; {переменные X и Y имеют тип Integer}
     J: Real;       {переменная J имеет тип Real}
Begin
  X:=7;            {переменной X присваивается значение}
  Y:=7;            {переменной Y присваивается значение}
  X:=Y+Y;          {значение переменной X изменяется}
  J:=0.075;        {переменной J присваивается значение с плавающей точкой}
End;
```

Значение, которое вычисляется справа от оператора присваивания, должно совпадать по типу с переменной, стоящей слева от знака присваивания. Таким образом, переменной целого типа, нельзя присвоить дробное значение. Аналогично, строковой переменной нельзя присвоить числовое значение.

Задание №1: Какое значение будет присвоено переменной t после выполнения фрагмента программы:

```
K:=1+2;
S:=2*k;
T:=6-s;
```

Задание №2: Определите, чему будет равно значение переменной F после выполнения фрагмента программы.

```
F:=30;
F:=F+4;
```

Задание №3: Определите, чему будет равно значение переменной A после выполнения фрагмента программы.

1. a:=100; a:=10*a+1;
2. a:=100; a:=-a;

Задание №4: Определите значение переменной b после выполнения фрагмента программы:

```
A:=(sqr(2)+1)*(20-sqr(2*2))-11;   B:=11 div (a-4);
```

ВВОД И ВЫВОД ДАННЫХ

В рассмотренных ранее примерах значения задавались прямо в коде программы. Если, к примеру, требуется решить следующую задачу:

Составить программу, которая вычисляет площадь прямоугольника. Пользователь вводит длину и ширину прямоугольника.

Возникает проблема: как предоставить пользователю возможность ввода данных и как вывести результат вычисления на экран.

Для ввода данных в Delphi используется компонент Edit. Значение, которое пользователь вводит, заносится в свойство Text. Для того, чтобы использовать введенное значение, необходимо его считать и присвоить какой-либо переменной. При этом, если нам необходимо работать со значением, как с числом, необходимо преобразовать тип. Т.к. значение в свойстве Text строкового типа.

Функции преобразования типов:

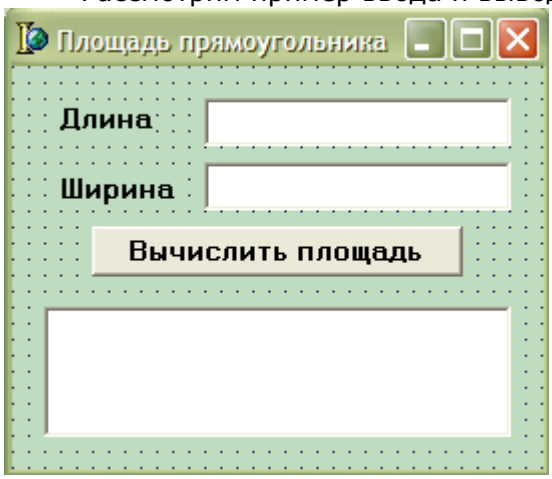
IntToStr(x:integer) – преобразование переменной X целого типа данных в переменную строкового типа.

FloatToStr(x:real) **-** преобразование переменной X вещественного типа в переменную строкового типа.

StrToInt(S:string) – преобразование переменной s строкового типа в переменную целого типа.

StrToFloat(S:string) - преобразование переменной s строкового типа в переменную вещественного типа.

Рассмотрим пример ввода и вывода данных на примере решения поставленной задачи:



Прежде, чем приступить к составлению программы, необходимо настроить интерфейс программы, установить все необходимее компоненты для ввода и вывода данных, а также для управления программой.

Для этого на форму необходимо поставить 2 компонента Edit, для ввода длины и ширины прямоугольника, 1 компонент Memo, для вывода результата и кнопку Button, для организации вычисления.

Для того, чтобы по нажатию на кнопку производилось вычисление, необходимо создать обработчик события нажатия кнопки onClick и в нем написать код программы. Для этого необходимо дважды щелкнуть по кнопке, в результате откроется окно редактора кода с готовой заготовкой обработчика.

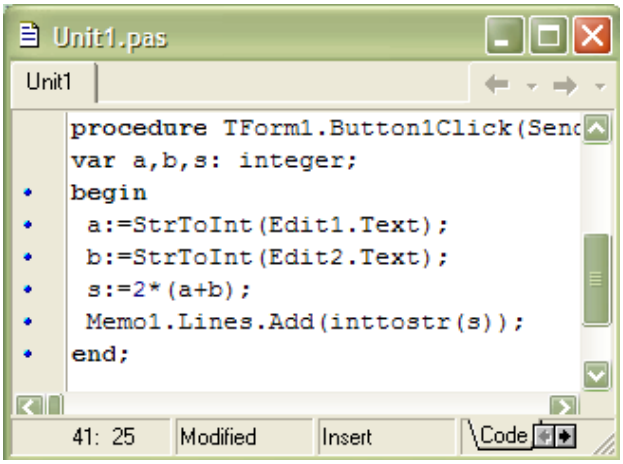
Код программы:

В программе используются три переменные:

- для хранения длины;
- для хранения ширины;
- для хранения вычисленной площади.

Прежде, чем вычислить площадь прямоугольника, необходимо считать значения длины и ширины введенные пользователем в Edit1 и в Edit2.

Обратите внимание, на преобразование типов, в момент считывания данных с компонентов. Значение, хранящееся в свойстве Text, строкового типа преобразуется в целое число.



Запись Edit1.Text означает обращение к свойству Text компонента Edit1. После имени компонента, через точку указывается его свойство.

После того, как значения считаны, можно приступать к вычислению площади. Для этого, значению переменной s присваивается значение 2*(a+b), по формуле площади прямоугольника.

Программа вычислит значение площади, но для того, чтобы пользователь увидел результат вычисления, необходимо значение переменной s вывести на экран. Для этого на форме расположен компонент Memo1. В его свойство Lines методом Add добавляется значение переменной s. Обратите внимание, на преобразование типа, теперь уже из числа в строку.

Задание №5: Даны x, y, z. Вычислить a, b, если

A	$a=\frac{\sqrt{\lvert x-1\rvert}-\sqrt{\lvert y\rvert}}{1+\frac{x^2}{2}+\frac{y^2}{4}}$	$b=X(arctgZ)+e^{-(x+3)}$

Домашнее задание: Даны x, y, z. Вычислить a, b, если

А	$a=\frac{3+e^{y-1}}{1+x^2\lvert y-tgz\rvert}$	$b=1+\lvert y-x\rvert+\frac{(y-x)^2}{2}+\frac{\lvert y-x\rvert^3}{3}$
Б	$a=(1+y)\frac{x+\frac{y}{x^2+4}}{e^{-x-2}+\frac{1}{x^2+4}}$	$b=\frac{1+\cos(y-2)}{\frac{x^4}{2}+\sin^2 z}$
В	$a=y+\frac{x}{y^2+\left\lvert\frac{x^2}{y+\frac{x^3}{3}}\right\rvert}$	$b=1+tg^2\frac{z}{2}$
Г	$a=\frac{2\cos(x-\frac{\pi}{6})}{1/2+\sin^2 y}$	$b=1+\frac{z^2}{3+\frac{z^2}{5}}$
Д	$a=\frac{1+\sin^2(x+y)}{2+\left\lvert x-\frac{2x}{1+x^2y^2}\right\rvert}+x$	$b=\cos^2(arctg\frac{1}{z})$
Е	$a=\ln\left\lvert(y-\sqrt{\lvert x\rvert})(x-\frac{y}{z+\frac{x^2}{4}})\right\rvert$	$b=x-\frac{x^2}{3!}+\frac{x^5}{5!}$

САМОСТОЯТЕЛЬНАЯ РАБОТА ПО КАРТОЧКАМ:

1 задание: вычислить формулу

2 и 3 задание: записать математическое выражение на языке программирования

4 задание: записать на математический язык выражение записанное на языке программирования

Вариант №1

1. 247 div 10, 247 mod 10
2. $a=\frac{3+e^{y-1}}{1+x^2\lvert y-tgz\rvert}$
3. $256+\frac{148+1{,}25^2}{\sqrt{15}}$
4. sqr(2.5)*2.5+132/(sin(25*pi/180)+sqrt(5))

Вариант №2

1. 25 div 4, 25 mod 4
2. $b=1+\lvert y-x\rvert+\frac{(y-x)^2}{2}+\frac{\lvert y-x\rvert^3}{3}$
3. $\frac{25{,}4^2-14{,}25}{155\sin^2 35-6.1}$
4. (5*a*sqr(x)*x)-3*b)/(sqrt(a)+3*b)

Вариант №3

1. 132 div 100 , 132 mod 10
2. $a=(1+y)\frac{x+\frac{y}{x^2+4}}{e^{-x-2}+\frac{1}{x^2+4}}$
3. $\frac{158-12.1^2}{5.1}\cdot 10{,}2$
4. (cos(5*3.14/180)+sqr(2.1)/(5+sqr(3))

Вариант №4

1. 535 div 10 , 535 mod 100
2. $b=\frac{1+\cos(y-2)}{\frac{x^4}{2}+\sin^2 z}$
3. $\frac{\sqrt{256{,}25-1{,}18^{2,1}}}{\cos^2 40\cdot 848{,}2-1{,}2}$
4. (sqr(5*x)+a)/(2*x +tan(x))

Вариант №10

1. 983 div 100 , 638 mod 10
2. $b = \cos^2(arctg\frac{1}{z})$
3. $\sqrt{144,4} + \frac{1825 \cdot \cos 40}{1,5^3 + 1,82^2}$
4. (sqr(a*b)+2*a)/(sqr(cos(a))+b)

Вариант №6

1. 76 div 10 , 176 mod 10
2. $b = 1 + tg^2\frac{z}{2}$
3. $\frac{\sin^2 30 - \cos^2 30}{\sqrt{0,045} - 2}$
4. (sqr(sin(2*a))+a*b)/(sqr(a*b)+b)

Вариант №7

1. 159 div 10 , 365 mod 10
2. $a = \frac{2\cos(x - \frac{\pi}{6})}{1/2 + \sin^2 y}$
3. $181 - \frac{15,6^3 - 8,2}{121}$
4. (sqr(2.3)+sqrt(157.5))/sqr(tan(40*pi/180))

Вариант №8

1. 69 div 10 , 69 mod 10
2. $b = 1 + \frac{z^2}{3 + \frac{z^2}{5}}$
3. $tg30 \cdot \frac{145 + 14,5^2}{\sqrt{256,25 + 2,1^3}}$
4. (b+sqr(cos(a)))/(sqr(a*b)+3*a)

Вариант №9

1. 395 div 10 , 198 mod 10
2. $a = \ln\left|(y - \sqrt{|x|})(x - \frac{y}{z + \frac{x^2}{4}})\right|$
3. $16,2^2 - \frac{526}{15,2^2 4,3}$
4. abs(sin(75*pi/180)-5)+(3.2+sqr(1.5))/2.3

Вариант №5

1. 17 div 3, 17 mod 3
2. $a = y + \frac{x}{y^2 + \left|\frac{x^2}{y + \frac{x^3}{3}}\right|}$
3. $\frac{148}{1,4^3 + 2,6} + 18,1$
4. (2*cos(60*pi/180))/(3+sqr(5))

Вариант №11

1. 123 div 10, 123 mod 10
2. $a = \frac{\sqrt{|x-1|} - \sqrt{|y|}}{1 + \frac{x^2}{2} + \frac{y^2}{4}}$
3. $\frac{14}{52 - 14,6^2} + 326,3$
4. 5*a*sqr(b)+(a + b)/(2*a)

Вариант №12

1. 352 div 100 , 352 mod 100
2. $b = X(arctgZ) + e^{-(x+3)}$
3. $4^{2,5} + \frac{\sqrt{1148,25 - 1,1^4}}{156 \cdot \cos 25 + 2,6}$
4. (3*sqr(x)+2*y)/(x*y-7.4)+2/3/y

УРОК № 4. УСЛОВНЫЙ ОПЕРАТОР IF

Цель:

Образовательная: Изучить разветвляющиеся алгоритмы, условный оператор IF. Объяснить принцип работы условного оператора и рассмотреть примеры его использования

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

Развивающая: развитие познавательных способностей

Тип занятия: Комбинированный урок

Форма организации учебного процесса: Комбинированный урок

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Повторение теоретического материала и проверка домашнего задания** (выборочно у доски и тетради). Проверка знаний с помощью тестовых заданий
3. **Изучение нового материала.** Лекция
4. **Устные упражнения**
5. **Рассмотрение примеров** применения условного оператора
6. **Решение задач**

   Домашнее задание: прочесть теоретический материал лекции и запомнить принцип работы условного оператора. Решить задачи:

7. 1. Составить программу вычисления значения функции: $y = \begin{cases} \dfrac{2x^3+1}{x-5}, & если\ x < 4 \\ 3x^2 - 2, & если\ x \geq 4 \end{cases}$
   2. Составить программу вычисления значения функции: $F = \begin{cases} x^2 - 3x + 9, & если\ x \leq 3 \\ \dfrac{1}{x^3+6}, & если\ x > 3 \end{cases}$
8. **Подведение итогов урока.**

Часто при решении задач приходиться делать выбор и в зависимости от условия выполнять те или иные действия. Для организации выбора в языках программирования высокого уровня используется команда ветвления.

Ветвление – это составная команда алгоритма, в которой в зависимости от условия выполняется или одно, или другое действие.

Ветвление в Object Pascal реализуется с помощью условного оператора IF.

Блок-схема условного оператора:

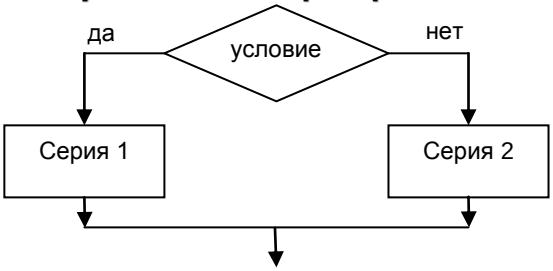


Схема условного оператора IF:

IF <условие> THEN <серия1>
ELSE <серия2>

Условие – два арифметических выражения, соединенные знаком сравнения, или условие равенства или неравенства строк. Серия – это один или несколько операторов. Если операторов несколько, то они заключаются в операторные скобки Begin End. Перед ELSE точку с запятой не ставят.

Порядок работы оператора IF:

1. Проверяется истинность <условия>.
2. Если условие истинно, то выполняются операторы серии 1, а операторы серии 2 не выполняются. Если условие ложно, то наоборот выполняются операторы серии 2, а операторы серии 1 не выполняются.

2 формы записи условного оператора:

1. Полная форма записи. В этой форме присутствуют обе ветви.
2. Неполная форма записи. В этой форме ветвь Else отсутствует, если в случае невыполнения условия ничего не надо делать.

Пример: составить программу, нахождения наибольшего из двух введенных чисел на экран.

Решение:

```
Var x, y: integer;
Begin
 x:=StrToInt(Edit1.text);
 y:=StrToInt(Edit1.text);
 if  x>y      then Memo1.Lines.Add(IntToStr(x))
              else Memo1.Lines.Add(IntToStr(y))
End;
```

Упражнения:

1. Определите, чему будет равно значение переменной S после выполнения фрагмента программы:

```
S:=6;
If s<0 then s:=s+10; s:=s+1;
```

2. Найдите ошибку в записи условного оператора:

```
d:=10;
if d<5 then x:=d;
       else y:=d;
```

3. Определите, чему будет равно значение переменной K после выполнения следующих фрагментов программ:
   - **a.** K:=20; k:=k+10; if k+10<>30 then k:=8 else k:=k-1;
   - **b.** K:=20; k:=k+10; if k+10=30 then k:=8 else k:=k-1;
4. Используя составной оператор, упростите следующий фрагмент:

```
If a>b then c:=1;
If a>b then d:=2;
If a<=b then c:=3;
If a<=b then d:=4;
```

Пример: составить блок-схему и программу вычисления значения функции:

$$y=\begin{cases} x^2, \text{если } x\le 0 \\ \sqrt{x}, \text{если } x>0 \end{cases}$$

Решение:

```
Var x: integer; y:real;
Begin
 x:=StrToInt(Edit1.text);
 if  x<=0     then y:=SQR(x)
              else y:=SQRT(x);
 Memo1.Lines.Add(FloatToStr(y))
End;
```

Решение задач:

1. Написать программу для подсчета суммы только положительных из 3-х данных чисел.
2. Составить программу, в которой из 3-х введенных с клавиатуры чисел, возводятся в квадрат положительные числа, а отрицательные остаются без изменения.

УРОК №5. РЕШЕНИЕ ЗАДАЧ ПО ТЕМЕ «УСЛОВНЫЙ ОПЕРАТОР»

Цель:

Образовательная: Получить навыки решения задач с использование условного оператора IF

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

Развивающая: развитие познавательных способностей

Тип занятия: Урок закрепления знаний и получения практических навыков

Форма организации учебного процесса: Практическая работа

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Актуализации прежних знаний.** Фронтальный опрос по изученной теме и выборочно выполнение тестовых заданий
3. **Практическая работа:** решение задач с использованием условного оператора
4. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Решение задач у доски

1. Определить, является ли данное целое число четным.
2. Пользователь вводит двузначное целое число. Программа печатает на экране наибольшую из цифр этого числа.

Задачи для самостоятельного решения

1. Составьте программу, определяющую, входит ли введённая Вами цифра в десятичную запись введённого Вами трёхзначного числа, и печатающую сообщение о том, входит ли эта цифра в запись числа или нет.
2. Пользователь вводит 3 числа. Программа печатает их в порядке убывания.
3. Составить программу для вычисления выражения Max(x+y+z, xyz)+3.
4. Составить программу для вычисления выражения Min(x^2+y^2, y^2+z^2)-4.

УРОК №6. ВЛОЖЕННЫЕ УСЛОВНЫЕ ОПЕРАТОРЫ

Цель:

Образовательная: Рассмотреть каким образом организуются вложенные условные операторы, и в каких случаях они используются. Научить составлять программы с использованием вложенных условных операторов.

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

Развивающая: развитие практических навыков решения задач по теме «Вложенные условные операторы».

Тип занятия: комбинированный урок

Форма организации учебного процесса: комбинированный урок

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Актуализация знаний.** Повторение принципа работы условного оператора. Фронтальный опрос
3. **Изучение нового материала:** принцип работы вложенного условного оператора
4. **Рассмотрение примера**
5. **Изучение нового материала:** составное условие, логические операции и таблица истинности
6. **Рассмотрение примеров**
7. **Решение задач**
8. **Домашнее задание:**

4.1.Составить блок-схему и программу для вычисления функции:

$$y=\begin{cases}\sqrt{x}, & если \quad x>5\\ x^3-3, & если \quad x=5\\ 25+x, & если \quad x<5\end{cases}$$

4.2.Меньшее из двух чисел a и b заменить на 0, а в случае их равенства заменить оба.

4.3.Даны действительные числа x, y, z. Получить Min (x, y, z).

9. **Подведение итогов урока.**

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Оператор IF может быть использован и для создания многоуровневой структуры операторов IF. Многоуровневая структура в общем виде записывается следующим образом:

IF <условие1> THEN <серия1>
ELSE
IF <условие2> THEN <серия2>
ELSE <серия3>

Принцип работы вложенного условного оператора:

Сначала проверяется <условие1>. Если оно соблюдается, то выполняется <серия1> и дальше управление передается операторам, записанным после конструкции. Если <условие1> не соблюдается, то проверяется <условие2> и при его соблюдении выполняется <серия2>. Если же оно не соблюдается, то выполняется <серия3>.

Во всех случаях после выполнения соответствующей серии команд управление передается ниже следующим командам.

Задание №1: чему будет равно значение переменной D после выполнения фрагмента программы:

```
D:=3; v:=10; if V<6 then if v<20 then d:=1 else d:=2;
```

Ответ зависит от того, к какому IF принадлежит Else (к первому или ко второму). Пройдем по тексту программы от Else к началу, и если по пути нам не встретится еще 1 Else, то 1-й же IF на который мы натолкнемся, будет искомый. Если встретится еще один ELSE, то забудем на время про 1-й ELSE. Найдем IF для нового ELSE. В нашем случае else принадлежит к второму If. Отсюда следует, что ответ d=3.

Правило: ELSE всегда относится к ближайшему слева оператору IF.

Задание №2: чему будет равно значение переменной D после выполнения фрагмента программы, если K=9 (если K=2):

```
D:=8; if k<8 then if k>3 then d:=3 else d:=1;
```

Пример 1: Даны три числа: A, B, C. Если выполняется неравенство A<B<C, то вычислить сумму чисел, иначе их произведение.

```
var a,b,c,s:integer;
begin
    a:=StrToInt(Edit1.Text);
    b:=StrToInt(Edit2.Text);
    c:=StrToInt(Edit3.Text);
    if a<b then
        if b<c then s:=a+b+c
            else s:=a*b*c
```

```
            else s:=a*b*c;
        Memo1.Lines.Add(IntToStr(s));
    end;
```

Достаточно громоздкая запись, несмотря на то, что производится вычисление только по одной формуле. Если при подобном условии потребовалось бы вычислять несколько выражений, происходило бы дублирование кода.

Для упрощения записи подобных условий используется составное условие.

Составное условие

Составное условие – условие (логическое выражение), состоящее из нескольких простых условий, объединенных логическими операциями и, или, не (and, or, not).

Логические операции: and (И), or (ИЛИ), not (НЕ, отрицание)

Значения операндов		**Результат операции**		
X	***Y***	***Not X***	***X and Y***	***X or Y***
Л	Л	И	Л	Л
Л	И	И	Л	И
И	Л	Л	Л	И
И	И	Л	И	И

Логические операции, операции отношения и арифметические операции часто встречаются в одном выражении. При этом отношения, стоящие слева и справа от знака логической операции должны стоять в скобках, т.к. логические операции имеют более высокий приоритет. Принят следующий **приоритет операций**:

1. not (самый высокий приоритет)
2. and, *, div, mod
3. or, xor, +, -
4. операции сравнения

Кроме того, порядок выполнения операций определяется скобками.

Например: A or B and not (A or B) *Ответ*: 4 3 2 1

Задание №3: Вычислите значение выражения:

1. (a>5)and(b>5)and(a<20)and(b<30)
2. not(a<15)or not(b<30)
3. (-3>=5)or not (7<9)and(0<3)

Используя составное условие рассмотренную ранее задачу можно решить так:

```
var a,b,c,s:integer;
begin
    a:=StrToInt(Edit1.Text);
    b:=StrToInt(Edit2.Text);
    c:=StrToInt(Edit3.Text);
    if (a<b)and(b<c)        then s:=a+b+c
                            else s:=a*b*c;
    Memo1.Lines.Add(IntToStr(s));
end;
```

При такой записи программы условие более наглядно и нет надобности повторять фрагменты программы.

Пример 2: Даны действительные числа a, b, c. Если $a \le b \le c$, то все числа заменить их квадратами, если a>b>c, то каждое число заменить наибольшим из них, в противном случае сменить знак каждого числа.

Решение:

Запишем условие задачи следующим образом:

$a:=a^2$, $b:=b^2$, $c:=c^2$ - если $a \le b \le c$

a:=c, b:=c - если a>b>c

a:=-a, b:=-b, c:=-c - в остальных случаях.

```
Var A, B, C: integer;
```

```
Begin
   A:=StrToInt(Edit1.text);
   B:=StrToInt(Edit2.text);
   C:=StrToInt(Edit3.text);
   If (a<=b) and (b<=c) Then
                        Begin
                              A:=sqr(A);    B:=sqr(B);    C:=sqr(C);
                        End
                  Else
                        If (A>B) and (B>C) Then
                                          Begin
                                           A:=C;  B:=C;
                                          End
                                         Else
                                          Begin
                                           A:=-A;  B:=-B;  C:=-C;
                                          End;
   Memo1.Lines.Add('a='+IntToStr(A)+', b='+ IntToStr(B)+', c='+IntToStr(C));
End;
```

Задача 1: Написать программу, определяющую, принадлежит ли число, введенное с клавиатуры, интервалу (0; 5). Использовать составное условие.

Задача 2: Составить программу нахождения произведения двух наибольших из трех вводимых с клавиатуры.

УРОК №7. ОПЕРАТОР ВЫБОРА

Цель:

Образовательная: Рассмотреть оператор выбора Case и принцип его работы. Научить составлять программы с использованием оператора выбора Case.

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

Развивающая: развитие практических навыков решения задач по теме «Оператор выбора Case».

Тип занятия: комбинированный урок

Форма организации учебного процесса: комбинированный урок

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Проверка домашнего задания.** Решение двух задач из домашнего задания у доски. От задач по теме ветвление перейти к рассмотрению задач по теме «Оператор выбора». Пояснить, что при решении задач с организацией множественного выбора в некоторых случаях удобнее использовать оператор выбора Case, чем вложенные условные операторы IF.
3. **Изучение новой темы.**
4. **Рассмотрение примера решения задачи.**
5. **Решение задач.**
6. **Домашнее задание:**

   4.4. Составить программу, которая по номеру месяца определяет соответствующее время года.

   4.5. Ученик вводит с клавиатуры букву русского алфавита. Программа должна ответить, какой звук эта буква: гласный, согласный звонкий, согласный глухой или какой-нибудь другой (можно и не знаю).
7. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Оператор выбора позволяет выбрать одно из нескольких возможностей продолжения программы.

```
Case  <ключ выбора>  of
```

```
        <список выбора>
        [else <оператор>]
    end;
```

Ключ выбора - параметр, по которому осуществляется выбор - выражение любого порядкового типа (Integer, Boolean, Char и т.д.). Возможные значения переменных порядкового типа можно выстроить по порядку и пересчитать.

Пример:

```
Var x : integer;
begin
    Case x of
      0: y:=2;
      1: y:=x;
    End;
End;
```

Порядок работы оператора выбора Case:

1. Вычисляется значение выражения <ключ выбора>. Это может быть переменная или некоторое выражение порядкового типа.
2. В последовательности операторов <список выбора> отыскивается такой, которому предшествует константа, равная вычисленному значению.
3. Найденный оператор выполняется, после чего оператор выбора Case завершает свою работу. Если в списке выбора нет константы, соответствующий вычисленному значению, то выполняется оператор, стоящий за словом else.

Задание №1: Чему будет равно значение переменной k после выполнения фрагмента программы:

```
Var a, k: integer;
Begin
. . .
  A:=3;
  Case a*a+1 of
        8, 3, 20: k:=0;
        7, 10:    k:=1;
        12..18:   k:=3;
  End;
End;
```

Пример 1: Школьная отметка имеет название и обозначается цифрой. Написать программу, которая определяет по значению отметки, записанной цифрой, ее название.

Пример 2: Для заданного числа X вычислить значение функции:

$$y=\begin{cases} x^2, & \text{если } x=5,6,7 \\ 2x-5, & \text{если } x=1,2,3,4 \\ \sqrt{x}, & \text{если } x=8,9,10 \end{cases}$$

Задачи:

1. Составить программу, которая по заданному номеру дня недели требуется вывести его название.
2. Составить программу, которая по номеру месяца определяет соответствующее время года.
3. Составить программу нахождения числа дней в месяце, если даны: номер месяца N – целое число от 1 до 12; целое число A, равное 1 для високосного года и равное 0 в противном случае.

```
Case N of
    1,3,5,7,8,10,12: D:=31;
    2: if A=1 then D:=29 else D:=28
    Else D:=30
End;
```

УРОК №8. ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ

Цель:

Образовательная: изучить принципы организации циклических алгоритмов, рассмотреть оператор цикла с параметром for, операторы цикла с постусловием Repeat … Until и с предусловием While … do.

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

Развивающая: развитие навыков самостоятельной работы над изучением теоретического материала

Тип занятия: урок формирования новых знаний

Форма организации учебного процесса: урок лекция

Наглядные пособия и ТСО: таблица «Циклические алгоритмы»

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Изучение нового материала.** Изучение таблицы «Циклические алгоритмы». Рассмотрение принципов работы цикла с параметром, циклов с постусловием и предусловием, а также отличий между ними. Правила записи циклических алгоритмов на языке программирования Object Pascal.
3. **Выполнение устных упражнений.**
4. **Рассмотрение примера решения задачи.** Три варианта решения задачи, с использованием трех разных циклов.
5. **Домашнее задание:** повторить теоретический материал. Знать правила записи циклов на языке программирования.
6. **Подведение итогов урока.** В течение урока мы рассмотрели три основных вида циклов, принципы их работы и отличия между ними. Цикл с параметром For применяется при организации циклических алгоритмов с заданным числом повторений, циклы с предусловием и постусловием применяются при организации циклических алгоритмов с заранее неизвестным числом повторений.

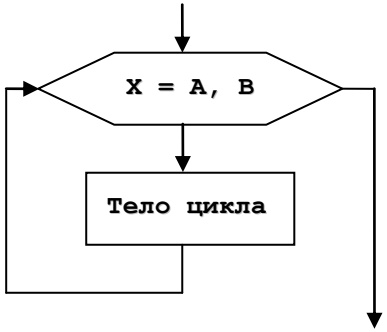
При решении задач часто возникает необходимость много кратного повторения однотипных действий при различных значениях параметров, определяющих эти действия.

Алгоритмы, реализующие такие действия, называются циклическими, а многократно повторяемая последовательность действий (тело цикла) – циклами. Использование циклов позволяет выполнять большие объемы вычислений при помощи компактных программ.

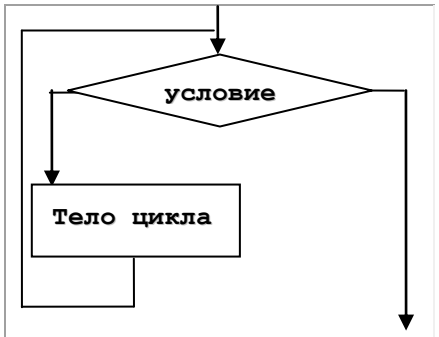
Различают циклы с заданным и заранее неизвестным числом повторений. В таблице «Циклические алгоритмы» собраны основные виды циклов:

1. Цикл с параметром
2. Цикл с предусловием
3. Цикл с постусловием

Вид цикла	Название цикла	Блок-схема	Запись на языке программирования	Пример	Порядок работы цикла

Цикл с известным числом повторений	**Цикл с параметром For**		**For X:= A to B do**
			begin
			<тело цикла>
			end;
			Переменная X – параметр цикла, которая последовательно принимает значения в интервале от A до B.
			A – начальное и **B** - конечное значение параметра
				T:=1;
				For i:=1 to 10 do
				Begin
				T:=T*5;
				End;
					Если A≤B, то параметр X постепенно принимает значения равные A, A+1, … B-1, B и для каждого значения выполняется <тело цикла>. Т.е. параметр X изменяется с шагом +1.
					Если A>B, то <тело цикла> не выполнится ни разу.

			For X:= A downto B do
			begin
			<тело цикла>
			End;
				T:=1;
				For i:=7 downto 1 do
				Begin
				T:=T*5;
				End;
					Выполняется таким же образом, как и прямой цикл с параметром. Если A≥B, то параметр X изменяется с шагом -1.
					Если A≤B, то <тело цикла> не выполнится ни разу.

Цикл с неизвестным числом повторений	**цикл с предусловием While … do**		**While <условие> do**
			begin
			<тело цикла>
			End;
			
			Пока условие истинно, выполняй тело цикла. Если условие ложно, завершай работу.
				T:=0;
				X:=5;
				While X<>0 do
				Begin
				T:=T+5;
				X:=X-1;
				End;
					Проверяется условие. Если условие истинно, то выполняется <тело цикла>, затем снова проверяется условие и т.д.
					Если условие не соблюдается, цикл прекращает свою работу.
					В теле цикла обязательно должен быть оператор, влияющий на соблюдение условия, в противном случае произойдет зацикливание.

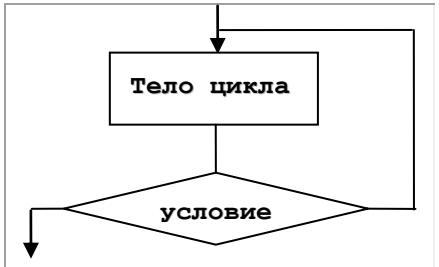
	Цикл с постусловием Repeat … Until		**Repeat** **<тело цикла>** **Until <условие>** Повторять <тело цикла>, пока условие ложно. Если условие истинно завершай работу цикла.	T:=0; X:=5; Repeat T:=T+5; X:=X-1; Until X=0;	Выполняется <тело цикла>. Затем проверяется условие. Если условие не выполняется, то <тело цикла> выполняется еще раз. Если условие выполняется, то цикл завершает свою работу.

Таблица «Циклические алгоритмы»

Устные упражнения:

По циклу For:

1. Сколько раз будет выполнено тело цикла в следующих фрагментах.
   1. `For k:=-1 to 1 do`
   2. `For k:=10 to 20 do`
   3. `For k:=20 to 10 do`
   4. `k:=5; r:=15; For i:=k+1 to r-1 do`
   5. `k:=r; For i:=k to r do`
2. Определите значение переменной s после выполнения следующих операторов:
```
s:=0; n:=10;
For i:=2 to n do s:=s+100 div i;
```
3. Проследите шаг за шагом, что происходит при выполнении приведенных ниже фрагментов программ.

```
for x:=7 to 9 do
  y:=x*3;
  x:=14;
```

```
s:=0;
for x:=1 to 4 do
  s:=s+1;
```

```
for x:=1 to 4 do
ShowMessage(inttostr(x)+''+inttostr(x-1)+''+inttostr(x+1));
```

```
Q:=5; r:=9; for p:=Q to r-1 do ShowMessage(p,'',Q,'',r);
```

По циклу с предусловием While

1. Сколько раз будет повторен цикл, и какими будут значения переменных a, b, s после завершения следующей последовательности операторов:
```
A:=1; b:=1;
While a+b<8 do
Begin a:=a+1; b:=b+2; end;
S:=a+b;
```
2. Какими будут значения переменных a,b после выполнения последовательности операторов:
```
A:=1; b:=1;
While a<=3 do a:=a+1; b:=b+1;
```
3. Определить значение переменной s после выполнения следующих операторов:
   a. `S:=0; i:=0; while i<5 do inc(i); s:=s+100 div I;`
   b. `S:=0; i:=1; while i>1 do begin s:=s+100 div I; dec(i); end;`

По циклу с постусловием Repeat:

1. Определите значение переменной s после выполнения следующих операторов:
```
S:=0; i:=1; repeat s:=s+5 div I; i:=i-1; until i<=1;
```

Пример: Составить программу для возведения числа 8 в пятую степень.

1 вариант решения: используется оператор цикла с параметром For

```
var i,p:integer;
begin
p:=1;
for i:=1 to 5 do
 p:=p*8;
Memo1.Lines.Add(IntToStr(p))
end;
```

2 вариант решения: используется оператор цикла с предусловием While

```
var i,p:integer;
begin
p:=1;
i:=1;
while i<=5 do
begin
 p:=p*8;
 i:=i+1;
end;
Memo1.Lines.Add(IntToStr(p))
end;
```

3 вариант решения: используется оператор цикла с постусловием Repeat

```
var i,p:integer;
begin
p:=1;
i:=1;
Repeat
 p:=p*8;
 i:=i+1;
Until i>5;
Memo1.Lines.Add(IntToStr(p))
end;
```

УРОК №9. РЕШЕНИЕ ЗАДАЧ ПО ТЕМЕ «ЦИКЛ С ПАРАМЕТРОМ»

Цель:

Образовательная: привить навыки решения задач по теме «Цикл с параметром»,

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

Развивающая: развитие навыков использования циклических алгоритмов при решении задач

Тип занятия: урок формирования новых умений

Форма организации учебного процесса: урок практических занятий

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Повторение теоретического материала.** (Тестовые задания, устный опрос или работа по карточкам)
3. **Рассмотрение примеров.**
4. **Решение задач.**
5. **Домашнее задание:**
   1. Подсчитать сумму всех чисел от 100 до 1 (в таком порядке).
   2. Подсчитать сумму всех нечетных чисел от 1 до 1000.
   3. Вводится некоторое натуральное число N. Вывести все натуральные трехзначные числа, сумма цифр которых равна введенному числу N.
   4. Составить программу вычисления суммы кубов чисел от 25 до 125.
   5. Среди двузначных чисел найти те, сумма квадратов цифр которых делится на 13.
   6. Найти все натуральные трехзначные числа, сумма кубов цифр которых равна самому числу. Например, 153 – одно из таких чисел, т.к. 1^3+5^3+3^3=1+125+27=153.
6. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Пример №1:

Условие задачи: Подсчитать сумму всех чисел от 1 до 100.

Вопросы для повторения:

1. Каким образом в программе можно перебрать все числа от 1 до 100? (в цикле)
2. Какой оператор цикла можно использовать, если нам известно начальное и конечное значение цикла? (оператор цикла с параметром For)
3. Каким образом оформляется цикл с параметром? (for i:=A to B do <тело цикла>)

Решение задачи:

В программе не вводятся никакие данные, требуется только подсчитать сумму чисел от 1 до 100 и вывести ее на экран. Установить на форму кнопку Button и в обработчике события нажатия кнопки внести следующий код программы:

```
Var I,s: integer;
Begin
 S:=0;
 For i:=1 to 100 do s:=s+I;
 Memo1.Lines.Add(IntToStr(s));
End;
```

В программе используется цикл с параметром, который последовательно перебирает все числа от 1 до 100, каждый раз увеличивая значение параметра i на 1 (1, 2, 3, 4 … 99, 100). Для хранения значения суммы в программе используется переменная s. Перед тем как начать цикл, следует обнулить значение переменной s. В теле цикла к текущему значению переменной s каждый раз прибавляется значение параметра I, т.е. последовательно прибавляются все числа от 1 до 100. После подсчета суммы, выводим значение переменной s на экран.

Пример №2: Подсчитать сумму всех четных чисел от 1 до 100.

Решение задачи:

```
Var I,s: integer;
Begin
 S:=0;
```

```
For i:=1 to 100 do
  If i mod 2 = 0 then s:=s+I;
Memo1.Lines.Add(IntToStr(s));
End;
```

Пример №3: Из двухзначных чисел вывести те, сумма цифр которых равна n.

Решение задачи:

```
Var I,N: integer;
Begin
N:=StrToInt(Edit1.Text);
For i:=10 to 99 do
Begin
 If (i mod 10)+(i div 10)=N then Memo1.Lines.Add(IntToStr(i));
End;
End;
```

Задачи:

1. Вводится натуральное число N. Вывести его факториал. (N!=1*2*3…N)
2. Найти все двухзначные числа, которые делятся на n или содержат n.
3. Подсчитать сумму кубов всех нечетных чисел от 1 до 10.

УРОК №10. РЕШЕНИЕ ЗАДАЧ ПО ТЕМЕ "ЦИКЛ С ПРЕДУСЛОВИЕМ И ЦИКЛ С ПОСТУСЛОВИЕМ WHILE"

Цель:

Образовательная: привить навыки решения задач по теме «Цикл с предусловием и с постусловием»,

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

Развивающая: развитие навыков использования циклических алгоритмов при решении задач

Тип занятия: урок формирования новых умений

Форма организации учебного процесса: урок практических занятий

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Повторение теоретического материала.** (Тестовые задания, устный опрос или работа по карточкам)
3. **Рассмотрение примеров.**
4. **Решение задач.**
5. **Домашнее задание:**
6. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Пример №1. Напечатать минимальное число, большее 200, которое нацело делится на 17.

Решение: Присвоим переменной x значение 200 и затем будем его увеличивать до тех пор, пока x не разделится на 17 нацело. Делимость на цело проверяем с помощью операции mod – если остаток от деления на 17 равен 0, значит, число x делится на 17 нацело.

```
Var x:integer;
begin
  X:=200;
  Repeat
     X:=x+1;
  Until x mod 17 =0;
  Memo1.lines.add(inttostr(x))
End;
```

Пример №2. Гражданин 1 марта открыл счет в банке, вложив 1000 руб. Через каждый месяц размер вклада увеличивается на 2 % от имеющейся суммы. Определить, через сколько месяцев размер вклада превысит 1200 руб.

Решение: Первоначально переменная x равна 1000, а количество месяцев - 0. В цикле будем увеличивать значение переменной на 2 процента и одновременно увеличивать значение переменной month (количество месяцев) на единицу.

```
Var x:real; month:integer;
begin
   X:=1000;
   Month:=0;
   While x<=1200 do
   Begin
    X:=x+x*2/100;
    month:= month+1;
   End;
   Memo1.lines.add('Через '+IntToStr(month)+' месяцев');
End;
```

Пример №3. Вводится натуральное число. Определить количество цифр в нем.

Решение: В цикле делим число на 10 до тех пор, пока не дойдем до нуля. При этом, каждый раз деля число на 10, мы откидываем последнюю цифру (например, 256 div 10 = 25), поэтому увеличиваем счетчик на 1.

```
Var x, count : integer;
Begin
  X:=StrToInt(Edit1.Text);
  Count:=0;
  While x>0 do
  Begin
        X:=x div 10;
        inc(Count)
  End;
  Memo1.Lines.Add(IntToStr(Count));
End;
```

Пример №4. Вводится натуральное число. Определить его первую цифру.

Решение: Уменьшаем данное число x до тех пор, пока не останется однозначное число. Его и выводим на экран.

```
Var x : integer;
Begin
  X:=StrToInt(Edit1.Text);
  While x>9 do
  Begin
        X:=x div 10;
  End;
  Memo1.Lines.Add(IntToStr(x));
End;
```

Задание №1: Найти максимальное из натуральных чисел, не превышающих 5000, которое нацело делится на 39.

Задание №2: Вводится натуральное число. Определить сумму его цифр.

Задание №3: Даны натуральные числа n,m . Получить сумму m последних цифр числа n.

УРОК №11. РЕШЕНИЕ ЗАДАЧ ПО ТЕМЕ "ЦИКЛ С ПРЕДУСЛОВИЕМ И ЦИКЛ С ПОСТУСЛОВИЕМ WHILE"

Цель:

Образовательная: привить навыки решения задач по теме «Цикл с предусловием и с постусловием»,

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

Развивающая: развитие навыков использования циклических алгоритмов при решении задач

Тип занятия: урок формирования новых умений

Форма организации учебного процесса: урок практических занятий

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Повторение теоретического материала.** (Тестовые задания, устный опрос или работа по карточкам)
3. **Рассмотрение примера.**
4. **Решение задач.**
5. **Домашнее задание:**
6. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Пример №1. Составить программу для вычисления и вывода значений функции $y = \frac{x^2 - 2x + 2}{x + 5}$ при изменении *x* от –4 до 4 с шагом 0,2.

Решение: Первоначально x присваиваем значение -4, а затем в цикле увеличиваем ее значение на 0,2, до тех пор, пока x меньше или равно 4. В цикле вычисляем очередное значение функции и выводим его на экран.

```
Var y : real; x : Currency;
Begin
   X:=-4;
   While x<=4 do
   Begin
    Y:=(sqr(x)-2*x+2)/(x+5);
    Memo1.Lines.Add('x = ' + FloatToStr(x)+', y = ' + FloatToStr(y));
    X:=x+0.2;
   End;
End;
```

Пример №2. Составить программу для определения K, при котором функция X^k/K становится меньше A , где K =1,2,3…

Решение:

```
Var A, K : integer; Y:real;
begin
A:=StrToInt(Edit1.Text);
K:=0;
Repeat
  K:=K+1;
  Y:=exp(ln(x)*k)/k;
Until Y<A;
Memo1.Lines.Add(FloatToStr(Y));
End;
```

Пример №3. Напечатать 20 первых чисел Фибоначчи. Числа Фибоначчи представляют собой последовательность: 0, 1, 1, 2, 3, 5, 8 … , т.е. каждое число, начиная с третьего, равняется сумме двух предыдущих.

Решение:

```
var A,B,X,T:integer;
Begin
A:=1;
B:=1;
Memo1.Lines.Add(IntToStr(A));
Memo1.Lines.Add(IntToStr(B));
T:=3;
while T<=20 do
```

```
 begin
  X:=A+B;
  Memo1.Lines.Add(IntToStr(X));
  A:=B;
  B:=X;
  T:=T+1;
 end;
End;
```

Задание №1: Вводится некоторое натуральное число N. Выводится N-е число Фибоначчи.

Задание №2: Составить программу для вычисления и вывода значений функции $y = \frac{\sin^2 x^2 - 2x}{\ln x - e^3}$ при изменении ***x*** от 1 до 6 с шагом 0,02.

Задание №3: Вводится натуральное число. Напечатать его цифровой корень. Цифровой корень образуется следующим образом: складываются все цифры числа. Если получено неоднозначное число, то цифры снова складываются и так до тех пор, пока не получим одну цифру. Например, ввели 12345. Сумма цифр=15. Снова складываем. Цифровой корень =6.

УРОК №12. РЕШЕНИЕ ЗАДАЧ ПО ТЕМЕ "ВЛОЖЕННЫЕ ЦИКЛЫ"

Цель:

Образовательная: привить навыки решения задач по теме «Цикл с предусловием и с постусловием»,

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

Развивающая: развитие навыков использования циклических алгоритмов при решении задач

Тип занятия: урок формирования новых знаний и умений

Форма организации учебного процесса: урок практических занятий

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Повторение теоретического материала.** (Тестовые задания, устный опрос или работа по карточкам)
3. **Рассмотрение примеров.**
4. **Решение задач.**
5. **Домашнее задание:**
6. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Циклы можно вкладывать один в другой, аналогично условному оператору.

Рассмотрим вложенные циклы на примерах задач, решаемых методом перебора.

Метод перебора состоит в том, что алгоритм генерирует все возможные комбинации исходных данных и отбирает те из них, которые удовлетворяют заданным условиям. В математике этот метод не получил широкого распространения, к тому же часто применение метода перебора приводит к очень большому количеству вариантов, перебрать которые за обозримое время не может даже современный компьютер. Однако применение этого метода в программировании оправдано тем, что для многих задач получить решение с помощью метода перебора гораздо проще, чем с помощью других методов. Кроме того, можно применять некоторые меры для уменьшения количества перебираемых вариантов за счет исключения заведомо неперспективных.

Пример №1: найдите все натуральные трехзначные числа, каждое из которых удовлетворяет двум следующим свойствам:

1. первая цифра в три раза меньше последней цифры;
2. сумма самого числа с числом, получающимся из него перестановкой второй и третьей цифр, делится на 8 без остатка.

Решение (вариант №1)

```
For i:=1 to 9 do
For j:=0 to 9 do
For k:=0 to 9 do
If (i=3*k) and ((100*i+10*j+k+100*i+10*k+j) mod 8=0)
        then Memo1.Lines.Add(inttostr(i)+inttostr(j)+inttostr(k));
```

Решение (вариант №2)

```
var a,b,c,i:integer;
begin
  for i:=100 to 999 do
  begin
          a:=i div 100;
          b:=(i div 10)mod 10;
          c:=i mod 10;
          if (a=3*c)and((i+a*100+c*10+b)mod 8 = 0) then memo1.Lines.Add(inttostr(i));
  end;
end;
```

Пример №2: Найти все пятизначные числа вида 5m27n (m и n - цифры), которые делятся на 15.

Решение: Легко организовать полный перебор всех возможных вариантов:

```
For n:=0 to 9 do
For m:=0 to 9 do
If (50000+m*1000+270+n) mod 15 = 0
  then Memo1.Lines.Add('5'+IntToStr(n*1000+270+m));
```

Уменьшим перебор. Число делится на 15, если оно делится на 5 и 3 (15=5*3). Число делится на 5, если оно оканчивается на 0 или 5. Следовательно, значениями n могут быть только цифры 0 или 5. Число делится на 3, если сумма его цифр делится на 3. Три цифры числа известны, их сумма равна 5+2+7=14.

- Если n=0, то сумма цифр равна 14, и нужно подобрать такое n, чтобы сумма была кратна 3. Отсюда: значение m изменяется от 1 до 9 с шагом 3.
- Если n=5, то сумма четырех цифр равна 19, следовательно, m изменяется от 2 до 9 с шагом 3.

```
m:=1;
while m<=9 do
begin
 memo1.Lines.Add('5'+inttostr(m)+'270');
 m:=m+3;
end;

m:=2;
while m<=9 do
begin
 memo1.Lines.Add('5'+IntToStr(m)+'275');
 m:=m+3;
end;
```

Задание: Составить программу для решения следующих ребусов:

1. $ABBA=AA^2+BB^2$
2. КИО*ИО=ТОКИО.

В каждый ребус вместо каждой буквы подставить некоторую цифру, причем одинаковым буквам должны соответствовать одинаковые цифры, а различным буквам – различные цифры. Найти все подходящие числа.

УРОК №13. ОДНОМЕРНЫЕ МАССИВЫ

Цель:

Образовательная: Рассмотреть диапазонный тип данных. Объяснить понятие одномерного массива. Рассмотреть способы ввода данных в массив и вывода данных из массива на экран. Обращение к элементам массива.

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

Развивающая: развитие познавательных способностей

Тип занятия: урок формирования новых знаний

Форма организации учебного процесса: урок-лекция

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Изложение нового материала:**
   - Одномерные массивы.
   - Способы ввода и вывода одномерного массива.
   - Пример использования массива.
3. **Решение задач.**
4. **Домашнее задание:** повторить теоретический материал

УЧЕБНЫЙ МАТЕРИАЛ

Диапазонный тип данных

Диапазонный тип данных определяет подмножество значений из перечисляемых типов, т.е. относящихся к целочисленным, символьным, булевым или перечислимым типам.

Общий вид записи:

Type имя_типа = Low .. High;

Имя_типа – идентификатор типа, Low и High – идентификаторы первого и последнего значения подмножества.

Пример:

```
Type
  SubChar = 'a'..'d';
  SubInteger = 1..100;
```

В приведенном примере создается диапазонный тип SubChar, переменные которого миогут принимать значения от a до d, и тип SubInteger, переменные которого могут принимать значения от 1 до 100. Попытка присвоить переменным этих типов значения, выходящие из определенного в типе диапазона, вызовет ошибку.

Диапазонный тип данных можно создать на основе перечислимого типа. Можно определить диапазонный тип, содержащий только те значения, которые характеризуют рабочее состояние светофора:

```
Type TrafficLightColor = Red .. Green;
```

Все переменные этого типа могут иметь только три значения: Red, Yellow или Green.

Одномерные массивы

Многие задачи, которые решаются с помощью компьютера, связаны с обработкой больших объемов информации. Для удобства обработки информация часто сводится в таблицу.

Каждому значению или элементу таблицы соответствует его порядковый номер, и следовательно, при задании порядкового номера, становится ясно, о каком элементе таблицы идет речь.

1	2	3	4	5
32	25	16	65	81

Данная таблица имеет 5 элементов. Каждый элемент имеет свой порядковый номер, например, элемент 81 имеет порядковый номер 5, а элемент под порядковым номером 2 имеет значение 25.

Очевидно, что при хранении таблицы порядковые номера хранить не требуется: зная начало нумерации, можно путем отсчета найти любой элемент. Кроме того, полезно знать и самый большой порядковый номер, так как это позволяет определить заранее размер таблицы.

Таким образом, для задания таблиц требуется задать тип элементов, ее имя, начальный и конечный порядковые номера.

При обработке больших объемов информации с помощью компьютера также используются таблицы, в программировании их называют массивами.

Массив – это ряд ячеек памяти, отведенных для хранения значений индексированной переменной (индексированные переменные – x_2, a_i).

Если какая-либо переменная в программе является массивом, то её необходимо описать в разделе переменных.

Для работы с одномерными массивами используется специальный тип данных – регулярный, который описывается в разделе описания переменных Var. Для описания массивов используется служебные слова Array, Of.

Общая форма описания одномерного массива:

Имя переменной: Array [1..100] of <тип компонентов массива>

Пример: **Var** massiv: array[1..10] of integer;

В описании массивов начальное и конечное значение индексов должны быть const.

Для обращения к элементам массива используется следующая форма записи элемента: a[номер], где номер – это порядковый номер элемента в массиве.

Ввод массива. Заполнение одномерного массива случайными числами. Для ввода произвольных значений в массив можно использовать генератор случайных чисел Random(W). Эта функция генерирует случайные числа в диапазоне от 1 до W-1.

```
const W=20;N=5;
var a:array [1..N] of integer; i:integer;
begin
    for i:=1 to N do a[i]:=random(W);
end;
```

Вывод одномерного массива в компонент Memo.

```
var a: array [1..5] of integer; i: integer;
Begin
       For i:=1 to 5 do Memo1.Lines.Add(IntToStr(a[i]));
end;
```

Пример №1: В зоопарке живет три удава. Известна длина каждого удава в сантиметрах (500, 400 и 600). Какая длина получится у трех удавов, вытянутых в одну линию.

Решение: обозначим длину первого удава через dlina[1], второго – dlina[2], третьего - dlina[3]. Отведем по жути переменные массив:

Var dlina : array [1..3] of Integer;

Программа полностью:

```
Var dlina : array [1..3] of Integer; summa:integer;
Begin
{в этот момент в ячейках памяти уже находятся числа и с ними можно выполнять арифметические действия}
  Summa: = dlina[1]+dlina[2]+dlina[3];
  ShowMessage(inttostr(summa));
End;
```

Теперь запишем эту же программу, в предположении, что длины удавов заранее неизвестны и мы их вводим при помощи компонентов Edit:

```
Var dlina : array [1..3] of Integer; summa:integer;
Begin
  Dlina[1]:= StrToInt(Edit1.Text);
  Dlina[2]:= StrToInt(Edit2.Text);
  Dlina[3]:= StrToInt(Edit3.Text);
  Summa: = dlina[1]+dlina[2]+dlina[3];
  ShowMessage(inttostr(summa));
End;
```

Теперь решим эту же задачу в предположении, что удавов не 3, а 1000.

```
Var dlina : array [1..1000] of Integer; summa, i:integer;
Begin
For i:=1 to 1000 do
 Begin
  dlina[i]:=StrToInt(InputBox('Ввод массива', 'Введите'+IntToStr(i)+'-ый элемент','2'));
 End;
Summa: = 0;
For i:=1 to 1000 do summa:= summa + dlina [i];
ShowMessage(inttostr(summa));
End;
```

Функция InputBox:

InputBox(Текст заголовка окна', 'текст в окне','значение по умолчанию')

Выводит на экран окно ввода значения с заголовком, указанном в первом параметре, текст в окне задан с помощью второго параметра, а значение, которое установлено по умолчанию, указывается с помощью третьего параметра.

Задание №1. Составить программу, которая выводит четвертый элемент массива.

Пример №2. Составить программу вычисления первых 30 чисел Фибоначчи. Эти числа определяются так: A[1]=1, A[2]=1, а каждое следующее число равно сумме двух предыдущих.

Решение:

```
Var A: Array [1..30] of integer;
Begin
 A[1]:=1; A[2]:=1;
 For i:=3 to 30 do A[i]:=A[i-1]+A[i-2];
 For i:=1 to 30 do Memo1.Lines.Add(inttostr(A[i]));
End;
```

Задание №2. Определить для заданного массива A сумму положительных, количество нулевых, произведение отрицательных элементов.

УРОК №14. ДВУМЕРНЫЕ МАССИВЫ

Цель:

Образовательная: Объяснить понятие двумерного массива. Рассмотреть способы ввода данных в массив и вывода данных из массива на экран. Обращение к элементам двумерного массива.

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

Развивающая: развитие познавательных способностей

Тип занятия: урок формирования новых знаний

Форма организации учебного процесса: урок-лекция

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Изложение нового материала:**
   - Двумерные массивы.
   - Компонент StringGrid.
   - Способы ввода и вывода двумерного массива.
   - Пример использования массива.
3. **Домашнее задание:** повторить теоретический материал

УЧЕБНЫЙ МАТЕРИАЛ

Компонент StringGrid используется для работы с массивами, причем элементы массива должны иметь строковый тип. StringGrid - таблица строк, в ячейках которой располагаются произвольные текстовые строки.

Главное свойство компонента - Cells - двухмерный массив ячеек, каждая из которых может содержать произвольный текст. Свойство Cells имеет тип String, поэтому программа может легко прочитать или записать содержимое нужной ячейки. Например:

Cells [1,1] := 'Левая верхняя ячейка рабочей зоны';

Свойства компонента:

Cells[ACol, ARow: Integer] :String;	Определяет содержимое ячейки с табличными координатами (ACol, ARow)
ColCount	Содержит количество столбцов таблицы
DefaultColWidth	Содержит значение ширины столбца
FixedCols	Определяет количество столбцов фиксированной зоны
FixedRows	Определяет количество рядов фиксированной зоны
RowCount	Содержит количество рядов таблицы

Свойство **Options** имеет подсвойства, с помощью которых можно управлять свойствами таблицы:

goEditing	Разрешено редактирование ячейки. Игнорируется, если включен элемент goRowSelect. Редактирование начинается после щелчка мыши или нажатия клавиши F2 и завершается при щелчке по другой ячейке или нажатии клавиши Enter

Двумерный массив представляет собой таблицу из нескольких строк и столбцов. Если массив двумерный, то для определения положения какого-либо элемента достаточно задания номера строки и столбца, на пересечении которых находиться рассматриваемый элемент. Так выделенные на рис. 1 в элементы массива имеют следующие обозначения B[1,1], B[1,4], B[5,2].

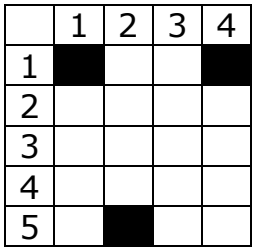


Рисунок 1

Рассмотрим массив размерностью NxN для N=3. Здесь количество строк и столбцов совпадает, поэтому такой массив называют квадратным.

$$a_{11} \quad a_{12} \quad a_{13}$$
$$a_{21} \quad a_{22} \quad a_{23}$$
$$a_{31} \quad a_{32} \quad a_{33}$$

Главной диагональю двухмерного массива называют прямую, мысленно проведенную из левого верхнего угла в правый нижний. В нашем примере элементы главной диагонали имеют вид: a_{11}, a_{22}, a_{33}.Если индекс строк обозначить i, а индекс столбцов – j, то в общем случае факт принадлежности элемента главной диагонали записывается так: i=j или так: i-j=0.

Для элементов, расположенных выше главной диагонали, справедливо соотношение индексов i<j.

Для элементов, расположенных ниже главной диагонали, справедливо соотношение индексов i>j.

Индексы элементов, расположенных на линиях, параллельных главной диагонали, удовлетворяют соотношениям |i-j|=k, где k – номер линии по отношению к главной диагонали.

Линию, соединяющую правый верхний угол массива с левым нижним, называют *побочной диагональю*. Для индексов элементов побочной диагонали a_{31}, a_{22}, a_{13}, справедливо соотношение i+j=n+1, где n – количество строк и столбцов массива (в примере n=3). Для индексов элементов расположенных выше побочной диагонали, справедливо соотношение i+j<n+1, а для индексов элементов, расположенных ниже побочной диагонали, справедливо соотношение i+j>n+1.

Общая форма описания двумерного массива:

Имя переменной: Array [1..100, 1..50] of <тип компонентов массива>

Пример: Var A: array [1..4,1..4] of real;

Для обращения к элементам массива используется следующая форма записи элемента: a[i,j].

Ввод массива

1. Ввод элементов двумерного массива с помощью функции InputBox.

```
const N=5;
var a:array[1..N,1..N] of integer;  i,j:integer;
begin
 for i:=1 to N do begin
 for j:=1 to N do begin
   a[i,j]:=StrToInt(InputBox('Ввод данных','Введите '+IntToStr(i)+'-ый элемент массива','2'));
   StringGrid1.Cells[j,i]:=IntToStr(a[i,j]);
```

```
  End; end;
end;
```

2. Ввод элементов двумерного массива из компонента StrinGrid (FixedCols=0, FixedRows=0, ColCount=5, RowCount=5)

```
const N=4;
var a:array[0..N,0..N] of integer; i,j:integer;
begin
 for i:=0 to N do begin
 for j:=0 to N do  begin
   a[i,j]:=StrToInt(StringGrid1.Cells[j,i]);
  end; end;end;
```

Вывод массива

1. Вывод двумерного массива в компонент StringGrid (FixedCols=0, FixedRows=0, ColCount=5, RowCount=5)

```
const N=4;
var a:array[0..N,0..N] of integer; i,j:integer;
begin
 for i:=0 to N do begin
 for j:=0 to N do  begin
   StringGrid1.Cells[j,i]:=IntToStr(a[i,j]);
  end; end;end;
```

Пример №1. Составить программу, выводящую на экран таблицу умножения.

```
var a:array [1..9,1..9] of integer;
i,j:integer;
begin
 for i:=1 to 9 do
 for j:=1 to 9 do
 a[i,j]:=i*j;

 for i:=1 to 9 do
 for j:=1 to 9 do
 StringGrid1.Cells[j,i]:=inttostr(a[i,j]);
End;
```

Задача №2. В данном двумерном массиве размерностью 5X5 поменять столбцы с номерами P и Q местами.

Задача №3. Дан двумерный массив. К элементам четных строк прибавить элемент первой строки соответствующего столбца. Из элементов нечетных столбцов вычесть элемент последнего столбца соответствующей строки.

Задача №4. В двухмерном массиве хранится информация о зарплате 18 сотрудников фирмы за каждый месяц года (в первом столбце – за январь, во втором – за февраль, и т.п.). Определить среднюю зарплату за каждый месяц.

УРОК №15. ТИПОВЫЕ АЛГОРИТМЫ ОБРАБОТКИ МАССИВОВ

Цель:

Образовательная: Рассмотреть типовые алгоритмы обработки одномерных и двумерных массивов: поиск элемента, перебор элементов, поиск элемента, удовлетворяющего некоторому условию, сортировка массива.

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

Развивающая: развитие познавательных способностей

Тип занятия: урок формирования новых навыков

Форма организации учебного процесса: урок практических работ

Литература: Ю.А.Аляев, В.П.Гладков, О.А.Козлов, Практикум по алгоритмизации и программированию на языке Паскаль, Москва, Финансы и статистика, 2004

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Решение задач**
3. **Домашнее задание:** повторить теоретический материал, решение задач

УЧЕБНЫЙ МАТЕРИАЛ

Вопросы для повторения по одномерным массивам:

1. Верно ли, что все элементы массива должны быть одного типа?
2. В чем преимущество объединения отдельных элементов в массив?
3. Что предпринимают, чтобы найти отдельный элемент массива?
4. Возможно ли обращение к пятому элементу массива, минуя предыдущие элементы?
5. В каком случае можно сказать, что два одномерных массива равны?
6. Каким образом можно поменять местами значения двух одномерных массивов?

Вопросы для повторения по двумерным массивам:

1. В чем разница между одномерным и двумерным массивами?
2. Можно ли обратится к элементу двухмерного массива, используя только один индекс?
3. Какой индекс при обращении к элементу двухмерного массива считается индексом строки, а какой – столбца?
4. Как можно переставить столбцы двухмерного массива?

Пример №1. Сформировать двухмерный массив nxn указанного вида, для произвольного n. Для n=4 массив имеет вид:

0	1	0	2
3	0	4	0
0	5	0	6
7	0	8	0

Решение. Ненулевое значение имеют элементы, сумма индексов которых нечетна. Для формирования значения можно использовать дополнительную переменную. Для перебора элементов будем использовать два вложенных цикла.

```
var a:array [1..6,1..6] of integer;
k,i,j:integer;
begin
 k:=1;
 for i:=1 to 6 do
 for j:=1 to 6 do
 begin
  if (i+j)mod 2 =1 then begin a[i,j]:=k; k:=k+1;end
  else a[i,j]:=0;
 end;

 for i:=1 to 6 do
 for j:=1 to 6 do
 StringGrid1.Cells[j,i]:=inttostr(a[i,j]);
end;
```

Решение задач

1. Дан массив A(10X10). Найти максимальный элемент массива.
2. Дан массив A(10X10). Найти сумму элементов массива.
3. Дан массив A(10X10). Вывести на экран элементы главной диагонали.
4. Дан массив A(10). Отсортировать массив по возрастанию. Использовать метод прямого выбора

Решение

```
var i,max,k,j:integer; a:array [1..10] of integer;
begin
    for i:=1 to 10 do  a[i]:=random(9)+1;
    for i:=1 to 10 do memo1.Lines.Add(inttostr(a[i]));

    For i:=10 downto 2 do
    Begin
     K:=I;
```

```
        Max:=a[i];
        For j:=1 to i do
              if a[j]>max then Begin K:=j; Max:=a[k];  End;
        If i<>k then Begin A[k]:=a[i]; A[i]:=max; End;
       end;
       for i:=1 to 10 do memo2.Lines.Add(inttostr(a[i]))
   end;
```

Самостоятельное решение задач

1. Дан массив A(10X10). Найти минимальный элемент массива.
2. Дан массив A(10X10). Найти сумму положительных элементов массива, произведение отрицательных элементов массива и количество элементов равных нулю.
3. Дан массив A(10X10). Вывести на экран элементы побочной диагонали.

Дополнительный материал

Задача 1. Вычислить сумму нескольких произвольных элементов массива, индексы которых вводятся с клавиатуры. Признаком окончания ввода является нуль.

В этом и следующих примерах будем считать, что описан одномерный массив с нижней границей индекса 1 и верхней — nn. В массиве используются только n элементов.

```
Const nn=50;
type mas=array  [1..nn]  of real;
var a:mas;   {исходный массив}
n:integer;     {количество     используемых элементов <=nn}
i,j,k:integer;   {индексы массива)
```

Решение. Пусть n=5 и массив a содержит следующие значения:: a[1]=-3, a[2]=17, a[3]=4, a[4]=-5, a[5]=81. Последовательность ввода имеет вид: 3,1,10,3,5,0. В этом случае искомая сумма вычисляется в следующем порядке:

1. s:=0;
2. s:=s+a[3]=0+4=4;
3. s:=s+a[1]=4+(-3)=1;
4. сообщение пользователю «В массиве нет элемента с индексом 10»;
5. s:=s+a[3]=1 + 4=5;
6. s:=s+a[5]=5 + 81=86;
7. 0 — конец ввода.

В результате получена сумма s=86.

Организуем цикл ввода элементов до нуля. В теле цикла будем поверять допустимость индекса и находить сумму или выдавать сообщение о неправильном индексе. Получим фрагмент алгоритма на языке Паскаль:

```
{Ввод массива}
Write ('Введите используемое количество элементов ');
Readln(n);
Write('Введите ',n,' элементов массива ');
for i:=1 to n do read(a[i]);
{вычисление суммы}
s:=0;
write('Введите индекс ');
read(i);
while i<>o do
begin
       if (K=i) and (i<=n) then s:=s+a[i]
                           else writeln{'В массиве нет элемента с индексом ', i) ;
       write{'Введите индекс ');
       read(i);
end;
{Вывод ответа}
writeln('s=' , s) ; .
```

Задача 2. Для каждого элемента массива просмотреть все последующие элементы. Здесь возможно несколько вариантов.

Вариант 1. Массив просматриваем по одному элементу слева направо. Подмассив просматриваем по одному элементу тоже слева направо. Обозначим: i — индекс массива, а j —

индекс подмассива, тогда схема примет вид:

```
for i:=1 to n-1 do  {у последнего элемента нет последующего}
for j:=i+1 to n do    {обработка a[j]}.
```

Вариант 2. Подмассив просматриваем справа налево.

```
For i:=1 to n-1 do
Begin
        J:=n;
        While j>I do
        Begin
                {обработка a[j]}
                J:=j-1;
        End;
End.
```

Могут быть использованы и схемы перебора парами, соседями и т.д.

Задача 3. Для каждого элемента просмотреть все предшествующие ему элементы. Пусть i - индекс массива, а j – индекс подмассива.

Решение.

Случай 1. Основной массив просматривается слева направо, подмассив просматривается так же.

```
for i:=2 to n do {у первого элемента нет предыдущего}
 for j :=1 to i-1 do {обработка a[j]}
```

Случай 2. Основной массив просматривается слева направо, а подмассив — справа налево.

```
for i:=2 to n do
begin
        j:=i-1;
        while j>0 do
        begin
           {обработка a [ j]}
            J:=j-1;
        end;
end.
```

Случай 3. Основной массив просматривается справа налево, а подмассив — слева направо.

```
i:=n;
while i>1 do
begin
        for j:=1 to i-1 do  {обработка a[j]}
        i:=i-1;
end.
```

Случай 4. Основной массив просматривается справа налево, подмассив просматривается так же.

```
i:=1;
while i>1 do
begin
        j:=i-1;
        while j>0 do
        begin
                {обработка a[j]}
                j:=j-1;
        end;
        i:=i-1;
end.
```

Задача 4. Найти сумму (произведение) элементов одномерного массива.

Решение. Каждый элемент массива прибавляется к сумме s:=s+a[i] или умножается на произведение предыдущих элементов p:=p*a[i]. Поскольку требуется перебрать все элементы массива, выберем схему перебора элементов по одному, двигаясь с начала или с конца массива.

{сумма элементов массива}	**{произведение элементов массива}**
s:=0; i:=1;	p:=1; i:=n;

while i<=n do
begin
  s:=s+a[i];
  i:=i+1;
end.
	while i>0 do
	begin
	  p:=p*a[i];
	  i:=i-1;
	end.

Задача 5. Задан одномерный массив. Нужно переставить элементы в обратном порядке. Другой массив использовать не разрешается. Для массива a={1,2,3,4,5,6,7,8,9} получим результат: a={9,8,7,6,5,4,3,2,1}.

Решение первое. При преобразовании меняются местами два элемента: первый и последний, второй и предпоследний и т. д. В случае нечетного количества элементов имеется центральный элемент, который должен остаться на месте. Меняются элементы, симметричные относительно центрального. Индексы каждой пары таких элементов определим по формулам: i и n+1-i. Всего в массиве должно произойти n div 2 обменов. Последняя фраза легко переводится на язык Паскаль: организовать арифметический цикл со счетчиком до n div 2, в теле которого меняются местами элементы a[i] и a[n+1-i].

```
for i:=1 to n div 2 do
begin
        r:=a[i];
        a[i]:=a[n+1-i];
        a[n+1-i]:=r;
end.
```

Решение второе. Введем обозначения для пары меняемых элементов: i — элемент, входящий в пару и расположенный в начале массива, j — соответствующий элемент пары, расположенный в конце массива. При таком соглашении i должно быть меньше j. Если i=j, то они указывают на один и тот же элемент, который можно не менять. Если i>j, то пар больше нет. Таким образом имеем итерационный цикл, который выполняется до тех пор пока есть пары элементов. В теле цикла a[i] и a[j] меняются местами.

```
i:=1;  j:=n;
while i<j do
begin
        r:=a[i];
        a[i]:=a[j];
        a[j]:=r;
        i:=i+1;
        j:=j-1;
end.
```

Задача 6. Из массива a получить такой массив b, в котором элементы расположены в таком порядке: $\mathbf{a_n a_1 a_{n-1} a_2 a_{n-2} a_3 \ldots}$

Решение. Элементы в массиве b чередуются следующим образом. На первом месте стоит последний элемент из массива a, на втором — первый, на третьем — предпоследний, т.е. на нечетных местах располагаются элементы с конца массива a в порядке убывания индексов. На четных местах в массиве b располагаются элементы с начала массива a в порядке возрастания индексов. Выбираем схему перебора по элементам массива b (цикл по выходному массиву). Элементы массива a перебираем по схеме «от обоих концов к середине». Запишем это решение на языке Паскаль:

```
i:=1;   {индексы массива a}
j:=n;
for k:=1 to n do
        if k mod 2=0 then begin b[k]:=a[j]; j:=j-1; end
                     else begin b[k]:=a[i]; i:=i+1; end.
```

Задача 7. В заданном одномерном массиве все нулевые элементы переписать в конец массива. Для массива получим a={1,0,2,3,0,4,5,0,0,0,6} a={1,2,3,4,5,6,0,0,0,0,0}.

Решение. Воспользуемся услугами вспомогательного массива, в который вначале перепишем ненулевые элементы, оставшуюся часть вспомогательного массива заполним нулями. Затем перепишем содержимое вспомогательного массива в исходный массив. Соответствующий вариант решения приведен ниже. Здесь: a - исходный массив, i - индекс массива a, b — вспомогательный массив, j — его индекс. Размерность обоих массивов - n.

```
Const n=11;
```

```
Var a,b:array[1..n] of integer;
Begin
…
j:=1; {указывает номер первой свободной позиции в массиве b}
{переписываем все ненулевые элементы из b в a}
for i:=1 to n do
if a[i]<>0 then begin b[j]:=a[i]; j:=j+1; end;
{заполняем остаток массива b нулями}
for i:=j  to n do b[i]:=0;
{переписываем массив b в a}
a:=b;
…
```

Главным недостатком полученного решения является применение вспомогательного массива — нерациональное использование оперативной памяти компьютера. Для того чтобы обойтись без него, запишем ненулевые элементы в начало исходного массива. Место записи элемента указывает переменная j, которая передвигается по массиву только после записи ненулевого элемента. При найденном нуле значение переменной j не изменяется. Переменная i будет указывать на проверяемый элемент массива, т.е. для одного массива используются два индекса: индекс i перебирает элементы массива a как входного (исходного), а индекс j — как выходного. Результат получается сразу в массиве a. Соответствующий вариант программы приведен ниже:

```
j:=1;
for i:=1 to n do
if a[i]<>0 then begin a[j]:=a[i];j:=j+1; end;
for i:=j  to n do a[i]:=0;
```

Задача 8. Провести поиск среди элементов одномерного массива элемента, равного заданному аргументу поиска a.

Существует несколько вариантов решения задачи.

Вариант 1 (линейный поиск). Если нет дополнительной информации о разыскиваемых данных, то остается последовательный просмотр массива с увеличением шаг за шагом той его части, где искомого элемента не обнаружено. Такой метод называется линейным поиском. Исходные данные для решения задачи — массив чисел. В результате поиска получаем либо номер элемента, совпадающего с a, либо ответ "Такого элемента нет". Существуют 2 варианта окончания поиска: 1) закончились элементы массива, а нужный элемент не найден; 2) найден нужный элемент.

Первый вариант окончания поиска определяется по истинности условия i>n, где i — текущий элемент массива, n - количество элементов в массиве.

Второй вариант окончания поиска определяется по истинности логической переменной f, где true соответствует найдено, a false — не найдено.

Таким образом, условие окончания поиска записывается в виде (i>n) or f, что соответствует словесному описанию «Просмотрены все элементы или не найдено». Просмотр элементов массива в цикле будет выполняться в случае ложности приведенного условия. Найдем его отрицание согласно закону де Моргана: отрицание дизъюнкции равно конъюнкции отрицаний. not((i>n) or f)=not(i>n) and not f=(i<=n) and not f.

Программа на языке Паскаль:

```
{линейный поиск элемента в одномерном массиве}
const  nn=12;   {константа  задает максимальный размер массива}
type mas1=array[1..nn]   of integer;   {тип массива}
var b:mas1;   {исходный массив}
n:integer;   {размер массива}
a:integer;   {число для поиска}
i:integer;   {индекс массива}
f:boolean;   {истина,  если искомый элемент найден}
begin
        write('Введите размер массива от 1 до  ',nn);
        repeat {вводим n до тех пор,}
                readln(n); {пока оно не попадет}
        until (1<=n) and (n<=nn); {на отрезок от 1 до nn}
        {Вводим элементы массива}
        for i:=1 to n do
```

```
        begin
                write('Введите элемент массив B[',i:2, ']=');
                readln(b[i])
        end;
write('Введите искомый элемент  ');
readln(a) ;
i:=1; {начинаем просмотр с первого элемента}
f:=false; {пока не найдено}
while (i<=n) and not f do
{просматриваем массив, пока есть элементы и не найден искомый}
        if b[i]=a {если совпали,}
        then f:=true {то нашли,}
        else i:=i+1; {иначе перейти к следующему элементу}
        if f  {вывод результатов поиска}
        then write('Номер найденного элемента ',i)
        else write('Не нашли');
end.
```

Тестирование.

Тест 1. n=8, B[1]=2, B[2]=8, B[3]=3, B[4]=1, B[5]=9, B[6]=2, B[7]=2, B[8]=2, a=5. Результат поиска: «Не нашли».

Тест 2. n=7, B[1]=2, B[2]=8, B[3]=3, B[4]=1, B[5]=9, B[6]=8, B[7]=2, a=8. Результат поиска: «Номер найденного элемента 2».

Вариант 2 (линейный поиск с барьером). Применяется широко распространенный прием фиктивного элемента, или барьера, расположенного в конце массива. Это позволяет упростить условие окончания цикла, так как заранее ясно, что хотя бы один элемент, равный a, в массиве есть.

```
Program POISK1;
{поиск с барьером в одномерном массиве}
const nn=12;   {константа  задает максимальный размер массива}
type mas1=array[1..nn] of integer; {тип массива}
Var b:mas1; {исходный массив}
n:integer; { размер массива }
a: integer; { число для поиска }
i: integer; { индекс массива }
begin
        write('Введите размер массива от 1 до ',nn-1);
        repeat  {вводим n до тех пор,}
                readln(n);   { пока оно не попадет }
        until (1<=n) and (n<=nn); {на отрезок от 1 до nn-1}
        {Вводим элементы массива}
        for i:=1 to n do
        begin
                write ('Введите элемент массива B [', i : 2, ' ]=');
                readln(b[i]);
        end;
        write('Введите искомый элемент  ');
        readln(a);
        b[n+1]:=a; {добавили барьер, равный a,  в конец массива}
        i:=1;   {начинаем просмотр с первого элемента}
        while b[i]<>a do i:=i + 1; {просматриваем массив,  пока не найдем}
        if i<=n  {вывод результатов поиска}
        then write('Номер найденного элемента ',i)
        else write('Не нашли');
end.
```

Тестирование.

Тест 1. n=8, B[1]=2, B[2]=8, B[3]=3, B[4]=1, B[5]=9, B[6]=2, B[7]=2, B[8]=2, B[9]=5, a=5. Результат поиска: «Не нашли».

Тест 2. n=7, B[1]=2, B[2]=8, B[3]=3, B[4]=1, B [5] =9, B [6] =8, B [7] =2, B [8] =8, a=8. Результат поиска: «Номер найденного элемента 2».

Вариант 3 (поиск методом деления пополам, или двоичный поиск). Предварительно

упорядочим (отсортируем) массив: делим его пополам и для сравнения выбираем средний элемент. Если он совпадет с искомым, то поиск заканчивается. Если средний элемент меньше искомого, то все элементы, расположенные левее, также будут меньше искомого. Следовательно, их можно исключить из дальнейшего поиска, оставив только правую часть массива. Если средний элемент больше искомого, то отбрасывается правая часть. Процедура деления массива пополам и отбор части массива для дальнейшего поиска продолжаются до тех пор, пока искомый элемент не будет найден или длина части массив не станет равной нулю. Каждый раз величина той части массива, где ведется поиск, уменьшается вдвое. Поэтому максимальное число требующихся сравнений равно] log2N [, где N — количество элементов в массиве.

Рассмотрим пример двоичного поиска.

Искомый элемент a=7. Исходный массив n=16.

Первый шаг поиска:

номер элемента:! 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16;
массив: B={1,3,5,6,7,10,12,14,15,16, 20,21,23,25,26,30} ;
начальная граница поиска: L=1,
конечная граница поиска: r=16;
середина части поиска: i= (L+R) div 2 = 8;
поскольку B[8]=14>a=7, то отбрасываем правую часть: L=1, R=i-1=7.

Второй шаг поиска:

номер элемента: 12 3 4 5 6 7;
массив: B={1, 3, 5, 6, 7,10,12} ;
начальная граница поиска: L=1, конечная граница поиска: R=7;
середина части поиска: i= (L+R) div 2 = 4;
поскольку B[4]=6<a=7, то отбрасываем левую часть: L=i+1=5, R=7.

Третий шаг поиска:

номер элемента: 5 6 7;
массив: B={7,10,12} ;
начальная граница поиска: L=5,
конечная граница поиска: R=7;
середина части поиска: i=(L+R) div 2 = б;
поскольку B[6]=10>a=7, то отбрасываем правую часть: L=5, R=i-1=5.

Четвертый шаг поиска:

номер элемента: 5;
массив: B={7};
начальная граница поиска: L=5,
конечная граница поиска: R=5;
Середина части поиска: i=(L+R) div 2 = 5;
Поскольку B [5] =7=a=7, то искомый элемент найден в позиции 5.

Фрагмент программы, реализующей двоичный поиск:

```
L:=1; r:=n;
F:=false;{не найдено}
While (L<=r) and not f do
Begin
        i:=(L+r) div 2;
        If b[i]=a then f:=true;{нашли}
        Else if b[i]<a then L:=L+1 else r:=i-1;
End.
```

Тестирование.

Тест 1. a=7, n=15,
b={1,3,5,7,9,10,12,14,16,18,21,23,25,27,29}.
Ответ: искомый элемент найден в позиции 4 .

Тест 2. a=6, n=15, b={1, 3, 5, 7, 9,10,12,14,16,18, 21, 23,25,27,29}.
Ответ: искомый элемент не найден.

УРОК №16. СТРОКОВЫЙ ТИП ДАННЫХ

Цель:

Образовательная: изучить строковый тип данных, основные процедуры и функции для обработки строк (копирование, удаление, определение позиции подстроки в строке, определение длины строки). Рассмотреть примеры обработки строк.

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

Развивающая: развитие познавательных способностей

Тип занятия: урок изучения нового материала

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Изучение нового материала**
3. **Рассмотрение примеров**
4. **Выполнение заданий**
5. **Решение задач**
6. **Подведение итогов урока.**

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Проблема: Пользователь вводит 2 строки. Заменить в строке, полученной из двух строк, все буквы ‘а’ на буквы ‘А’ и вывести полученную строку на экран.

Переменная строкового типа должна содержать строку. Разберем процедуры и функции для работы со строками.

Процедуры

Delete(S, N, M);	Удаляет M символов из строки S, начиная с позиции N.
Insert(SubS, S, N);	Вставляет подстроку SubS в строку S, начиная с позиции N.

Функции

Chr(X): Char;	Возвращает символ с заданным порядковым номером X.
Ord(X): LongInt;	Возвращает порядковый номер символа X в таблице кодов символов.
Concat(S1[,S2,…,SN]): String;	Выполняет сцепку (конкатенацию) последовательности строк.
Copy(S, N, M): String ;	Возвращает подстроку из строки S, начиная с позиции N и длиной M символов.
Length(S): Byte;	Возвращает количество символов в строке S.
Pos(SubS, S): Byte;	Возвращает номер позиции, начиная с которой в строке S располагается подстрока SubS (если значение функции равно нулю, то S не содержит SubS).

Примеры:

Исходные данные	Операция	Результат	Пояснение
S1:=’Мото’; S2:=’Роллер’	S3:=s1+s2	S3=’мотороллер’	Операция + над двумя строками соединяет две строки в одну
S5:=’Мотороллер’	K:=pos(‘рол’,S5)	K=5	Функция Pos возвращает позицию, на которой находится строка ‘рол’ в строке S5.
S3:=’Мотороллер’	I:=Length(S3)	I=10	Функция Length (длина) возвращает количество символов в строке
S3:=’астроном’	S4:=Copy(S3, 3,	S4=’трон’	Функция Copy возвращает часть

	4)		строки длиной 4 начиная с третьего символа.
S5:='Коробочка'	Delete(s5, 4, 2)	S5='Корочка'	Процедура Delete удаляет из строки S5 два символа начиная с четвертого.
S6:='Рука'; S7:='баш';	Insert(S7, S6, 3)	S6='Рубашка'	Процедура Insert вставляет в строку s6 строку s7 начиная с третьего символа.

Задание №1: Даны слова: S1='шар', S2='ада', S3='ик'. Комбинируя операции копирования и соединения, получить из одних текстов другие.

Задание	Ответ
S1+s2	'шар'+'ада'='шарада'
S1+s3	'шар'+'ик'='шарик'
Copy(s1,3,1)+s3+copy(s1,1,2)	'р'+'ик'+'ша'='рикша'
Copy(s1,3,1)+s2+copy(s1,3,1)	'р'+'ада'+'р'='радар'

Задание №2: Даны слова: s1='ку', s2='шка', s3='юшон', s4='пила'. Написать конструкцию из этих слов образующих фразу 'кукушка кукушонку купила капюшон'.

Ответ:

```
S:=s1+s1+s2+'`+s1+s1+copy(s3,2,3)+s1+'`+s1+s4+'`+copy(s2,2,2)+copy(s4,1,1)+s3
```

Если задана строка s='Банка', то считается автоматически заданным массив символов с тем же именем: s[1]='Б', s[2]='а',s[3]='н',s[4]='к',s[5]='а'. Тогда после выполнения оператора s[3]:='р' мы получим s='Барка'.

Сравнение строк

Строки можно сравнивать. Условие s1=s2 считается выполненным, если обе строки абсолютно одинаковы, включая и пробелы. Сравнение идет посимвольно слева направо. Поэтому считается, что 'панк'<'парк', так как первый несовпадающий символ 'р' имеет больший номер, чем 'н'.

Пример №1. Составить программу замены в данной строке всех букв 'а' на букву 'А'.

Решение: Методом перебора каждого символа строки проверяем, является очередной символ символом 'а', и если является, то заменяем его на 'А'.

```
Var s:string; i:integer;
Begin
S:=Edit1.Text;
For i:=1 to length(s) do
 If s[i]='а' then s[i]:='А';
Edit2.text:=s;
End;
```

Задание №1: Написать программу подсчета количества букв Н в тексте.

Задание №2: Составить программу, удваивающую каждую букву в заданном тексте.

ЛЕКЦИЯ №17. ТИПОВЫЕ АЛГОРИТМЫ ОБРАБОТКИ СТРОК

Цель:

Образовательная: получить навыки решения задач обработки строк.

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

Развивающая: развитие практических навыков решения задач по теме «Строковый тип данных».

Тип занятия: урок формирования новых умений

Форма организации учебного процесса: практическая работа

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Актуализация знаний.** Повторение теоретического материала
3. **Решение задач**
4. **Подведение итогов урока.**

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Пример №1. Составить программу, которая подсчитывает количество слов в строке. Учесть, что слова могут разделяться не одним пробелом, а несколькими.

Решение:

```
Var s:string; k, i:integer;
Begin
  S:=Edit1.Text+’ ‘;
  For i:=1 to length(s)-1 do
  Begin
        If (s[i]<>’ ‘)and(s[i+1]=’ ‘) then k:=k+1;
  End;
  Edit2.text:=’Количество слов в строке = ’+inttostr(k);
End;
```

Пример №2. Составить программу, которая в заданной строке заменяет слово ‘да’ на слово ‘нет’.

Решение:

```
Var   Text, Slovo1, Slovo2 : String;
      i, DlinaSlova, P     : Integer;
Begin
  Text:=Edit1.Text;  //Исходная строка
  Slovo1:=Edit2.text;//Слово, которое заменяем
  Slovo2:=Edit3.Text;//Слово, на которое заменяем
  Memo1.Lines.Add('Ответ: Исходный текст: '+ Text);
  DlinaSlova:=Length(Slovo1);
  P:=Pos(Slovo1,Text); {номер позиции, с которой в строке Text в первый раз встречается подстрока
Slovo1}
  While P>0 do          {цикл продолжается до тех пор, пока подстрока Slovo1 встречается в строке Text}
   begin
     Delete(Text, P, DlinaSlova); {удаление подстроки Slovo1, начинающейся с позиции P, из строки Text}
     Insert(Slovo2, Text, P);  {вставка подстроки Slovo2 в строку Text с позиции P}
     P:=Pos(Slovo1, Text); {номер позиции, с которой подстрока Slovo1 встречается в строке Text в
очередной раз}
   end;
  Memo1.Lines.Add('Новый текст: '+Text);
End;
```

Пример №3. Дана строка из слов. Найти самое длинное слово и поставить его в начало строки.

Решение: Для одинаковой обработки символов добавим в конец строки пробел. Как только обнаружится конец слова, вычислим его длину и проверим на максимум:

```
var s,ss,smax : string; i : integer;
begin
smax:='';
s:=edit1.Text+' ';
ss:='';
for i:=1 to length(S)-1 do
if (s[i]<>' ')and(s[i+1]=' ') then
                          begin
                          ss:=ss+s[i];
                          if length(smax)<length(ss) then smax:=ss;
                          ss:='';
                          end
                         else
                          if s[i]<>' ' then ss:=ss+s[i];
```

```
edit2.text:=smax+' '+s;
end;
```

Задание №1. Задана строка, состоящая из слов, разделенных пробелами. Удалить все слова, которые начинаются на букву "c".

Задание №2. Задана строка, состоящая из слов, разделенных пробелами. Напечатать слова, которые содержат хотя бы одну букву "d".

Задание №3. Дана строка из слов. Если все слова в строке имеют одинаковую длину, вывести сообщение «Длина одинакова».

УРОК №18. ПЕРЕЧИСЛИМЫЙ И ВАРИАНТНЫЙ ТИПЫ ДАННЫХ

Цель:

Образовательная: объяснить перечислимый и вариантный типы данных, способы их задания и использования. Научить использовать перечислимый и вариантный типы данных при составлении программ для решения задач.

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

Развивающая: развитие логических способностей учащихся.

Тип занятия: урок формирования новых знаний

Форма организации учебного процесса: урок-лекция

ХОД УРОКА

1. **Организационный момент.** Раскрытие темы урока, целей, задач и плана урока.
2. **Зачет по разделу «Циклические алгоритмы»** (тестовые задания, устный ответ и решение задач).
3. **Изучение нового теоретического материала.** Перечислимый тип.
4. **Рассмотрение примера.**
5. **Изучение нового теоретического материала.** Диапазонный тип.
6. **Рассмотрение примера.**
7. **Подведение итогов урока.**

УЧЕБНЫЙ МАТЕРИАЛ

Перечислимый тип данных

Перечислимый тип определяет упорядоченный набор идентификаторов. Эти идентификаторы обозначают конкретные значения, которые могут принимать переменные этого типа.

Общий вид записи:

Type имя_типа = (знач1, знач2, .. значN);

Имя_типа – идентификатор типа, знач1, знач2, .. значN – набор идентификаторов.

Пример №1:

```
Type TrafficLight = (Red, Yellow, Green, Dead);
```

В приведенном примере описан тип TrafficLight (светофор), который может иметь следующие значения: Red, Yellow, Green и Dead (не работает). После такого определения все переменные типа TrafficLight, могут прнимать только одно из этих значений.

Задача. Известно, сколько дней в каждом месяце года. Сколько дней летом?

Сначала запишем программу традиционным способом.

```
const day : array [1..12] of byte = (31,28,31,30,31,30,31,31,30,31,30,31);
var s,i:integer;
begin
  s:=0;
  for i:=6 to 8 do s:=s+day[i]; {летние месяцы 6, 7, 8}
  memo1.Lines.Add(inttostr(s));
end;
```

Недостаток приведенной программы – не самая лучшая наглядность, к тому же приходится самому на пальцах вычислять номера месяцев начала и конца лета (6 и 8). Для повышения наглядности и удобства программы используем перечислимый тип данных.

```
type month = (January, February, March, April, May, June, July, August, September, October, November, December);
const day:array[January .. December] of byte = (31,28,31,30,31,30,31,31,30,31,30,31);
var s:integer;
   i: month;
begin
 s:=0;
 for i:=june to august do s:=s+day[i];
 memo1.Lines.Add(inttostr(s));
end;
```

В разделе **Type** в данном примере мы ввели свой собственный перечислимый тип данных **Month**.

Основное отличие данного примера от предыдущего и его преимущество заключается в том, что в операторе For можно написать June to august вместо 6 to 8, а при определении массива Day можно написать array[January..December] вместо array[1..12]. Для этого пришлось определить специальный перечислимый тип данных Month, перечислив в скобках произвольные имена месяцев, а переменную цикла I задать типом Month, а не Integer.

Значения перечислимого типа можно использовать так же свободно, как и значения порядковых типов, например: If i=February then day[i]:=29;

Вариантный тип данных Variant

Это специальный тип данных, переменные которого могут хранить любые данные – численные, строковые, объектные и т.д.

Пример:

```
Var V:Variant;
Begin
 V:=1; //целочисленное значение
 V:=1234.5678; //действительное значение
 V:='Привет'; //строковое значение
End;
```

Использование вариантных переменных, является заметное снижение быстродействия. Программный код, в котором имеются переменные этого типа, работает очень медленно, поэтому можно рекомендовать по возможности избегать их применения.