МИНИСТЕРСТВО ЭНЕРГЕТИКИ И ПРОМЫШЛЕННОСТИ РЕСПУБЛИКИ ТАДЖИКИСТАН
МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ ТАДЖИКИСТАН
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ТАДЖИКИСТАНА
СОВМЕСТНЫЙ ТАДЖИКСКО – РОССИЙСКИЙ ФАКУЛЬТЕТ
КАФЕДРА СЕТЕВЫХ ТЕХНОЛОГИЙ И КОМПЬЮТЕРНОГО ДИЗАЙНА

Краткий курс лекций

по дисциплине «**ОСНОВЫ АЛГОРИТМА И ИНФОРМАТИКИ**»
ДЛЯ СТУДЕНТОВ 1-ГО КУРСА СПЕЦИАЛЬНОСТИ
1-45 01 03 01

Душанбе – 2015

ЛЕКЦИЯ №1

Тема: ОБЩИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Большинство действий, совершаемых человеком, выполняются по определенным правилам. Их эффективность во многом зависит от того, насколько он представляет, что делать в каждый момент времени, в какой последовательности, каким должен быть итог его действий. Так, например, применение в производстве и быту различных автоматов, компьютеров требует от человека строгого соблюдения определенной последовательности действий при их использовании, что невозможно без предварительного составления алгоритмов.

Таким образом, осмысление и разработка алгоритмов выполняемых действий становится существенным компонентом деятельности человека, составной частью его культуры мышления и поведения. Алгоритм – одно из фундаментальных понятий, которое используется в различных областях знания, но изучается оно в математике и информатике. Его освоение начинается уже в начальной школе на уроках математики, где ученики овладевают алгоритмами арифметических действий, знакомятся с правилами вычитания числа из суммы, суммы из числа и др.

Немного из истории появления термина «алгоритм»

Происхождение термина «алгоритм» связано с математикой. История его возникновения такова. В IX веке в Багдаде жил ученый ал(аль)-Хорезми (полное имя – Мухаммед бен Мусаал-Хорезми), математик, астроном, географ. В одном из своих трудов он описал десятичную систему счисления и впервые сформулировал правила выполнения арифметических действий над целыми числами и обыкновенными дробями. Арабский оригинал этой книги был утерян, но остался латинский перевод XII в., по которому Западная Европа ознакомилась с десятичной системой счисления и правилами выполнения арифметических действий. Правила в книгах ал-Хорезми в латинском переводе начинались словами «Алгоризми сказал». В других латинских переводах автор именовался как Алгоритмус. Со временем было забыто, что Алгоризми (Алгоритмус) – это автор правил, и эти правила стали называть алгоритмами. Многие столетия разрабатывались алгоритмы для решения все новых и новых классов задач, но само понятие алгоритма не имело точного математического определения. В настоящее время понятие алгоритма уточнено.

Алгоритм – понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение цели.

Основные свойства и способы представления алгоритма

Любой алгоритм должен обладать следующими свойствами:

• определенностью – за конечное число шагов либо должен быть получен результат, либо доказано его отсутствие;

• результативностью – обязательным получением некоторого результата (числа, таблицы, текста, звука, изображения и т. д.) или сигнала о том, что данный алгоритм неприменим для решения поставленной задачи;

• массовостью – возможностью получения результата при различных исходных данных для некоторого класса сходных задач;

• формальностью – отвлечение от содержания поставленной задачи и строгое выполнение некоторого правила, инструкции;

• дискретностью — возможностью разбиения алгоритма на отдельные элементарные действия.

Существуют следующие формы представления алгоритма:

• Словесная (вербальная) на неформальном языке.

• На языках программирования.

• Графическая.

Словесная форма представления алгоритма имеет ряд недостатков. Для достаточно сложных алгоритмов описание становится слишком громоздким и не наглядным. Эта форма представления обычно используется лишь на начальных стадиях разработки алгоритма. Пример словесной формы описания алгоритма: Чтобы перейти улицу, нужно посмотреть налево, убедиться в отсутствии приближающегося транспорта, дойти до середины улицы, посмотреть направо, убедиться в отсутствии близко идущего транспорта, продолжить движение через улицу. При наличии движущихся транспортных средств нужно ждать, когда транспорт проедет.

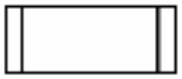
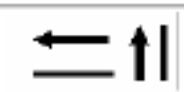
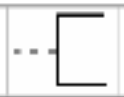
Алгоритм, записанный на языке программирования, называется программой.

Графическая форма представления алгоритмов является более наглядной и строгой. Алгоритм изображается в виде последовательности связанных между собой блоков, каждый из которых соответствует выполнению одного или нескольких операторов. Такое графическое представление называется блок-схемой алгоритма.

Условные графические обозначения символов, используемых для составления блок-схемы алгоритма, стандартизированы [2]. Некоторые, часто используемые обозначения, приведены в табл. 1.

Таблица 1

Условные графические обозначения символов

Название блока	**Обозначение**	**Название блока**	**Обозна-чение**
Начало или конец алгоритма		Решение	
Процесс (действие или серия действий)		Предопределенный процесс (вспомогательный алгоритм)	
Ввод/вывод данных		Модификация (заголовок цикла)	
Линии потока		Комментарии	

Представление алгоритма в виде блок-схемы является промежуточным, так как алгоритм в таком виде не может быть непосредственно выполнен ЭВМ, но помогает пользователю при создании (написании) программы для ПК. Использование блок-схем дает возможность:

• наглядно отобразить базовые конструкции алгоритма;

• сосредоточить внимание на структуре алгоритма, а не на синтаксисе языка;

• анализировать логическую структуру алгоритма;

• преобразовывать алгоритм методом укрупнения (сведения к единому блоку) или детализации – разбиения на ряд блоков;

• использовать принцип блочности при коллективном решении сложной задачи;

• осуществить быструю проверку разработанного алгоритма (на уровне идеи);

• разобрать большее число учебных задач.

Составление блок-схемы алгоритма является важным и в большинстве случаев необходимым этапом решения сложной и большой задачи на ЭВМ, значительно облегчающим процесс составления программ.

ЛЕКЦИЯ №2

ТЕМА: БАЗОВЫЕ СТРУКТУРЫ ПРОГРАММИРОВАНИЯ

Выделяют три основные структуры алгоритмов:

1. Линейная.

2. Разветвляющаяся (альтернатива «если–то–иначе» или «если–то»).

3. Циклическая (повторение).

Линейная структура – является основной. Она означает, что действия выполняются друг за другом (рис. 1).

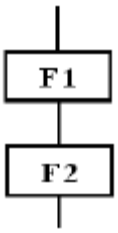


Рис. 1. Линейная структура

Прямоугольник, показанный на рисунке, может представлять как одну единственную команду, так и множество операторов, необходимых для выполнения сложной обработки данных, где F1 и F2 – некоторые команды для соответствующего исполнителя. Команды записываются с помощью операции присваивания.

Присваивание переменной какого-либо значения или присваивание одной переменной значения другой переменной является наиболее часто выполняемым действием в программе, написанной на любом языке программирования. В языке программирования присваивание является операцией и обозначается как «:=». Это означает, что при выполнении этой операции происходит не только присваивание значения определенной переменной, но и возвращается некоторое значение.

Разветвляющаяся структура (ветвление) – это структура, обеспечивающая альтернативный выбор в зависимости от заданного условия.

Выполняется проверка условия, а затем выбирается один из путей (рис. 2), где Р – это условие, в зависимости от истинности (Да)1 или ложности (Нет) которого управление передается по одной из двух ветвей.

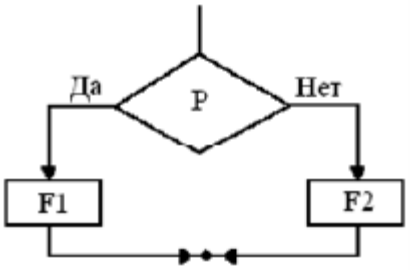


Рис. 2. Ветвление

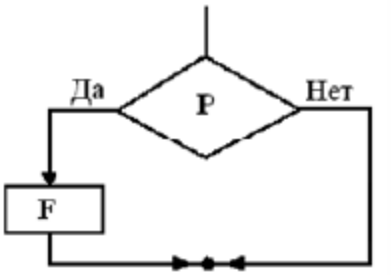


Рис. 3. Неполное ветвление

Может оказаться, что для одного из результатов проверки условия ничего предпринимать не надо. В этом случае можно применять только один обрабатывающий блок (рис. 3).

Эта структура называется также ЕСЛИ –ТО – ИНАЧЕ. Каждый из путей (ТО или ИНАЧЕ) ведет к общей точке слияния, так что выполнение программы продолжается независимо от того, какой путь был выбран.

Циклическая структура (или повторение) предусматривает повторное выполнение некоторого набора действий. В литературе часто обозначают вместо слова «Да» знак «+», а вместо слова «Нет» пишут «–».

Циклы позволяют записать длинные последовательности операций обработки данных с помощью небольшого числа повторяющихся команд.

Итерационным называется цикл, число повторений которого не задается, а определяется в ходе выполнения цикла. В этом случае одно повторение цикла называется итерацией. Рекурсия – это такая ситуация, когда некоторый алгоритм непосредственно или через другие алгоритмы вызывает себя в

качестве вспомогательного. Сам алгоритм при этом называется рекурсивным.

В качестве примера использования рекурсии рассмотрим задачу поиска файлов. Пусть нужно получить список всех файлов, например, с расширением bmp, которые находятся в указанном пользователем каталоге и во всех подкаталогах этого каталога.

Словесно алгоритм обработки каталога может быть представлен так:

1. Вывести список всех файлов, удовлетворяющих критерию запроса.

2. Если в каталоге есть подкаталоги, то обработать каждый из этих каталогов.

Приведенный алгоритм, блок-схема которого представлена на рис. 4, является рекурсивным. Для того чтобы обработать подкаталог, процедура обработки текущего каталога должна вызвать сама себя.

Рис. 4

Различают цикл с предусловием и цикл с постусловием.

Цикл начинается с проверки логического выражения «P». Если оно истинно, то выполняется «F», затем все повторяется снова, до тех пор, пока логическое выражение сохраняет значение «истина». Как только оно становится ложным, выполнение операций «F» прекращается и управление передается по программе дальше. Так как выражение, управляющее циклом, проверяется в самом начале, то в случае, если условие сразу окажется ложным, операторы циклической части «F» могут вообще не выполняться. Операторы циклической части «F» должны изменять переменную и(ли

переменные), влияющую на значение логического выражения, иначе программа «зациклится» – будет выполняться бесконечно. Рассмотренная

циклическая конструкция называется цикл «пока», или цикл с предусловием (см. рис. 5).

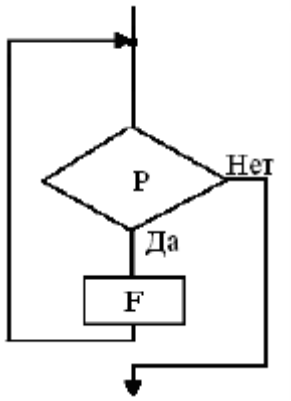


Рис. 5

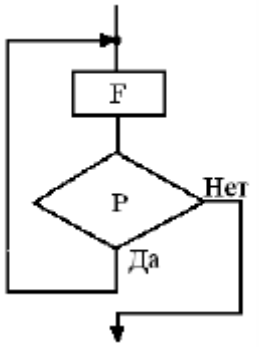


Рис. 6

Существует и иная конструкция цикла, которая предусматривает проверку

условия после выполнения команд, встроенных внутрь цикла. Это цикл с

постусловием (см. рис. 6).

Циклические структуры можно комбинировать одну с другой – как путем организации их следований, так и путем создания суперпозиций (вложений одной структуры в другую). Схематические изображения

нескольких суперпозиций базовых алгоритмических структур представлены на рис. 7–10.

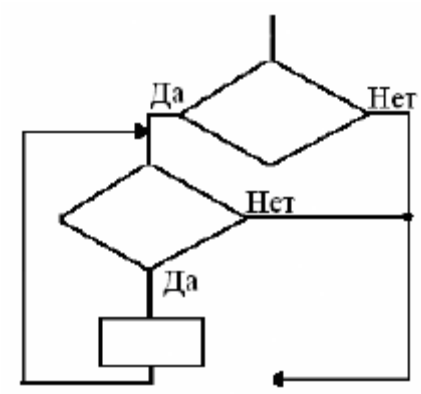


Рис. 7. Алгоритм типа «цикл, вложенный в неполную развилку»

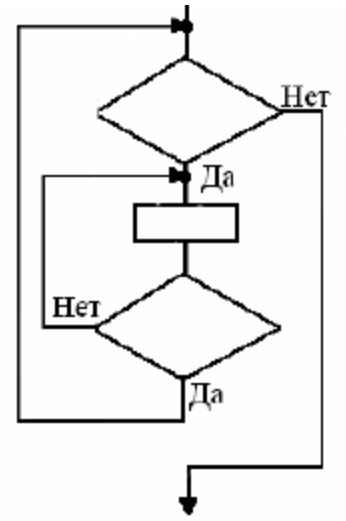


Рис. 8. Алгоритм типа «цикл в цикле»

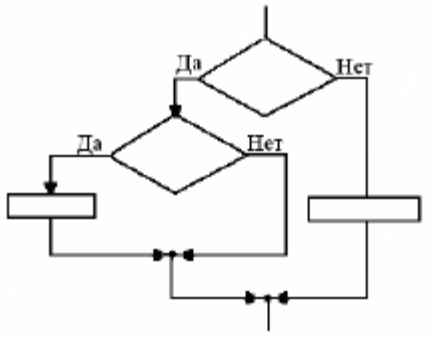


Рис. 9. Алгоритм типа «развилка в развилке»

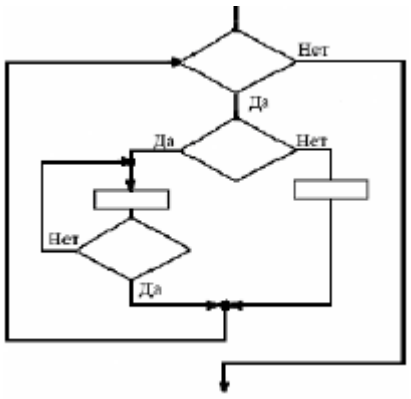


Рис. 10. Иллюстрация трехкратного вложения одной базовой структуры в другую

ЛЕКЦИЯ 3.

Тема: ОСНОВНОЕ ПОНЯТИЕ *VISUAL BASIC*

Систему программирования Visual Basic называют средой проектирования, и в практических работах настоящего практикума предлагается выполнять проекты в этой среде.

Объект – это некая отдельная сущность, отличающаяся от других сущностей особыми свойствами, поведением, взаимодействием с другими объектами приложения. При объектно-ориентированном подходе в программировании любое приложение представляет собой набор взаимосвязанных объектов, реализующих необходимые функциональные требования, предъявленные к приложению. Форма, кнопки, метки, текстовые поля и т.п. – это объекты.

Класс – это совокупность объектов, обладающих общими свойствами и поведением. Например, кнопка на форме со всеми своими конкретными свойствами и действием является объектом класса Command Button.

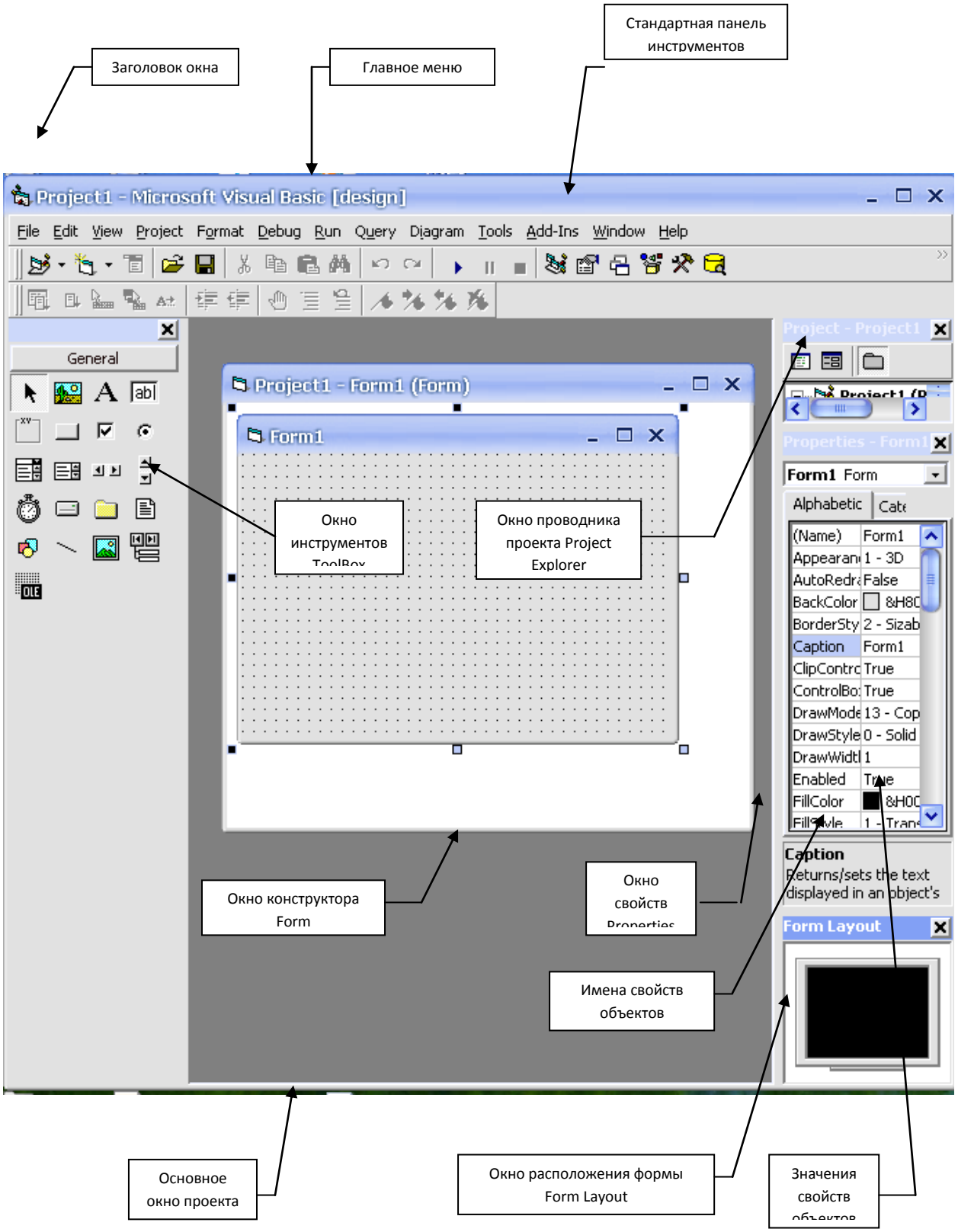
Событие – это средство взаимодействия объектов друг с другом. Событие может создаваться пользователем или возникать в результате воздействия других программных объектов. Объекты могут генерировать события и выполнять действия в ответ на заданные события. Примеры событий – загрузка формы (Load), щелчок мышью по объекту (Click), двойной щелчок (DblClick).

Метод – это процедура, которая реализует возможные с объектом действия. В результате этих действий в объекте что-либо меняется. Методы работают как процедуры, но принадлежат конкретным объектам так же, как и свойства.

Оператор – это конструкция языка программирования, задающая одну или несколько операций, производимых над операндами. В качестве *операндов* могут выступать константы, переменные, выражения, функции. Любой оператор записывается в соответствии со строгими синтаксическими правилами (форматами). Например, в *операторе присваивания* «=» читается «присвоить» и означает, что в ячейку памяти записывается значение, равное выражению справа от знака «=».

Переменная – это важная составляющая данная любого языка программирования: она представляет собой зарезервированное место в оперативной памяти для хранения данных. Каждая переменная имеет собственное имя. После того как переменной присвоено значение, она может использоваться в программе вместо самого значения. Для вводимых в программу переменных надо описывать их тип.

Рис.1 Вид экрана после загрузки программы Visual Basic



ЛЕКЦИЯ № 4

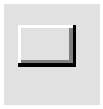
ТЕМА: ОСНОВНЫЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ VISUAL BASIC

Элементы управления - это объекты, которые служат для организации интерфейса между пользователем и компьютером. Например: *кнопки, списки, переключатели*.

Рис.2 Панель элементов управления

Label (Метка**)**  - этот объект используется для размещения в форме текста, заголовков, надписей к полям, поясняющей информации. Текст, задаваемый объектом *Label*, не может быть изменен пользователем приложения. Но это не значит, что его нельзя изменить при выполнении приложения. Изменить текст можно программно.

TextBox (Текстовое поле**)**  – этот объект предназначен для отображения информации и ввода данных в форме типа поле. Текстовые поля - это наиболее часто встречающийся тип элементов управления, т. к. их можно использовать не только для просмотра информации, как с помощью рассмотренных ранее элементов управления типа *Label*, но и для ввода информации пользователями приложения.

CommandButton (Кнопка**)**  – этот объект имеет важную роль в форме. Нажатие кнопки, размещенной в форме, позволяет выполнить процедуру обработки события ***Click***. Это может быть, например, печать данных или проведение определенных вычислений.

CheckBox (Флажок**)**  – этот объект используется для размещения в форме данных, которые могут иметь только одно из двух допустимых значений, которые называется *флажком.* Флажки позволяют пользователю дать ответ на поставленный вопрос. В случае положительного ответа пользователь устанавливает флажок, и он приобретает вид квадрата, в котором размещена галочка. При неустановленном флажке он имеет вид пустого квадрата, обозначая отрицательный ответ на поставленный вопрос. Возможно еще одно состояние флажка, при котором он недоступен. В этом состоянии он имеет вид галочки на сером фоне. Флажки могут использоваться в форме по одному или группами.

OptionButton (Переключатель)  – эти объекты называются *переключателями,* так как располагаемые в группах, они позволяют выбрать одно из нескольких значений.

ListBox  - создает в форме список, в котором элементы расположены в одну или несколько колонок. В случае, если элементы списка не помещаются в созданном объекте ListBox, то в нем появляются полосы прокрутки, располагаемые снизу и/или с правой стороны.

ComboBox  - создает в форме раскрывающийся список. Этот тип списка позволяет пользователю осуществлять выбор значения, вводимого в размещаемое сверху поле ввода, или выбирать значение из списка, открываемого нажатием кнопки со стрелкой, размещаемой с правой стороны. Список данного типа

удобно использовать в том случае, если вводимых значений много, а места в форме для расположения обычного списка не хватает.

Line (Линия**)**  – этот объект используется для добавления в форму линии.

Shape (Контур**)**  – этот объект используется для добавления в форму контуры. Она позволяет создавать в форме прямоугольник, квадрат, овал, окружность, прямоугольник и квадрат с округленными углами.

Таблица 1. *Значения свойства Shape*

Значение свойства	**Контур**
0-Rectangle	Прямоугольник
1-Square	Квадрат
2-Oval	Овал
3-Circle	Окружность
4-Rounded Rectangle	Прямоугольник с округленными углами
5-Rounded Square	Квадрат с округленными углами

VScrollBar и HScrollBar(Полосы прокрутки**)**  – эти объекты используются для размещения в форме полосы прокрутки. Элементы управления VScrollBar и HScrollBar похожи на полосы прокруток программы Microsoft Word и другими программными продуктами, работающими в среде Windows, а также при работе с многострочными текстовыми полями и списками, в которых информация не помещается целиком в окне просмотра. Эти элементы управления отличаются от полос прокруток программы Word и другими программами работающие в среде Windows, так

как существуют самостоятельно и используются для управления вводом параметра, значение которого может меняться в некотором диапазоне.

Frame  – этот объект является контейнером и служит для объединения других элементов в группу, после чего помещенными в него объектами можно управлять как единым целым.

Timer (Таймер**)**  – этот объект, который обрабатывает данные системных часов. Его можно использовать для выполнения определенных действий через заданный интервал времени. Интервал активизации объекта в миллисекундах может принимать значение от 0 до 64767 (от 0 до 64,8 секунды).

Image (Изображение**)**  **–** это объект позволяет разместить в форме графические изображения.

PictureBox (Графическое окно**)**  – этот объект обладает более широким набором свойств и методов, чем объект *Image*. Он может использоваться для следующих целей:

- для отображения графических изображений;
- в качестве контейнера для других элементов управления;
- в виде графического окна для вывода текста, графических элементов, анимации.

ЛЕКЦИЯ № 5

Тема: ТИПЫ ДАННЫХ

Для хранения чисел в Visual Basic используется разные типы данных такие как:

Числовые (Integer, Long, Single, Double, Currency);

Integer (числовой) - служит для хранения целых чисел и может принимать значения в диапазоне от - 32768 до 32767.

Long(числовой)- служит для хранения больших целочисленных чисел и может принимать значения в диапазоне от -2 147483648 до 2 147483648.

Single(действительный) - служит для хранения чисел с плавающей точкой и может принимать значения в диапазоне - 3,4^38 ... -1,4^-45 для отрицательных чисел и 1,4^-45 ... 3,4^38 для положительных чисел.

Double(действительный) - служит для хранения чисел с плавающей точкой и может принимать значения в диапазоне -1,7^-308 ... -4,9^-324 для отрицательных чисел и 4,9^-324 ... l,7^+308 для положительных чисел.

Currency (денежный) - служит для хранения чисел с фиксированной точкой. Целая часть числа может содержать до 15 цифр, а дробная - до 4.

String (строковой) - используется для переменных строкового типа, и может хранить строки как фиксированной (до 2^{16} символов), так и переменной длины (до 2^{31} символов).

Date (типа дата) - служит для хранения даты и времени.

Byte (байтовый) - служит для хранения двоичных данных и используется в массивах.

Boolean(логический) - служит для логического типа данных и может принимать одно из двух значений: **True** (Истина) или **False** (Ложь).

Variant (произвольный) - является универсальным типом данных и переменная этого типа может хранить любой из выше описанных типов данных.

Object (объектный) - служит для хранения ссылок на объекты.

ЛЕКЦИЯ 6.

ТЕМА: ПЕРЕМЕННЫЕ. ОБЪЯВЛЕНИЯ ПЕРЕМЕННЫХ

Переменные используются для хранения необходимых в программе данных. Переменная имеет *Имя* и *Значение*. *Имя* переменной не измена, а *Значение* может меняться в процессе выполнения программы. В Visual Basic используется явное и неявное объявление переменной. Любая переменная должна быть объявлена в программе. Явное объявление осуществляется операторами Dim, Private, Static, Public, которые имеют следующий синтаксис:

Dim, Private, Static, Public имяПеременной [As типДанных]

Операторы Dim, Private, Static, Public определяют область действия переменной. С помощью одного оператора вы можете объявлять несколько переменных, разделяя их запятыми.

При использовании явных объявлений переменных рекомендуется установить такой режим трансляции программы, при котором допускается только явное объявление переменных. Для этого необходимо в начале модуля (или кода) вставить оператор ***Option Explicit (Явное объявление)***.

ЛЕКЦИЯ 7.

ТЕМА: ДИАЛОГОВЫЕ ОКНА

В Visual Basic 6.0. существует специальный вид окон, которые называются ***диалоговыми***. Диалоговые окна бывают двух типов:

1. Модальные диалоговые окна

Модальные диалоговые окна - эти окна, из которого нельзя перейти в другое окно, не закрыв текущее. Данный вид диалоговых окон используется для выдачи сообщений о ходе работы приложения, его настройки или ввода каких-либо данных, необходимых для работы.

2. Немодальные диалоговые окна

Немодальные диалоговые окна - эти окна, позволяют перемещать фокус на другое окно или форму без закрытия текущего окна. Данный тип диалоговых окон используется редко.

7.1. Диалоговое окно сообщения - MsgBox

Диалоговое окно сообщения *MsgBox* не требует проектирования и вызывается из программы командой *MsgBox* или с помощью аналогичной функции *MsgBox*, имеющий следующий синтаксис:

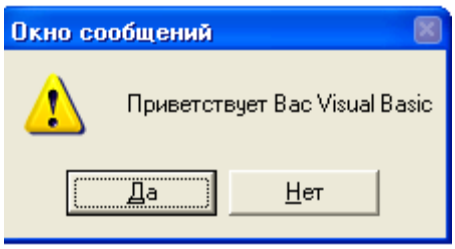
MsgBox (prompt[, buttons] [, title])

где:

- prompt - текст сообщения в диалоговом окне;
- buttons - числовое выражение, которое задает параметры для кнопок управления и значков в диалоговом окне;
- title - текст заголовка диалогового окна;

Например, в командном окне среды проектирования **Immediate** вводим следующую команду: MsgBox “Приветствует Вас Visual Basic”, vbYesNo+vbExclamation, “Окно сообщений” и нажимаем на клавишу **Enter.**

Рис.3 Диалоговое окно сообщения “MsgBox”



7.2. Диалоговое окно ввода информации - InputBox

Часто в диалоговых окон необходимо не только нажать кнопки выбора действия, но и ввести определенную информацию, которая затем анализируется программой. Для выполнения такого рода действий в Visual Basic можно использовать диалоговое окно ввода информации **InputBox**. Функция *InputBox* имеет следующий синтаксис:

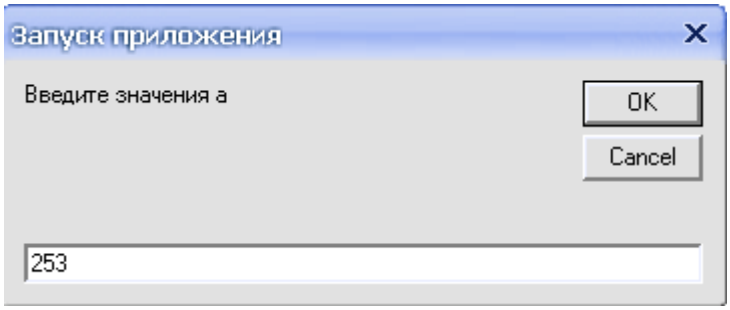
InputBox (prompt [, title] [, default])

где:

- prompt - текст сообщения в диалоговом окне;
- title - текст заголовка диалогового окна;
- default - значение текстового поля ввода по умолчанию. Если параметр отсутствует, строка остается пустой;

Например, в командном окне среды проектирования **Immediate** вводим следующую команду: a=InputBox(“Введите значения **a”,** **“**Запуск приложения**”**) и нажимаем на клавишу **Enter.**

Рис. 4 Диалоговое окно ввода InputBox



В отличие от диалогового окна *MsgBox***,** в окне *InputBox* всегда имеется только две кнопки управления: **ОК** и **Cancel.** Кнопка **ОК** подтверждает ввод данных, кнопка **Cancel** - закрывает диалоговое окно без ввода данных.

ЛЕКЦИЯ 8.

ТЕМА: УПРАВЛЯЮЩИЕ ОПЕРАТОРЫ

В Visual Basic, как и во всех языках программирования, существуют управляющие операторы, предназначенные для управления порядком выполнения команд. Различают два основных типа управляющих операторов:

- If
- Select Case

В свою очередь, управляющий оператор If бывает однострочной и многострочной:

- If…Then
- If…Then…Else

8.1. Оператор If…Then, If…Then…Else

Оператор **If…Then** применяется в том случае, когда необходимо выполнить один или группу операторов при соблюдении определенного условия, т.е. когда значение заданного условия равно **True**. Существует две разновидности данного оператора: однострочный и многострочный. Однострочный оператор имеет следующий синтаксис:

If **условие** Then **конструкция**

В этом операторе условие и выполняемые при соблюдении действия располагаются в одной строке.

Если при выполнении условии требуется выполнение блока операторов, используется многострочный оператор, имеющий следующий синтаксис:

If **условие** Then

конструкции

End If

Оператор **If…Then…Else** аналогичен конструкции **If…Then**, но позволяет задать действия, исполняемые как при выполнении условий, так и в случае их невыполнения. Конструкция имеет следующий синтаксис:

If **условие** Then

конструкции для обработки истинного условия

Else

конструкции для обработки ложного условия

End If

Если заданное в конструкции условие не выполняется (результат проверки равен **False**), и конструкция содержит ключевое слово **Else**, Visual Basic выполнит последовательность конструкций, расположенных следом за Else.

8.2. Оператор выбора Select Case

Оператор **Select Case** позволяет обрабатывать в программе несколько условий и аналогичен блоку оператора If. Это конструкция состоит из анализируемого выражения и набора операторов Case на каждое возможное значение выражения.

Сначала Visual Basic вычисляет значение заданного в конструкции выражения. Затем полученное значение сравнивается со значениями, задаваемыми в операторах Case конструкции. Если найдено искомое значение, выполняются команды, приписанные данному оператору Case. После завершения выполнения конструкций управление будет передано конструкции, следующей за ключевым словом End Select. Синтаксис конструкции Select Case следующий:

Select Case **сравниваемоеЗначение**

CASE **значение1**

конструкция1

CASE значение2

конструкция2

…

End Select

В начале конструкции расположены ключевое слова **Select Case**, указывающий на то, что расположенный рядом с ним параметр **сравниваемоеЗначение** будет проверяться на несколько значений. Далее следуют группа команд, начинающиеся с ключевого слова **Case** где, выполняются команды, расположенные между этим и следующим за ключевым словом **Case.**

ЛЕКЦИЯ № 9

ТЕМА: ЦИКЛИЧЕСКИЕ ОПЕРАТОРЫ

На языке Visual Basic для выполнения повторяющихся действий используются циклы. Бывают следующие виды циклических операторов:

- For…Next
- Do…Loop
- While…Wend

9.1. Оператор For…Next

Конструкция **For…Next** выполняет последовательность команд определенное число раз. Такую конструкцию называют *циклом,* а выполняемые ею программные коды – *телом цикла.* Синтаксис конструкции **For…Next** следующий:

For **счетчик** = **начЗначение** To **конЗначение** [Step **шаг**]

конструкции

Next [**счетчик**]

Первый аргумент конструкции «**счетчик»** – определяет имя переменной, которая будет “считать” количество выполнений цикла. Параметр **начЗначение** указывает числовое значение, которое присваивается переменной счетчику перед первым проходом цикла. Цикл выполняется до тех пор, пока значение счетчика не превысит конечного значения, указанного после ключевого слова **To**. После каждого прохода цикла значение счетчика изменяется на величину **шаг**, указанную за ключевым словом **Step**. Ключевое слово **Next** обозначает конец тела цикла и является обязательным.

9.2. Оператор Do…Loop

Цикл, задаваемый конструкцией Do…Loop, выполняется до тех пор, пока истинно задаваемое в цикле условие. Синтаксис конструкции Do…Loop следующий:

Do **условие**

конструкции

Loop

Аргумент конструкции ***условие*** является логическим выражением, значение которого проверяется перед каждым проходом цикла. Если это значение равно True, то выполняется последовательность команд, которые расположены между Do While и ключевым словом Loop. Эти конструкции образуют тело цикла. Если при очередном проходе цикла условие равно False, то происходит выход из цикла и управление передается конструкции, следующий за Loop. Возможна ситуация, при которой операторы цикла не выполняются ни разу. Она возникает в том случае, если при первой проверке условия оказывается ложным. Если в предыдущей конструкции условие, по которому выполняется цикл, расположено в заголовке, то в этой конструкции условие располагается в конце цикла:

Do

конструкции

Loop While **условие**

Есть еще две разновидности конструкции цикла Do…Loop. Они аналогичны рассмотренным раннее, но отличаются тем, что цикл выполняется до тех пор, пока условие ложно, а не истинно. Данные операторы имеют следующий синтаксис:

Do Until **условие**

конструкции

Loop

и

Do

конструкции

Loop Until **условие**

ЛЕКЦИЯ № 10

ТЕМА: МАССИВЫ. ВИДЫ МАССИВОВ

Для хранения величин кроме простых переменных можно использовать массивы. *Массив* представляет собой набор переменных с одним именем и разными индексами. Каждая такая переменная называется *элементом массива.* Количество хранящихся в массиве элементов называются *размерами массива.* Размер массива ограничен объемом оперативной памяти и типом данных элементов массива. В массиве индекс элемента указывается в круглых скобках после имени массива. Например: A(l), A(2), A(10)

где:

1, 2, 10 - являются элементами массива.

A – именем массива.

Массивы бывают двух видов:

10.1. Одномерные массивы

Одномерные массивы имеют неизменный размер, заданный при его объявлении. При объявлении массива после его имени в круглых скобках указывается верхняя граница массива. По умолчанию нижней границей массива является 0.

Например: Dim A(20) As Integer

В этом коде, задается массив из 21 элемента. Индекс элементов массива изменяется от 0 до 20:

Можно явно задать нижнюю границу массива, используя ключевое слово ***To***:

Например: Dim A(1 To 20) As Integer

В этом случае задается массив из 20 элементов. Индекс элементов массива изменяется от 1 до 20.

10.2. Многомерные массивы

Многомерные массивы могут изменять размер в процессе выполнения.

В следующем коде объявляется многомерный массив размерностью 21x21:

```
Dim A(20, 20) As Integer
```

При использовании многомерных массивов, как и в случае одномерных, можно явно задавать нижнюю границу:

```
1. Dim A(1 To 20, 1 To 20) As Integer
```

В первой строке задана верхняя и нижняя граница обеих размерностей.

```
2. Dim A(20, 1 To 20) As Integer
```

Во второй строке задана верхняя и нижняя граница только для второй размерности.

ЛЕКЦИЯ № 11

ТЕМА: ОПЕРАТОР REDIM

С помощью выполняемого оператора *ReDim* указывается размерность массива в виде числа или выражения. Например, размерность массива может быть задана любым из следующих способов:

```
ReDim A(x)

ReDim A(20)

ReDim A(1 To 20)
```

При выполнении оператора *ReDim* данные, размещенные в массиве ранее, теряются. Это удобно в том случае, если данные вам больше не нужны и вы хотите переопределить размерность массива и подготовить его для размещения новых данных. Если вы хотите изменить размер массива, не потеряв при этом данных, то необходимо воспользоваться оператором *ReDim* с ключевым словом *Preserve*. Например, приведенный ниже программный код увеличивает размер массива на единицу без потери хранящихся в массиве данных: ReDim Preserve A(X + 1)

ЛЕКЦИЯ №12

ТЕМА: ПРОЦЕДУРЫ

При программировании широко используются процедуры, позволяющие разбивать программные коды на небольшие логические блоки, которые, во первых, легче отлаживать, а во вторых, можно в свою очередь использовать при создании других процедур. В Visual Basic-е используются следующие виды процедур:

- Sub
- Function
- Property

12.1. ПРОЦЕДУРА *SUB*

Процедура *Sub* не возвращает значения и наиболее часто используется для обработки связанного с ней события. Ее можно помещать в стандартные модули, модули класса и форм. Она имеет следующий синтаксис:

[Private] [Public] [Static} Sub **имяПроцедуры (аргументы)**

операторы

End Sub

Между ключевыми словами Sub и End Sub в процедуре располагаются выполняемые при ее вызове операторы программного кода. Параметр **аргументы** можно применять для объявления передаваемых в процедуру переменных.

Процедура Sub подразделяется на общие процедуры и процедуры событий. Общие процедуры служат для размещения повторяющихся операторов, используемых процедурами по обработке событий, тем самым, разгружая их и исключая дублирование часто встречающихся кодов, что в свою очередь облегчает поддержку приложения.

Процедуры обработки событий связанны с объектами, размещенными в формах Visual Basic, или с самой формой и выполняются при наступлении события, с которыми они связаны. Для события, связанного с формой, процедура Sub имеет следующий синтаксис:

Private Sub **Form_имяСобытия (аргументы)**

операторы

End Sub

Для события, связанного с элементом управления формы, процедура обработки событий Sub имеет следующий синтаксис:

Private Sub **имяЭлементаУправления_имяСобытия (аргументы)**

Операторы

End Sub

12.2. ВЫЗОВ И ПЕРЕДАЧА ПАРАМЕТРОВ ПРОЦЕДУР

Вызов процедуры Sub можно осуществлять двумя способами. Первый способ предполагает ключевого слова *Call.* Например, процедуру с именем *NameProc* можно вызвать оператором

Call NameProc (**аргумент1, аргумент2, … аргументN**)

Второй способ позволяет вызвать процедуру Sub по ее имени.

NameProc **аргумент1, аргумент2, … аргументN**

Передача параметров процедуры SUB

Переменные, передаваемые процедуре, называются параметрами процедуры. Передача параметров в процедуре может осуществляться двумя способами: *по значению* (by value) и *по ссылке* (by reference). В первым случае в качестве переменной передается не сама переменная, а ее копия. Поэтому изменение параметра в процедуре затрагивает не переменную, а ее копию. Для передачи параметров по ссылке используется ключевое слово *ByVal*:

Sub NameProc (ByVal strArg as String)

тело процедуры

End Sub

ЛЕКЦИЯ № 13

ТЕМА: ПРОЦЕДУРА FUNCTION

Процедура Function в отличие от процедуры Sub может возвращать значение в вызывающую процедуру. Синтаксис процедуры Function выглядит следующим образом:

[Private] [Public] [Static} Function **имяПроцедуры** (**аргументы**) [As type]

операторы

End Function

Процедура *Function*, как и переменные, имеет тип, задаваемый с помощью ключевого слова As. Если тип процедуры не задан, по умолчанию ей присваивается тип *Variant.* Тип процедуры определяет в свою очередь тип возвращаемого ею значения.

Возвращаемое процедурой значение присваивается имени процедуры, и может быть использовано в выражениях программного кода аналогичному стандартным функциям Visual Basic.

Вызов процедуры Function

Вызов процедуры Function аналогичен вызову встроенных функций Visual Basic. Кроме этого процедуру Function можно вызвать так же, как и процедуру *Sub*.

ЛЕКЦИЯ № 14

ТЕМА: РАБОТА СО СТРОКАМИ. ОСНОВНЫЕ ФУНКЦИИ

Для работы в Visual Basic со строками используется *оператор объединения,* называемый также *оператором конкатенации,* или встроенными функциями. Познакомимся с операциями, которые можно совершать над строками и наиболее распространенными встроенными функциями обработки строк.

Таблица 3. *Функции, предназначенные для работы со строками*

Функция	Назначение
Asc	Возвращает ASCII-код символа
Chr	Преобразовывает ASCII-код в символ
InStr, InStrRev	Осуществляют поиск одной строки в другой
LCase	Изменяет регистр букв исходной строки на нижний
UCase	Изменяет регистр букв исходной строки на верхний
Len	Возвращает количество символов в строке
LTrim, RTrim, Trim	Удаляют пробелы, расположенные соответственно в начале, в конце и с обеих сторон символьной строки
Mid	Возвращает заданное количество символов из произвольного места строки
Right	Возвращает указанное количество символов с конца строки
Left	Возвращает указанное количество символов с начала строки
StrReverse	Изменяет порядок следования символов в строке на обратный

StrConv	Изменяет регистр букв символьной строки
Str, CStr	Преобразовывают числовое выражение в строку
Val	Преобразовывает строку в числовое выражение

Функция *Str и Val*

Функция *Str* преобразовывает численное значение в символьное представление. Синтаксис функции следующий:

*Str **(число)***

Функция *Val* преобразовывает символьную строку в численное значение. Синтаксис функции следующий:

*Val **(символьноеВыражение)***

Выделение подстроки

Используя функции *Left, Right* и *Mid* можно выделить подстроку заданной символьной строки. Функции *Left* и *Right* выделяют строку начиная с крайнего левого или крайнего правого символа, а функция *Mid* позволяет выбрать любую подстроку. Синтаксис этих функций следующее:

Left(выражение, числоСимволов)

Right(выражение, числоСимволов)

Mid(выражение, номерПозиции, числоСимволов)

Ниже приведены примеры использования этих функций и возвращаемые ими значения:

a= "**Технологический Университет**"

Print Left(a, 3) ' Возвращает "**Тех**"

Print Right (a, 5) ' Возвращает "**ситет**"

Print Mid (a, 17,3) ' Возвращает "**Уни**"

Преобразование строки

Функции *UCase* и *LCase* используются в Visual Basic для преобразования строчных символов в заглавные и заглавных в строчные.

Функции *UCase, LCase*

Функция *Ucase* преобразует все строчные буквы в заглавные. Синтаксис функции следующий: *UCase (символьнаяСтрока)*

Например:

a = "**университет**"

Print UCase(a) ' Возвращает "**УНИВЕРСИТЕТ**"

Print UCase$(a) ' Возвращает "**УНИВЕРСИТЕТ**"

Функция *LCase* возвращает заданную символьную строку, в которой все заглавные буквы преобразованы в строчные. Синтаксис функции следующий: *LCase(символьнаяСтрока).*

Например:

a= "**УНИВЕРСИТЕТ**"

Print LCase(a) ' Возвращает " **университет** "

Print LCase$(a) ' Возвращает " **университет** "

Функции LTrim, RTrim, Trim

Функции *LTrim, RTrim* удаляют пробелы в начале и в конце строки, а функция Trim удаляет пробелы в начале и в конце строки.

Например:

a= "**УНИВЕРСИТЕТ**"

Print LTrim(a) ' Возвращает " _УНИВЕРСИТЕТ "

Print RTrim(a) ' Возвращает "УНИВЕРСИТЕТ_ "

Print Trim(a) ' Возвращает "_УНИВЕРСИТЕТ_ "

Функция ASCII

Функция *Asc* возвращает ASCII – код начальной буквы строки.

Например:

a= "Hello"

Print Asc(a) ' Возвращает "72"

Функция Chr(Charcode)

Функция *Charcode* принимает значения от 0 до 255 символов. Значения от 0 до 31 соответсвует так называемым управляющим кодам.

Например:

Print Chr(86) ' Возвращает "V"

Print Chr(Asc(“э”+1) ' Возвращает "ю"

Функция InStr

Функция *InStr* возвращает позицию первого вхождения подстроки в данную строку, начиная соответственно с ее начала и с конца.

Например:

a= "VISUAL BASIC"

b= "A"

Print InStr(a,b); InStr(7,a,b) ' Возвращает "59"

Функция Len

Функция *Len* определяет длину строки.

Например:

```
a= "VISUAL BASIC"
Print Len(a) ' Возвращает "12"
```

ЛЕКЦИЯ № 15

ТЕМА: РАБОТА С ФАЙЛАМИ

В Visual Basic как и в других языках программирования существует понятие типа файла, который определяется организационной структурой хранения информации в файле и способом доступа к этой информации. Для этих целей Visual Basic6.0. предоставляет полный набор функций, работающих с файлами, папками и устройствами, дающий возможность производить с ними все необходимые действия. Различают следующие типы файлов:

Файлы последовательного доступа. Эти файлы называются последовательными, потому что могут читаться только с самого начала по порядку. Запись нельзя прочитать из середины файла. Каждая запись является строкой текста, которая заканчивается специальными символами – разделителями.

Файлы произвольного доступа. Это структурированные файлы, которые содержат информацию в виде записей. Примером могут служить файлы баз данных.

Двоичные (бинарные) файлы. В принципе, это те же файлы с последовательным доступом, но информация в них не организована в строки.

Традиционный подход при работе с файлами

Традиционный подход при работе с любыми файлами состоит из трех этапов:

- открытие файла;
- чтение или запись информации в файл;
- закрытие файла.

Таблица 4. *Функции и операторы для работы с файлами*

Функции, операторы	**Описание**

Open	Открывает файл
Close	Закрывает все файлы
Reset	Закрывает все открытые файлы
Print	Записывает данные в файл
FileCopy	Копирует файл
EOF	Определяет метку конца файла
FileAttr	Возвращает режим доступа открытого файла
FileDateTime	Возвращает дату и время создания файла
FileLen	Возвращает размер файла в байтах
FreeFile	Возвращает номер свободного идентификатора файла (дескриптора)
GetAttr	Получает атрибуты файла
SetAttr	Устанавливает атрибуты файла
Loc	Возвращает номер текущей позиции в файле
LOF	Возвращает размер открытого файла в байтах
Seek	Устанавливает номер позицию или запись в файле
Dir	Возвращает содержимое текущей папки
Kill	Удаляет файл
Lock	Блокирует файл
Unlock	Снимает блокировку файла
Name	Задает (переименовывает) имя файла
Get#	Читает данные из файла
Input #	Читает данные из файла
LineInput#	Читает строку из файла
Put#	Записывает данные в файл
Write#	Записывает данные в файл

Открытие файлов

Работа с каждым из типов файлов имеет свои особенности. Однако есть два действия, общие для всех типов файлов - их открытие и закрытие. Понятно, что перед тем как записать данные

в файл или прочитать данные из файла, необходимо сначала открыть этот файл. Открытие файла выполняется оператором ***Open***:

Open ИмяФайла For Режим доступа As #НомерФайла

где:

- *ИмяФайла* - имя открываемого файла.
- *Режим доступа* - определяет способ доступа к файлу и для файлов последовательного доступа может иметь следующие значения:

- *Input* – файл открывается для чтения информации из него**;**

- *Output* – файл открывается для записи в него информации;

- *Append* – файл открывается для записи в него информации;

- *Номер файла* – целое число от 1 до 511. После открытия файла операторы и функции, работающие с файлом, обращаются к нему не по имени, а по номеру. Если номер открываемого файла специально не контролируется и не задается программой, его можно узнать с помощью функции *FreeFile*, возвращающей последний свободный номер открываемого файла.

Закрытие файлов

Закрытие файлов выполняется очень просто. Для этого необходимо использовать оператор ***Close***, имеющий следующий синтаксис:

Close [#НомерФайла][, #НомерФайла] [, #НомерФайла]…

где:

- *НомерФайла* - список закрываемых файлов.

ЛЕКЦИЯ № 16

ТЕМА: ФАЙЛЫ ПОСЛЕДОВАТЕЛЬНОГО ДОСТУПА

Чтения информации из текстового файла последовательного доступа выполняется с помощью операторов ***Input #*** и ***Line Input #.***

Оператор ***Input #*** имеет следующий синтаксис:

Input # НомерФайла, ИмяПеременной

где:

- *НомерФайла* - номер файла, аналогичный номеру файла в операторе Open;
- *ИмяПеременной* - список переменных.

Оператор ***Input#*** может читать не всю текстовую строку, а только ее часть. Переменные, используемые в операторе, могут быть как строкового типа, так и числового. Оператор *Input#* обычно используют для чтения файлов, созданных с помощью оператора *Write#.*

Для построчного чтения данных из последовательного файла применяется оператор *Line Input #*. Синтаксис этого оператора следующий:

Line Input # НомерФайла, ИмяПеременной

где:

- *НомерФайла* - номер файла, аналогичный номеру файла в операторе Open;
- *ИмяПеременной* - имя переменной;

Оператор *LineInput#* читает из файла строку текста и присваивает прочитанное значение переменной типа String. Признаком конца строки служит символ перехода на новую строку. При просмотре текстового файла в программе Блокнот эти символы не видны на экране. Если файл создавался с помощью программы, написанной на Visual Basic, то символ добавляется в файл операторами Write# или Print#, отвечающими за запись информации в открытый файл.

Запись в файл данных

Запись в файл последовательного доступа осуществляется с помощью оператора Print# и Write#. Синтаксис оператора Print# следующий:

Print #НомерФайла, [СписокВывода]

где:

- *НомерФайла* - номер файла, аналогичный номеру файла в операторе Open;
- *СписокВывода* – перечень строковых или числовых выражений, значения которых записываются в файл;

Оператор *Print#* после каждого элемента списка вставляет в файл один пробел, если выражения в списке вывода разделены точкой с запятой, или несколько пробелов, если выражения в списке вывода разделены запятой. Файл, созданный с помощью оператора *Print #,* читается обычно оператором *Line Input* **#.**

Синтаксис оператора Write# следующий:

Write #НомерФайла, [СписокВывода]

где:

- *НомерФайла* - номер файла, аналогичный номеру файла в операторе Open;
- *СписокВывода* – перечень строковых или числовых выражений, значения которых записываются в файл;

Оператор Write# после каждого элемента списка вставляет в файл запятую. Строка текста записывается в кавычках. Файл, созданный с помощью оператора *Print#,* читается обычно оператором *Input #.*

Файлы произвольного доступа

Файл с произвольным доступом обладает заданной структурой и состоит из записей. Каждая запись в файле имеет определенный размер и номер. Доступ к данным в файле с произвольным доступом осуществляется именно по номеру записи. Данные из файла такого типа читаются и записываются записями.

Открытие файла произвольного доступа

Для открытия файла с произвольным доступом используется оператор *Open*. Синтаксис оператора *Open* выглядит следующим образом:

Open ИмяФайла [For Random] As #НомерФайла Len = ДлинаЗаписи

где:

- *ИмяФайла* - полное имя файла;
- *For Random* – этот режим не обязателен, так как этот параметр устанавливается по умолчанию;
- *НомерФайла* - номер файла;
- *ДлинаЗаписи* - длина записи в байтах.

Как видно, в отличие от файла с последовательным доступом при открытии файла с произвольным доступом необходимо обязательно указывать длину записи. Закрытие файла осуществляется с помощью оператора *Close*. Формат оператора такой же, как и для файлов с последовательным доступом.

Чтение данных из файла

Данные из файла произвольного доступа, как правило, считываются записями. Для этого используется оператор *Get#,* который имеет следующий синтаксис:

Get #НомерФайла, [НомерЗаписи], Переменная

где:

- *НомерФайла* - номер файла;
- *НомерЗаписи* - номер записи в файле;
- *Переменная* – имя переменной.

Если параметр *НомерЗаписи* в функции *Get#* не указан, считывается текущая запись, на которой позиционирован указатель.

Запись в файл произвольного доступа

Для записи данных в файл произвольного доступа используется оператор *Put #*, имеющий следующий синтаксис:

Put #НомерФайла, [НомерЗаписи], Переменная

где:

- *НомерФайла* - номер файла, аналогичный номеру в операторе Open;

- *НомерЗаписи* - целочисленное выражение, которое задает номер записи в файле;
- *Переменная* - переменная, указывающая источник записываемых данных.

При использовании оператора *Put #* необходимо иметь в виду, что данные в записи с указанным в операторе номером будут заменены теми, которые записываются в файл. для добавления записей в файл необходимо указывать номер записи.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ С ПРИМЕРАМИ

Программирования линейных алгоритмов

Пример:

Вычислить значение выражений по формуле: x-10Sinx+$|x^4 - x^5|$

Решение примера состоит из трех частей.

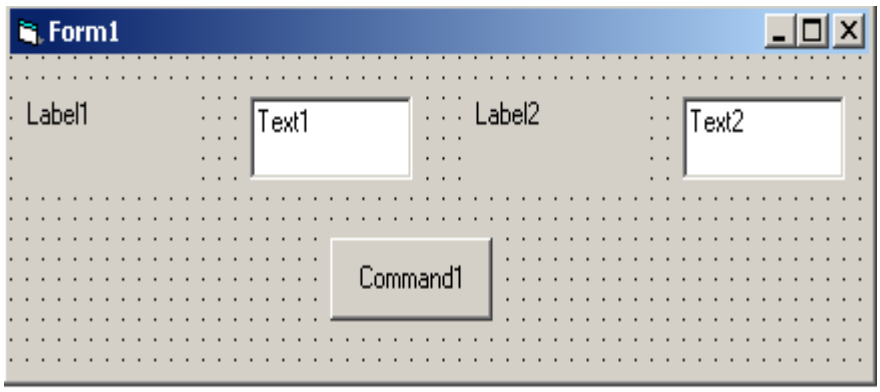
В первой части примера приведены количества объектов использованных в форме для вычисления линейного алгоритма. Форма состоит из следующих элементов (объектов):

1. *Text1* – для ввода значения “**X**”
2. *Text2* – для вывода “**Результата**”
3. *Label1* – указывает метку “**X**”
4. *Label2* – указывает метку “**Результата**”
5. *CommandButton* – “**Вычисляет**” значение выражений алгоритма

Во второй части примера приведен код программы на языке VisualBasic.

В третьей части примера приведен результат написанной программы.

Часть I



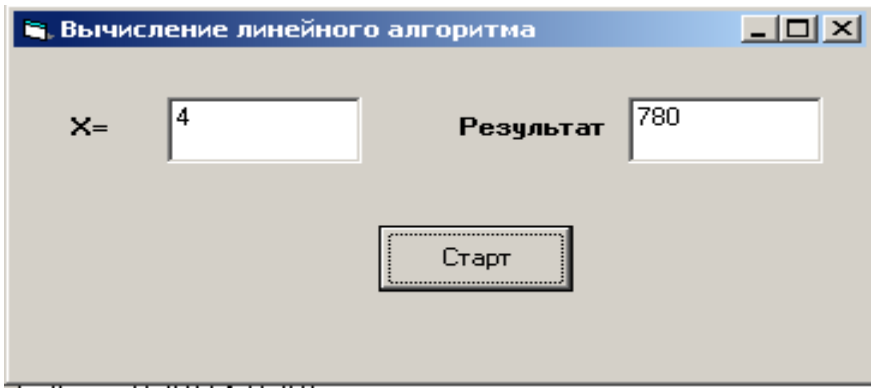
Часть II

Код программы:

```
Option Explicit

Private Sub Command1_Click()

Dim x As Integer

Dim z As Integer

x = Text1.Text

z = (x - 10 * Sin(x)) + Abs(x ^ 4 - x ^ 5)

Text2.Text = z

End Sub
```

Часть III

Результат:



Примеры линейных алгоритмов:

Вычислить значения выражений по формулам:

1. $\frac{b+\sqrt{b^2+4ac}}{2a}-a^3c+b^{-2}$;

2. $\frac{a}{c}\cdot\frac{b}{d}-\frac{ab-c}{cd}$;

3. $\frac{Sinx+Cosy}{Cosx-Siny}\cdot tgxy$;

4. $\frac{3+e^{y-1}}{1+x^2\left|y-tgx\right|}$;

5. $\ln\left|(y-\sqrt{|x|})\left(x-\frac{y}{x+\frac{x^2}{4}}\right)\right|$;

6. $\frac{\ln|Cosx|}{(1+x^2)}$;

7. $\frac{1+Sin^2(x+y)}{2+\left|x-\frac{2x}{1+x^2y^2}\right|}+x$;

8. x-$10^{Sinx}+Cos(x-y)$;

9. $\frac{1+Sin\sqrt{x+1}}{Cos(12y-4)}$;

10. $\frac{Cosx}{\pi-2x}+16x\cdot Cos(xy)-2$;

11. $z=\frac{x^5\cdot\sqrt[3]{ax+\sqrt{|a+x|}}}{4\cos^2(ax)}+g^2+wt,\ где\ g=x+24{,}8$

12. $v=e^{aw}+p,\ где\ w=\frac{a^5+Sinab}{\sqrt{ab}+tgb},\ a=Cosb+5{,}1\cdot10^{-3}$

13. $y = \left[\sqrt[5]{m^4 + m - Cos^2(x+m)/(m-m^2) + \sqrt{m}} \right] m + x, \text{ где } m = \frac{\mu}{3},\ x = \mu^2$

14. $y = \lg\left|arctg(x) - Sin(ax)\right|, \text{ где } x = 10^{-a+Cos(a+1)}$

15. $I = \frac{1}{Sinx} + \lg\left|ctg\left(\frac{x+1}{x-1}\right) + 2{,}5\right|$

16. $y = ax^2 - Cos(bx) + z/2, \text{ где } x = (a+b)/2, \qquad z = \frac{1+x}{1+\frac{1+x}{x}}$

Программирования циклических алгоритмов

Пример:

Дано натуральное число n. Вычислить: S=1!+2!+3!+…+n! (n>1)

Решение примера состоит из трех частей.

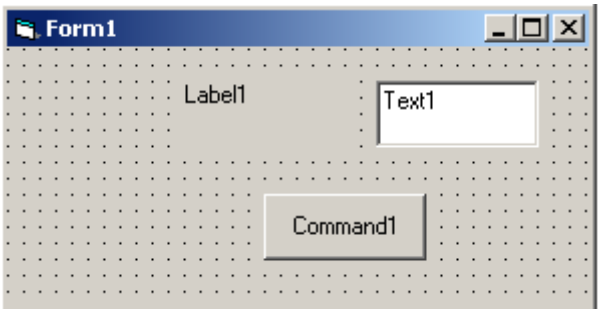
В первой части примера приведены количества объектов использованных в форме для вычисления циклического алгоритма. Форма состоит из следующих элементов (объектов):

1. *Text1* – для ввода значения “**N**”
2. *Label1* – указывает метку “**N**”
3. *CommandButton* – “**Вычисляет**” значение выражений алгоритма
4. *Print* – оператор для вывода результата в форме.

Во второй части примера приведен код программы на языке Visual Basic.

В третьей части примера приведен результат написанной программы.

Часть I



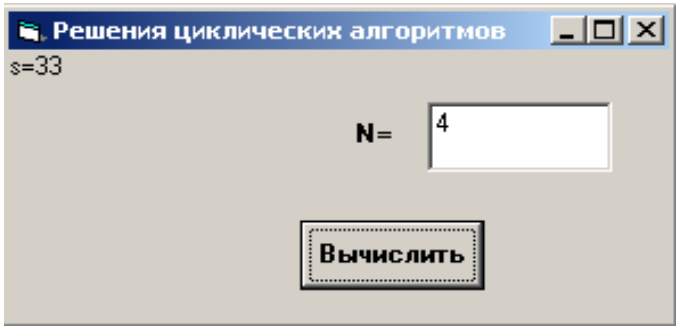
Часть II

Код программы:

```
Option Explicit
Private Sub Command1_Click()
Dim fact As Integer, N As Integer
Dim i As Integer, s As Long
Form1.Cls
N = Val(Text1.Text)
s = 0
fact = 1
For i = 1 To N
    fact = fact * i
    s = s + fact
Next i
Print "s=" & s
End Sub
```

Часть III

Результат:



Примеры циклических алгоритмов:

1. Дано натуральное число N. Вычислить: S=1-

$\frac{1}{2}+\frac{1}{4}-\frac{1}{8}+...+(-1)^n\cdot\frac{1}{2^n}$;

2. Дано натуральное число N. Вычислить:

S=$\frac{1}{Sin1}+\frac{1}{Sin1+Sin2}+...+\frac{1}{Sin1+Sin2+...+SinN}$;

3. Дано натуральное число N. Вычислить произведение первых N сомножителей: P= $\frac{2}{3}\cdot\frac{4}{5}\cdot\frac{6}{7}\times...\times\frac{2N}{2N+1}$;

4. Дано действительное число X. Вычислить:

x-$\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}-\frac{x^{11}}{11!}+\frac{x^{13}}{13!}$;

5. Дано действительное число X.

Вычислить: $\frac{(x-1)(x-3)(x-7)\times...\times(x-63)}{(x-2)(x-4)(x-8)\times...\times(x-64)}$;

6. Дано натуральное число N. Вычислить:

P=$\left(1-\frac{1}{2^2}\right)\left(1-\frac{1}{3^2}\right)\times\ldots\times\left(1-\frac{1}{n^2}\right)$, где n>2;

7. Дано натуральное число N. Вычислить:

S=$\frac{1}{3^2}+\frac{1}{5^2}+\frac{1}{7^2}+\ldots+\frac{1}{(2n+1)^2}$;

8. Дано натуральное число N. Вычислить:

S=1·2+2·3·4+…+n·(n+1)$\times\ldots\times 2n$;

9. Даны действительное число a, натуральное число n. Вычислить:

P=a(a-n)(a-2n)$\times\ldots\times$(a-n^2).

10. Даны действительное число a, натуральное число n. Вычислить:

S=$\frac{1}{a}+\frac{1}{a^2}+\frac{1}{a^4}+\ldots+\frac{1}{a^{2n-2}}$;

Программирования разветвляющихся алгоритмов

Пример:

Для данного X вычислить значение функций: F(x)=$\begin{cases}3x-9, если x\le 7\\ \frac{1}{x^2-4}, если x>7\end{cases}$

Решение примера состоит из трех частей.

В первой части примера приведены количества объектов использованных в форме для вычисление алгоритма. Форма состоит из следующих элементов (объектов):

1. *CommandButton* – “**Вычисляет**” значение выражений алгоритма

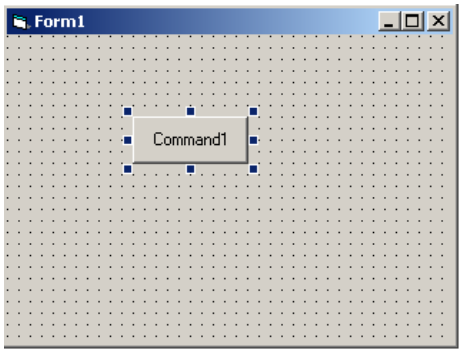
2. *InputBox* – диалоговое окно ввода, для значения “**X**”

3. *MsgBox* – диалоговое окно сообщений для вывода “**Результата**”.

Во второй части примера приведен код программы на языке Visual Basic.

В третьей части примера приведен результат написанной программы.

Часть I



Часть II

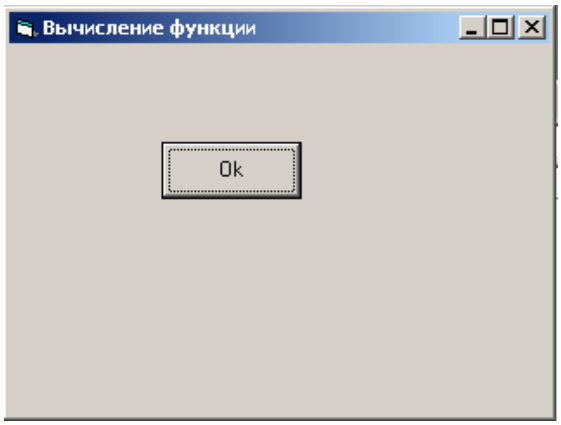
Код программы:

```
Option Explicit

Private Sub Command1_Click ()

Dim x As Integer

Dim F As Single

x = InputBox ("x")

If x <= 7 Then
```

```
F = 3 * x - 9
If x > 7 Then
F = 1 / (x ^ 2 - 4)
End If
End If
MsgBox "F=" & F ,, "Ответ"
End Sub
```

Часть III

Результат:



Примеры разветвляющихся алгоритмов:

Для данного X вычислить значение функций:

1. $F(x)=\begin{cases} x^2-3x+9, если\ \chi>3 \\ \dfrac{1}{x^3+6}, если x\le 3 \end{cases}$ 2.

$F(x)=\begin{cases} 9, если\ \chi\le -3 \\ \dfrac{1}{x^2+1}, если x>-3 \end{cases}$

3. $F(x)=\begin{cases} -3x+9, если\ \chi\le 7 \\ \dfrac{1}{x\ -7}, если x>7 \end{cases}$ 4.

$F(x)=\begin{cases} x^2, если\ 0\le\chi\le 3 \\ 4, если x>3 или x<0 \end{cases}$

5. $F(x)=\begin{cases} 0, если\ \chi\le 1 \\ \dfrac{1}{x+6}, если x>1 \end{cases}$ 6.

$F(x)=\begin{cases} x^2+4x+5, если\ \chi\le 2 \\ \dfrac{1}{x^2+4x+5}, если x>2 \end{cases}$

7. $F(x)=\begin{cases} x^2-x, если\ 0\le\chi\le 1 \\ x^2-Sin\pi x^2, если x>1 или x<0 \end{cases}$ 8. $F(x)=\begin{cases} Sinx, если\ \chi\le 0 \\ Cosx, если x>0 \end{cases}$

9. $F(x)=\begin{cases} x^2\sqrt{y+x^3}-x, если\ 0\le\chi\le 1 \\ x^2-Tg\pi x^2-\lambda^3, если x>1\ или\ x<0 \end{cases}$

10. $F(x)=\begin{cases} \sqrt{\dfrac{Cosx+Sinx}{e^{x^2+x}}}\cdot x^2-x, если\ 0\le\chi\le 1 \\ x\ -Arctg\pi x\ , если x>1 или x<0 \end{cases}$

Программирование с использованием оператора “SELECT CASE”

Пример:

Составить программу, которая по данному числу (1-4) выводит название времена года, по данному числу (1-12) выводит название соответствующего ему месяца.

Решение примера состоит из трех частей.

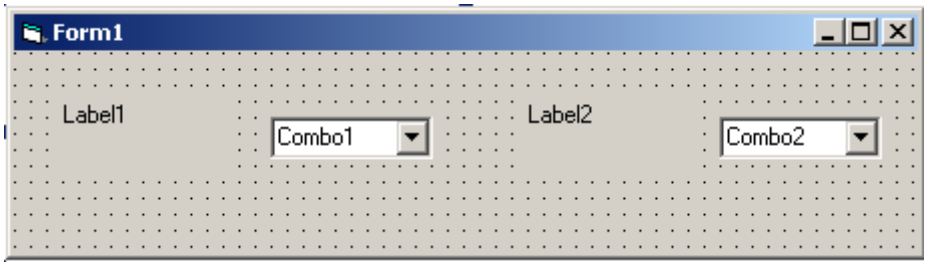
В первой части примера приведены количества объектов использованных в форме для вывода названии времен года, и соответствующего ему месяца. Форма состоит из следующих элементов (объектов):

1. *Label1* – указывает метку “**Времена года**”
2. *Label2* – указывает метку “**Месяцы**”
3. *ComboBox1* – выводит “**Времена года**”
4. *ComboBox2* – выводит “**Месяцы**” всех времен года

Во второй части примера приведен код программы на языке Visual Basic.

В третьей части примера приведен результат написанной программы.

Часть I



Часть II

Код программы:

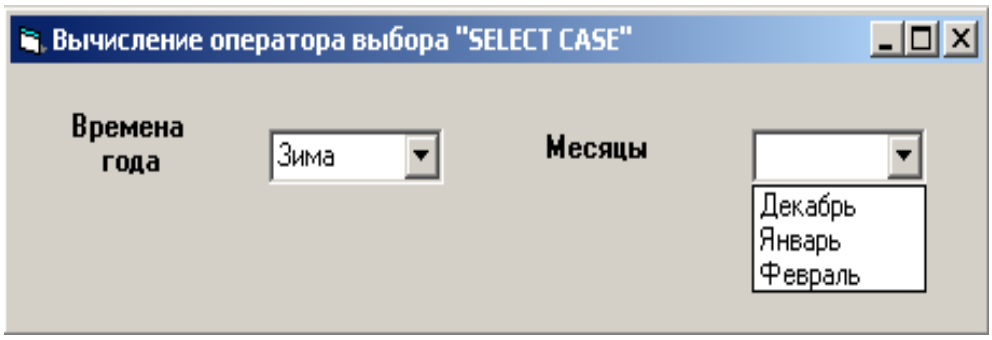
```
Option Explicit

Private Sub Combo1_Click()

Dim A As String
```

```
Combo2.Clear
A = Trim(Combo1.Text)
Select Case A
Case "Зима"
Combo2.AddItem "Декабрь"
Combo2.AddItem "Январь"
Combo2.AddItem "Февраль"
Case "Весна"
Combo2.AddItem "Март"
Combo2.AddItem "Апрель"
Combo2.AddItem "Май"
Case "Лето"
Combo2.AddItem "Июнь"
Combo2.AddItem "Июль"
Combo2.AddItem "Август"
Case "Осень"
Combo2.AddItem "Сентябрь"
Combo2.AddItem "Октябрь"
Combo2.AddItem "Ноябрь"
End Select
End Sub
Private Sub Command1_Click()
'Dim n As Integer
```

```
'n = Val(Text1.Text)
'Select Case n
'Case 1
'Combo1.AddItem "Зима"
'Case 2
'Combo1.AddItem "Весна"
'Case 3
'Combo1.AddItem "Лето"
'Case 4
'Combo1.AddItem "Осень"
'End Select
End Sub
Private Sub Form_Load()
Combo1.AddItem "Зима"
Combo1.AddItem "Весна"
Combo1.AddItem "Лето"
Combo1.AddItem "Осень"
End Sub
```

Часть III

Результат:



Примеры с использованием оператора выбора " SELECT CASE"

1. Дано целое число в диапазоне 1–7. Вывести строку — название дня недели, соответствующее данному числу (1 — «понедельник», 2 — «вторник» и т. д.).

2. Дано целое число *K*. Вывести строку-описание оценки, соответствующей числу *K* (1-«плохо», 2 — «неудовлетворительно», 3 — «удовлетворительно», 4 — «хорошо», 5 — «отлично»). Если *K* не лежит в диапазоне 1–5, то вывести строку «ошибка».

3. Дан номер месяца — целое число в диапазоне 1–12 (1 — январь, 2 — февраль и т. д.). Вывести название соответствующего времени года («зима», «весна», «лето», «осень»).

4. Дан номер месяца — целое число в диапазоне 1–12 (1 — январь, 2 — февраль и т. д.). Определить количество дней в этом месяце для невисокосного года.

5. Арифметические действия над числами пронумерованы следующим образом: 1 — сложение, 2 — вычитание, 3 —

умножение, 4 — деление. Дан номер действия N (целое число в диапазоне 1–4) и вещественные числа A и B (B не равно 0). Выполнить над числами указанное действие и вывести результат.

6. Единицы длины пронумерованы следующим образом: 1 — дециметр, 2 — километр, 3 — метр, 4 — миллиметр, 5 — сантиметр. Дан номер единицы длины (целое число в диапазоне 1–5) и длина отрезка в этих единицах (вещественное число). Найти длину отрезка в метрах.

7. Единицы массы пронумерованы следующим образом: 1 — килограмм, 2 — миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер. Дан номер единицы массы (целое число в диапазоне 1–5) и масса тела в этих единицах (вещественное число). Найти массу тела в килограммах.

8. Даны два целых числа: D (день) и M (месяц), определяющие правильную дату невисокосного года. Вывести значения D и M для даты, предшествующей указанной.

9. Даны два целых числа: D (день) и M (месяц), определяющие правильную дату невисокосного года. Вывести значения D и M для даты, следующей за указанной.

10. Робот может перемещаться в четырех направлениях («С» — север, «З» — запад, «Ю» — юг, «В» — восток) и принимать три цифровые команды: 0 — продолжать движение, 1 — поворот налево, 1 — поворот направо. Дан символ C — исходное направление робота и целое число N — посланная ему команда.

Вывести направление робота после выполнения полученной команды.

Программирование с использованием оператора “IF…THEN”

Пример:

Даны три числа. Вывести в начале наименьшее, а затем наибольшее из данных чисел.

Решение примера состоит из трех частей.

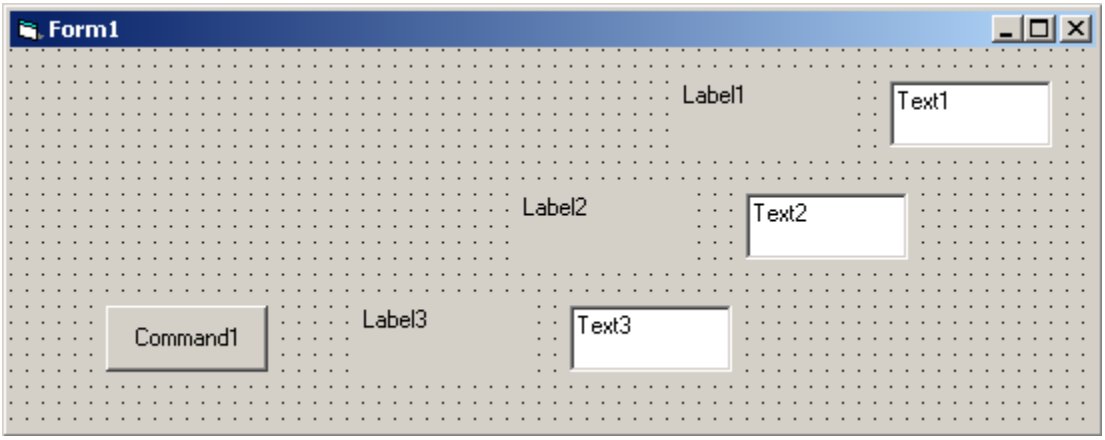
В первой части примера приведены количества объектов, использованных в форме для вывода в начале наименьшего, а затем наибольшего из данных чисел. Форма состоит из следующих элементов (объектов):

1. *Label1* – указывает метку “**A**”
2. *Label2* – указывает метку “**B**”
3. *Label3* – указывает метку “**C**”
4. *Text1* – для ввода значения “**A**”
5. *Text2* – для ввода значения “**B**”
6. *Text3* - для ввода значения “**C**”
7. *Command1* – для вызова основной процедуры

Во второй части примера приведен код программы на языке Visual Basic.

В третьей части примера приведен результат выполнения программы.

Часть I



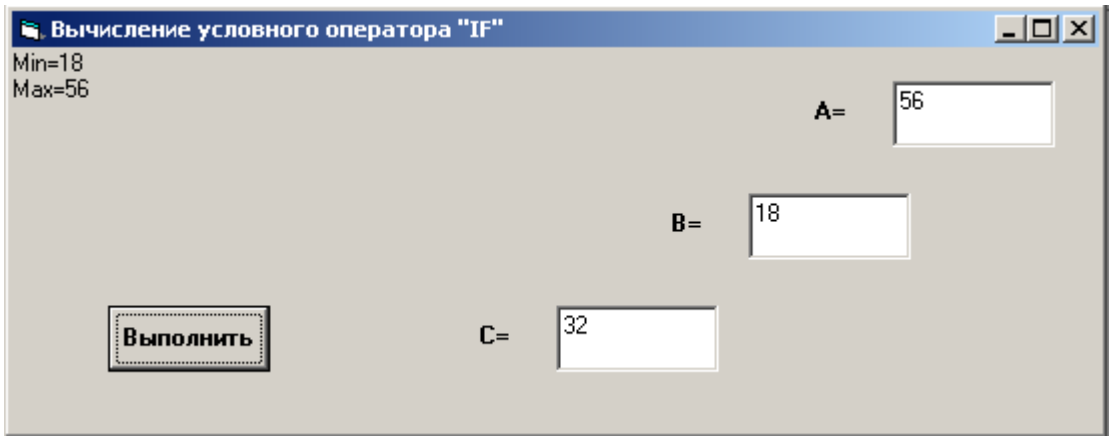
Часть II

Код программы:

```
Option Explicit
Private Sub Command1_Click()
Dim A As Integer
Dim B As Integer
Dim C As Integer
Dim Max, Min As Integer
A = Val(Text1.Text)
B = Val(Text2.Text)
C = Val(Text3.Text)
If A > B Then Max = A: Min = B
If B > C Then Max = B: Min = C
If C > A Then Max = C: Min = A
Print "Min=" & Min
Print "Max=" & Max
End Sub
```

Часть III

Результат:



Примеры с использованием условного оператора "If…Then"

1. Даны три действительных числа. Возвести в квадрат те из них, значения которых неотрицательны, и в четвертую степень – отрицательные.

2. Определить делителем каких чисел A, B, C является число K.

3. Перераспределить значения переменных X и Y так, чтобы в X оказалось большее из этих значений, а в Y меньшее.

4. Из трех данных вещественных чисел X, Y, Z выбрать наименьший.

5. Подсчитать количество целых чисел среди чисел A, B, C.

6. Написать программу нахождения суммы большего и меньшего из трех чисел.

7. Определить, является ли целое число N четным двузначным числом.

8. Подсчитать количество положительных чисел среди чисел A, B, C.

9. Даны три числа. Найти сумму двух наибольших из них.

10. Даны три числа. Найти среднее из них (то есть число, расположенное между наименьшим и наибольшим).

Программирование с использованием оператора “FOR…NEXT”

Пример:

Даны целые числа K и N (N>0). Вывести N раз число K.

Решение примера состоит из трех частей.

В первой части примера приведены количества объектов, использованных в форме для вывода N раз числа K. Форма состоит из следующих элементов (объектов):

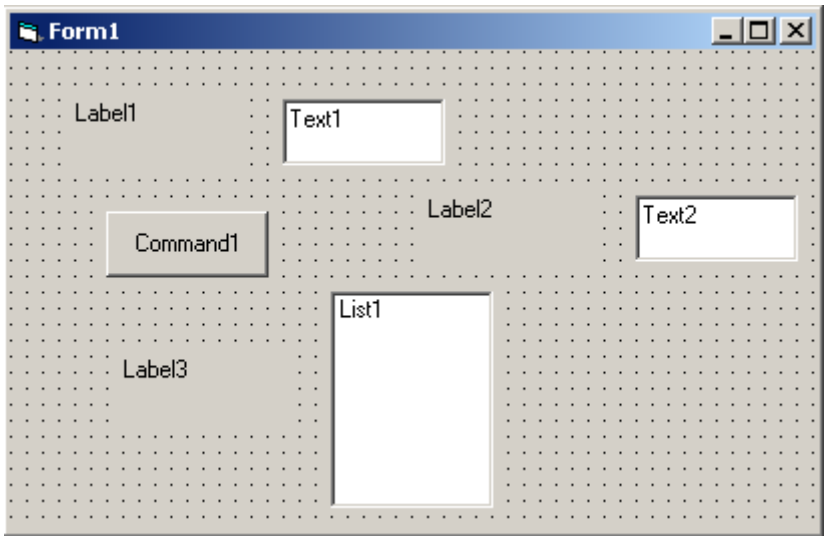
1. *Label1* – указывает метку “**N**”
2. *Label2* – указывает метку “**K**”
2. *Label3* – указывает метку “**Результат**”

4. *Text1* – для ввода значения “**N**”
5. *Text2* – для ввода значения “**K**”
6. *ListBox1* - для вывода “**N**” раз число “**K**”
7. *Command1* – для вызова основной процедуры

Во второй части примера приведен код программы на языке Visual Basic.

В третьей части примера приведен результат выполнения программы.

Часть I



Часть II

Код программы:

```
Option Explicit

Private Sub Command1_Click()

Dim N As Integer

Dim K As Integer

Dim i As Integer
```

```
N = Val(Text1.Text)
K = Val(Text2.Text)
For i = 1 To N
List1.AddItem K
Next i
End Sub
```

Часть III

Результат:

Примеры с использованием оператора “For…Next”

1. Дано вещественное число-цена 1кг конфет. Вывести стоимость 1, 2,…, 10кг конфет.

2. Дано вещественное число-цена 1кг конфет. Вывести стоимость 0.1, 0.2,…, 1кг конфет.

3. Даны два целых числа A и B(A<B). Найти сумму всех целых чисел от A до B включительно.

4. Даны два целых числа A и B(A<B). Найти произведение всех целых чисел от A до B включительно.

5. Дано целое число N(>0). Найти значение выражения 1.1-1.2+1.3-… (N слагаемых, знаки чередуются). Условный оператор не использовать.

6. Дано вещественное число A и целое число N(>0). Найти A в степени N: $A^N=A\cdot A\cdot\ldots\cdot A$ (числа A перемножаются N раз).

7. Дано вещественное число A и целое число N(>0). Используя один цикл, вывести все целые степени числа A от 1 до N.

8. Дано вещественное число A и целое число N(>0). Используя один цикл. Найти сумму $1+A+A^2+A^3+\ldots+A^N$.

9. Даны два целых числа A и B(A<B). Вывести в порядке возрастания все целые числа, расположенные между A и B(включая сами числа A и B), а также количество N этих чисел.

10. Даны целые числа K и N (N>0). Вывести N раз число K.

Программирование с помощью конструкции “DO WHILE”

Пример:

Дано целое число N(>0). Используя операции деления нацело и взятия остатка от деления, найти количество и сумму его цифр.

Решение примера состоит из трех частей.

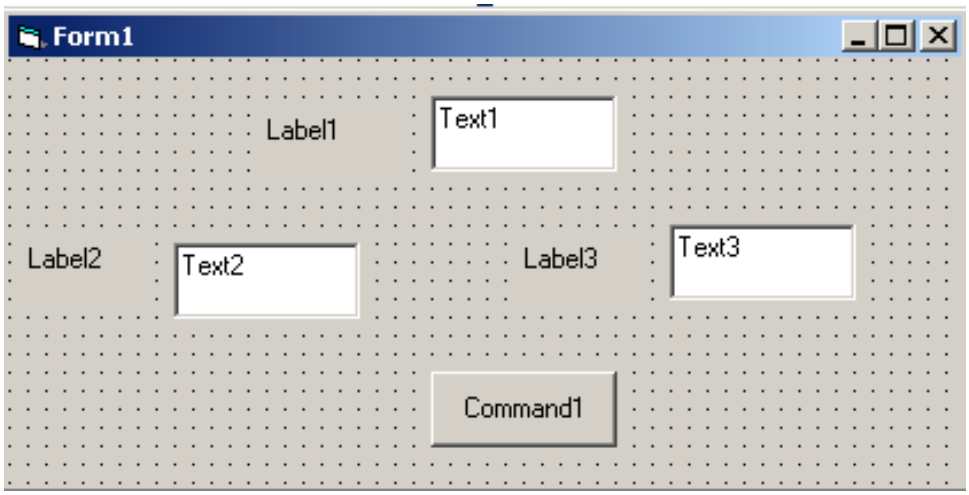
В первой части примера приведены количества объектов использованных в форме для вычисления количество и сумму цифр данного числа. Форма состоит из следующих элементов (объектов):

1. *Label1* – указывает метку “**N**”
2. *Label2* – указывает метку “**K**”
2. *Label3* – указывает метку “**S**”
4. *Text1* – для ввода значения “**N**”
5. *Text2* – выводить значения “**S**”-сумму цифр
6. *Text3* - выводить значения “**K**” – количество цифр
7. *Command1* – для вызова основной процедуры

Во второй части примера приведен код программы на языке VisualBasic.

В третьей части примера приведен результат выполнения программы.

Часть I



Часть II

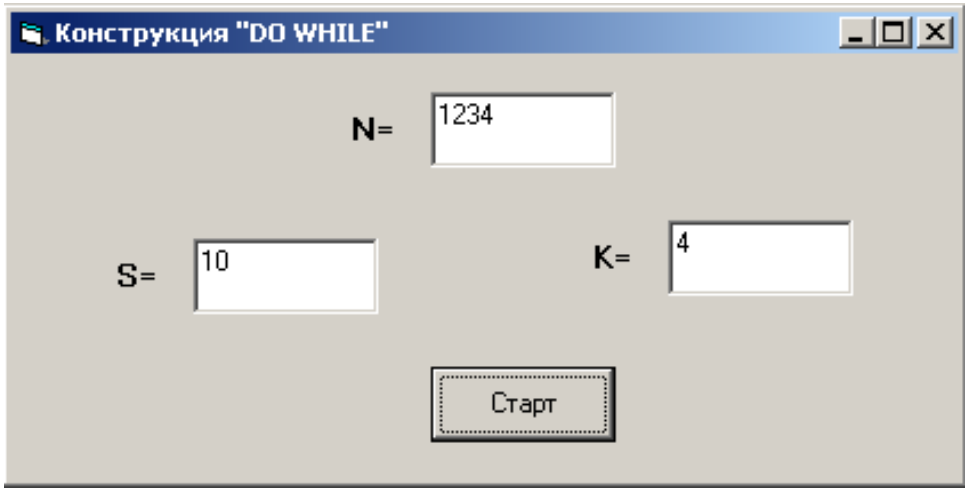
Код программы:

```
Private Sub Command1_Click()
Dim n As Integer
Dim k As Integer
Dim s As Integer
Form1.Cls
n = Val(Text1.Text)
s = 0: k = 0
Do While n > 0
s = s + (n Mod 10)
k = k + 1
n = n \ 10
Loop
Text2.Text = s
Text3.Text = k
```

End Sub

Часть III

Результат:



Примеры с использованием конструкции “DO WHILE”:

1. Даны целые положительные числа N и K. Используя только операции сложения и вычитания, найти частное от деления нацело N на K, а также остаток от этого деления.

2. Дано целое число N(>0), являющееся некоторой степенью числа 2: $N=2^K$. Найти целое число K-показатель этой степени.

3. Дано целое число N(>1). Вывести наименьшее из целых чисел K, для которых сумма 1+2+…+K будет больше или равна N, и саму эту сумму.

4. Дано целое число N(>1). Вывести наименьшее из целых чисел К, для которых сумма 1+2+…+К будет меньше или равна N, и саму эту сумму.

5. Дано целое число А(>1). Вывести наименьшее из целых чисел К, для которых сумма 1+1/2+…+1/К будет больше А, и саму эту сумму.

6. Дано целое число А(>1). Вывести наименьшее из целых чисел К, для которых сумма 1+1/2+…+1/К будет меньше А, и саму эту сумму.

7. Дано целое число N(>0). С помощью операций деления нацело и взятия остатка от деления определить, имеются ли в записи числа N

нечетные цифры. Если имеются, то вывести True, если нет – вывести False.

8. Дано целое число N(>0). Если оно является степенью числа 3, то вывести True, если не является – вывести False.

9. Начальный вклад в банке равен 1000 Сомони. Через каждый месяц размер вклада увеличивается на Р процентов от имеющейся суммы (Р-вещественное число, 0<P<25). По данному Р определить, через сколько месяцев размер вклада превысит 1100 Сомони, и вывести найденное количество месяцев К(целое число) и итоговый размер вклада S(вещественное число).

10. Спортсмен-лыжник начал тренировки, пробежав в первый день 10км. Каждый следующий день он увеличил длину пробега на P процентов от пробега предыдущего дня (P- вещественное, 0<P<50). По данному P определить, после какого дня суммарный пробег лыжника за все дне превысит 200км, и вывести найденное количество дней K(целое) и суммарный пробег S(вещественное число).

Программирование одномерных массивов

Пример:

Даны массивы A и B одинакового размера N. Поменять местами их содержимое и вывести вначале элементы преобразованного массива A, а затем – элементы преобразованного массива B.

Решение примера состоит из трех частей.

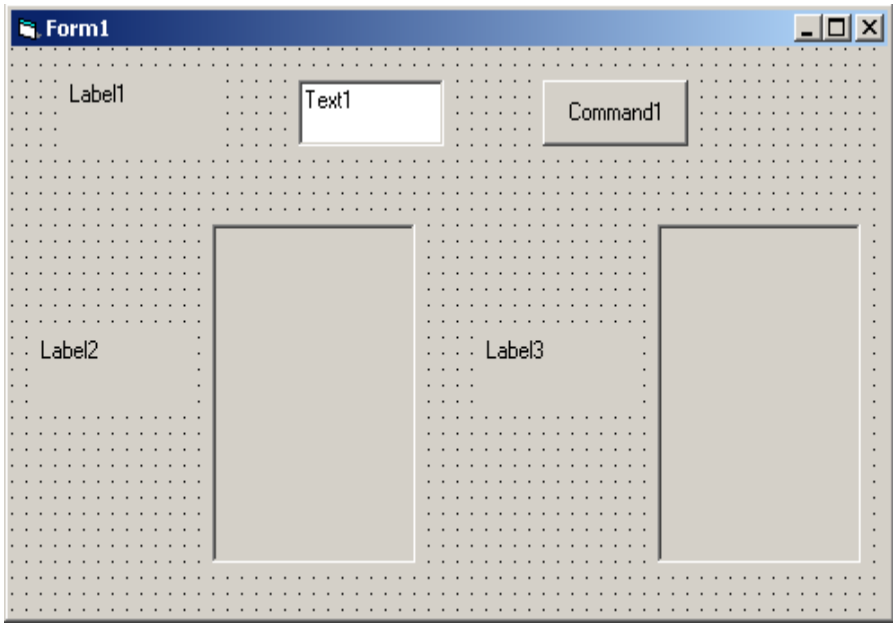
В первой части примера приведены количества объектов, использованных в форме для вычисления одномерного массива. Форма состоит из следующих элементов (объектов):

1. *Label1* – указывает метку “**N**”
2. *Label2* – указывает метку “**Матрицу A**”
2. *Label3* – указывает метку “**Матрицу B**”
4. *Text1* – для ввода значения “**N**”
5. *PictureBox1*– выводит значения “**Матрицу A**”
6. *PictureBox2* - выводит значения “**Матрицу A**”
7. *Command1* – для вызова основной процедуры

Во второй части примера приведен код программы на языке Visual Basic.

В третьей части примера приведен результат выполнения программы.

Часть I



Часть II

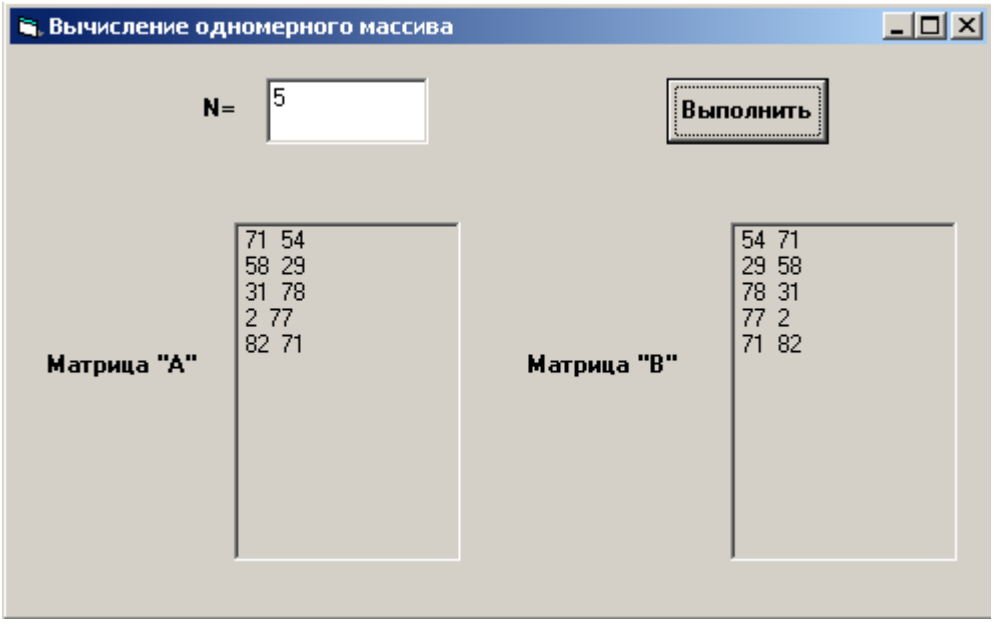
Код программы:

```
Option Explicit

Private Sub Command1_Click()

Dim n As Integer, temp As Integer, i As Integer

Dim a() As Integer, b() As Integer

Picture1.Cls

Picture2.Cls

n = Val(Text1.Text)

ReDim a(n)
```

```
ReDim b(n)
For i = 1 To n
a(i) = Int(Rnd() * 100) + 1
b(i) = Int(Rnd() * 100) + 1
Picture1.Print a(i);
Picture1.Print b(i)
Next i
Picture1.Print
For i = 1 To n
temp = a(i)
a(i) = b(i)
b(i) = temp
Picture2.Print a(i);
Picture2.Print b(i)
Next i
Picture2.Print
End Sub
```

Часть III

Результат:



Примеры по одномерным массивам:

1. Дан массив размера N. Вывести его элементы в обратном порядке.

2. Дано целое число N (> 0). Сформировать и вывести целочисленный массив размера N, содержащий N первых положительных нечетных чисел: 1, 3, 5, … .

3. Дан массив A размера N. Найти минимальный элемент из его элементов с четными номерами A_1, A_3, A_6, … .

4. Даны целые числа N>2, A и B. Сформировать и вывести целочисленный массив размера N, первый элемент которого равен A, второй равен B, а каждый последующий элемент равен сумме всех предыдущих.

5. Дано целое число N>0. Сформировать и вывести целочисленный массив размера N, содержащий N первых положительных нечетных чисел: 1, 3, 5, … .

6. Дан целочисленный массив размера *N*. Найти количество различных элементов в данном массиве.

7. Дана последовательность целых чисел a_1, a_2, …a_n. Выяснить, какое число встречается раньше-положительное или отрицательное.

8. Дана массив натуральных чисел. Найти сумму элементов, кратных данному К.

9. Дан целочисленный массив размера *N*. Вывести вначале все содержащиеся в данном массиве четные числа в порядке возрастания их индексов, а затем — все нечетные числа в порядке убывания их индексов.

10. Дана последовательность действительных чисел a_1, a_2, …a_n. Выяснить, будет ли она возрастающий.

Программирование двумерных массивов

Пример:

Дана матрица размера M x N. Для каждой строки матрицы найти сумму ее элементов.

Решение примера состоит из трех частей.

В первой части примера приведено количества объектов использованных в форме для вычисления двумерных массивов. Форма состоит из следующих элементов (объектов):

1. *Label1* – указывает метку “**M**”
2. *Label2* – указывает метку “**N**”

2. *Label3* – указывает метку “**Вывода матрицы**”

3. *Label4* – указывает метку “**Сумму каждой строки матрицы**”

4. *Text1* – для ввода значения “**M**”

5. *Text2* – для ввода значения “**N**”

6. *PictureBox1*– выводит значения “**Элементы матрицы**”

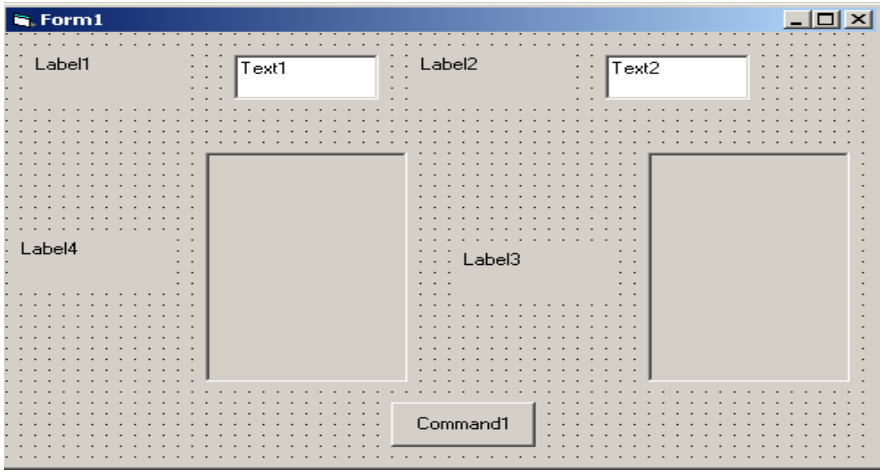
7. *PictureBox2* - выводит значения “**Сумму каждой строки матрицы**”

8. *Command1* – для вызова основной процедуры

Во второй части примера приведен код программы на языке VisualBasic.

В третьей части примера приведен результат выполнения программы.

Часть I



Часть II

Код программы:

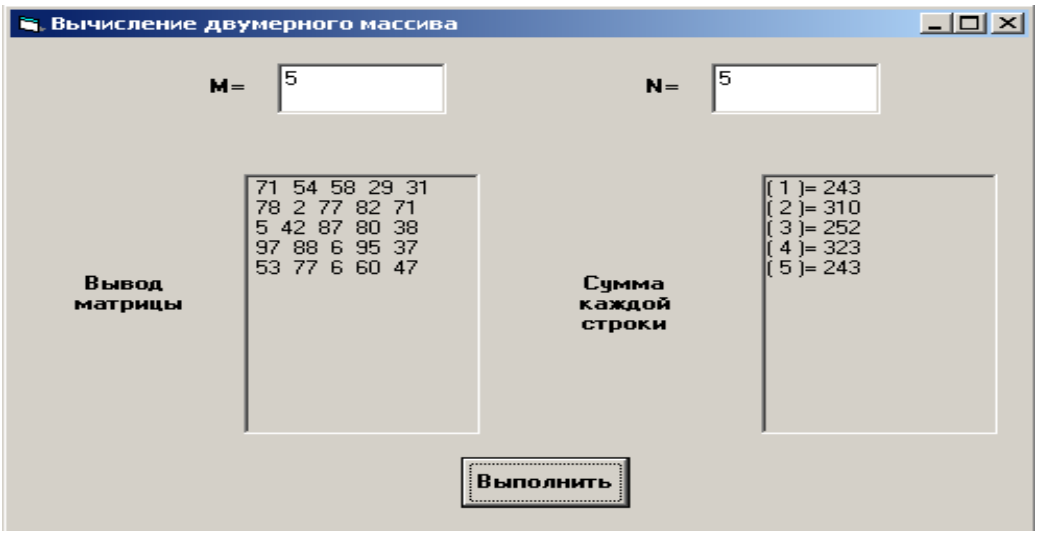
```
Option Explicit

Dim m As Integer, n As Integer

Dim i As Integer, j As Integer
```

```
Dim a() As Integer, b() As Integer
Private Sub Command1_Click()
Picture1.Cls
Picture2.Cls
m = Text1.Text
n = Text2.Text
ReDim a(m, n)
ReDim b(m)
For i = 1 To m
For j = 1 To n
a(i, j) = Int(Rnd() * 100) + 1
Picture1.Print a(i, j);
Next j
Picture1.Print
Next i
For i = 1 To m
b(i) = 0
For j = 1 To n
b(i) = b(i) + a(i, j)
Next j
Picture2.Print "("; i; ")="; b(i)
Next i
Picture2.Print
```

End Sub

Часть III

Результат:



Примеры по двумерным массивам:

1. Даны целые положительные числа M и N. Сформировать целочисленную матрицу M x N, у которой все элементы I–ой строки имеют значения 10·I(I=1,…,M).

2. Даны целые положительные числа M, N и набор из N чисел. Сформировать матрицу размера M x N, у которой в каждой строке содержаться все числа из исходного набора (в том же порядке).

3. Даны целые положительные числа M и N. Сформировать целочисленную матрицу M x N, у которой все элементы J–ого столбца имеют значения 5·J(J=1,…,N).

4. Дана матрица размера M x N и целое число K($1 \le K \le M$). Найти сумму и произведение элементов K-го столбца данной матрицы.

5. Дана матрица размера M x N. В каждом столбце матрицы найти максимальный элемент.

6. Дана матрица размера M x N. Удалить столбец, содержащую максимальный элемент матрицы.

7. Дана матрица размера M x N. Удалить столбец, содержащую максимальный элемент матрицы.

8. Дана матрица размера M x N. Для каждого столбца матрицы с четным номером (2, 4, …). Найти сумму его элементов. Условный оператор не использовать.

9. Дана матрица размера M x N. Для каждой строки матрицы с нечетным номером (1, 3, …) найти сумму его элементов. Условный оператор не использовать.

10. Дана матрица размера M x N. Продублировать строку матрицы, содержащую ее максимальный элемент.

Программирование с использованием функций

Пример:

Описать функцию IsLeapYear(Y) логического типа, которая возвращает True, если год Y(целое положительное число) является високосным, и False в противном случае. Вывести значение функции IsLeapYear для пяти данных значений параметра Y. Високосным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400.

Решение примера состоит из трех частей.

В первой части примера приведены количества объектов, использованных в форме для вычисления функции. Форма состоит из следующих элементов (объектов):

1. *Command1* – “**Вычисляет**” значение функций

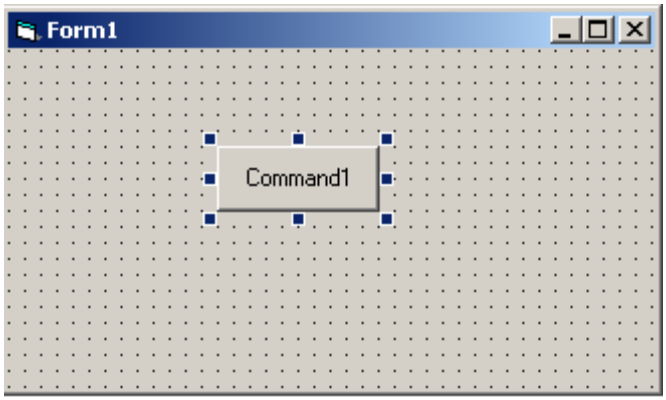
2. *InputBox* – диалоговое окно ввода для значения “**Y**”

3. *MsgBox* – диалоговое окно сообщений для вывода результата.

Во второй части примера приведен код программы на языке Visual Basic.

В третьей части примера приведен результат выполнения программы.

Часть I



Часть II

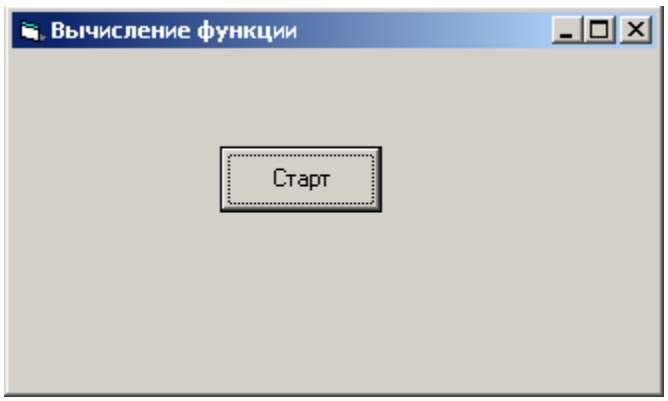
Код программы:

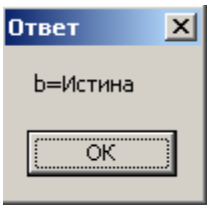
```
Option Explicit

Function IsLeapYear(y As Integer) As Boolean

If y Mod 4 = 0 Then

If y Mod 100 = 0 And y Mod 400 <> 0 Then

IsLeapYear = False

Else
```

```
IsLeapYear = True

End If

Else

IsLeapYear = False

End If

End Function

Private Sub Command1_Click()

Dim y As Integer, i As Integer

Dim b As Boolean

For i = 1 To 5

y = Val(InputBox("y", “Введите значения y”))

b = IsLeapYear(y)

MsgBox "b=" & b

Next i

End Sub
```

Часть III

Результат:





Примеры с использованием функции:

1. Описать функцию CircleS(r) вещественного типа находящую площадь круга радиуса R(r-вещественное). С помощью этой функции найти площади трех кругов с данными радиусами. Площадь круга радиуса R вычисляется по формуле $S=\pi \cdot R^2$. В качестве значения π использовать 3.14.

2. Описать функцию Fact(N) вещественного типа, вычисляющую значение факториала $N!=1 \cdot 2 \cdot \ldots \cdot N$ (N>0-параметр целого типа; вещественное возвращаемое значение используется для того, чтобы избежать целочисленного переполнения при больших значения N).

3. Описать функцию RingS(R_1, R_2) вещественного типа, находящую площадь кольца, заключенного между двумя окружностями с общим центром и радиусами R_1 и R_2 (R_1 и R_2 - вещественные, $R_1 > R_2$). С ее помощью найти площади трех колец,

для которых даны внешние и внутренние радиусы. Воспользоваться формулой площади круга радиуса R: $S=\pi \cdot R^2$. В качестве значения π использовать 3.14.

4. Описать функцию Power1(A,B) вещественного типа, находящую A^B по формуле $A^B=\exp(B \cdot \ln(A))$ (параметры A и B вещественные). В случае нулевого и отрицательного параметра A функция возвращающего 0. С помощью этой функции найти степени A^P, B^P, C^P если даны A, B, C, P.

5. Описать функцию Roots Count(a,b,c)целого типа, определяющую количество корней квадратного уравнения $ax^2+bx+c=0$ (a, b, c-вещественные параметры, $a \neq 0$). С ее помощью найти количество корней для каждого из трех квадратных уравнений с данными коэффициентами. Количество корней определять по значению дискриминанта: $d=b^2-4 \cdot a \cdot c$.

6. Описать функцию Sign(x) целого типа, возвращающую для вещественного числа X следующие значения: -1, если X<0; 0, если X=0; 1, если X>0. С помощью этой функции найти значение выражения Sign(A)+Sign(B) для данных вещественных чисел A и B.

7. Используя функцию IsLeapYear из решенного примера, описать функцию MonthDays(M,Y) целого типа, которая возвращает количество дней для M-го месяца года Y($1 \le M \le 12$, Y>0-целые числа). Вывести значение функции MonthDays для данного года Y и месяцев M_1, M_2, M_3.

8. Описать функцию Fact2(N) вещественного типа, вычисляющую двойной факториал:

N!!=1·3·5·…·N, если N-нечетное;

N!!=2·4·6·…·N, если N-четное;

(N>0- параметр целого типа; вещественное возвращаемое значение используется для того, чтобы избежать целочисленного переполнения при больших значениях N). С помощью этой функции найти двойные факториалы пяти данных целых чисел.

9. Описать функцию Fib(N) целого типа, вычисляющую N–й элемент последовательности чисел Фибоначчи F_k, которая описывается следующими формулами:

$F_1=1, F_2=1, \quad F_k=F_{k-2}+F_{k-1}, \quad K=3,4,\ldots .$

Используя функцию Fib, найти пять чисел Фибоначчи с данными номерами $N_1, N_2,\ldots,N_5$.

10. Описать функцию Event(K) логического типа, возвращающую True, если целый параметр К является четным, и False в противном случае. С ее помощью найти количество четных чисел в наборе из 10 целых чисел.

Программирование с помощью процедуры

Пример:

Описать процедуру ShiftLeft3(A,B,C), выполняющую левый циклический сдвиг: значение А переходит в С, значение С - в В, значение В - в А(А, В, С -вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой

процедуры выполнить левый циклический сдвиг для двух данных наборов из трех чисел: (A_1B_1 C_1) и (A_2 B_2 C_2).

Решение примера состоит из трех частей.

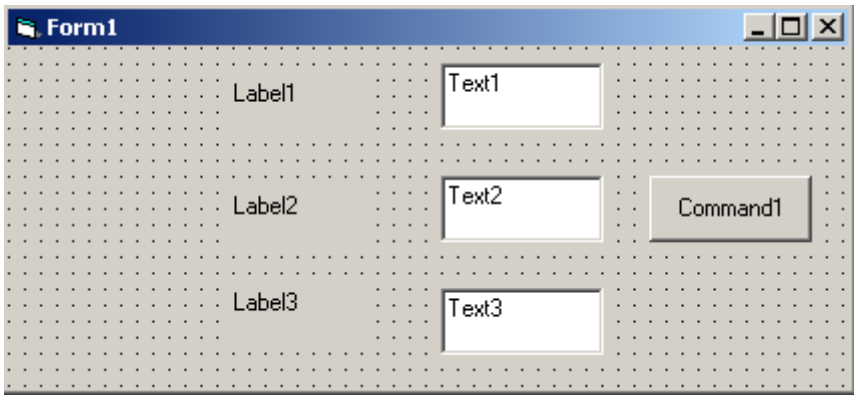
В первой части примера приведены количества объектов использованных в форме для вычисления процедуры. Форма состоит из следующих элементов (объектов):

1. *Label1* – указывает метку “**A**”
2. *Label2* – указывает метку “**B**”
3. *Label3* – указывает метку “**C**”
4. *Text1* – выводит значение “**A**”
5. *Text2* – выводит значение “**B**”
6. *Text3*– выводит значение “**C**”
7. *Command1* – для вызова основной процедуры

Во второй части примера приведен код программы на языке Visual Basic.

В третьей части примера приведен результат выполнения программы.

Часть I



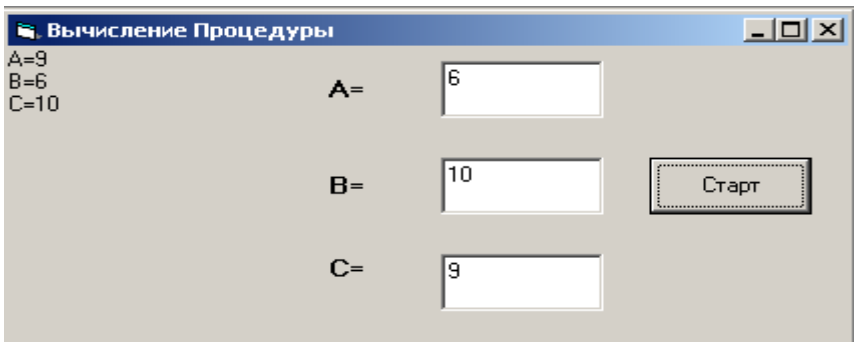
Часть II

Код программы:

```
Option Explicit
Private Sub ShiftLeft3(A, B, C)
Dim D As Single
D = A
A = C
C = B
B = D
End Sub
Private Sub Command1_Click()
Dim A As Single, B As Single, C As Single
A = Val(Text1.Text)
B = Val(Text2.Text)
C = Val(Text3.Text)
Call ShiftLeft3(A, B, C)
Print "A=" & A
Print "B=" & B
Print "C=" & C
End Sub
```

Часть III

Результат:



Примеры с использованием процедуры:

1. Описать процедуру TrianglePS(a, P, S), вычисляющую по стороне a равностороннего треугольника его периметр P=3·a и площадь $S=a^2\cdot(3)^{1/2}/4$ (a-входной, P и S-выходные параметры; все параметры являются вещественными). С помощью этой процедуры найти периметры и площади трех равносторонних треугольников с данными сторонами.

2. Описать процедуру Mean(X,Y,AMean,GMean), вычисляющую среднее арифметическое AMean=(x+y)/2 и среднее геометрическое $GMean=(x*y)^{1/2}$ двух положительных чисел X и Y (XиY-входные, AMean и GMean-выходные параметры вещественного типа). С помощью этой процедуры найти среднее арифметическое и среднее геометрическое для пар (A,B), (A,C), (A,D) если даны A,B,C,D

3. Описать процедуру DigitCountSum(*K*, *C*, *S*), находящую количество *C* цифр целого положительного числа *K*, а также их сумму *S* (*K* — входной, *C* и *S* — выходные параметры целого типа). С помощью этой процедуры найти количество и сумму цифр для каждого из пяти данных целых чисел.

4. Описать процедуру Swap(X, Y), меняющую содержимое переменных X и Y (X и Y-вещественные параметры, являющиеся одновременно входными и выходными). С ее помощью для данных

переменных A, B, C, D последовательно поменять содержимое следующих пар: A и B, C и D, B и C и вывести новые значения A, B, C, D.

5. Описать процедуру SortDec3(A,B,C), меняющую содержимое переменных A,B,C таким образом, чтобы их значения оказались упорядоченными по убыванию (A,B,C-вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры упорядочить по убыванию два данных набора из трех чисел: (A_1, B_1, C_1) и (A_2, B_2, C_2).

6. Описать процедуру ShiftRight3(A,B,C), выполняющую правый циклический сдвиг: значение A переходит в B, значение B- в C, значение C- в A (A,B,C-вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры выполнить правый циклический сдвиг для двух данных наборов из трех чисел: (A_1, B_1, C_1) и (A_2, B_2, C_2).

7. Описать процедуру PowerA3(A, B), вычисляющую третью степень числа A и возвращающую ее в переменной B(A-входной, B-выходной параметр; оба параметра являются вещественными). С помощью этой процедуры найти третьи степени пяти данных чисел.

8. Описать процедуру PowerA234(A, B, C, D), вычисляющую вторую, третью и четвертую степень числа A и возвращающую эти степени соответственно в переменных B, C и D (A-входной, B, C,

D-выходные параметры; все параметры являются вещественными). С помощью этой процедуры найти вторую, третью и четвертую степень пяти данных чисел.

9. Описать процедуру DigitCountSum(K, C, S), находящую количество С цифр целого положительного числа К, а также их сумму S(К-входной, С и S–выходные параметры целого типа). С помощью этой процедуры найти количество и сумму цифр для каждого из пяти данных целых чисел.

10. Описать процедуру Minmax(X, Y), записывающую в переменную X минимальное из значений X и Y, а в переменную Y-максимальное из этих значений (X и Y- вещественные параметры, являющиеся одновременно входными и выходными). Используя четыре вызова этой процедуры, найти минимальное и максимальное из данных чисел A, B, C, D.

Программирование с основными функциями строк

Пример.

Дана строка. Подсчитать общее количество содержащихся в ней строчных латинских и русских букв.

Решение примера состоит из трех частей.

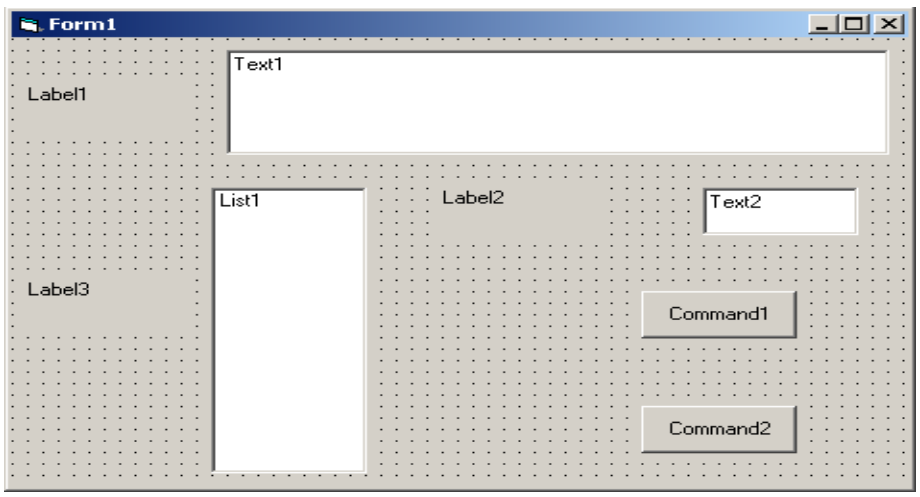
В первой части примера приведено количества объектов, использованных в форме для вычисления общего количество строчных латинских и русских букв. Форма состоит из следующих элементов (объектов):

1. *Label1* – указывает метку “**S**”
2. *Label2* – указывает метку “**Количество букв**”
3. *Label3* – указывает метку “**Количество букв и символов**”
4. *Text1* – для вывода “**Текста**”
5. *Text2* – для вывода “**Количество строчных латинских и русских букв**”
6. *ListBox1*– для вывода “**Количество букв и символов**”
7. *Command1* – для вычисления “**Количество букв и символов**”
8. *Command2* – для вычисления “**Результата**”

Во второй части примера приведен код программы на языке Visual Basic.

В третьей части примера приведен результат выполнения программы.

Часть I



Часть II

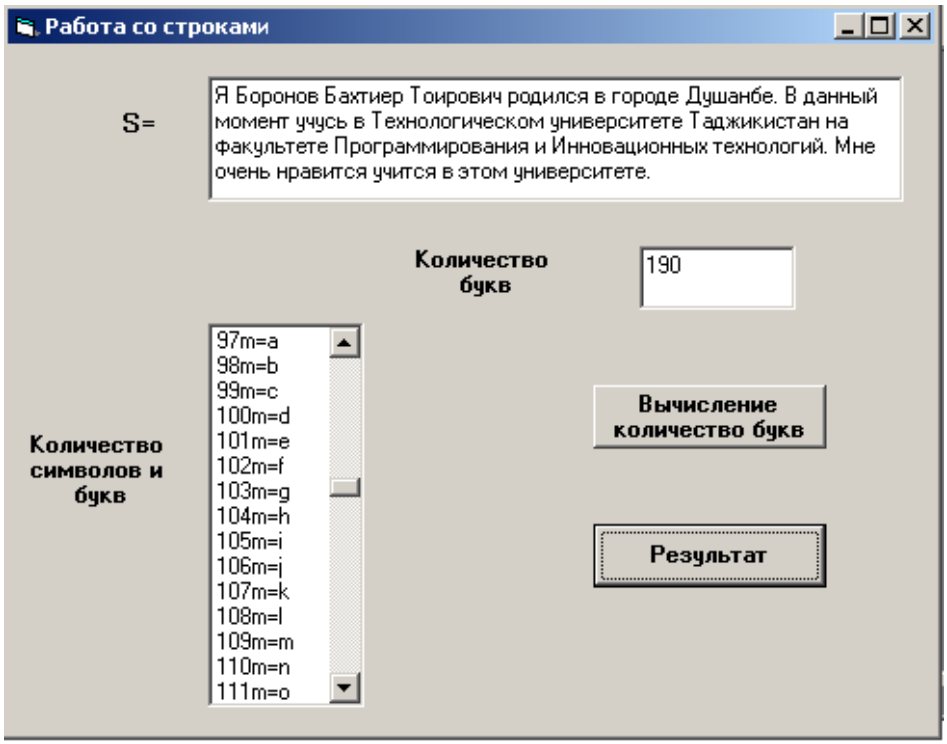
Код программы:

```
Option Explicit

Private Sub Command1_Click()

Dim m As String

Dim i As Integer

For i = 0 To 255

m = Chr(i)

List1.AddItem i & "m=" & m

Next i

End Sub

Private Sub Command2_Click()

Dim S As String

Dim i As Integer

Dim n As Integer, n1 As Integer, n2 As Integer

S = Text1.Text

n = 0

n1 = Len(S)

For i = 1 To n1

n2 = Asc(Mid(S, i, 1))

If (n2 >= 97 And n <= 122) Or (n2 >= 224 And n2 <= 255) Then
```

```
n = n + 1

End If

Next i

Text2.Text = n

End Sub
```

Часть III

Результат:



Примеры с использованием функции строк:

1. Дана строка. Подсчитать количество содержащихся в ней цифр.

2. Дана строка. Подсчитать количество содержащихся в ней прописных латинских букв.

3. Дана строка. Преобразовать в ней все прописные латинские буквы в строчные.

4. Дана строка. Преобразовать в ней все строчные буквы (как латинские, так и русские) в прописные.

5. Дана строка. Преобразовать в ней все строчные буквы (как латинские, так и русские) в прописные, а прописные - в строчные.

6. Дано целое число N(>0)и символ C. Вывести строку длины N, которая состоит из символов C.

7. Дана строка, состоящая из русских слов, разделенных пробелами(одним или несколькими). Найти количество слов в строке.

8. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые начинаются и заканчиваются одной и той же буквой.

9. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти длину самого короткого слова.

10. Дана строка-предложение на русском языке. Подсчитать количество содержащихся в строке гласных букв.

Программирование с файлами

Пример:

Дана строка S и текстовый файл. Добавить строку S в начало файла.

Решение примера состоит из трех частей.

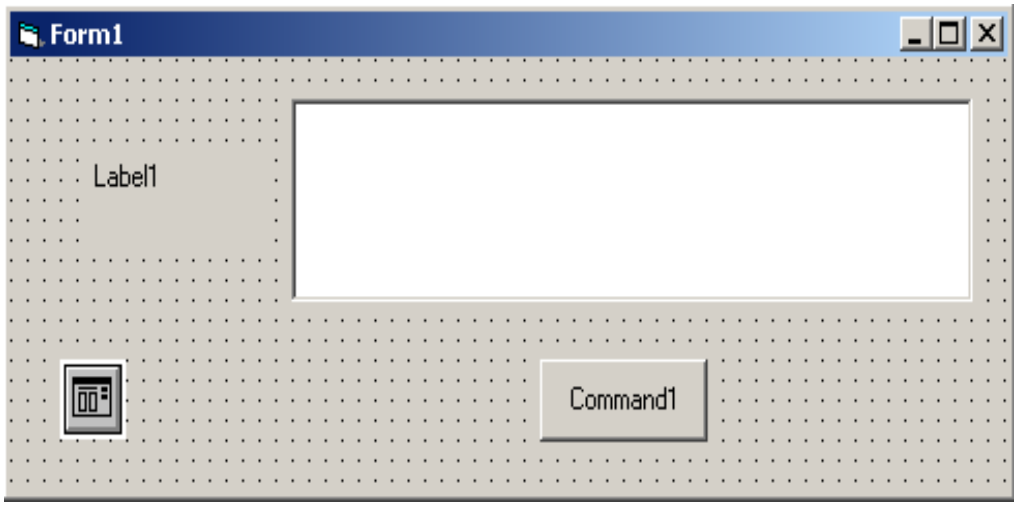
В первой части примера приведено количества объектов, использованных в форме для добавления строки в начало файла. Форма состоит из следующих элементов (объектов):

1. *Label1* – указывает метку “**S**”
2. *Text1* – для ввода “**Текста**”
3. *CommonDialog1* – для “**Сохранения файла**”
4. *Command2* – для открытия “**Файла**”

Во второй части примера приведен код программы на языке Visual Basic.

В третьей части примера приведен результат выполнения программы.

Часть I



Часть II

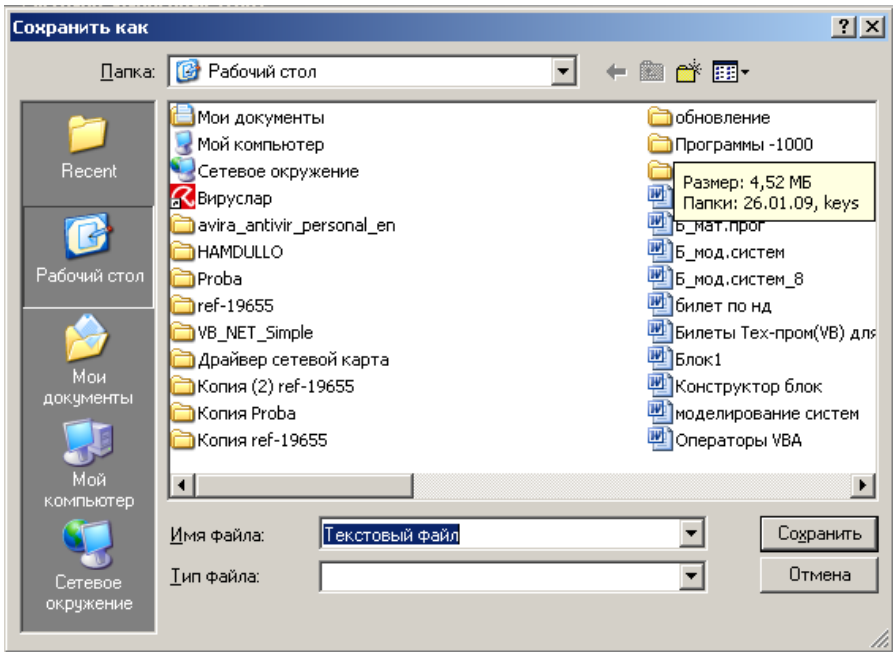
Код программы:

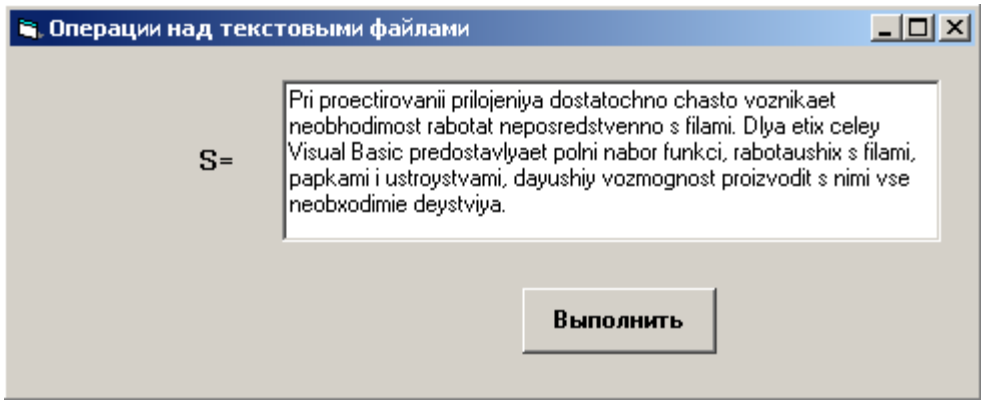
```
Option Explicit

Dim S As String

Dim strFileName As String

Dim strFileContent As String

Dim nFreeFile As Integer

Private Sub Command1_Click()

S = Text1.Text

nFreeFile = FreeFile

CommonDialog1.ShowSave

strFileName = CommonDialog1.FileName

If strFileName <> "" Then

Open strFileName For Output As nFreeFile
```

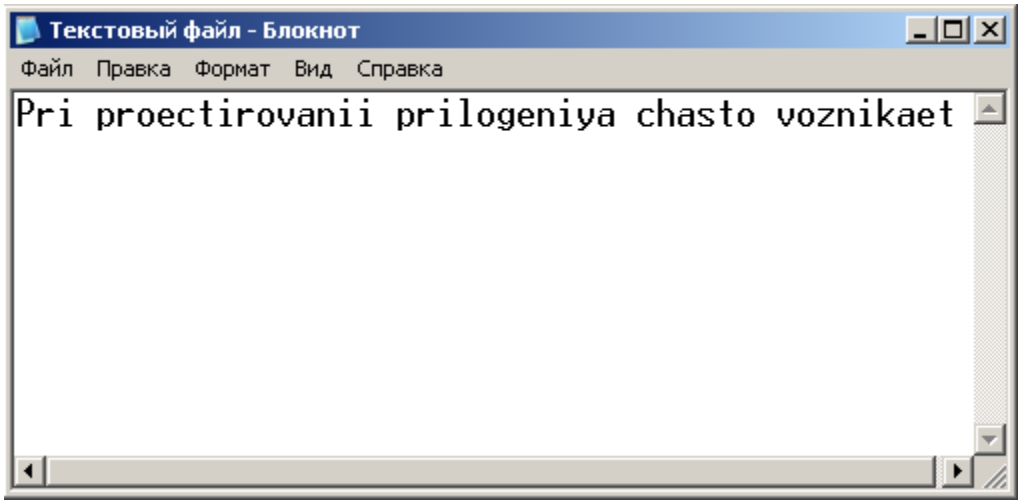
```
strFileContent = Text1.Text

Print #nFreeFile, strFileContent

Close

End If

End Sub
```

Часть III

Результат:







Примеры с использованием текстовых файлов:

1. Дана строка S и текстовый файл. Добавить строку в конец файла.

2. Даны два текстовых файла. Добавить в начало первого файла содержимое второго файла.

3. Даны два текстовых файла. Добавить в конец первого файла содержимое второго файла.

4. Дан текстовый файл. Заменить в нем все прописные русские буквы на строчные, а все строчные – на прописные.

5. Дан текстовый файл, содержащий более трех строк. Удалить из него последние три строки.

6. Дан текстовый файл. Заменить в нем все подряд идущие пробелы на один пробел.

7. Дан текстовый файл. Продублировать в нем все пустые строки.

8. Дан непустой текстовый файл. Удалить из него первую строку.

9. Дано целое число К, и текстовый файл. Удалить из файла строку с номером К. если строки с таким номером нет, то оставить файл без изменения.

10. Дано имя файла и целые положительные числа N и K. Создать текстовый файл с указанным именем и записать в него N строк, каждая из которых состоит из К символов «*»(звездочка).