МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кузбасский государственный технический университет имени Т. Ф. Горбачева»
Институт профессионального образования
Кафедра информационных и автоматизированных производственных систем

Составитель О.С. Семенова

ОП.04 Основы алгоритмизации и программирования

Методические указания к практическим работам

Рекомендовано цикловой методической комиссией
информационных систем и программирования
в качестве электронного издания для использования
в образовательном процессе

Кемерово
2018

Рецензенты:

И.В. Чичерин, к.т.н., доцент кафедры информационных и автоматизированных производственных систем

Семенова Ольга Сергеевна
Основы алгоритмизации и программирования: методические указания к практическим работам [Электронный ресурс]: для студентов специальности СПО 09.02.07 «Информационные системы и программирование» / сост. О.С. Семенова, КузГТУ. – Электрон. издан. – Кемерово, 2018.

Методические указания к практическим работам по курсу «Основы алгоритмизации и программирования» позволяют углубить знания, полученные в ходе аудиторных занятий; способствуют закреплению теоретических положений; развивают навыки по их практическому применению.

Практическая работа №1
Алгоритмизация вычислительного процесса

Цель работы: Усвоить понятия: алгоритм как фундаментальное понятие информатики, способы описания, основные типы алгоритмов. Освоить принципы решения задач с использованием основных алгоритмических конструкций.

Теоретические положения

Решение любой задачи на ЭВМ можно разбить на следующие этапы: разработка алгоритма решения задачи, составление программы решения задачи на алгоритмическом языке, ввод программы в ЭВМ, отладка программы (исправление ошибок), выполнение программы на ПК, анализ полученных результатов.

Первый этап решения задачи состоит в разработке алгоритма. ***Алгоритм*** – это точная конечная система правил, определяющая содержание и порядок действий исполнителя над некоторыми объектами (исходными и промежуточными данными) для получения после конечного числа шагов искомого результата. Алгоритм состоит из шагов. ***Шаг*** – отдельное законченное действие.

Виды алгоритмов:

- Линейный алгоритм – описание последовательности действий, которые выполняются однократно в заданном порядке.
- Циклический алгоритм – описание действий, которые повторяются заданное число раз или пока не выполнится некоторое условие. Тело цикла – перечень повторяющихся действий. Циклические алгоритмы подразделяют на алгоритмы с предусловием, постусловием и алгоритмы с конечным числом повторов. В алгоритмах с предусловием сначала выполняется проверка условия окончания цикла и затем, в зависимости от результата проверки, выполняется (или не выполняется) так называемое тело цикла.
- Разветвляющийся алгоритм – алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий.

Основные свойства алгоритмов:

- Дискретность (прерывность) – разбиение алгоритма на шаги;
- Результативность – получение из исходных данных результата за конечное число шагов;
- Массовость – пригодность для решения не какой-либо одной, а целого

класса задач;

• Детерминированность (определенность) – совпадение получаемых результатов независимо от пользователя и применяемых технических средств;

• Выполнимость и понятность – каждый шаг алгоритма должен быть понятен исполнителю.

Способы описания алгоритма:

1. словесно-формульный
2. структурный или блок-схемный
3. с использованием специальных алгоритмических языков
4. с помощью сетей Петри (рис. 1)
5. с помощью граф-схем

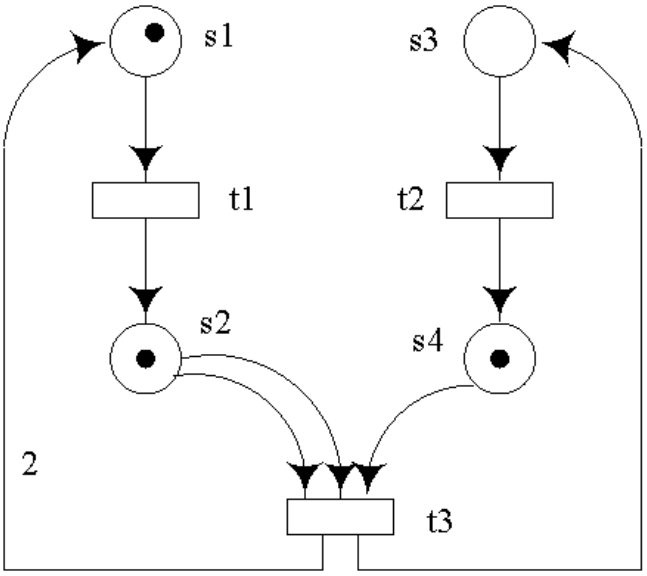


Рис. 1 – Описание алгоритма с помощью сетей Петри

Следует заметить, что блок-схема – наиболее распространенный тип схем, описывающий алгоритмы или процессы, изображая шаги в виде блоков различной формы, соединенных между собой стрелками. Основные элементы блок-схем регламентируются соответствующими ГОСТами: ГОСТ 10.002-80 ЕСПД, ГОСТ 10.003-80 ЕСПД (рис. 2).

Пример: Найти наибольшее из 3-х чисел. Вывести ответ на экран.

Данный алгоритм словесно можно описать следующим образом: Ввести 3 числа: A, B, C. Проверить, какое число больше: A или B? Если A больше B, то проверить, какое число больше: A или C? В случае, если A, то оно и будет максимальным из 3-чисел, иначе максимальным будет число C. Если же самое первое условие ложно (т.е A<B), тогда аналогичные

операции проводим с числами B и C. Блок-схема алгоритма - на рис. 3.

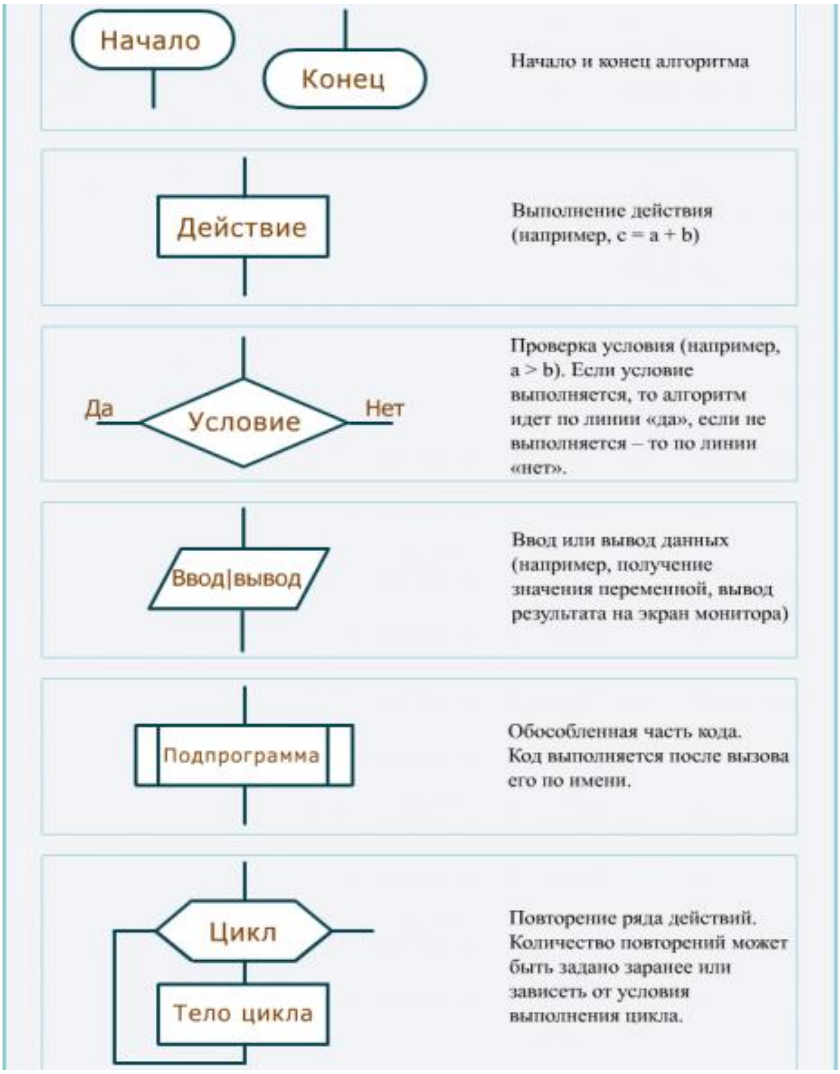


Рис. 2 – Блоки структурного алгоритма

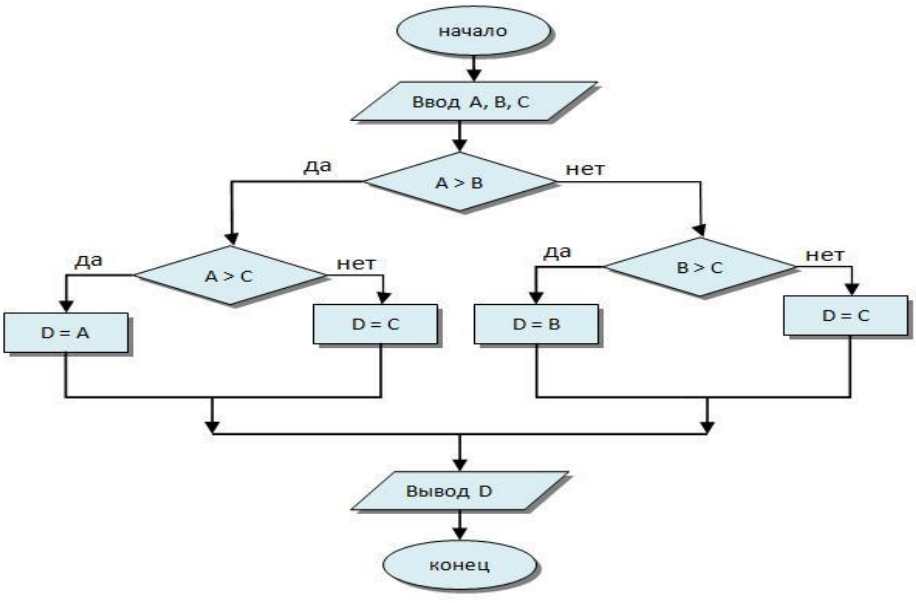


Рис. 3 – Решение примера

Задания к практической работе №1

1. Даны натуральные числа a, b. Вычислить произведение a*b, используя в программе лишь операции +, -, =
2. Дано натуральное n, вычислить n!
3. Составить алгоритм программы, выводящий на экран квадраты всех натуральных чисел от 0 до заданного натурального n.
4. Составить блок-схему алгоритма для вычисления выражения:

$$y = \begin{cases} -40x^2 + 16, & x < 4 \\ 14(x^2 - 6), & x \geq 4 \end{cases}$$

Проверить при x=1; x=12

5. Составить блок-схему алгоритма для решения задачи: вывести только те из 130-и введённых чисел, целая часть которых равна 50.
6. Ввести с клавиатуры 140 чисел. Найти сумму тех из них, которые являются целыми.
7. Найти сумму только тех из 150 введенных с клавиатуры чисел, которые меньше 23.
8. Введенное с клавиатуры предложение вывести на экран 100 раз в случайным образом определённые позиции.
9. Введенное с клавиатуры число напечатать на экране 15 раз по диагонали.
10. Дан массив A(5,7). Найти сумму всех элементов, находящихся на нечетных строках массива.
11. Вывести на экран первых 4 положительных элемента массива B(21,6).
12. Найти сумму элементов массива G(3,8), находящихся на двух первых строках массива.
13. Проверить, является ли заданное натуральное число n простым.
14. Имеется квадратная таблица a[1..n, 1..n]. Известно, что для некоторого i строка с номером i заполнена одними нулями, а столбец с номером i - одними единицами (за исключением их пересечения на диагонали, где стоит неизвестно что). Найти такое i (оно, очевидно, единственно).
15. Даны два возрастающих массива x: array1 [1..k] и y: array2 [1..l]. Найти количество общих элементов в этих массивах (т. е. количество тех целых t, для которых t = x[i] = y[j] для некоторых i и j).

Практическая работа №2
Изучение интегрированной среды разработчика

Цель работы:

В данной практической работе студент должен ознакомиться со средой разработчика программ Visual Basic 6.0 (VB) или VBA, изучить основные окна.

Теоретические положения

После запуска оболочки Visual Basic 6.0 на экран выводится окно New Project, в котором пользователю предоставляется возможность создать новый проект (т.е. программу на языке VB), загрузить ранее созданный проект или осуществить выбор проекта из созданных за последнее время.

К основным окнам VB относятся: главное окно (содержит строку заголовка, строку меню; строку инструментов); панель инструментов (содержит объекты, которые могут быть помещены в созданное окно путем перетаскивания); окно Project Explorer (содержит список файлов проекта); окно Properties (содержит набор свойств форм и элементов управления); окно Form Designer (содержит все формы проекта); окно формы, окно Form Layout (определяет положение форм на экране при запуске программы).

Программа VB выгружается из памяти путем нажатия на кнопку “Закрыть”, находящейся в строке заголовка, либо путем выбора опции File/Exit (выход). Перед выходом из VB необходимо сохранить созданный проект с помощью опции File/Save Project as.

VBA – это подмножество визуального языка программирования Visual Basic (VB), которое включает почти все средства создания приложений VB. VBA отличается от языка программирования VB тем, что система VBA предназначена для непосредственной работы с объектами Office, в ней нельзя создавать проект независимо от приложений Office. Таким образом, в VBA языком программирования является VB, а инструментальная среда программирования реализована в виде редактора VB, который может активизироваться из любого приложения MS Office.

Например, для того, чтобы открыть редактор VBA из приложения PowerPoint необходимо нажать на вкладке «Разработчик» кнопку «Visual Basic». Вернуться из редактора в приложение можно, выбрав команду Microsoft PowerPoint в меню Вид или комбинацией клавиш Alt + F11.

Задание к практической работе №2

1. Для интегрированной среды разработчика Visual Basic 6.0 (VB)

- Загрузить VB. Назвать все окна, выведенные на экран;
- Убрать с экрана окна Properties и Project Explorer. Переместить окна в более удобные положения, изменить размер окон, где это нужно. Развернуть основное окно и окно Form Designer. Вернуть изменения;
- Найти справку по окну Project Explorer;
- Сохранить проект под именем “Практическая работа”;
- Выйти из VB.

2. Для интегрированной среды разработчика Visual Basic for Aplication (VBA)

- Загрузить любое офисное приложение. Открыть интегрированную среду разработчика VBA.
- Убрать с экрана окна Properties и Project Explorer. Переместить окна в более удобные положения, изменить размер окон, где это нужно. Развернуть основное окно и окно Form Designer. Вернуть изменения;
- Найти справку по окну Project Explorer;
- Сохранить файл с поддержкой макросов «Практическая работа.xlsm»;
- Выйти из VB.

Практическая работа №3
Изучение основных объектов и их свойств

Цель работы: Научиться создавать новые объекты на форме с помощью элементов управления, устанавливать свойства объектов.

Теоретические положения

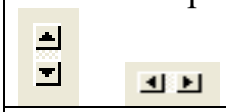
Окна (на этапе разработки называемые формами) и элементы управления являются объектами. У всех объектов есть свойства – элементы информации, описывающие определенный объект. От установки свойств зависят шрифт и цвет текстового окна, заголовок формы, положение кнопки на экране и т. д.

После загрузки VB на экране появляется либо новая форма, либо уже созданная ранее. Для того чтобы самостоятельно добавить к существующему проекту новую форму необходимо выбрать пункт меню File/Add Form.

Любое окно проекта содержит элементы управления, основная задача которых – выводить на экран информацию либо предоставлять возможность пользователю выполнять определенные действия. Элементы управления могут быть добавлены на форму с помощью двойного щелчка по соответствующему элементу на панели инструментов. Краткое описание элементов управления содержится в таблице 1.

Таблица 1 – Элементы управления

Название элемента управления	Описание элемента управления
Командная кнопка 	Используется для выполнения действий. К каждой кнопке присоединяется процедура, которая выполняется тогда, когда пользователь щелкает по кнопке
Ярлык (надпись) 	Ярлык добавляет к форме любой текст
Текстовое окно 	Служит для ввода пользователем любых данных
Кнопки–переключатели 	Обеспечивают пользователю возможность выбрать одну опцию из нескольких
Контрольные индикаторы 	Обеспечивают пользователю возможность выбрать любое количество опций
Рамка 	Позволяет объединять элементы управления в

	группы, которые будут работать независимо друг от друга
Линейки прокрутки 	Используются для вертикальной или горизонтальной прокрутки данных в текстовых окнах
Окна списков 	Позволяет пользователю выбирать из списка возможностей
Комбинированные списки 	Комбинация текстового окна и окна списка. Элемент может быть выбран из списка либо щелчком по нему, либо впечатывание имени в текстовое поле
Изображение 	Выводит на экран содержимое графического файла
Данные 	Осуществляет доступ к данным, размещённых во внешних структурированных файлах

Наиболее часто используемые свойства объектов приведены в таблице 2.

Таблица 2 – Объекты и их свойства

Наименование свойства объекта	**Описание свойства**	**Опции**
Общие свойства		
Name (имя)	Название объекта	-
Caption (заголовок)	Текст, появляющийся на объекте при отображении его на экране	-
Border Style (стиль границы)	Определяющий тип границы и элементов, находящихся на площади заголовка	0– None 1–Fixed Single 2–Sizable 3–Fixed Dialog
Locked	Определяет, будет ли объект реагировать на действия пользователя. Внешний вид объекта не изменяется.	True False
Left (левая граница)	Расстояние от левой границы объекта до левого края экрана	
Top (верхняя граница)	Расстояние от верхней границы объекта до верхнего края экрана	
Width (ширина)	Ширина объекта	
Height (высота)	Высота объекта	
Visible (видимый)	Определяет видимость объекта на	True

	экране	False
Enabled (доступный)	Определяет доступность объекта	True False
Font (шрифт)	Определяет внешний вид текста	
BackColor (цвет)	Устанавливает цвет фона	
TabStop	Определяет, может ли объект быть активным	True False
TabIndex	Определяет последовательность, в которой объекты активизируются при нажатии клавиши <Tab>	
Свойства формы		
Control Box (Системное меню)	Присоединение к форме системного меню	True False
Max Button (Кнопка развертывания)	Присоединение к форме кнопки развертывания	
Min Button (кнопка свертывания)	Присоединение к форме кнопки свертывания	
StartUpPosition	Определяет, в каком месте экрана появляется форма при загрузке	0-Manual 1-в центре окна 2- в центре экрана
SpecialEffect	Придание форме трехмерного вида	
MDIChild	Определяет, является ли форма дочерней	True False
WindowList	Возвращает список дочерних форм (только для MDI)	
Свойства командной кнопки		
Default	Идентифицирует кнопку, которая будет задействована, когда пользователь нажмет <Enter>	True False
Cancel	Идентифицирует кнопку, которая будет задействована, когда пользователь нажмет <Esc>	True False
Value	Показывает, что кнопка была нажата. Установка значения .T. инициализирует щелчок по кнопке.	True False
ToolTipText	Всплывающая подсказка для кнопок	
Свойства ярлыка		
Alignment	Определяет положение текста внутри площади, выделенной для ярлыка	
Auto Size	Автоматическая установка размера	True

	объекта	False
WordWrap	Автоматическое расширение объекта по мере увеличения длины текста	True False
Свойства текстового окна		
Text	Содержит введенный пользователем текст	
MultiLine	Создание текстового окна с несколькими строками текста	True False
MaxLength	Устанавливает предельное число символов, которые могут быть введены в окно	
ScrollBar	Устанавливает наличие линеек прокрутки	0/1/2/4
Password Char	Задание символа для отображения его на экране	
SelText	Строка подсвеченных символов	
SelLength	Длина подсвеченной строки	
SelStart	Текущее положение курсора	
Свойства кнопок переключателей		
Value	Показывает, может ли кнопка быть выбрана	True False
Alignment	Определяет положение текста относительно кнопки	0/1
Свойства контрольных индикаторов(флажков)		
Value	Показывает, может ли кнопка быть выбрана	0/1/2
Свойства линеек прокрутки		
Min и Max	Определяют предельные значения, которые могут быть использованы линейкой прокрутки	
Value	Определяет текущее положение маркера на линейке прокрутки	
Small Change	Определяет величину, на которую изменится Value при щелчке по стрелке на конце линейки прокрутки	
Large Change	Определяет величину изменения Value при щелчке по самой линейке прокрутки	
Свойства окон списков		
ListCount	Определяет количество элементов в списке. Доступно во время работы	

	программ	
ListIndex	Возвращает индекс количества элементов, которые в данный момент подсвечены. Доступно во время работы программ	
Text	Содержит текст выбранного элемента. Доступно во время работы программ	
Multiselect	Определяет количество элементов списка, которое может быть выбрано	0/1/2
Columns	Определяет количество колонок в списке	
IntegralHeight	Отвечает за вывод списка на экран	True False
Sorted	Расположение элементов в алфавитном порядке	True False
Свойства комбинированных списков		
Style	Определяет внешний вид комбинированного списка	0/1/2
List	Перечень элементов, входящих в список	
Sorted	Сортировка элементов списка в алфавитном порядке	
Text	Значение (текст) выбранного элемента	
Listindex	Позиция выбранного элемента	
ListCount	Количество элементов в списке	
Свойства окон с рисунками		
Picture	Имя файла с рисунком	
Stretch	Рисунок подстраивает свой размер под ширину объекта	True False
Redraw	Перерисовка изображения при закрытии окна, закрывающего текущее изображение	True False

Задание к практической работе №3

1. Откройте проект. Создайте форму следующего вида (рис.4):

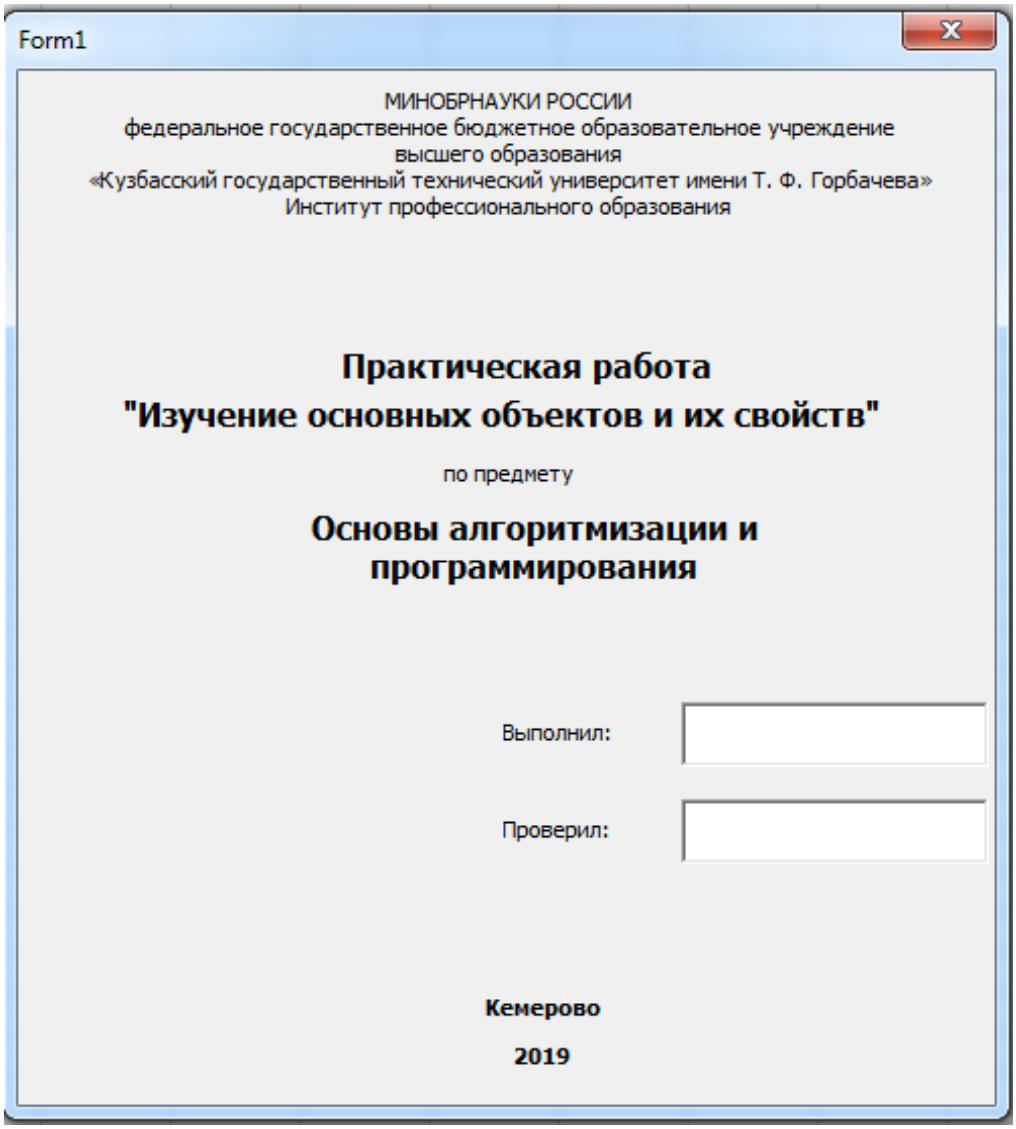


Рис. 4 – Форма

2. Добавьте к проекту новую форму для ввода пароля (рис.5).

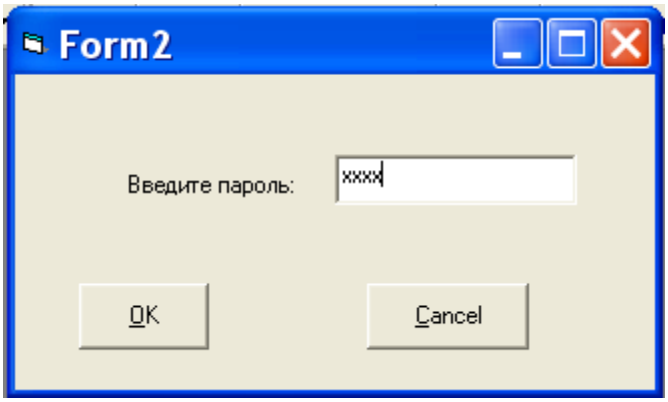


Рис. 5 – Форма

3. Добавьте к проекту новую форму (рис.6).

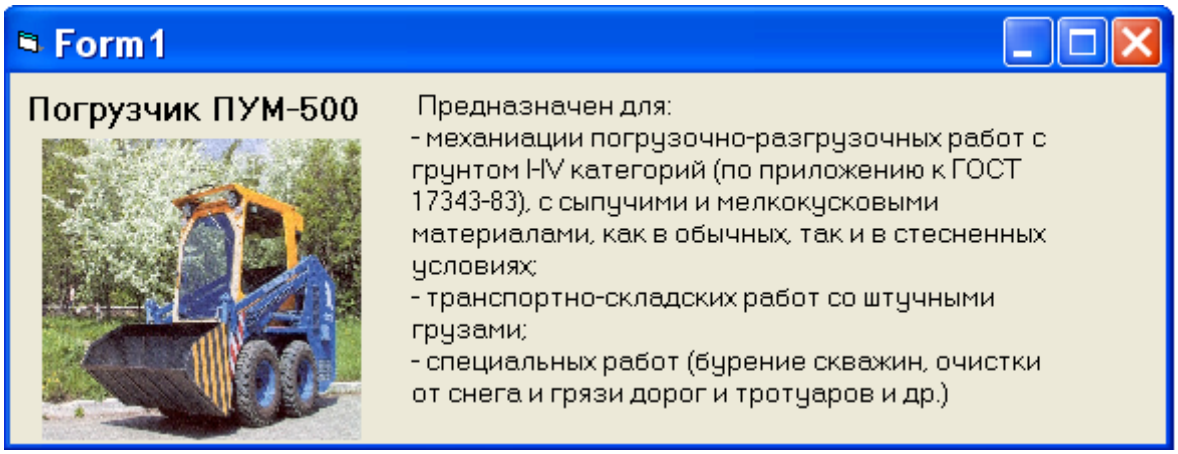

Form1
Погрузчик ПУМ-500
Предназначен для:
- механиации погрузочно-разгрузочных работ с грунтом I-IV категорий (по приложению к ГОСТ 17343-83), с сыпучими и мелкокусковыми материалами, как в обычных, так и в стесненных условиях;
- транспортно-складских работ со штучными грузами;
- специальных работ (бурение скважин, очистки от снега и грязи дорог и тротуаров и др.)

Рис. 6 – Форма

Практическая работа №4
Свойства и методы объектов. Реакция объектов на события

Цель работы: Изучение основных событий VB. Знакомство с написанием процедур отклика на события. Ссылки на свойства. Методы.

Теоретические положения

Для каждого объекта существует ряд возможных событий. Программный код отклика на эти события содержится в процедурах. Основные события приведены в таблице 3.

Таблица 3 – События

Событие	Описание
Click (щелчок)	Происходит при щелчке пользователя кнопкой мыши по объекту
Double Click (двойной щелчок)	Происходит при двойном щелчке пользователя кнопкой мыши по объекту
Mouse Down	Происходит при нажатии кнопки мыши
Mouse Up	Происходит при отпускании кнопки мыши
Mouse Move	Происходит при прохождении указателя мыши по объекту
Drag Over	Происходит при перетаскивании элемента формы через объект
Drag Drop	Происходит в конце операции перетаскивания, когда кнопка мыши отпускается
Got Focus	Происходит, когда элемент управления получает активность
Lost Focus	Происходит, когда элемент управления теряет активность
Load	Происходит при каждой загрузке формы в память
Initialize	Возникает в момент создания экземпляра формы (до ее загрузки в память компьютера и отображения на экране)

Процедуры создаются в *кодовом окне*. Каждая процедура должна иметь имя, которое должно состоять из имени объекта, символа подчеркивания и имени события. Процедура записывается в следующем виде:

Private Sub объект_событие()

операторы

……

End Sub

В процедуре можно использовать любые свойства элементов управления. Для того чтобы в процессе работы программы изменить свойство объекта, необходимо записать выражение следующего вида:

объект . свойство = значение

Пример: При щелчке мыши по командной кнопке «ОК» (*cmdOK*) ярлык (*lblText*) изменяет свой текст на «Добро пожаловать в VB».

Private Sub cmdOK_Click()

lblText.Caption=«Добро пожаловать в VB»

End Sub

Каждый объект имеет некоторое количество доступных ему методов. Методы – это встроенные процедуры, оказывающие некоторое действие на объект. Некоторые методы приведены в таблице 4.

Таблица 4 – Методы

Метод	**Описание**
Set Focus	Активизация объекта
Show	Загрузка в память формы и вывод ее на экран
Hide	Сокрытие формы с экрана
AddItem элемент	Применяется для окон списков. Добавление элемента в список во время работы программы
RemoveItem элемент	Применяется для окон списков. Удаляет элемент из списка

Для того чтобы выполнить один из методов необходимо записать выражение следующего вида: *объект.метод*

Пример: Добавление фамилий студентов в комбинированный список. Выбор фамилии из списка (рис. 7).

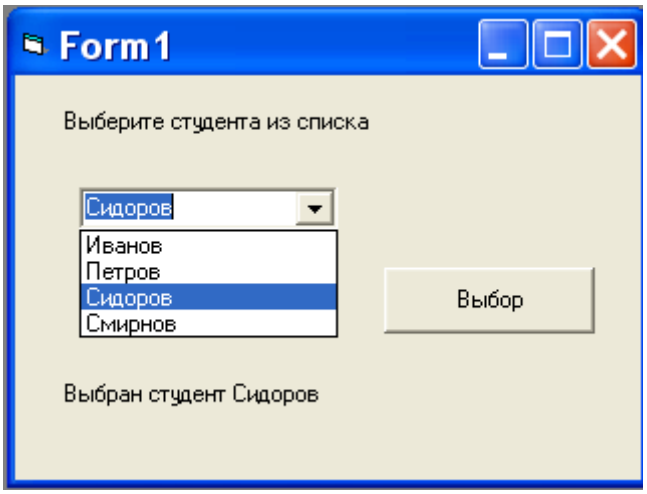


Рис. 7 – Форма

```
Private Sub form_initialize ()
cmbStudent.AddItem «Иванов»
cmbStudent.AddItem «Петров»
cmbStudent.AddItem «Сидоров»
cmbStudent.AddItem «Смирнов»
End Sub

Private Sub vibor_Click()
Label1.Caption=”Выбран студент” & cmbStudent.Text
End Sub
```

Задание к практической работе №4

1. К формам, созданным в предыдущей практической работе, добавьте кнопки перехода от одной формы к другой. Напишите программный код, позволяющий проверить правильность ввода пароля. Добавьте возможность выхода из приложения при нажатии на кнопку «Выход», размещенную на последней форме.
2. Разработайте интерфейс формы, напишите программный код, позволяющий все данные, вводимые пользователем в текстовое окно, автоматически при нажатии на кнопку «ОК» отображать на ярлыке.
3. Создайте блок кнопок - «Красный», «Желтый», «Зеленый», «Синий», «Белый», «Черный». При нажатии на кнопку форма окрашивается соответствующим цветом.
4. При нажатии на кнопку «Min» размер формы уменьшается на 10 твипов, «Max» – увеличивается на 10 твипов (1 пункт = 20 твипов. Пункт – единица, используемая для измерения размера шрифта).

5. Создать следующую форму (рис. 8) для ввода текста:

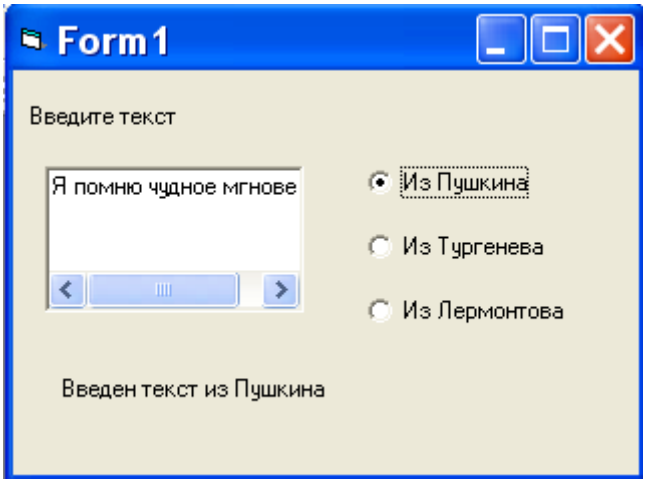


Рис. 8 – Форма

6. Создать форму для сбора данных (рис. 9). После нажатия на кнопку “*Далее*”, введенные данные выводятся на экран с помощью функции MsgBox().

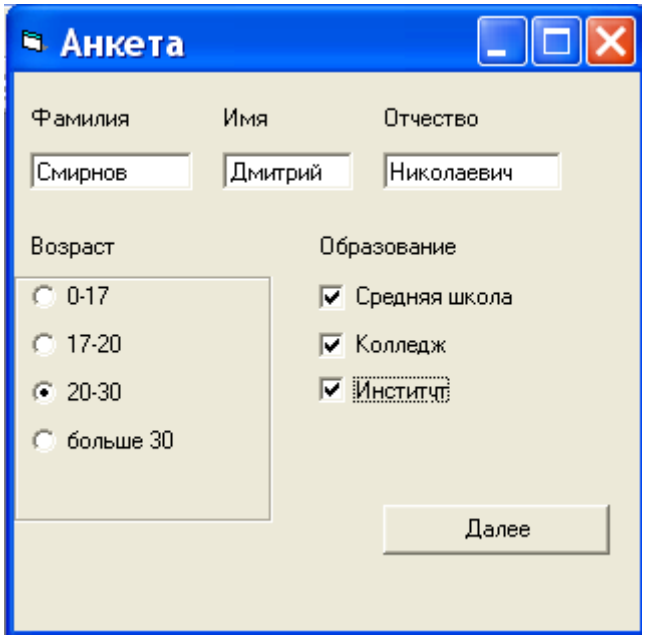


Рис. 9 – Форма

7. Создайте форму для добавления/удаления данных о студентах в комбинированный список, состоящий из 3 колонок (рис. 10).

Примечание: Для добавления нового элемента используется метод .AddItem, для записи данных в колонки необходимо использовать свойство List. Например, для помещения числа «17» в 0-й строку и 1-ю колонку комбинированного списка cmbStudent необходимо записать: cmbStudent.List(0, 1) = "17".

8. Создать проект “Выбор шрифта”, позволяющий выбрать с помощью списков тип шрифта, размер, начертание (свойства): жирный, подчёркнутый, курсив. Выбранные параметры должны применяться к метке Образец.

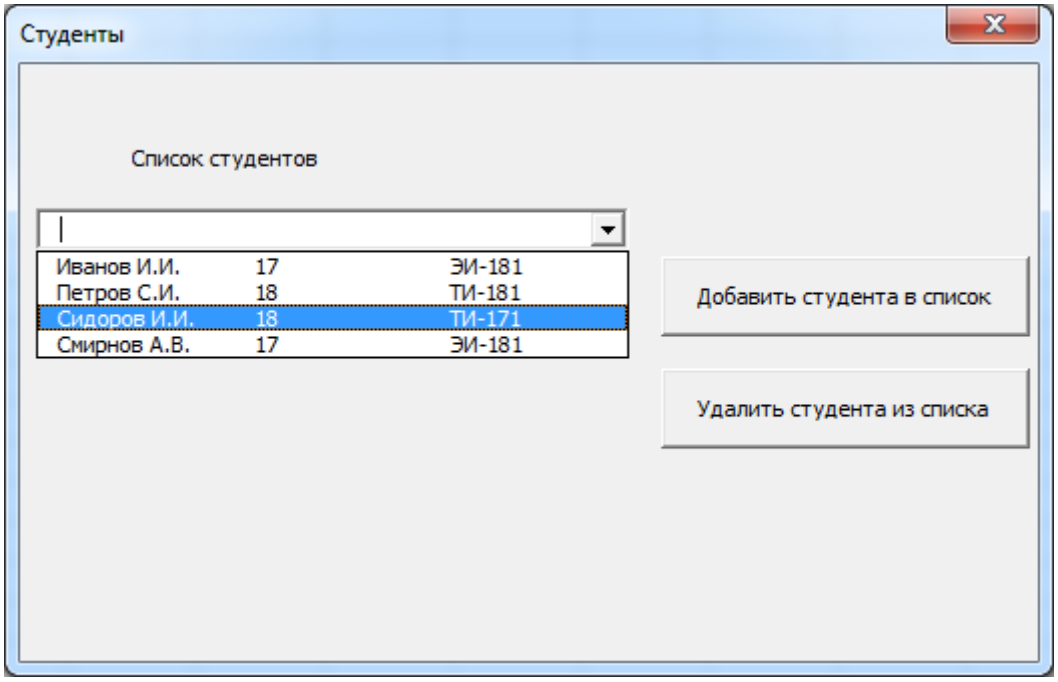


Рис. 10 – Форма

Практическая работа №5
Построение выражений

Цель работы: Знакомство с написанием выражений. Освоение навыков использования арифметических и логических операторов. Изучение основных встроенных функций языка программирования Visual Basic 6.0.

Теоретические положения

Выражение – это формальное правило для вычисления некоторого значения. Выражение строится как совокупность операндов (констант, переменных, функций), объединенных знаками операций, выполнение которых приводит к вычислению значения выражения. Главное свойство выражений – возможность иметь значение. В зависимости от типа операндов и используемых операций выражения делятся на *арифметические*, *логические* и *строковые*.

Арифметическими называются выражения, содержащие операнды только арифметического типа и знаки математических операций. В таблице 5 приведены арифметические операторы и выполняемые ими функции.

Таблица 5 – Арифметические операторы

Оператор	Выполняемая операция
+	сложение
-	вычитание
*	умножение
/	деление
\	целочисленное деление
mod	остаток от деления нацело
^	возведение в степень

Если в выражении отсутствуют скобки, то операторы выполняются в следующем порядке: 1) возведение в степень, 2) умножение и деление, 3) деление нацело, 4) взятие остатка от деления, 5) сложение и вычитание.

Порядок вычислений в выражении можно изменить, используя круглые скобки. Например, в формуле (8-3*(4-2))/(3+2) сначала выполняется операция 4-2, затем умножение на 3, затем вычитание из 8 предыдущего результата, сложение 3+2 и, наконец, деление.

При построении арифметических выражений можно использовать встроенные функции, часть которых приведена в таблице 6.

При работе со случайными числами кроме функции RND использу-

ется оператор RANDOMIZE. RANDOMIZE [выражение] – математический оператор, включающий генератор случайных чисел. Если генератор случайных чисел не запущен, то функция RND будет выдавать одну и ту же последовательность чисел при каждом запуске программы.

Таблица 6 – Математические функции

Функция	**Описание**
ABS(x)	абсолютная величина числа *x*
FIX(x)	число, полученное отбрасыванием дробной части числа *x*
CINT(x)	число, полученное округлением числа *x*
INT(x)	целая часть числа *x*
COS(x)	косинус числа *x*
SIN(x)	синус числа *x*
TAN(x)	тангенс числа *x*
ATN(x)	арктангенс числа *x*
SQR(x)	корень квадратный из числа *x*
EXP(x)	e^x
LOG(x)	ln *x*
RND [(x)]	псевдослучайное число в диапазоне [0,1)

В состав *строковых* выражений могут входить переменные и константы строкового типа, строки и строковые функции. Строками являются любые последовательности символов, заключенные в кавычки. В строковых выражениях можно использовать операцию конкатенации (строкового сложения), которая обозначается знаком «&». Например, пусть строковое выражение будет включать строку «ин», строковую переменную strA, значением которой является строка «форма», и строку «тика»: «ин»+strA+«тика». Тогда значение строкового выражения будет «информатика».

В состав *логических* выражений кроме логических операндов могут входить также числа, числовые выражения и операции сравнения. Логическое выражение может принимать два значения: «Истина» (True) или «Ложь» (False). В логических выражениях используются операторы сравнения, приведенные в таблице 7.

Пример:

5>3 – истинно;

2*2=5 – ложно.

Таблица 7 – Операторы сравнения

Оператор	Назначение
=	равно
>	больше
<	меньше
<>	не равно
>=	больше или равно
<=	меньше или равно

Над логическими выражениями можно совершать действия логической математики, при этом используются логические операторы (таблица 8).

Таблица 8 – Логические операторы

Логические операторы	Описание
AND	возвращает значение «Истина», если все участвующие в операции выражения тоже имеют значение «Истина»
OR	возвращает значение «Истина», если хотя бы одно из участвующих в операции выражений имеет значение «Истина».
XOR	возвращает значение «Истина», если только одно из участвующих в операции выражений имеет значение «Истина»
NOT	возвращает обратное для значения выражения значение.

Логические выражения используются в *управляющих конструкциях* IF и SELECT CASE, предназначенных для управления порядком выполнения команд. Синтаксис конструкции IF следующий:

IF логическое_выражение THEN действие1 ELSE действие2

Данная конструкция позволяет проверить значение логического выражения. Если логическое_выражение возвращает значение «Истина», то выполняется действие1, иначе – действие2.

Пример: *IF a>0 THEN form1.print «a>0» ELSE form1.print «a≤0»*

Блочная структура оператора IF используется в том случае, если необходимо выполнить не одно действие, а группу:

IF логическое_выражение THEN
действие11
действие12

действие13
ELSE
действие21
действие22
End If

Конструкция SELECT CASE позволяет обрабатывать несколько значений логического выражения. Эта конструкция состоит из анализируемого выражения и набора операторов CASE на каждое возможное значение выражения. Синтаксис конструкции SELECT CASE следующий:

SELECT CASE логическое_выражение
CASE значение1
действие11
действие12
CASE значение2
действие21
действие22
. . .
CASE ELSE
действие31
действие32
END SELECT

Пример: Найти значения синуса и косинуса угла, введенного пользователем. Если синус или косинус угла равен единице, то закрыть форму Form1.

```
Private Sub Command1_Click ()
угол = Val(Inputbox("Введи значение угла"))
Text1.Text= "sin(" & угол &")= " & sin(угол)
Text2.Text= "cos(" & угол &")= " & cos(угол)
If sin(угол)=1 or cos(угол)=1 then Form1.Hide
End Sub
```

Пример: Определить случайное число *x* в интервале [100, 3000). Уменьшить ширину формы на *x*.

```
Private Sub Command1_Click ()
x = Rnd * 2900 + 100
Form1.Width= Form1.Width - x
End Sub
```

Задание к практической работе №5

1. Создать форму (рис. 10). Найти сумму чисел a и b, проверить на равенство, найти большее, вычислить значение c по формуле $c = \sqrt{a^2 + b^2 + \cos(a+b)}$. Результат вывести с помощью функции MsgBox.

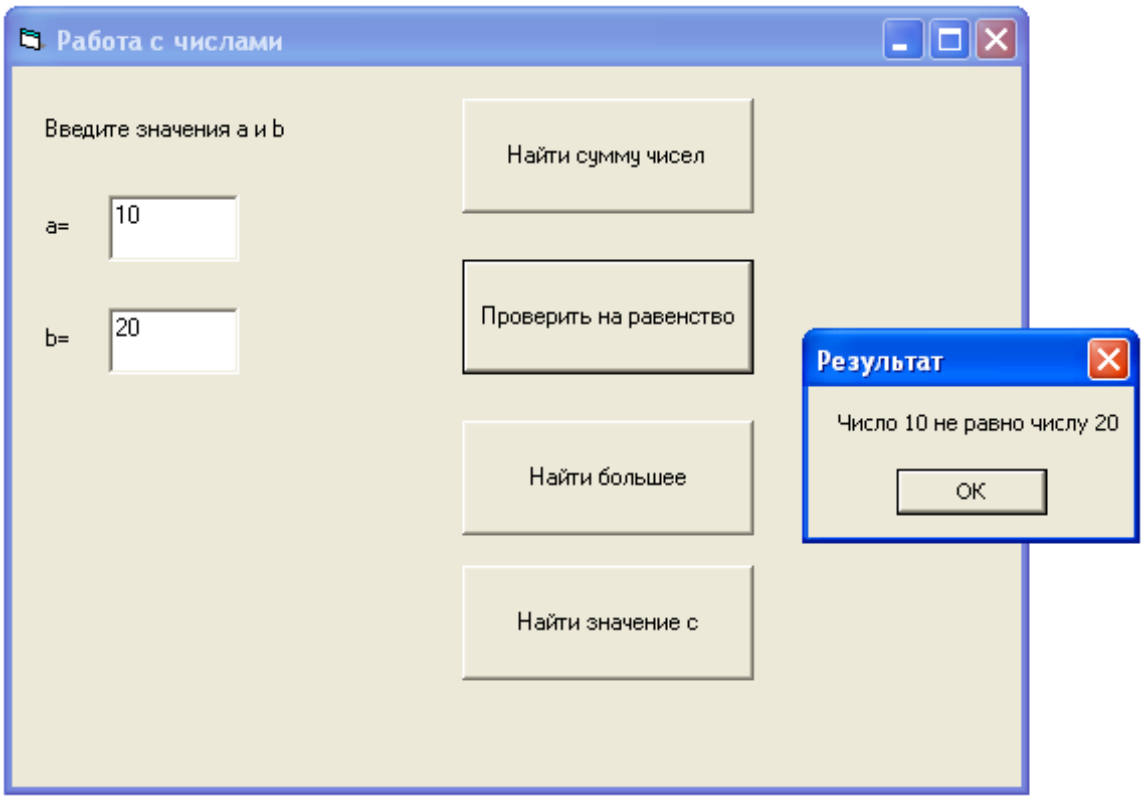


Рис. 10 – Форма «Работа с числами»

2. Создать форму (рис. 11). Создать объект Shape1 на форме. При нажатии на командные кнопки:

- перемещать объект по горизонтали на 100 точек;
- перемещать объект по вертикали на 100 точек;
- перемещать объект в координату A(x, y), где x, y – случайные числа;
- перемещать объект по синусоиде в течение 5 секунд;
- закрыть форму и выйти из программы;
- дополнительно к существующей форме добавить комбинированные списки для выбора размера и цвета объекта;
- добавить возможность изменять размер формы.

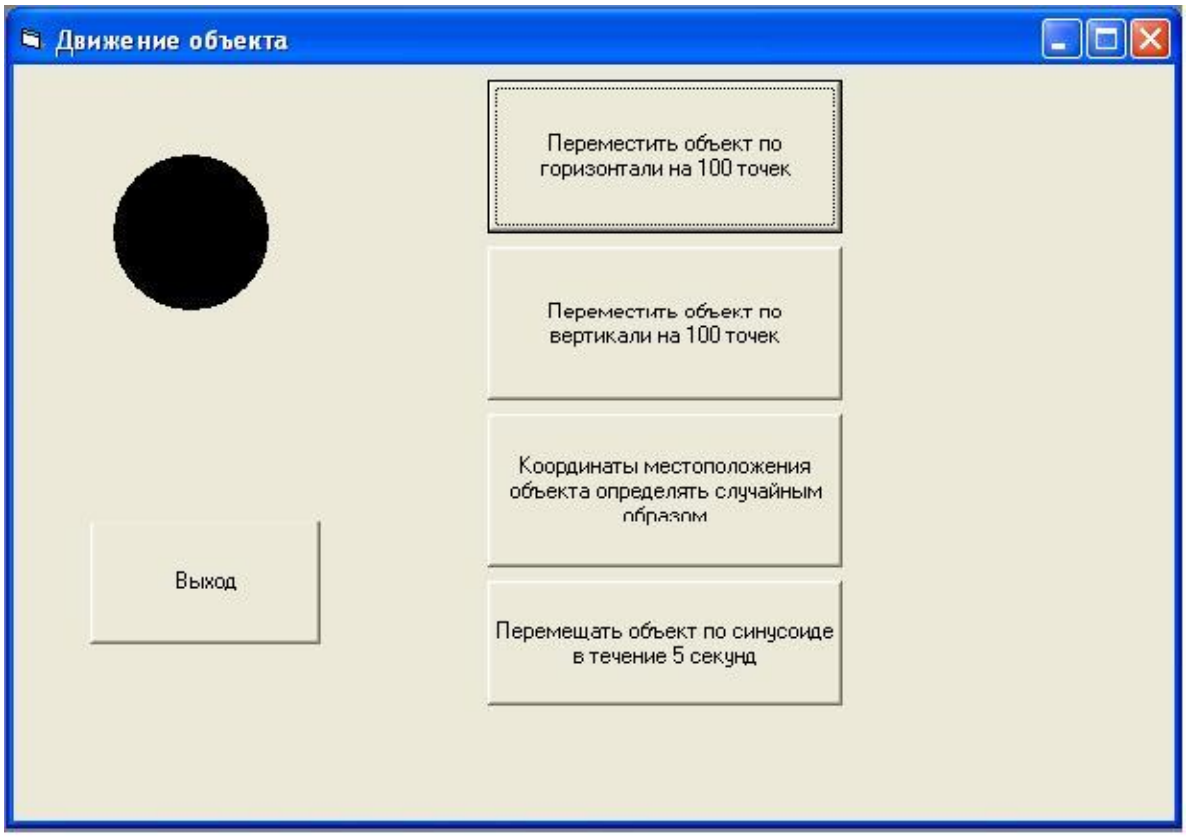


Рис. 11 – Форма «Движение объекта»

Практическая работа №6
Строки

Цель работы: Научиться работать со строковым типом данных.

Теоретические положения

Строковый тип – тип данных, значениями которого является произвольная последовательность символов алфавита. Каждая переменная такого типа может быть представлена фиксированным количеством байтов или иметь произвольную длину.

В состав *строковых* выражений могут входить переменные и константы строкового типа, строки и строковые функции. Строками являются любые последовательности символов, заключенные в кавычки. В VBA для работы со строками можно использовать только один оператор – оператор объединения. С помощью данного оператора можно объединять несколько строк в одну. Этот оператор обозначается символом амперсанда (&).

Например, объединение строк удобно использовать при формировании полного адреса, если известен индекс, город и улица. В следующем примере показан результат объединения фамилии, имени и отчества:

```
sLastName = “Иванов ”
sFirstName = “Иван ”
sSecondName = “Иванович”
sName = sLastName & sFirstName & sSecondName
Form1.Print sName ' Возвращает “Иванов Иван Иванович”
```

Список наиболее часто используемых функций для работы со строками приведен в таблице 9.

Пример: Найти количество цифр в строке, вводимой пользователем с клавиатуры.

```
Private Sub Command1_Click ()
a=Inputbox(“Введи строку”)
For  i=1 To Len(a)
b= MID (a, i ,1)
If Asc (b)>=48 And  Asc (b)<=57 Then
kol=kol+1
End If
Next  i
Text1.Text=“Количество цифр в строке = ” & kol
End Sub
```

Таблица 9 – Строковые функции

Функция	**Назначение**
AsC(*x*)	Возвращает ASCII-код символа *x*
Chr(*x*)	Преобразовывает ASCII-код *x* в символ
InStr, InStrRev	Осуществляют поиск одной строки в другой
LCase (*строка*)	Изменяет регистр букв исходной строки на нижний
Left (*строка, n*)	Возвращает *n* символов с начала строки
Len (*строка*)	Возвращает количество символов в строке
Trim (*строка*)	Удаляют пробелы, расположенные с обеих сторон строки
Mid (*строка, n, m*)	Возвращает *m* символов строки, начиная с *n*
Right (*строка, n*)	Возвращает *n* символов с конца строки
Str (*число*)	Преобразовывает числовое выражение в строку
StrReverse (*строка*)	Изменяет порядок следования символов в строке на обратный
StrConv (*строка*)	Изменяет регистр букв символьной строки
Val (*строка*)	Преобразовывают строку в числовое выражение
UCase (*строка*)	Изменяет регистр букв исходной строки на верхний

Пример: Вывести в текстовое окно все гласные буквы из введенного пользователем предложения.

```
Private Sub Command1_Click()
строка=Inputbox("Введи строку")
For  i=1 To Len(строка)
    символ= MID (строка, i ,1)
    If символ= "а" or  символ= "е" or символ= "у" or символ= "ы" or _
    символ= "о" or символ= "э" or символ= "я" or символ= "и" or _
    символ= "ю" Then Text1.Text=Text1.Text & символ
Next  i
End Sub
```

Задание к практической работе №6

1. Вывести в текстовое окно таблицу кодов ASCII (рисунок 12).
2. Создать форму для работы со строковыми данными (рисунок 13). Найти:

- количество букв в предложении;

- количество букв “А” в предложении;
- количество слов в предложении;
- перевести предложение в верхний регистр;
- перевернуть предложение.

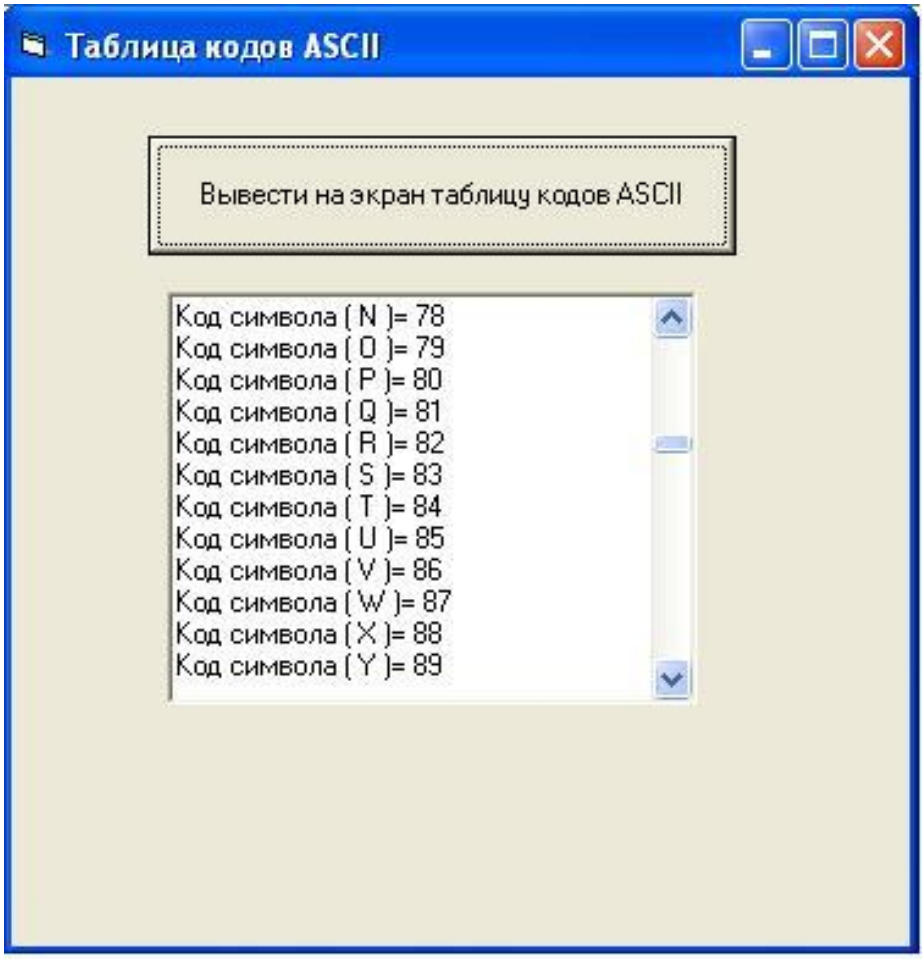


Рис. 12 – Форма «Таблица кодов ASCII»

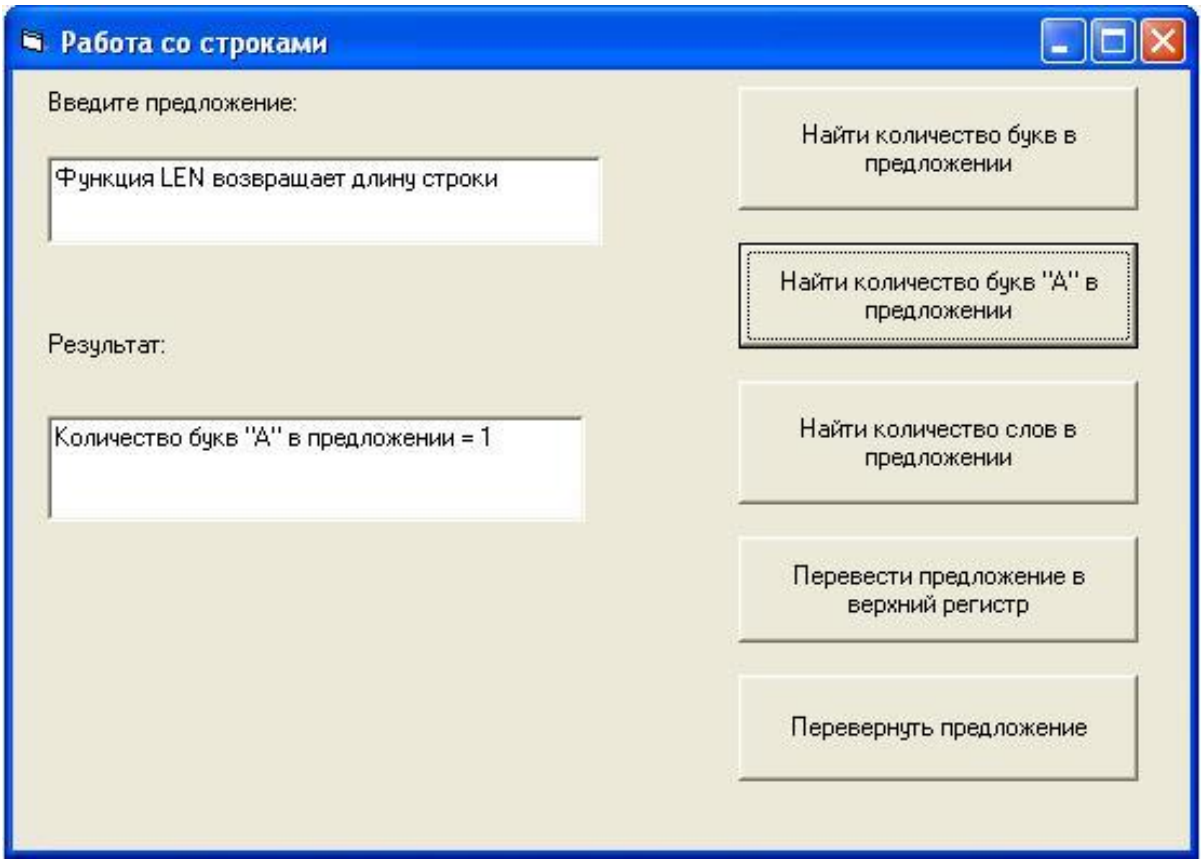


Рис. 13 – Форма «Работа со строками»

Практическая работа №7
Работа с массивами

Цель работы: Освоение навыков создания статических и динамических массивов. Заполнение массивов данными, вводимыми пользователем с клавиатуры, случайными числами. Извлечение данных из массивов.

Теоретические положения

Для хранения данных 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 называется *размером массива.* Размер массива ограничен объемом оперативной памяти и типом данных элементов массива.

Все элементы массива имеют одинаковый тип. Однако если массив имеет тип variant, то отдельные элементы могут содержать данные разных типов. Например, одни элементы могут быть числами, другие – строками или объектами.

Индекс элемента указывается в круглых скобках после имени массива. Например, strNames(l), strNames(2), strNames(10) являются элементами массива с именем strNames. Каждый из элементов массива можно использовать точно так же, как и простую переменную.

В Visual Basic существуют массивы *фиксированного размера (статические) и динамические массивы*. Массив фиксированного размера имеет неизменный размер, заданный при его объявлении. Динамические массивы могут изменять размер в процессе выполнения.

Объявление массива фиксированного размера зависит от области его видимости и осуществляется следующим образом:

- глобальный массив объявляется с помощью оператора public в секции Declaration модуля;
- массив уровня модуля – с помощью оператора private в секции Declaration модуля;
- локальный массив – с помощью оператора private в процедуре.

При объявлении массива после его имени в круглых скобках указывается верхняя граница массива. По умолчанию нижней границей массива является 0. Например, в приведенном ниже коде, который вводится в секцию Declaration модуля, задается массив из 21 элемента. Индекс элементов массива изменяется от 0 до 20:

Dim A (20) As Integer

Для создания глобального массива такого же размера необходимо использовать следующий код:

Public B (20) As Integer

Можно явно задать нижнюю границу массива, используя ключевое слово To: *A(1 To 20) As Integer*, в этом случае задается массив из 20 элементов. Индекс элементов массива изменяется от 1 до 20.

Visual Basic позволяет использовать многомерные массивы. Например, в следующем коде объявляется двумерный массив размерностью 21x21:

Dim B (20, 20) As Integer

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

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

Dim B (20, 1 To 20) As Integer

В верхней строке кода явно заданы верхняя и нижняя граница обеих размерностей. В нижней строке задана верхняя и нижняя граница только для второй размерности.

В случае, когда размер массива заранее неизвестен, Visual Basic позволяет использовать динамические массивы, размеры которых можно изменять во время выполнения. Применение динамических массивов позволяет эффективно управлять памятью, выделяя память под большой массив лишь на то время, когда этот массив используется, а затем освобождая ее.

Создание динамического массива осуществляется следующим образом:

1. Объявляется массив с помощью ключевых слов, используемых при создании массива фиксированного размера. Список размерностей массива остается пустым. Например, Dim A( ) As Integer.

2. С помощью выполняемого оператора ReDim указывается размерность массива в виде числа или выражения. Синтаксис оператора ReDim аналогичен синтаксису оператора объявления массива фиксированного размера:

ReDim A (x)

ReDim B (1 To 20, 30)

При выполнении оператора ReDim данные, размещенные в массиве ранее, теряются. Это удобно в том случае, если данные больше не нужны. Если необходимо изменить размер массива, не потеряв при этом данных,

то оператор ReDim используется 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)

Использование оператора ReDim с ключевым словом Preserve позволяет изменять только верхнюю границу последней размерности многомерных массивов.

В большинстве случаев массивы заполняются в циклах For…Next, при этом счётчик цикла используется в качестве индекса элемента массива.

Пример: Создать массив A(25). Заполнить массив случайными числами в диапазоне [10,100). Массив вывести в текстовое окно Text1.

```
Private Sub Command1_Click()
Dim A(25) As Integer
Randomize Timer
For  k=1 To 25
A(k)=Rnd*90+10
Text1.Text= Text1.Text & A(k)
Next  k
End Sub
```

Пример: Вывести на экран массив A(10,2), заполненный введенными с клавиатуры числами. Найти минимальный элемент массива.

```
Private Sub Form_Load()
Dim A(10,2) As Single
'заполнение массива
For  i=1 To 10
For  j=1 To 2
A(i,j)=Val (Inputbox ("Введи элемент массива"))
Text1.Text= Text1.Text & A(i, j)
Next  j
Text1.Text= Text 1.Text & Chr(13) & Chr(10)
Next  i
 'поиск минимального элемента массива
Минимум=A(1,1)
For  i=1 To 10
For  j=1 To 2
If A(i, j)< Минимум Then
Минимум = A(i, j)
```

```
Номер_строки = i
Номер_столбца = j
End If
Next  j
Next  i
MsgBox “Минимальный элемент массива, равный ” & Минимум &_
“Находится на” & Номер_строки & “ строке, в “ & Номер_столбца_
& “ столбце.”
End Sub
```

Задание к практической работе №7

1. Создать одномерный массив A(20). Заполнить его случайными числами в заданном диапазоне. Вывести массив на экран (рис. 14).
2. Создать двумерный массив A(3,6). Заполнить массив целыми случайными числами. Вывести массив на экран (рис. 15). Найти:

- сумму тех элементов массива, которые превышают число “10”;
- наибольший элемент массива;
- количество элементов в массиве, значение которых превышает наперед заданное число;
- вывести на экран первую строку массива в отдельное текстовое окно.

3. Создать динамический массив (рис. 16). Заполнить массив случайными числами и вывести массив на экран. Найти:

- сумму всех элементов массива;
- найти разность 3-го и 5-го элемента массива;
- найти количество четных чисел в массиве.

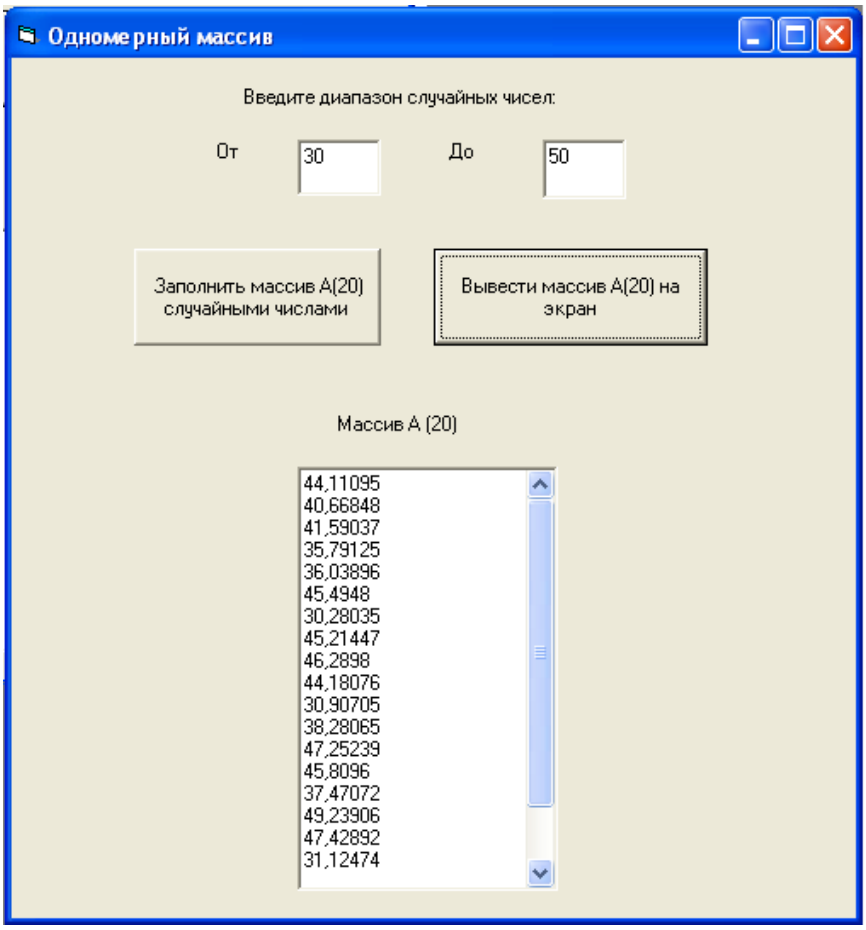


Рис.14 – Форма «Одномерный массив»

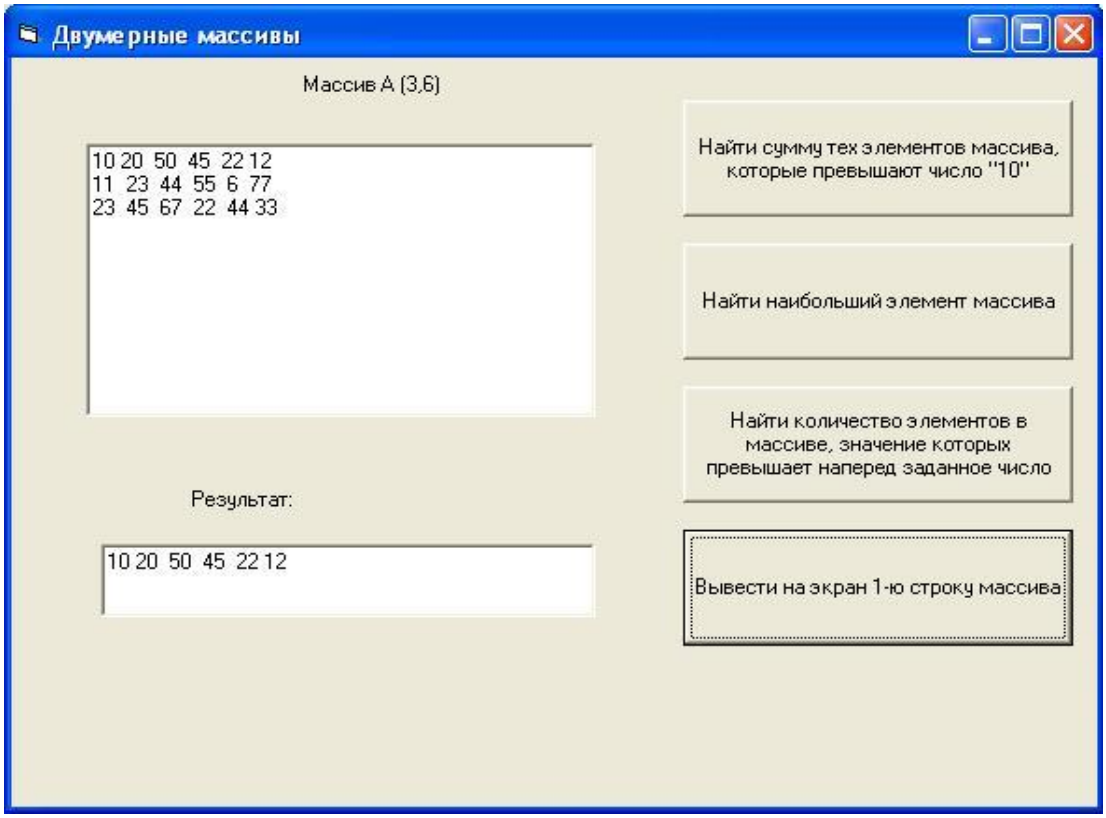


Рис. 15 – Форма «Двумерные массивы»

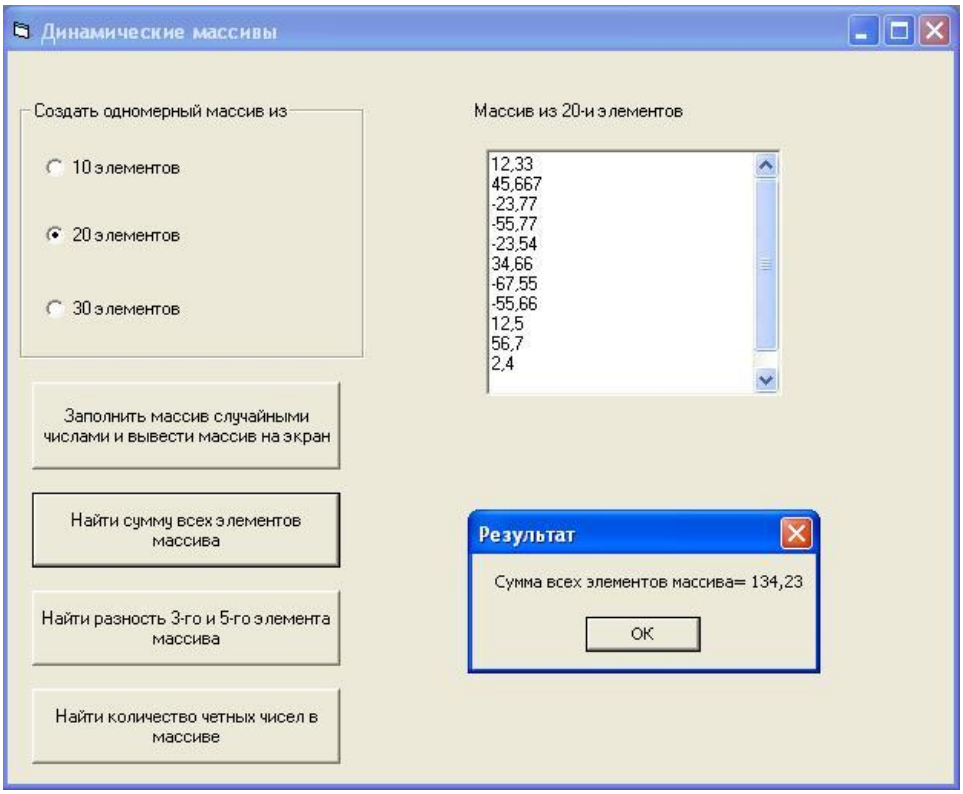


Рис.16 – Форма «Динамические массивы»

Практическая работа №8
Программирование модуля. Организация процедур и функций

Цель работы: Изучение основ программирования стандартных модулей, организации процедур и функций.

Теоретические положения

Процедуры VBA бывают двух типов:

• процедуры обработки событий;

• общие процедуры.

Имя процедуры обработки события, связанного с элементом управления, состоит из имени элемента управления, символа подчеркивания и имени события, например *Закрытъ_ click* – процедура обработки нажатия кнопки *Закрыть* в форме.

Общие процедуры VBA могут храниться в любом типе модулей VBA, так как они не связаны с конкретным объектом. Они выполняются только тогда, когда явно вызываются другими процедурами. Обычно эти процедуры реализуют какие-то общие действия, которые могут вызываться разными процедурами обработки событий.

Процедуры, как и переменные, должны быть объявлены до того, как они могут быть вызваны. Объявления общих процедур помещаются в разделе General (Общая область) модуля. Процедуры обработки событий хранятся в разделах модуля формы или отчета, соответствующих связанным с этими процедурами объектам.

В свою очередь, процедуры VBA делятся на ***подпрограммы*** и ***функции***. Они являются фрагментами программного кода, который заключается между операторами Sub и End Sub или между Function и End Function соответственно. ***Процедуры-подпрограммы*** выполняют действия, но не возвращают значение, поэтому они не могут быть использованы в выражениях. Процедуры обработки событий представляют собой процедуры-подпрограммы. ***Процедуры-функции*** всегда возвращают значение, поэтому они обычно используются в выражениях. Общие процедуры могут быть как процедурами-подпрограммами, так и процедурами-функциями.

Синтаксис процедуры-подпрограммы VBA:

Sub <имяПроцедуры> (<аргумент1>, <аргумент2>, ...) <оператор1>
<оператор2>
End Sub

Список аргументов у процедуры может отсутствовать и может содер-

жать необязательные аргументы.

Объявление каждого аргумента имеет следующий синтаксис:
<имяАргумента> [As <типДанных> [=<значениеПоУмолчанию>]],
где *<имяАргумента>* – идентификатор, составленный согласно правилам создания имен и представляющий аргумент в теле процедуры;
<типДанных> – это либо встроенный тип данных, либо тип, определенный пользователем. Тип данных аргумента может не указываться, и тогда считается, что он имеет тип Variant. Аргументом процедуры может быть и массив. Тогда после имени аргумента должны стоять круглые скобки. Для необязательного аргумента может быть указано *<значение по умолчанию>,* которое будет использоваться, если этот аргумент будет опущен. Если значение по умолчанию не указано, необязательный аргумент инициируется точно так же, как переменная, т. е. числовой аргумент – в 0, строковый – в строку нулевой длины и т. д.
Описание функции:
Function <имяФункции> (<аргумент1>, <аргумент2>, ...) [As <тип3начение>]
<оператор1>
<оператор2>
<имяФункции> = <возвращаемоеЗначение>
End Function

Кроме того что ключевое слово Sub заменяется на Function, в теле функции обязательно присутствует оператор присваивания имени функции какого-нибудь значения. Это значение и возвращается функцией. В заголовке функции может быть описан тип возвращаемого значения. Если этот тип не указан, функция возвращает значение Variant.

Вызов подпрограмм и функций

Чтобы использовать написанную подпрограмму или функцию, ее нужно *вызвать.* Вызов процедуры-подпрограммы отличается от вызова процедуры-функции.

Обычно подпрограмма вызывается из другой подпрограммы или функции с помощью специального оператора VBA. Если она имеет аргументы, ей передается список *фактических параметров.*
Оператор вызова подпрограммы может использоваться в двух формах:
<имя Процедуры><список фактических параметров>
или
Call <имя Процедуры> (<список фактических параметров>).

В первом случае список фактических параметров задается без скобок, во втором – использование скобок обязательно. Но всегда список фактических параметров должен полностью соответствовать списку аргументов, заданному в объявлении подпрограммы. Все фактические параметры для обязательных аргументов должны быть перечислены в том порядке, в каком они присутствуют в описании подпрограммы, после чего могут идти параметры для необязательных аргументов.

Вызов функции имеет следующий вид:

<имя переменной>=<имя функции>(<список фактических параметров>).

Список фактических параметров при вызове функции должен обязательно заключаться в кавычки.

Например, вызов функции SummaVar может выглядеть следующим образом:

```
Dim intS As Integer
intS = SummaVar(1,2,3,4,5,6,7,8,9) 'суммируем целые числа
```

Задание к практической работе №8

1. Создать программу, работающую с процедурами и функцией, параметрами которых являются значения, вводимые в текстовые поля формы. После нажатия кнопки *Счет* на форме основная программа вызывает процедуры и функцию и выводит полученные результаты в три соответствующие метки.

Процедуры и функция выполняют следующие действия:

а) первая процедура производит суммирование двух первых введенных значений a и b;

б) вторая процедура производит умножение третьего и четвертого введенных значений c и d;

в) функция вычисляет выражение a+b-c*d.

2. Создать программу, которая, используя данные, считанные из текстовых полей формы, выполняет вычисление значений двух выражений: $y = \sin(a + b) + 5$ и $z = \mathrm{tg}(c + + 3) – \cos(3 * d)$, где *a, b, c, d* – значения переменных, введенных в соответствующие текстовые поля. Данные значения *y* и *z* вычисляются в соответствующих процедурах (процедуре) или функциях. Основной блок программы сравнивает полученные значения, и об этом выдается сообщение на форму.

3. Создать программу (рис. 17), рассчитывающую значения выражений в зависимости от выбранного переключателя на форме и введенных

значений в текстовые поля. Основная программа после нажатия на кнопку *Вычислить* вызывает процедуру *Shet*, рассчитывающую все необходимые выражения. После нажатия на кнопку *Закрыть* разработанное приложение закрывается.

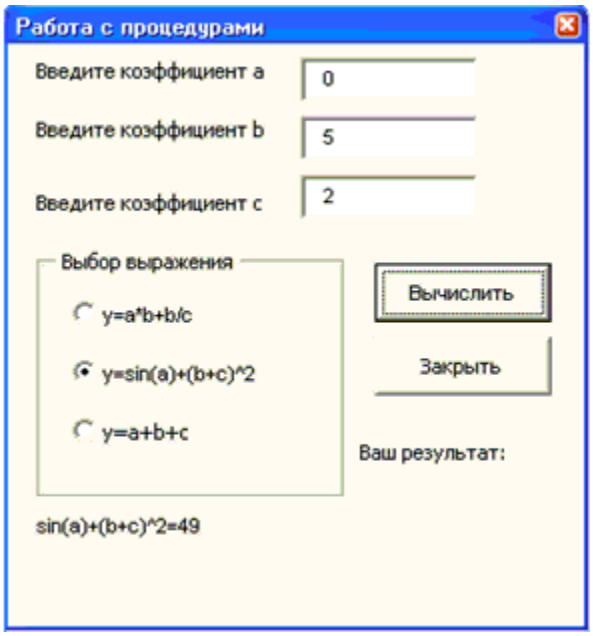


Рис.17 – Форма

4. Разработать приложение «Калькулятор».

Практическая работа №9
Организация доступа к объектам табличного процессора

Цель работы: Изучение способов обращения к основным объектам табличного процессора.

Теоретические положения

Термин **Объекты Excel** (понимаемый в широком смысле, как объектная модель Excel) включает в себя элементы, из которых состоит любая рабочая книга Excel. Это, например, рабочие листы (**Worksheets**), строки (**Rows**), столбцы (**Columns**), диапазоны ячеек (**Ranges**) и сама рабочая книга Excel (**Workbook**) в том числе. Каждый объект Excel имеет набор свойств, которые являются его неотъемлемой частью.

Например, объект **Worksheet** (рабочий лист) имеет свойства **Name** (имя), **Protection** (защита), **Visible** (видимость), **Scroll Area** (область прокрутки) и так далее. Таким образом, если в процессе выполнения макроса требуется скрыть рабочий лист, то достаточно изменить свойство **Visible** этого листа.

В Excel VBA существует особый тип объектов – **коллекция**. Как можно догадаться из названия, коллекция ссылается на группу (или коллекцию) объектов Excel. Например, коллекция **Rows** – это объект, содержащий все строки рабочего листа.

Доступ ко всем основным объектам Excel может быть осуществлён (прямо или косвенно) через объект **Workbooks**, который является коллекцией всех открытых в данный момент рабочих книг. Каждая рабочая книга содержит объект **Sheets** – коллекция, которая включает в себя все рабочие листы и листы с диаграммами рабочей книги. Каждый объект **Worksheet** состоит из коллекции **Rows** – в неё входят все строки рабочего листа, и коллекции **Columns** – все столбцы рабочего листа, и так далее.

В следующей таблице перечислены некоторые наиболее часто используемые объекты Excel.

Таблица 10 – Объекты Excel

Объект	**Описание**
Application	Приложение Excel.
Workbooks	Коллекция всех открытых в данный момент рабочих книг в текущем приложении Excel. Доступ к какой-то конкретной рабочей книге может быть осуществлён через объект

	Workbooks при помощи числового индекса рабочей книги или её имени, например, **Workbooks(1)** или **Workbooks(«Книга1»)**.
Workbook	Объект **Workbook** – это рабочая книга. Доступ к ней может быть выполнен через коллекцию **Workbooks** при помощи числового индекса или имени рабочей книги (см. выше). Для доступа к активной в данный момент рабочей книге можно использовать **ActiveWorkbook**. Из объекта **Workbook** можно получить доступ к объекту **Sheets**, который является коллекцией всех листов рабочей книги (рабочие листы и диаграммы), а также к объекту **Worksheets**, который представляет из себя коллекцию всех рабочих листов книги Excel.
Sheets	Объект **Sheets**– это коллекция всех листов рабочей книги. Это могут быть как рабочие листы, так и диаграммы на отдельном листе. Доступ к отдельному листу из коллекции **Sheets** можно получить при помощи числового индекса листа или его имени, например, **Sheets(1)** или **Sheets(«Лист1»)**.
Worksheets	Объект **Worksheets** – это коллекция всех рабочих листов в рабочей книге (то есть, все листы, кроме диаграмм на отдельном листе). Доступ к отдельному рабочему листу из коллекции **Worksheets** можно получить при помощи числового индекса рабочего листа или его имени, например, **Worksheets(1)** или **Worksheets(«Лист1»)**.
Worksheet	Объект **Worksheet** – это отдельный рабочий лист книги Excel. Доступ к нему можно получить при помощи числового индекса рабочего листа или его имени (см. выше). Кроме этого Вы можете использовать **ActiveSheet** для доступа к активному в данный момент рабочему листу. Из объекта **Worksheet** можно получить доступ к объектам **Rows** и **Columns**, которые являются коллекцией объектов **Range**, ссылающихся на строки и столбцы рабочего листа. А также можно получить доступ к отдельной ячейке или к любому диапазону смежных ячеек на рабочем листе.
Rows	Объект **Rows** – это коллекция всех строк рабочего листа. Объект **Range**, состоящий из отдельной строки рабочего листа, может быть доступен по номеру этой строки, например,

	Rows(1).
Columns	Объект **Columns** – это коллекция всех столбцов рабочего листа. Объект **Range**, состоящий из отдельного столбца рабочего листа, может быть доступен по номеру этого столбца, например, **Columns(1)**.
Range	Объект **Range** – это любое количество смежных ячеек на рабочем листе. Это может быть одна ячейка или все ячейки листа. Доступ к диапазону, состоящему из единственной ячейки, может быть осуществлён через объект **Worksheet** при помощи свойства **Cells**, например, **Worksheet.Cells(1,1)**. По-другому ссылку на диапазон можно записать, указав адреса начальной и конечной ячеек. Их можно записать через двоеточие или через запятую. Например, **Worksheet.Range(«A1:B10»)** или **Worksheet.Range(«A1», «B10»)** или **Worksheet.Range(Cells(1,1), Cells(10,2))**. Обратите внимание, если в адресе **Range** вторая ячейка не указана (например, **Worksheet.Range(«A1»)** или **Worksheet.Range(Cells(1,1))**, то будет выбран диапазон, состоящий из единственной ячейки.

Приведённая выше таблица показывает, как выполняется доступ к объектам Excel через родительские объекты. Например, ссылку на диапазон ячеек можно записать вот так:

Workbooks("Книга1").Worksheets("Лист1").Range("A1:B10")

Присваивание объекта переменной

В Excel VBA объект может быть присвоен переменной при помощи ключевого слова **Set**:

Dim DataWb As Workbook

Set DataWb = Workbooks("Книга1.xlsx")

Активный объект

В любой момент времени в Excel есть активный объект **Workbook** – это рабочая книга, открытая в этот момент. Точно так же существует активный объект **Worksheet**, активный объект **Range** и так далее.

Сослаться на активный объект **Workbook** или **Sheet** в коде VBA можно как на **ActiveWorkbook** или **ActiveSheet**, а на активный объект **Range** – как на **Selection**.

Если в коде VBA записана ссылка на рабочий лист, без указания к какой именно рабочей книге он относится, то Excel по умолчанию обращается к активной рабочей книге. Точно так же, если сослаться на диапазон, не указывая определённую рабочую книгу или лист, то Excel по умолчанию обратится к активному рабочему листу в активной рабочей книге.

Таким образом, чтобы сослаться на диапазон **A1:B10** на активном рабочем листе активной книги, можно записать просто:

```
Range("A1:B10")
```

Смена активного объекта

Если в процессе выполнения программы требуется сделать активной другую рабочую книгу, другой рабочий лист, диапазон и так далее, то для этого нужно использовать методы **Activate** или **Select** вот таким образом:

```
Sub ActivateAndSelect()
  Workbooks("Книга2").Activate
  Worksheets("Лист2").Select
  Worksheets("Лист2").Range("A1:B10").Select
  Worksheets("Лист2").Range("A5").Activate
End Sub
```

Методы объектов, в том числе использованные только что методы **Activate** или **Select**, далее будут рассмотрены более подробно.

Свойства объектов

Каждый объект VBA имеет заданные для него свойства. Например, объект **Workbook** имеет свойства **Name** (имя), **RevisionNumber** (количество сохранений), **Sheets** (листы) и множество других. Чтобы получить доступ к свойствам объекта, нужно записать имя объекта, затем точку и далее имя свойства. Например, имя активной рабочей книги может быть доступно вот так: **ActiveWorkbook.Name**. Таким образом, чтобы присвоить переменной **wbName** имя активной рабочей книги, можно использовать вот такой код:

```
Dim wbName As String
wbName = ActiveWorkbook.Name
```

Ранее мы показали, как объект **Workbook** может быть использован для доступа к объекту **Worksheet** при помощи такой команды:

```
Workbooks("Книга1").Worksheets("Лист1")
```

Это возможно потому, что коллекция **Worksheets** является свойством объекта **Workbook**.

Некоторые свойства объекта доступны только для чтения, то есть их

значения пользователь изменять не может. В то же время существуют свойства, которым можно присваивать различные значения. Например, чтобы изменить название активного листа на «**Мой рабочий лист**«, достаточно присвоить это имя свойству **Name** активного листа, вот так:

ActiveSheet.Name = "Мой рабочий лист"

Методы объектов

Объекты VBA имеют методы для выполнения определённых действий. **Методы объекта** – это процедуры, привязанные к объектам определённого типа. Например, объект **Workbook** имеет методы **Activate**, **Close**, **Save** и ещё множество других.

Для того, чтобы вызвать метод объекта, нужно записать имя объекта, точку и имя метода. Например, чтобы сохранить активную рабочую книгу, можно использовать вот такую строку кода:

ActiveWorkbook.Save

Как и другие процедуры, методы могут иметь аргументы, которые передаются методу при его вызове. Например, метод **Close** объекта **Workbook** имеет три необязательных аргумента, которые определяют, должна ли быть сохранена рабочая книга перед закрытием и тому подобное.

Чтобы передать методу аргументы, необходимо записать после вызова метода значения этих аргументов через запятую. Например, если нужно сохранить активную рабочую книгу как файл **.csv** с именем «Книга2», то нужно вызвать метод **SaveAs** объекта **Workbook** и передать аргументу **Filename** значение **Книга2**, а аргументу **FileFormat** – значение **xlCSV**:

ActiveWorkbook.SaveAs "Книга2", xlCSV

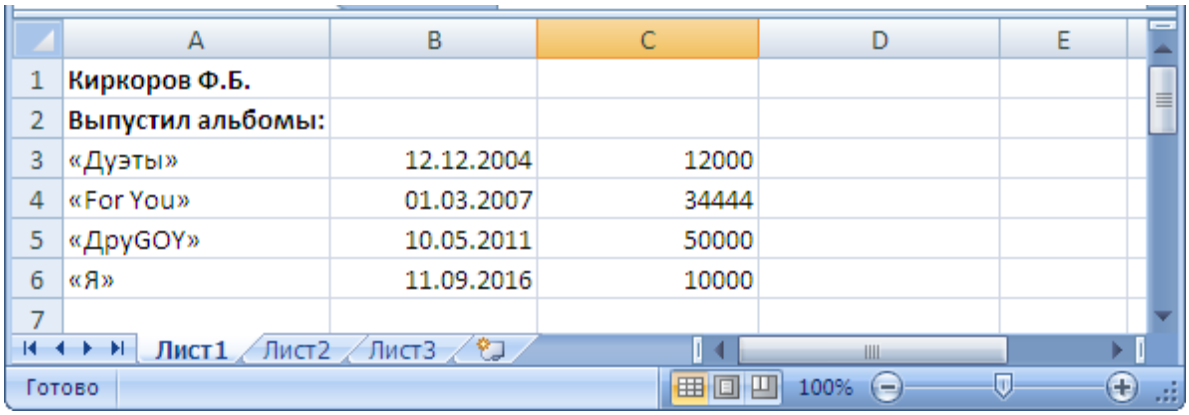
Чтобы сделать код более читаемым, при вызове метода можно использовать именованные аргументы. В этом случае сначала записывают имя аргумента, затем оператор присваивания «**:=**» и после него указывают значение. Таким образом, приведённый выше пример вызова метода **SaveAs** объекта **Workbook** можно записать по-другому:

ActiveWorkbook.SaveAs Filename:="Книга2", [FileFormat]:=xlCSV

В окне **Object Browser** редактора Visual Basic показан список всех доступных объектов, их свойств и методов. Чтобы открыть этот список, запустите редактор Visual Basic и нажмите **F2**.

Задание к практической работе №9

1. Найти сумму всех положительных чисел, записанных на Листе1.
2. На листе имеется область, ограниченная рамкой. Необходимо залить эту область определенным цветом.
3. Написать программу, которая вводимые пользователем данные об ***n*** альбомах исполнителя выводит на Лист1 Excel (рис. 18).

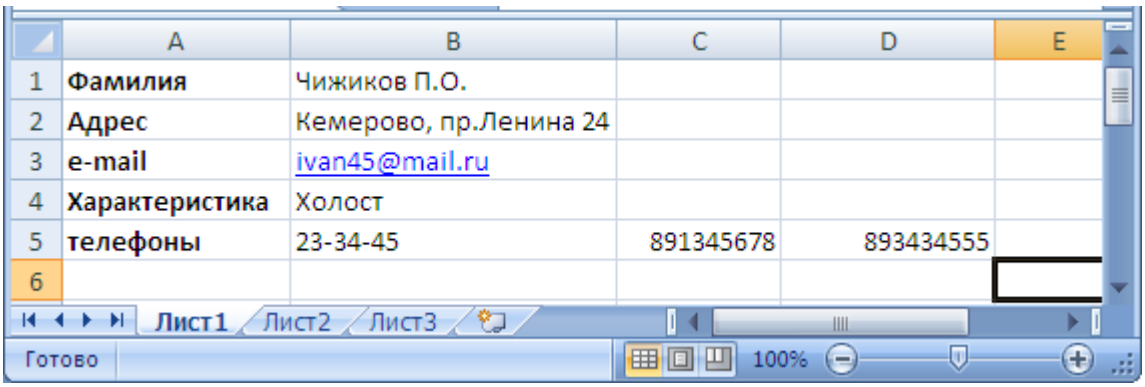


	A	B	C	D	E
1	**Киркоров Ф.Б.**				
2	**Выпустил альбомы:**				
3	«Дуэты»	12.12.2004	12000		
4	«For You»	01.03.2007	34444		
5	«ДруGOY»	10.05.2011	50000		
6	«Я»	11.09.2016	10000		
7					

Лист1 Лист2 Лист3 Готово 100%

Рис.18 – Лист Excel

4. Сформировать файл Excel для 5 сотрудников (рис. 19). Данные по каждому сотруднику разместить на отдельных листах Лист1…Лист15. Написать программу на VBA для считывания данных из файла Excel и вывода ее на экран.



	A	B	C	D	E
1	**Фамилия**	Чижиков П.О.			
2	**Адрес**	Кемерово, пр.Ленина 24			
3	**e-mail**	ivan45@mail.ru			
4	**Характеристика**	Холост			
5	**телефоны**	23-34-45	891345678	893434555	
6					

Лист1 Лист2 Лист3 Готово 100%

Рис.19 – Лист Excel

Практическая работа №10
Автоматизация стандартных текстовых документов

Цель работы: Изучение основ работы со стандартными текстовыми документами.

Теоретические положения

Ключевым в объектной модели Word является объект Application, так как он содержит все остальные объекты Word. Его элементами на разных уровнях иерархии являются около 180 объектов. Сам корневой объект Application имеет более сотни элементов: свойств, методов и событий.

Свойства объекта Word.Application

Свойства любого объекта делятся на две группы: свойства-участники (объекты) и терминальные свойства (обычные переменные VBA).

Единую систему организации панелей меню и инструментальных кнопок обеспечивает объект CommandBars, справок – Assistant, поиска – FileSearch.

Центральными объектами Word являются коллекции Documents и Templates, точнее составляющие их элементы, сам документ и шаблоны. Объект AutoCorrect поддерживает работу по автоматической коррекции набираемых текстов. Его возможности эквивалентны команде Автозамена меню Сервис.

Объект Browser позволяет перемещать точку вставки, указывающую на объекты в документе. Коллекция объектов Dialogs представляет совокупность диалоговых окон, встроенных в Word. Добавлять новые или удалять элементы этой коллекции программным путем нельзя. Но соответствующие окна можно открыть и показать на экране дисплея и тем самым организовать диалог пользователем по теме, заданной соответствующим окном.

Три объекта, связанные с проверкой грамматики и орфографии: Languages, Dictionaries, SpellingSuggestions, – позволяют установить нужный язык, выбрать словарь, в том числе пользовательские словари, а также работать со списком слов, предлагаемых для исправления при обнаружении ошибки правописания. Команды Правописание и Язык меню Сервис предоставляют аналогичные, функциональные возможности при работе с документом вручную.

С помощью объекта Options можно программным путем установить различные опции приложения и документа аналогично тому, как если бы

вы выбрали команду Параметры в меню Сервис.

Работа с документами и класс Document

Когда открывается приложение, создается коллекция документов Documents, содержащая все открытые документы. В начальный момент коллекция содержит минимум один новый или ранее существовавший документ. Новый документ добавляется методом Add, а уже существующий – методом Open объекта Documents. Чтобы добраться до нужного документа, достаточно указать его индекс – имя файла, хранящего документ, или его порядковый номер в коллекции. Для той же цели можно использовать и метод Item, но обычно он опускается. Метод Save позволяет сохранить документ, а метод Close, сохраняя документ в файле, закрывает его и удаляет из коллекции.

Глобальное свойство Dialogs возвращает коллекцию диалоговых окон. Константа wdDialogFileOpen задает конкретное диалоговое окно – объект класса Dialog.

Классы, задающие структуризацию текста документа

Текст – это основа большинства документов. Его можно структурировать, оперируя различными единицами при решении тех или иных задач преобразования. Минимальной единицей текста обычно является символ. Кроме этого, существуют следующие единицы: слова, предложения, абзацы, а также более крупные образования: страницы, параграфы, главы.

Классы Characters, Words, Statements, Paragraphs, Sections позволяют работать с последовательностями (коллекциями) символов, слов, предложений, абзацев и разделов. Самой крупной единицей после абзаца выступает раздел. Элементом коллекций Characters, Words и Statements является объект класса Range. Объект Range позволяет работать как с одним элементом, так и с произвольной последовательностью элементов. Документы, поддокументы, абзацы, разделы – все они имеют метод или свойство Range, возвращающее интервал, связанный с объектом. Поэтому работа с текстом так или иначе ведется через методы и свойства объекта Range.

Объект Document

Объект Document может реагировать на три события, возникающие в результате действий пользователя. События объекта Document: Open, New, Close.

Рассмотрим основные классы, определяющие структуру документа.

1. Subdocuments (Subdocument) – коллекция и сам поддокумент. Есть некоторый разумный предел размера одного документа. Если в документе

больше 10–20 страниц, работать с ним становится неудобно. В этом случае в нем выделяют главный документ и поддокументы. Главный документ в этом случае имеет коллекцию поддокументов, каждый из них является, по сути, документом, с которым можно работать независимо. Метод AddFromRange класса SubDocuments создает поддокумент, выделяя из главного документа область, заданную параметром Range.

2. Tables (Table), TablesOfAuthoritiesCategories (T.O.A.C), TablesOfAuthorities (TableOfAuthorities), TablesOfContents (TablesOfContent), TablesOfFigures (TablesOfFigure). Класс Table определяет «обычные» таблицы с произвольным количеством строк и столбцов и произвольным заполнением полей. Остальные классы задают таблицы специального вида.

3. Shapes(Shape), InlineShapes(InlineShape) – эти две коллекции с их элементами позволяют добавлять в документ рисунки, но не только их. ActiveX– и OLE-объекты также являются элементами этих коллекций. Элементы этих двух коллекций отличаются тем, как они привязаны к документу: первые могут свободно перемещаться, вторые жестко привязаны к заданной области документа.

4. Lists(List), ListParagraphs(ListParagraph), listTemplates (ListTemplate) – списки удобно вводить в документ, когда имеешь дело с перечислением. Списки можно оформлять в соответствии с шаблоном. Существуют две группы шаблонов: нумерованные списки и списки-бюллетени. Коллекция ListTemplates содержит шаблоны оформления списков, а класс ListTemplate описывает конкретный шаблон. Шаблон применяется к списку абзацев и придает ему структуру, заданную шаблоном. Коллекция Lists содержит те списки документа (списки абзацев), что оформлены как нумерованные списки или списки-бюллетени. Коллекция ListParagraphs представляет список абзацев всех списков документа. Свойством ListParagraphs, которое возвращает объект соответствующего класса, обладает не только документ, но и объекты List и Range. Так что при наличии списка – объекта List можно выделить список абзацев. Чаще приходится выполнять обратную операцию – применять к списку абзацев один из возможных шаблонов, придав ему «настоящую» структуру списка. Тогда используют объект ListFormat.

5. Comments(Comment), Bookmarks(Bookmark), FootNotes (FootNote), EndNotes(EndNote), Fields(Field) – эти коллекции и их элементы отражают независимые, но близкие по духу понятия. Это части документа, косвенно связанные с ним. При нормальном просмотре документа они могут быть и

не видны.

• Коллекция comments и класс comment задают комментарии. Комментарии, как известно, вводятся для пояснения тех или иных терминов или понятий документа. Формально они приписываются некоторой области – объекту range.

• Большой документ, к отдельным частям которого приходится часто обращаться, стоит снабдить закладками. Коллекция bookmarks задает все закладки данного документа.

• Еще один способ комментирования – сноски. Они могут быть двух видов: подстраничные (внизу страницы) и концевые (в конце документа). Первые собраны в коллекцию footnotes, вторые – endnotes.

6. Fields (Field) – эта коллекция позволяет работать с полями документа. Одна из особенностей полей состоит в том, что их значения обновляются автоматически в зависимости от изменившихся внешних условий или контекста.

7. Story Ranges (Range) – эта коллекция представляет совокупность частей документа, называемых фрагментами (Story). Количество различных фрагментов документа фиксировано. Нельзя добавлять элементы в эту коллекцию обычным способом, используя метод Add. Фрагменты появляются в коллекции, когда создается соответствующая часть документа. Фрагменты имеют тип, задаваемый константами из перечисления wdStoryType. Главный фрагмент – текст документа, тип которого задается константой wdMainTextStory. Комментарии, ссылки, колонтитулы составляют фрагменты других типов, т. е. сам фрагмент является объектом Range. Так что благодаря фрагментам можно, например, работать с коллекцией комментариев как с единой областью.

8. Variables (Variable) – с документом можно связать коллекцию переменных типа Variant. Это важная для программистов коллекция, так как время жизни переменных, в нее входящих, совпадает со временем жизни документа. Тем самым появляется возможность сохранять информацию о работе той или иной процедуры между сеансами. Например, можно иметь счетчики, подсчитывающие число вызовов макроса, и в зависимости от этого по-разному определять его дальнейшую работу.

Объекты Range и Selection

Объект Document имеет метод Range, возвращающий объект Range, и метод Select, создающий объект Selection. Метод Range – это функция, возвращающая как результат объект Range; метод Select – это процедура

без параметров, которая создает объект Selection в качестве побочного эффекта. Объект Range имеет метод Select, превращающий область объекта Range в выделенную. Тем самым метод Select определяет новый объект Selection. Симметрично, объект Selection имеет свойство Range, возвращающее объект Range, соответствующий выделенной области.

Большинство ранее описанных частей документа являются и частями (свойствами) объектов Range и Selection. Объект Range напоминает матрешку: в каждую область вложена область поменьше. Вот пример корректного (хоть и не самого эффективного) задания объекта Range:
ActiveDocument.Range.Sections(1).Range.Paragraphs(l).Range.Sentences(1). Words(1).Characters(1)

Работа с текстом

Объекты Range и Selection позволяют выполнять основные операции над текстом: «выделить», «добавить», «заменить», «удалить». У наших объектов большой набор методов, позволяющих реализовать эти операции. Все рассматриваемые здесь методы принадлежат обоим объектам, если не сделана специальная оговорка.

Выделение

Выделить некоторую часть текста по существу означает определить объект Range или Selection. Объекты задают некоторую область в тексте документа, а их свойства Start и End позволяют установить начало и конец этой области. Меняя значения свойства, можно задать нужную область выделения.

Move является основным методом перемещения точки вставки. Остальные методы этой группы – в той или иной степени его модификации. Метод Move(Unit, Count) сжимает область в точку, стягивая ее в начало или конец, а затем перемещает точку вставки. Параметр Unit определяет единицы перемещения, а Count – количество этих единиц (по умолчанию 1). Знак переменной Count задает направление стягивания и перемещения. Положительные значения этого параметра задают стягивание к концу и перемещение вперед, отрицательные – стягивание в начало и перемещение назад. Чистое стягивание без перемещения точки вставки задается как перемещение на одну единицу. Метод возвращает количество единиц, на которое фактически произошло перемещение, или 0, если оно не осуществлено. Параметр Unit принимает значения wdCharacter (по умолчанию), wdWord, wdSentence, wdParagraph, wdSection, wdStory, wdCell, wdColumn, wdRow и wdTable.

Методы перемещения на сам текст не влияют – лишь изменяют область, заданную объектами Range и Selection. Поэтому эти методы применимы только к переменным типа Range, но не к фиксированным областям. Например, запись *ActiveDocument.Paragraphs(l).Range.Move*
не имеет эффекта, поскольку область первого абзаца – вещь неизменяемая. Метод Move стягивает область в точку, которая и перемещается, поэтому после его выполнения область исчезает, остается только точка вставки. Методы MoveStart и MoveEnd перемещают начальную или конечную точку области, обычно тем самым расширяя область.

Удаление текста

Метод Delete позволяет удалить текст. Вызванный без параметров, он удаляет вызывающий его объект Range или Selection. Если он применен в форме Delete(Unit,Count), удаляется часть текста в указанной области. Параметр Unit задает единицы, но при удалении возможны только два значения: wdWord и wdCharacter. Параметр Count задает количество удаляемых единиц. Если область стянута в точку, удаляются символы перед точкой вставки или после нее в зависимости от знака параметра Count.

Вставка текста

Группа методов Insert объектов Range и Selection позволяет осуществлять вставки в документ. Для вставки текста используются методы InsertBefore(Text) и InsertAfter(Text). Параметр text типа string задает текст, вставляемый до или после области, заданной объектами range или selection. После вставки текста область автоматически расширяется, включая в себя добавляемый текст.

Свойство Text позволяет заменять текст в выделенной области, поэтому нет нужды вызывать метод Insert(Text). Методы InsertBefore и InsertAfter безопасны, так как текст добавляется, не изменяя содержимого области. Совсем иное дело – методы вставки, которые далеко не безопасны. При вставке внутрь области, например при использовании метода InsertSymbol или InsertParagraph, заменяется содержимое области.

Работа с буфером

Метод Copy, не имеющий параметров, копирует объект (содержимое области) в буфер. Метод cut, действуя аналогично, должен бы заодно и удалять объект. Но сам объект не удаляется – только стягивается в точку, так что над ним возможны дальнейшие операции.

Иногда в буфер копируют не текст, а его формат. Этим занимается метод CopyFormat, копирующий формат по первому символу объекта se-

lection. Если этот символ – метка абзаца, копируется формат абзаца. Методом CopyFormat обладает только объект selection.

Метод Paste позволяет поместить («приклеить») содержимое буфера в область, заданную объектами Range и Selection. Эта операция опасна, так как происходит замена, а не добавление текста. Поэтому обычно метод Paste применяется к объектам Range и Selection, предварительно стянутым в точку вставки. Метод PasteFormat применяет форматирование, хранящееся в буфере, к объекту Selection.

Наиболее важной особенностью работы на VBA в Word является вставка текста в документ при работе с приложениями. Для этого служат объекты Range и Selection, которые являются главными для практически любых операций, которые можно выполнять с помощью Word VBA. Некоторые из этих действий можно применять к документам в целом, но в общем случае вам необходим диапазон или выделенная область, прежде чем вносить изменения. Мы, однако, рассмотрим действия с документом при его создании.

Открытый документ Word уже содержит объекты Range, соответствующие многим его элементам. Каждый абзац, таблица, ячейка таблицы, комментарий и т. д. определяют диапазоны. Например, для того чтобы вставить некоторый текст в уже существующий документ, необходимо прописать код:

ActiveDocument.Paragraphs(1).Range.Text = «Здравствуй, мир!»

Причем данная строка будет расположена в конце существующего параграфа. С другой стороны, используя объект Selection, можно также вставить некоторый текст в документ, используя метод Add и присвоение свойства Text объекту Selection:

If Documents.Count = 0 Then Documents.Add

Selection.Text = "Изучение работы с текстом в документе Word является важной составной частью умения программировать в VBA, « + TextBox1.Text +», и отвечает запросам всех программистов!»

Примечание:

определение цвета

Selection.Font.Color =

wdColorRed – красный

wdColorDarkRed – бордовый

wdColorDarkTeal – бирюзовый

wdColorBlue – синий

wdColorGreen – зеленый
wdColorBlack – черный
wdColorOrange – оранжевый
определение жирности
Selection.Font.Bold =wdToggle – жирность
определение начертания
Selection.Font.Italic=wdToggle – курсив
определение выравнивания
Selection.ParagraphFormat.Alignment=
wdAlignParagraphRight – выравнивание по правому краю
wdAlignParagraphCenter – выравнивание по центру
wdAlignParagraphJustify – выравнивание по левому краю
вставка в текст конкретного предложения
Selection.TypeText Text:="Пример работы с текстом"
вставка новой пустой строки
Selection.TypeParagraph
установка размера букв
Selection.Font.Size = 14

Задание к практической работе №10

1. Написать программу, которая позволяет найти количество слов "WORD" в текстовом документе. Изменить начертание, жирность и размер шрифта всех найденных слов на заданное пользователем.
2. Написать программу, которая позволяет вывести любой текст в текстовый документ в режиме "печатной машинки".
3. На лист табличного процессора поместить список студентов (ФИО, адрес, дата рождения). Всем военнообязанным студентам автоматически сформировать письма, написав соответствующую программу на VBA (см. рис. 20). Дата явки определяется автоматически по формуле: Текущая дата+15дней.

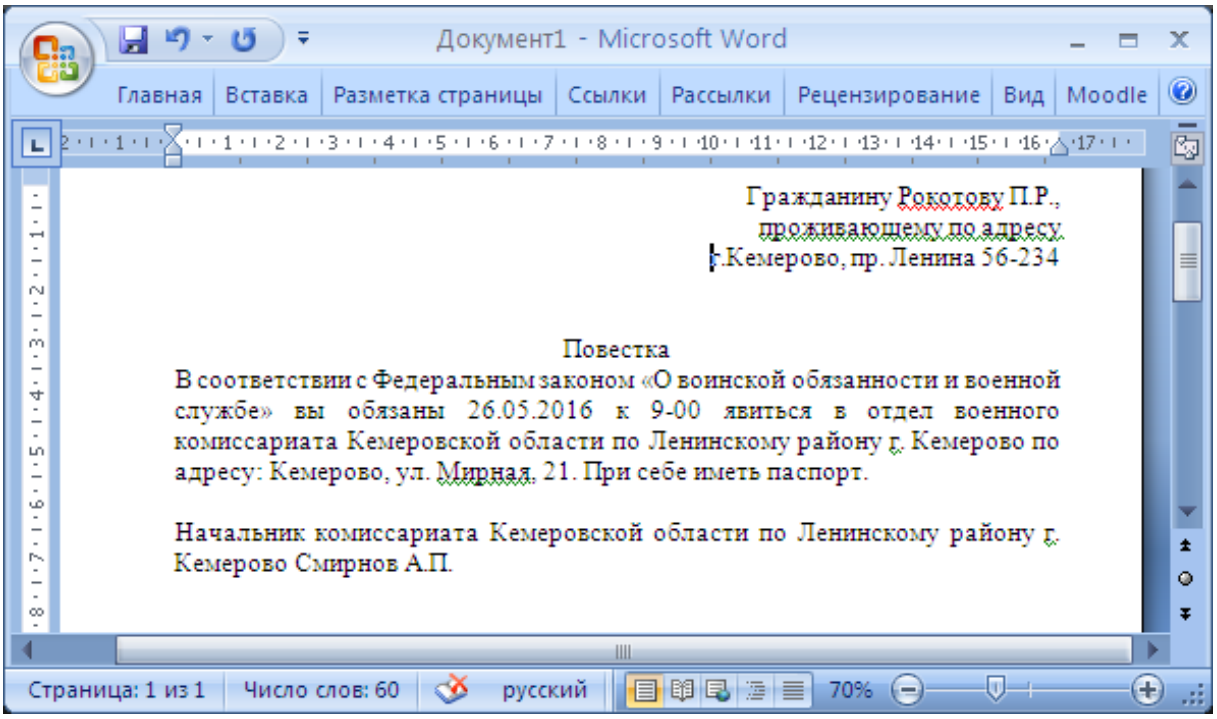


Рис.20 – Лист Excel

Практическая работа №11
Организация доступа к объектам базы данных

Цель работы: Изучение способов обращения к основным объектам баз данных.

Теоретические положения

Программирование с использованием интерфейса DAO

Объекты доступа к данным (DAO, Data Access Object) – это иерархия объектов, обеспечивающая доступ к структуре базы данных и ее содержимому. В программах Visual Basic пользователь имеет возможность использовать объектный интерфейс DAO для выполнения следующих задач:

- осуществления доступа к данным в локальных и удаленных базах данных Access и внешних источниках;
- управления базой данных и ее объектами;
- изменения структуры таблиц и схемы данных;
- управления системой защиты;
- создания, настройки и синхронизации реплик.
- Обе иерархии объектов DAO: для рабочей области Jet и для рабочей области **ODBCDirect** – начинаются с объекта DBEngine. Этот объект содержит свойства и методы, позволяющие управлять рабочими областями. Свойство **DefaultType** объекта DBEngine позволяет определять или устанавливать тип рабочей области, создаваемой по умолчанию.
- Объект DBEngine в качестве свойства содержит семейство Workspaces всех открытых рабочих областей. Можно выбрать элемент этого семейства, указав индекс или имя, чтобы получить доступ к конкретной рабочей области.
- Чтобы создать новую рабочую область ядра Jet или рабочую область ODBCDirect, используют метод CreateWorkspace объекта DBEngine с соответствующим параметром.

Несколько рабочих областей разных типов могут быть открыты одновременно. В момент создания рабочей области начинается сеанс работы с ней. После выполнения всех необходимых действий в рабочей области нужно завершить сеанс, закрыв рабочую область с помощью метода Close объекта Workspace.

Чтобы открыть базу данных, используйте существующий объект Database или создайте новый. Объект Database представляет собой базу

данных Jet (файл MDB), базу данных ISAM или источник данных ODBC, подключенный через Jet.

Доступ к текущей базе данных осуществляется с помощью объекта типа **Database**, возвращаемого методом **CurrentDb** объекта Application (который представляет приложение Access). Метод **CurrentDb** входит в набор глобальных методов, поэтому для его вызова можно использовать сокращенную ссылку без префикса Application с точкой.

Открыть существующую базу данных можно двумя способами:

- с помощью метода **OpenDatabase** объекта Workspace. В этом случае база данных будет открыта в заданной рабочей области;
- с помощью метода **OpenDatabase** объекта DBEngine. В этом случае база данных будет открыта в рабочей области, используемой по умолчанию.

Метод **OpenDatabase** объекта DBEngine входит в набор глобальных методов, поэтому при использовании сокращенной ссылки на этот метод без явного указания объекта (**DBEngine** или **Workspace**) используется метод объекта DBEngine. Метод **OpenDatabase** возвращает ссылку на созданный объект Database и имеет следующие параметры: Database *OpenDatabase(<имя>, <параметры>, <режим>, <соединение>)*

Метод **CreateDatabase** создает новый объект Database, добавляет его в семейство Databases открытых баз данных в рабочей области, сохраняет базу данных на диске и возвращает открытый объект Database. Этот метод используется только в рабочей области ядра Microsoft Jet. Метод **CreateDatabase** имеет следующие параметры:

Database CreateDatabase (<имя>, <порядок>, <параметры>)

Чтобы получить доступ к данным в открытой одним из перечисленных способов базе данных, необходимо открыть набор записей. Набор записей может представлять собой все записи таблицы или часть записей таблицы, удовлетворяющих указанному условию, или результат выборки из нескольких таблиц. Чтобы открыть набор записей в базе данных, используйте метод **OpenRecordset** объекта Database. Этот метод возвращает ссылку на созданный объект Recordset и имеет следующие параметры (табл. 11):

Recordset OpenRecordset(<источник>, <тип>, <параметры>, <блокировки>)

Таблица 11 – Параметры метода OpenRecordset

Параметр	Тип	Обязательный или нет	Описание
<источник>	String	Обязательный	Источник записей для нового объекта Recordset. В качестве источника записей можно указать имя таблицы или запроса, а также инструкцию SQL, которая возвращает записи
<тип>	<константа>	Необязательный	Константа, указывающая тип открываемого объекта Recordset.
<параметры>	<константы>	Необязательный	Произвольная комбинация определенных констант, задающих характеристики нового объекта Recordset. Эти константы приведены в справочной системе Access
<блокировки>	<константа>	Необязательный	Константа, определяющая тип блокировки объекта Recordset. Возможные константы перечислены в справке Access

помощью объектов DAO можно осуществлять всевозможные манипуляции с данными. Например, такие как:

- получение набора записей из таблицы;
- выполнение запроса SQL для получения набора записей;
- использование запроса, сохраненного в базе данных в виде объекта;
- использование набора методов объекта Recordset, предоставляющего широкие возможности по обработке данных: чтение, анализ и изменение данных без составления инструкций на языке SQL.

В следующих примерах приведены приемы программного изменения данных в открытом наборе записей. Переменная rs соответствует открытому набору записей – объекту типа Recordset.

Пример. Добавление записи в таблицу

'Добавляем сообщение в таблицу сообщений клиента
rs.AddNew 'Создание новой записи
rs!ИмяИгрока = playerName 'Запись значения в поле ИмяИгрока
rs!Сообщение = message 'Запись значения в поле Сообщение

rs.Update 'Сохранение изменений в источнике
rs.Bookmark = rs.LastModified
'Перемещение курсора на новую запись

Пример. Изменение текущей записи в таблице

'Увеличиваем счет игрока, сделавшего ход
'Делаем текущей запись, содержащую данные для нужного игрока
rs.FindFirst "[ИмяИгрока] = '" & newPlayer & "'"
rs.Edit 'Переводим запись в режим правки
rs!Счет = rs!Счет + newCount 'Изменяем значение поля Счет
rs.Update 'Сохраняем изменения
rs.Bookmark = rs.LastModified 'Делаем измененную запись текущей

Пример. Удаление текущей записи в таблице

playerQueryCode = rs!КодЗаявки 'Сохраняем параметры заявки
playerTrial = rs!Значение 'во временных переменных
rs.Delete 'и удаляем заявку из таблицы

Для перемещения по записям используются методы ***Move***, ***MoveFirst***, ***MoveNext***, ***MovePrev***, ***MoveLast*** объекта Recordset. Метод MoveLast перемещает курсор на последнюю запись в наборе, что приводит к загрузке в набор всех записей. После этого можно прочитать значение свойства **Count** объекта Recordset. Оно будет соответствовать общему количеству записей в наборе.

Задание к практической работе №11

Написать программу на языке VBA. Вся информация по сотрудникам находится в базе данных.

1. Вывести на Лист1 табличного процессора телефоны всех сотрудников.
2. Вывести на Лист2 табличного процессора всех сотрудников отдела, название которого введено с клавиатуры.
3. Найти среднюю зарплату сотрудников экономического отдела. Результат вывести на экран в диалоговое окно.
4. Найти номера телефонов и их количество у всех сотрудников банка. Результат вывести в текстовый файл.
5. Вывести количество сотрудников экономического отдела, имеющие должность «инженер». Результат вывести на экран в диалоговое окно.
6. Вывести количество оценок 3, 4, 5 по ТБ у сотрудников финансового отдела. Результат вывести на экран в диалоговое окно.

Практическая работа №12
Разработка приложения для решения прикладных задач

Цель работы: Изучение основ разработки прикладных программ.

Теоретические положения

Разработка прикладного программного обеспечения включает в себя:

- математическую постановку задачи, описание ее в содержательных терминах;
- выбор численного метода и разработка алгоритма решения;
- составление блок-схемы алгоритма;
- составление, ввод и отладка программы;
- ввод исходных данных и решение задачи;
- анализ полученных результатов и коррекция структуры алгоритма и программы при необходимости.

Рассмотрим следующую прикладную задачу: моделирование движения транспортных средств по участку улично-дорожной сети (УДС).

Математическая постановка задачи предусматривает формирование условий, ограничений и зависимостей, определяющих ее математическое описание. Математизация задачи сводится обычно к записи задачи в виде формул, определению величин начальных данных и последовательности использования формул.

При моделировании движения транспортного потока необходимо учитывать, что на движение автомобилей влияет множество случайных факторов, изменяющихся как в пространстве, так и во времени. Под воздействием этих факторов случайно меняются характеристики движения транспортных потоков: интенсивность, состав и скорости движения, ускорения, траектории и т.д. При статическом моделировании на компьютере потока автомобилей большое значение имеет моделирование прибытия транспортного средства в рассматриваемое сечение дороги, а также моделирование движения одиночных автомобилей для расчета скоростей движения в свободных условиях.

Для моделирования прибытия автомобилей необходимо использовать законы распределения интервалов во времени между движущимися в потоке друг за другом автомобилями. Для моделирования могут быть использованы как теоретические законы распределения (Пуассона, показательное), так и фактические статистические данные, полученные в резуль-

тате натурных экспериментов.

Распределение Пуассона наиболее широко применяется в теории вероятностей и, особенно, в теории массового обслуживания. Вероятность появления n событий, поступающих с интенсивностью λ в интервале времени $(0, t)$ определяется из уравнения

$$P_n(t) = e^{-\lambda t} \cdot \frac{(\lambda t)^n}{n!}. \quad (1)$$

Величины интервалов во времени по данному закону распределения определяются с помощью статистического моделирования на основе использования случайных чисел:

$$x = \frac{\ln(1-y)}{\lambda}, \quad (2)$$

где y – случайная величина, распределенная по закону равномерной плотности на интервале [0,1]; λ – средняя интенсивность транспортного потока, авт/ч; x – величина, распределенная по закону Пуассона.

Для моделирования случайной величины, распределенной по экспоненциальному (показательному) закону используют уравнение

$$\mathrm{P(t)} = \lambda \cdot e^{-\lambda t}, \quad (3)$$

где λ – средняя интенсивность транспортного потока, авт/ч; t – интервал времени между моментами прибытия автомобилей, ч.

Величина интервала времени между следующими друг за другом автомобилями определяется из соотношения

$$x = \frac{-\ln(y)}{\lambda}, \quad (4)$$

где x – величина, распределенная по экспоненциальному закону; y – случайная величина, распределенная по закону равномерной плотности на интервале [0,1].

Экспоненциальное и распределение Пуассона представляет собой удовлетворительную модель для описания интервалов времени между последовательными автомобилями в свободном потоке небольшой интенсивности. При большой интенсивности движения значения, предсказываемые с помощью этих распределений, отличаются от реально наблюдаемых. Поэтому для моделирования движения автомобиля в плотных потоках используют смещенное экспоненциальное распределение или двухкомпонентное распределение Шула.

Выбор типа автомобиля осуществляется по закону равномерной

плотности в соответствии с заданным составом транспортного потока. В языке программирования Visual Basic такое распределение случайной величины можно получить с помощью оператора RND.

Экспериментальными исследованиями установлено, что изменение скоростей движения автомобилей в транспортном потоке может быть описано нормальным законом распределения. Моделирование одномерной случайной величины, распределенной по нормальному закону, может производиться на основе следующего соотношения:

$$Xn = \sqrt{\frac{12}{n}} \sum_{i=1}^{n} (y_i - 0{,}5), \qquad (5)$$

где y_i – величина, распределенная по закону равномерной плотности; X_n – величина, имеющая приближенное нормальное распределение с нулевым средним и единичной дисперсией.

Можно использовать алгоритм программы, изображенный на рис.21, для генерирования нормально распределенных чисел на основе соотношения (7), учитывая, что *OTKL* задает среднее квадратичное отклонение, а *Mx* – математическое ожидание.

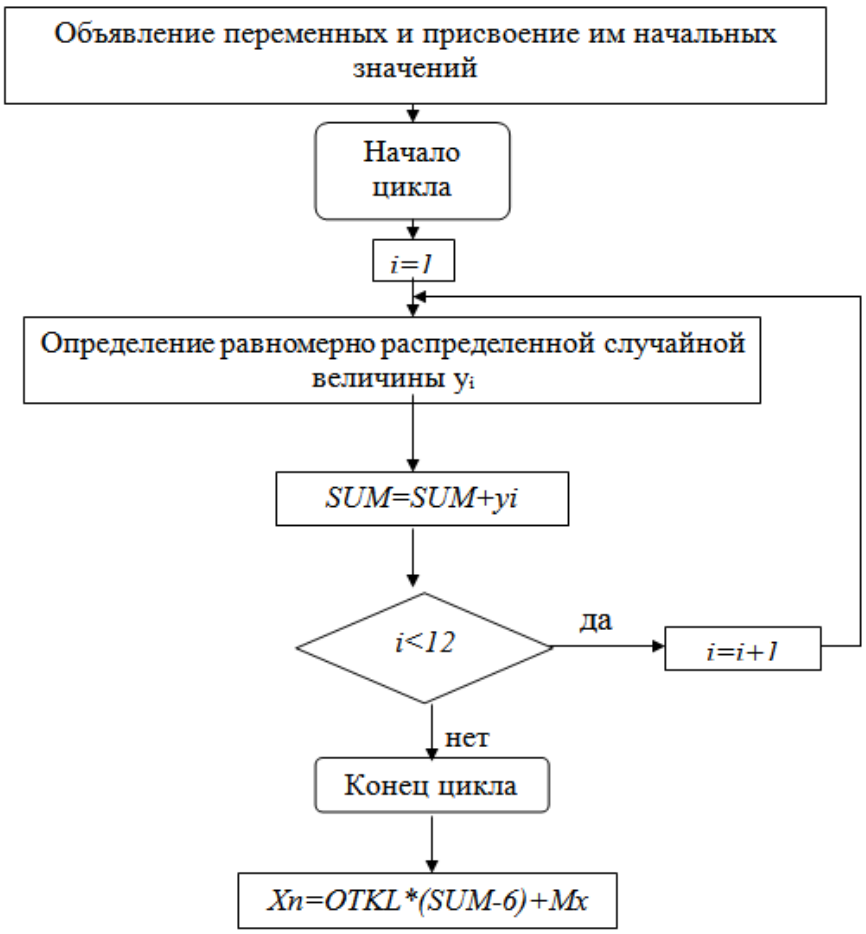


Рис. 21 – Алгоритм подпрограммы

Таким образом, определяется скорость движения автомобиля в момент появления на участке улично-дорожной сети для заданных значений математического ожидания и среднего квадратичного отклонения. Следует учитывать, что величина среднего квадратичного отклонения оказывает влияние на стабильность режима движения: чем меньше среднее квадратичное отклонение, тем меньше неравномерность движения.

При моделировании движения автомобиля следует ориентироваться на следующее соотношение между средним значением пространственной скорости и средним квадратичным отклонением:

$$\sigma_v/V = 0{,}2 \div 0{,}25.$$

При построении алгоритма программы необходимо учитывать, что величина X_n, смоделированная на основе соотношения (7), не должна превышать максимальное значение скорости данного автомобиля, а также максимально допустимого значения скорости на участке моделирования. Кроме того, необходимо учитывать, что максимально возможная скорость движения на участке моделирования ограничивается некоторым значением, зависящим от уровня удобства движения. Характеристика уровней удобства движения приведена в таблице.

Таблица 12 – Характеристика уровней удобства движения

Уровень удобства движения	**Скорость движения**	**Интенсивность**
А	$V \geq 0{,}9V_{своб.дв}$	$q \leq 0{,}2q_{max}$
Б	$V = (0{,}9 \div 0{,}7)V_{своб.дв}$	$q = (0{,}2 \div 0{,}45)q_{max}$
В	$V = (0{,}7 \div 0{,}55)V_{своб.дв}$	$q = (0{,}45 \div 0{,}7)q_{max}$
Г	$V < 0{,}55V_{своб.дв}$	$q > 0{,}7q_{max}$

Скорости движения связанных транспортных средств можно определить исходя из ситуационной модели. Алгоритм основан на отнесении текущей ситуации к одной из пяти взаимоисключающих групп, каждой из которых соответствует вполне определенная реакция ведомого автомобиля.

а) Относительная скорость между ведомым автомобилем и лидером равна нулю, и сохраняется желаемая дистанция. В этой ситуации никаких управляющих воздействий не предпринимается.

б) Дистанция между автомобилями меньше желаемой и продолжает уменьшаться. В этой ситуации производится торможение ведомого автомобиля до тех пор, пока относительная скорость между автомобилями

не станет равной нулю. Расчет замедления производится по формуле

$$a = -\frac{(V_{n-1} - V_n)^2}{2(H - \gamma \cdot \delta)}, \quad (6)$$

где V_{n-1} – скорость лидирующего автомобиля, м/с; V_n – скорость автомобиля следующего за лидером, м/с; H – дистанция между автомобилями (расстояние, измеренное между передними бамперами лидирующего и ведомого автомобилей), м; γ – коэффициент пропорциональности (γ = 1,2); δ – желаемая дистанция между автомобилями.

При длине автомобиля L минимальная желаемая дистанция $\delta_{min} = L$.

в) Дистанция между автомобилями меньше желаемой, но постоянно увеличивается. Корректировка скорости ведомого автомобиля не происходит до тех пор, пока дистанция меньше желаемой. Когда дистанция превысит желаемую, происходит ускорение ведомого автомобиля до скорости лидера:

$$a = -\frac{(V_{n-1} - V_n)^2}{2(H - \delta)}. \quad (7)$$

Значение ускорения не может быть больше максимального для данного транспортного средства.

г) Дистанция между транспортными средствами больше желаемой и постоянно увеличивается. В этой ситуации ведомый автомобиль осуществляет разгон с ускорением

$$a = \frac{(V_{n-1} - V_n)^2}{2H(\xi - 1)}, \quad (8)$$

где ξ – коэффициент пропорциональности. Рекомендуется принимать ξ в диапазоне 1,1÷1,2.

д) Дистанция больше желаемой, но скорость ведомого автомобиля больше скорости лидера. В этом случае ведомый автомобиль приближается к лидеру на расстояние δ, после чего относительная скорость принимает значение ноль. Замедление рассчитывается по формуле

$$a = -\frac{(V_{n-1} - V_n)^2}{2(H - \delta)}. \quad (9)$$

Для определения желаемой дистанции можно использовать следующее уравнение

$$\delta = L + \alpha \cdot V_{n-1}, \quad (10)$$

где L – длина лидирующего автомобиля, м; V_{n-1} – скорость лидера,

м/с; α – положительная константа, характеризующая водителя и автомобиль (α = 1,12÷1,22).

После постановки задачи осуществляется выбор численного метода, который в большой степени определяет алгоритм решения задачи, точность результатов, а также время на подготовку программы. Так как поставленная задача является прикладной, студент обязан самостоятельно разработать метод ее решения, а, следовательно, и алгоритм. Разработка алгоритма программы осуществляется с учетом следующего ограничения: автомобили движутся по одной полосе, движение происходит без обгонов.

Примерная блок-схема алгоритма моделирования движения транспортного потока на участке моделирования приведена на рис. 22. Студенту необходимо самостоятельно детализировать данный алгоритм.

После составления алгоритма программы необходимо написать ее на заданном алгоритмическом языке программирования. При написании программы необходимо учитывать, что при работе с большим количеством данных рациональнее поместить их в отдельный файл (базу данных).

Ввод и отладка программы осуществляется непосредственно на компьютере. Отладка программы заключается обычно в исправлении синтаксических ошибок.

После ввода исходных данных и решения задачи проверяется наличие в программе логических ошибок, приводящих к некорректным вычислениям. Такие ошибки устраняются посредством тестирования программы, то есть выполнения ее с тщательно проверенными исходными данными. При обнаружении смысловых (логических) ошибок необходимо их исправить.

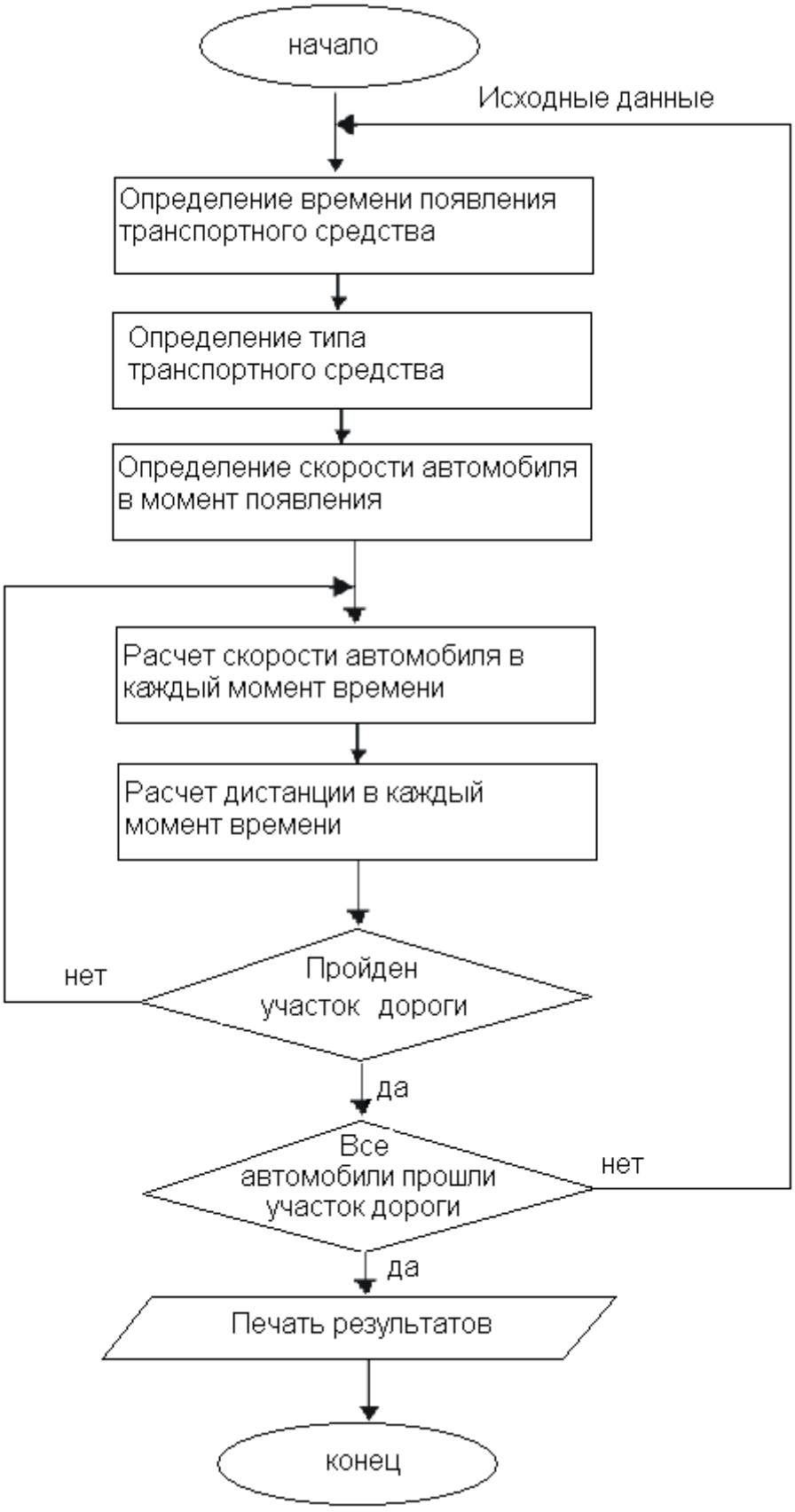


Рис. 22 – Блок-схема алгоритма программы

Задание к практической работе №12

1. Разработать программное обеспечение для моделирования движения транспортных средств по участку улично-дорожной сети.
2. Разработать программное обеспечение для автоматизации выбора подвижного состава для осуществления перевозок грузов между грузоотправителем (заказчиком автотранспортных услуг) и грузополучателем (потребителем автотранспортных услуг). Учесть, что при доставке груза от грузоотправителя к грузополучателю возможно возникновение следующих ситуаций:

А) Весь объем груза Q может перевезти одно транспортное средство за одну ездку. В этом случае выбирается автомобиль с номинальной грузоподъемностью $q_н \geq Q$.

Б) Весь объем груза Q может перевести одно транспортное средство за n ездок в течение времени в наряде $T_н$.

В) Весь объем груза Q могут перевезти несколько автомобилей в течение времени в наряде $T_н$.

Г) Все транспортные средства АТП не могут перевезти заданный объем груза за время в наряде.

Список литературы

1. Семакин, И. Г. Основы алгоритмизации и программирования [Текст] : учебник для образовательных организаций, реализующих программы среднего профессионального образования по специальностям "Информационные системы и программирование", "Сетевое и системное администрирование", "Обеспечение информационной безопасности автоматизированных систем", "Обеспечение информационной : [для студентов СПО] / И. Г. Семакин, А. П. Шестаков. – Москва : Академия, 2017. – 304 с. – Доступна электронная версия: http://www.academia-moscow.ru/catalogue/4831/345450/
2. Колдаев, В. Д. Основы алгоритмизации и программирования. – Москва : НИЦ ИНФРА-М, 2015. – 416 с. – Режим доступа: http://znanium.com/go.php?id=484837. – Загл. с экрана. (14.12.2018)
3. Колдаев, В. Д. Основы алгоритмизации и программирования. – Москва : НИЦ ИНФРА-М, 2019. – 414 с. – Режим доступа: http://znanium.com/go.php?id=980416. – Загл. с экрана. (14.12.2018)
4. Цветкова, М. С. Информатика [Электронный ресурс] : учебник для использования в учебном процессе образовательных учреждений СПО на базе основного общего образования с получением среднего общего образования / М. С. Цветкова, И. Ю. Хлобыстова. – Москва : Академия, 2017. – 352 с. – Режим доступа: http://academia-moscow.ru/reader/?id=227485#copy. – Загл. с экрана. (14.12.2018)
5. Гаврилов, М. , В. Информатика и информационные технологии 4-е изд., пер. и доп.[электронный ресурс]. – Москва : Юрайт, 2018. – 383 с. – Режим доступа: https://biblio-online.ru/book/informatika-i-informacionnye-tehnologii-413451. – Загл. с экрана. (14.12.2018)

Содержание

Практическая работа №1 3
Практическая работа №2 7
Практическая работа №3 9
Практическая работа №4 16
Практическая работа №5 21
Практическая работа №6 27
Практическая работа №7 31
Практическая работа №8 37
Практическая работа №9 41
Практическая работа №10 47
Практическая работа №11 56
Практическая работа №12 60
Список литературы 68