МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
“ХАРЬКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ”

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам
«Основы программирования в среде Visual Basic»
из раздела «Программирование в среде Visual Basic. Интегрированная среда разработки»
дисциплины «Информатика»
для студентов направления подготовки
6.050801 «Микро- и наноэлектроника»

Харьков НТУ «ХПИ» 2013

Методические указания к лабораторным работам «Основы программирования на Visual Basic» из раздела «Программирование в среде Visual Basic. Интегрированная среда разработки» дисциплины «Информатика» для студентов направления подготовки 6.050801 «Микро- и наноэлектроника» / Состав.: В.И. Шкалето, Р.В. Зайцев. – Харьков: НТУ «ХПИ», 2013. – 60 с.

Составители: В.И. Шкалето,
Р.В. Зайцев

Рецензент проф. О.О. Булгаков

Кафедра физического материаловедения для электроники и гелиоэнергетики

ВВЕДЕНИЕ

Методические указания к лабораторным раотам по разделу «Программирование в среде Visual Basic. Интегрированная среда разработки» дисциплины «Информатика» касаются трех лабораторних работ: «Ознакомление с рабочей средой Visual Basic. Первый проект», «Синтаксис программного кода Visual Basic. Область определения и время жизни переменных» и «Выражения и встроенные функции Visual Basic».

Так что же такое Visual Basic? Слово "Visual" относится к методу, используемому для создания того, что видит пользователь — графического пользовательского интерфейса, или GUI (graphical user interface). Слово "Basic" относится к аббревиатуре BASIC (Beginners All-Purpose Symbolic Instruction Code — многоцелевой код символьных инструкций для начинающих) языка программирования, который используется программистами намного чаще, чем любой другой язык в истории вычислений. Для создания различных полезных программ достаточно изучить лишь некоторые из его возможностей. Приведенные ниже лабораторные работы содержат материалы для начинающих программистов в среде Visual Basic 6.0 (VB6.0), в каждой из которых изучаются определенные вопросы технологии работы со средой программирования VB6.0 и решаются различные варианты учебной задачи.

В результате изучения теоретического материала и выполнения лабораторных работ студент должен изучить:

- этапы разработки программ;
- основы алгоритмического языка VB6.0;
- назначение, функции и структуру среды программирования VB6.0;

а также уметь:

- проводить физический и математический анализ простейших прикладных задач и разрабатывать алгоритмы их решения;
- вести отладку программ в среде программирования VB6.0.

Указанное позволит преорести первоначальные навыки работы в среде программирования VB6.0

ЛАБОРАТОРНАЯ РАБОТА 1

СИНТАКСИС ПРОГРАММНОГО КОДА VISUAL BASIC. ОБЛАСТЬ ОПРЕДЕЛЕНИЯ И ВРЕМЯ ЖИЗНИ ПЕРЕМЕННЫХ

Цель работы – Научиться писать программный код на языке Visual Basic 6. Научиться объявлять различные типы данных. Научиться определять область определения и время жизни переменных. Научиться объявлять массивы различных типов данных. Научиться работать с массивами. Научиться объявлять данные и структуры данных, определяемые пользователем.

1.1 Общие сведения

1.1.1 Синтаксис программного кода

Чтобы Visual Basic понимал исходный код, следует придерживаться определенных правил написания программ. В каждой строке кода помещается оператор, который может иметь дополнительные параметры:

```
[Ключевое_Слово] [Параметры]
```

Как использовать отдельные операторы и как задавать параметры, зависит от команды. Например, простейший оператор присваивания:

```
[Let] A = 1
```

Эта строка также содержит оператор — знак равенства “=”. Параметры указываются перед (A) и после (1) оператора. Для оператора присваивания ключевое слово Let – необязательно.

Можно разделять логическую строку, а значит и оператор, на несколько физических строк. Разделителем строк служит пробел, следующий перед символом подчеркивания (_).Это дает возможность форматировать длинные, трудно обозримые строки так, чтобы они полностью помещались на странице экрана.

Строка программы в Visual Basic может содержать максимум 1023 символа и не более десяти разделителей — этого обычно достаточно. В одной строке можно также объединять несколько операторов, которые разделяются двоеточием.

В Visual Basic, как и в большинстве языков программирования, могут быть использованы комментарии. Комментарии предназначены для пояснения отдельных фрагментов программы и игнорируются Visual Basic при выполнении программы. Для выделения начала комментария можно использовать или верхнюю запятую ('), или команду Rem — их действие одинаково. Rem представляет собой оператор и поэтому должен находиться в отдельной строке. Верхняя запятая может ставиться в любом месте строки, при этом текст комментария располагают справа.

Число строк кода (формы, модуля и т.п.) ограничивается 65534. Это ограничение не существенно, поскольку число строк в большинстве программ меньше.

1.1.2 Типы данных

1.1.2.1 Переменные

Переменные - представляют собой нечто вроде небольшого контейнера с определенным содержимым, например символами или числами. Этому контейнеру присваивается имя, т.е. имя переменной. Чтобы сослаться на содержимое, достаточно указать имя переменной.

В зависимости от содержимого различают переменные разных типов. Visual Basic поддерживает следующие типы переменных:

Данные типа Boolean могут содержать только значения True или False. Значению True соответствует 1,а False — 0.Если переменной этого типа присваивается значение 0, то переменная содержит False. Все другие значения подразумевают True.

Данные типа Byte, Integer, Long содержат лишь целые цифровые значения из различных диапазонов. Если переменной такого типа присваивается 0.4, то возвращается 1, если 1.5 — возвращается 2.

Данные типа Single и Double содержат числа с плавающей запятой из разных диапазонов значений.

Данные типа Currency также служат для представления чисел с плавающей запятой, но число разрядов после запятой ограничено четырьмя. Этого достаточно при выполнении денежных расчетов.

В Visual Basic в качестве разделителя целой и дробной частей используется точка.

Данные типа Date специально предназначены для обработки информации о дате и времени. Чтобы было понятно, что под указанным значением подразумевается дата и/или время, нужно поместить его между двумя знаками #. При вводе следует пользоваться американским форматом.

Если же при вводе данных этого типа использовать кавычки ("), что допустимо, то следует применять установленный в системе формат даты и времени.

Данные типа String служат для хранения строк (String). Каждый символ, сохраненный в переменной типа String, занимает 1 байт памяти. Поэтому операционные системы разных платформ поддерживают различную максимальную длину строки. Для того чтобы Visual Basic отличал строку от имени переменной, строка заключается в парные кавычки.

Данные типа Object служат для хранения других объектов и будут подробнее позже.

Тип данных Variant — это хамелеон. Он устанавливает тип данных в зависимости от содержимого. Если в такой переменной содержится число, то переменная типа Variant принимает соответствующий тип данных.

Если ее содержимое — число 5, то она принимает тип Integer; если 1.2 - Double; если текст, то String. Переменная типа Variant изменяет свой тип во время выполнения программы.

Возможные значения переменных для разных типов данных приведены в таблице 1.1.

Таблица 1.1 - Типы данных

№ п.п.	Тип данных	Занимаемая память, байт	Диапазон значений
1	Boolean	1	True (не 0) или False (0)
2	Byte	1	0 – 255
3	Integer	2	от –32768 до 32767
4	Long	4	от –2147483648 до 2147483647
5	Single	4	от –3.402823E+38 до –1.401298E-45 для отрицательных значений, от 1.401298E-45 до 3.402823E+38 для положительных значений, и 0.
6	Double	8	от –1.79769313486232E+308 до –4.94065645841247E-324 для отрицательных значений, от 4.94065645841247E-324 до 1.79769313486232E+308 для положительных значений, и 0.
7	Currency	8	от –922337203685477.5808 до 922337203685477.5807 (с плавающей точкой и 4 знаками после запятой)
8	Date	8	Значения даты и времени
9	String	<= 65535	1 байт на символ ASCII

1.1.2.2 Явное объявление

Во многих языках программирования все используемые переменные должны быть объявлены. Этой процедурой системе программирования сообщается имя и тип переменной. После объявления переменной система знает, каково ее содержимое, и, что особенно важно, сколько памяти нужно зарезервировать для нее.

При написании программы в Visual Basic пользователь решает сам, нужно объявлять переменную или нет. Для явного объявления переменной используют оператор Dim, который имеет следующий синтаксис:

```
Dim Имя_переменной [As Тип_данных]
```

Имя переменной можно выбирать произвольно, но при этом следует соблюдать следующие правила:

• имя переменной должно начинаться с буквы;

• максимальная длина имени — 255 символов;

• имена могут содержать буквы, цифры и символ подчеркивания (_). Все другие символы не допускаются;

• имя не может быть зарезервированным в Visual Basic словом (например, Print).

Длина переменной типа String обычно ограничивается лишь операционной системой. Но при необходимости ее можно указать явно. Для этого после слова String добавляют звездочку и максимальное число символов:

```
Dim Имя_переменной [As String] [*Число знаков]
```

Указывать тип данных при объявлении не обязательно. Тип данных при объявлении может устанавливаться просто добавлением знака типа к имени переменной (таблица 1.2).

Таблица 1.2 - Знаки типов переменных

Тип переменной	Знак	Пример
Integer	%	Counter%
Long	&	Nr&
Single	!	Result!
Double	#	Number#
Currency	@	Surnma@
String	$	FirstName$

1.1.2.3 Неявное объявление

Visual Basic, в отличие от других языков программирования, не требует явного объявления переменных. Оператор Dim явно задает имя и тип переменной. Но переменная может объявляться автоматически, когда она появляется в коде. Это так называемое неявное объявление переменной.

Исходя из этого, следующие коды эквивалентны:

```
Dim Price As Currency
Price=523
```

или

```
Price@=523
```

В качестве оператора объединения строк в Visual Basic можно использовать как знак суммирования (+), так и знак "коммерческое и" (&). Однако для лучшей читаемости кода следует применять только &, так как знак плюса используется обычно при суммировании числовых значений. Но не следует забывать и о том, что знак & функционирует и как идентификатор типа для переменных Long, если он находится в конце имени.

Visual Basic всегда по умолчанию применяет тип Variant. Чтобы переменные всегда объявлялись явно, используйте опцию Explicit. В этом случае Visual Basic будет требовать явного объявления переменных, что устраняет возможные ошибки при написании программы.

1.1.3 Область определения

Весьма важной характеристикой переменных является область их определения. В Visual Basic есть три вида областей определения, характеризующих доступность переменной:

•локальная: переменная доступна только в текущей процедуре;

•контейнера: переменная доступна только в текущей форме, модуле или классе;

•глобальная: переменная доступна во всем проекте.

Локальными являются переменные, определяемые внутри процедуры или функции. Они доступны только внутри этой процедуры.

Переменные контейнера определяются в секции (General) (Declarations) и доступны только внутри соот-ветствующего контейнера, т.е. формы, модуля или класса. При этом вместо оператора Dim используется заре-зервированное слово Private.

Глобальные переменные определяются в секции (General) (Declarations) модуля. При этом вместо оператора Dim используется зарезервированное слово Public. Глобальные переменные доступны во всех модулях и процедурах проекта.

1.1.4 Время жизни переменных

Локально объявленные переменные при выходе из процедуры удаляются из памяти, а при новом вызове процедуры инициализируются заново. Их содержимое при этом не сохраняется, что не всегда желательно. Этого можно было бы избежать путем расширения области определения, т.е. объявив переменную глобальной или, как минимум, переменной контейнера. Но это разрешает доступ к переменной из других процедур.

Visual Basic дает возможность объявлять статические переменные. При выходе из процедуры содержимое статической переменной сохраняется. При новом вызове этой процедуры переменной присваивается значение, которое она имела при последнем выходе из этой процедуры. Содержимое переменной сохраняется в течение всего времени, пока существует в памяти форма или модуль.

Для объявления переменной как статической нужно просто вместо оператора Dim использовать слово Static:

```
Static Имя_переменной [As Тип_переменной]
```

1.1.5 Объявление массивов

Массив - это упорядоченный набор элементов определенного типа, каждый из которых имеет свой порядковый номер, называемый индексом. Различают статические и динамические массивы.

Границы статического массива устанавливаются на этапе разработки и могут изменяться только в новой версии программы.

Динамические массивы изменяют свои границы в ходе выполнения программы. С их помощью можно динамически задавать размер массива в соответствии с конкретными условиями. Однако следует учесть, что работа с динамическими массивами требует дополнительных затрат на программирование.

Для объявления массива используется оператор Dim с указанием в круглых скобках после имени масси-ва его максимального индекса:

```
Dim aName (150) As String
```

В этом случае элементы переменной aName различают не по имени, а по индексу.

То, что уже говорилось о времени жизни и области определения переменных, относится большей частью и к массивам. Однако статические массивы нельзя определить локально внутри процедуры, а только глобально или для контейнера:

```
[Static | Public | Dim ] Имя_переменной (Верхняя_граница ) As Тип_данных
```

При использовании массивов не следует забывать, что в Visual Basic индексирование всегда начинается с нуля, т.е. индекс 0 обозначает первый элемент массива, индекс 1 — второй и т.д.

Оператор Option Base позволяет задать индексацию массива с 1:

```
Option Base 1
```

Этот оператор должен находиться в секции (General)(Declarations) контейнера (формы, модуля, класса).

Допустимыми значениями для Option Base являются только 0 и 1.

Для установки других границ массива необходимо использовать следующий синтаксис:

```
[Static | Public | Dim ] Имя_переменной ([Нижн_предел To ] Верхн_предел) As Тип_данных
```

Указанием верхней и нижней границ можно задать любые диапазоны индекса.

Visual Basic позволяет также создавать многомерные массивы. При объявлении многомерного массива верхние границы каждой размерности разделяются запятыми:

```
Dim aName (10,25) As String
```

В Visual Basic значения размерности массивов могут достигать 60. Для отдельных значений размеров могут указываться диапазоны индексов:

```
Dim aArray (10,80 To 120,40 To 45,1 To 256,1,1997 To 2050)
```

Иногда при объявлении массива его размер не известен. В этом случае следует объявлять динамический массив, что позволяет изменять его размер или размерность во время выполнения приложения.

Динамический массив создается в два этапа. Сначала массив определяют в секции (General)(Declarations) контейнера (формы, модуля, класса) без указания размера:

```
Dim aArray() As Variant
```

Затем с помощью оператора ReDim устанавливают фактический размер массива:

```
'(General)(Declarations)
Dim aArray ()As Variant
Private Sub Command_Click()
ReDim aArray (50,10)
'Код
End Sub
```

Синтаксис оператора ReDim:

```
ReDim [Preserve] Имя_переменной (Границы)[As Тип_данных]
```

В отличие от обычного Dim, оператор ReDim используется только в процедурах. При этом тип данных указывать не обязательно, особенно если он уже определен оператором Dim. Вы можете использовать оператор ReDim для изменения числа элементов или размерности массива. Однако вы не можете объявить массив с данными одного типа, затем использовать ReDim для приведения массива к другому типу, за исключением случая, если массив содержит переменные типа Variant. В этом случае приведение массива к явному типу данных допустимо.

Таким образом, размерность массива можно при необходимости изменить. Но тогда возникает опасность потерять его содержимое, т.к. как после изменения размерности элементам массива присваиваются значения по умолчанию. Однако Visual Basic предоставляет возможность изменять размерность массив без потери содержимого. Для этого следует использовать ReDim вместе с ключевым словом Preserve:

```
Dim aArray () As Variant
Private Sub Comroandl_Click()
ReDim Preserve aArray (50,15)
'Код
End Sub
```

Использование с оператором ReDim зарезервированного слова Preserve позволяет сохранить содержимое массива при изменении его размера или размерности. Но следует учитывать, что для многомерных масси-вов можно изменять только последнее измерение.

Начиная с Visual Basic 6.0, вы имеете возможность присвоить содержимое одного массива другому так же, как вы присваиваете значение одной переменной другой. Например, вам необходимо скопировать массив байтов. Это можно сделать, копируя байт за байтом.

```
Sub ByteCopy(oldCopy()As Byte, newCopy() As Byte)
Dim i As Integer
ReDim newCopy (Lbound(oldCopy)To UBound(oldCopy))
For i=Lbound(oldCopy)To Ubound(oldCopy)
newCopy(i)=oldCopy(i)
Next i
End Sub
```

Однако гораздо проще и привлекательней это выглядит, если присвоить один массив другому:

```
Sub ByteCopy(oldCopy()As Byte, newCopy() As Byte)
newCopy = oldCopy()
End Sub
```

Существуют определенные правила присвоения переменных, которые не стоит забывать. Например, если присвоение переменной, объявленной как Long, значения переменной тип Integer осуществляется без всяких проблем, то присвоение значения переменной тип Long переменной тип Integer легко может вызвать ошибку переполнения (Overflow). В дополнение к правилам работы с типами данных, при присвоении массивов необходимо учитывать размерности массивов, количество элементов в массивах разных размерностей и тип массив - статический или динамический.

Попытка присвоения массивов различных размерностей и типов данных может быть успешной или неуспешной в зависимости от следующих факторов:

• типа массива, используемого в левой части оператора присваивания (статический (Dim x(1 to 10) As Integer) или динамический (Dim x() As Integer));

• совпадения или несовпадения количества размерностей массивов в левой и правой части оператора присваивания;

• совпадения или несовпадения количества элементов в каждой размерности массивов независимо от значения параметра оператора Option Base (0 или 1);

• типов данных элементов массивов (типы должны быть совместимыми, как и при присваивании обыч-ных переменных).

Ошибки могут происходить, как во время компиляции, так и во время выполнения проекта (например, если типы данных не могут быть согласованы или если присвоение делает попытку к переопределению (ReDim) размера статического массива). Как программист Вы должны предусмотреть обработку такой ошибки или проверить (программно), что массивы совместимы, перед попыткой присвоения.

1.1.6 Типы данных, определяемые пользователем

Кроме встроенных типов данных, таких как Integer, Long и т.п. Visual Basic поддерживает также типы данных, определяемые пользователем. Они могут быть созданы как на основе встроенных типов данных, так и на основе ранее определенных пользователем.

Для определения пользовательского типа данных используется ключевое слово Type:

```
[Private | Public] Type Имя_типа
Елемент1[([Размерность ])] As Тип1
[Елемент2[([Размерность ])] As Тип2]
End Type
```

Определение общего (Public) собственного типа данных возможно только в секции (General) (Declarations) модуля. В этом случае этот тип

будет доступен во всех процедурах всех форм, модулей и модулей классов. Для определения пользовательского типа данных в форме или модуле класса следует использовать ключевое слово Private, поскольку объявление общего типа в данной ситуации не допускается. При этом область видимости такого типа будет ограничена тем контейнером, где он объявлен. Определив собственный тип данных, вы можете использовать его для объявления переменных этого типа. Эти переменные могут быть локальными, глобальными или переменными контейнера.

1.2 Порядок выполнения работы

1.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу prLab2_2.exe
3. Ответить на вопросы, предложенные программой.
4. Сохранить в файле и/или распечатать протокол ответов.
5. Запустить на выполнение программу «Visual Basic 6.0».
6. Открыть проект “prLab2_2Template.vbp”.
7. Дополните проект объявлениями переменных в модуле mdLab2_2Template и на форме frmLab2_2Template в секции (General) (Declarations). Вы имеете право добавлять свои программные строки только между следующими строками – комментариями:

```
        ' Программный код процедуры A Вашего варианта
        ' следует добавлять после этой строки
Оператор 1: Оператор 2: Rem
Операторы, добавляемые при выполнении работы
Оператор 3 ' и т.д.
        ' Ниже этой строки ничего не исправлять!!!!!!!
```

Для перехода между программными кодами модуля и форм используйте окно проекта Project.

8. Дополните соответствующие процедуры (модуль mdLab2_2Template.bas процедуры: Main, A; форма frmLab2_2Template.frm процедуры: B, C, D; форма frmLab2_2Template1.frm процедуры: E, F) проекта строками задания для Вашего варианта. Строки задания в вариантах размещены между многоточиями.

Например, в варианте задания указаны строки:

```
... i = 16: j = 6 ...
... k = j + i ...
```

Это строки задания, которые необходимо вставить в соответствующее место программного кода проекта.

9. Последнюю строку варианта задания необходимо вставить в процедуре Main модуля mdLab2_2Template.bas между кавычками в строке: strExecute = "".

Например, у Вас должна получиться строка, аналогичная следующей:

```
StrExecute = "Main -> C -> C -> E -> F -> D"
```

10. Исправьте в именах (свойство Name) проекта, формы и модуля символы Template на свою фамилию.

11. Сохраните проект в папке с выполняемой лабораторной работой.

12. Запустите проект на выполнение. Если все сделано правильно, а вы следовали указаниям, появляющимся в строке состояния, то в ListBox'е Вы получите протокол работы программы, который следует сохранить в файле или распечатать на принтере. Если при работе проекта возникают ошибки, то запустите проект в режиме пошагового выполнения. Проследите за порядком выполнения процедур проекта. Запишите последовательность выполнения процедур проекта. Найдите и исправьте ошибку. Повторите п. 12 сначала.

13. Изучите процедуры проекта prLab2_2Template.vbp. Опишите назначение всех процедур и операторов.

1.2.2 Порядок выполнения

В заготовку проекта prLab2_2Template.vbp, в формы «frmLab2_2Template.frm» и «frmLab2_2Template1.frm», а также в программный модуль «frmLab2_2Template.bas» внесите изменения согласно Вашему варианту задания так, как это указано в пункте 2.2.1.

1.2.3 Варианты заданий

Вариант № 1

В програмном модуле modMy.bas имеются строки:

```
Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 3: j = 8 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub
```

В модуле формы frmMy1.frm имеются строки:

```
Private i as Integer, j as Integer
Private Sub B()
... i = 9: j = 1 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 6: j = 7 ...
... k = k + j + i ...
End Sub
```

В модуле формы frmMy2.frm имеются строки:

```
Private Sub E()
... i = 10: j = 10 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub
```

Чему равно k, если проект выполнялся в последовательности:

Main -> D -> B -> F -> E -> C

Вариант № 2

В програмном модуле modMy.bas имеются строки:

Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 2: j = 1 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub

В модуле формы frmMy1.frm имеются строки:

Private i as Integer, j as Integer
Private Sub B()
... i = 4: j = 9 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 6: j = 5 ...
... k = k + j + i ...
End Sub

В модуле формы frmMy2.frm имеются строки:

Private Sub E()
... i = 7: j = 7 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub

Чему равно k, если проект выполнялся в последовательности:
Main -> A -> B -> E -> B -> F

Вариант № 3

В програмном модуле modMy.bas имеются строки:

Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 7: j = 5 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub

В модуле формы frmMy1.frm имеются строки:

Private i as Integer, j as Integer
Private Sub B()
... i = 2: j = 9 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 4: j = 8 ...
... k = k + j + i ...
End Sub

В модуле формы frmMy2.frm имеются строки:

Private Sub E()
... i = 3: j = 6 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub

Чему равно k, если проект выполнялся в последовательности:
Main -> B -> E -> A -> C -> A

Вариант № 4

В програмном модуле modMy.bas имеются строки:

Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 6: j = 4 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub

В модуле формы frmMy1.frm имеются строки:

Private i as Integer, j as Integer
Private Sub B()
... i = 7: j = 0 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 5: j = 8 ...
... k = k + j + i ...
End Sub

В модуле формы frmMy2.frm имеются строки:

Private Sub E()
... i = 10: j = 1 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub

Чему равно k, если проект выполнялся в последовательности:
Main -> C -> C -> C -> C -> A

Вариант № 5

В програмном модуле modMy.bas имеются строки:

Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 3: j = 4 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub

В модуле формы frmMy1.frm имеются строки:

Private i as Integer, j as Integer
Private Sub B()
... i = 1: j = 6 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 2: j = 0 ...
... k = k + j + i ...
End Sub

В модуле формы frmMy2.frm имеются строки:

Private Sub E()
... i = 5: j = 8 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub

Чему равно k, если проект выполнялся в последовательности:
Main -> B -> A -> F -> F -> A

Вариант № 6

В програмном модуле modMy.bas имеются строки:

```
Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 9: j = 2 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub
```

В модуле формы frmMy1.frm имеются строки:

```
Private i as Integer, j as Integer
Private Sub B()
... i = 0: j = 5 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 4: j = 4 ...
... k = k + j + i ...
End Sub
```

В модуле формы frmMy2.frm имеются строки:

```
Private Sub E()
... i = 1: j = 3 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub
```

Чему равно k, если проект выполнялся в последовательности:

Main -> C -> B -> B -> D -> A

Вариант № 7

В програмном модуле modMy.bas имеются строки:

```
Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 3: j = 9 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub
```

В модуле формы frmMy1.frm имеются строки:

```
Private i as Integer, j as Integer
Private Sub B()
... i = 10: j = 6 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 7: j = 2 ...
... k = k + j + i ...
End Sub
```

В модуле формы frmMy2.frm имеются строки:

```
Private Sub E()
... i = 8: j = 0 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub
```

Чему равно k, если проект выполнялся в последовательности:

Main -> E -> E -> D -> C -> D

Вариант № 8

В програмном модуле modMy.bas имеются строки:

```
Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 2: j = 4 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub
```

В модуле формы frmMy1.frm имеются строки:

```
Private i as Integer, j as Integer
Private Sub B()
... i = 7: j = 6 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 9: j = 5 ...
... k = k + j + i ...
End Sub
```

В модуле формы frmMy2.frm имеются строки:

```
Private Sub E()
... i = 8: j = 3 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub
```

Чему равно k, если проект выполнялся в последовательности:

Main -> F -> B -> E -> E -> C

Вариант № 9

В програмном модуле modMy.bas имеются строки:

```
Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 9: j = 9 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub
```

В модуле формы frmMy1.frm имеются строки:

```
Private i as Integer, j as Integer
Private Sub B()
... i = 7: j = 8 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 2: j = 1 ...
... k = k + j + i ...
End Sub
```

В модуле формы frmMy2.frm имеются строки:

```
Private Sub E()
... i = 10: j = 10 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub
```

Чему равно k, если проект выполнялся в последовательности:

Main -> C -> B -> A -> A -> F

Вариант № 10

В програмном модуле modMy.bas имеются строки:

```
Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 6: j = 9 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub
```

В модуле формы frmMy1.frm имеются строки:

```
Private i as Integer, j as Integer
Private Sub B()
... i = 10: j = 0 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 3: j = 1 ...
... k = k + j + i ...
End Sub
```

В модуле формы frmMy2.frm имеются строки:

```
Private Sub E()
... i = 4: j = 4 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub
```

Чему равно k, если проект выполнялся в последовательности:

Main -> C -> D -> F -> C -> A

Вариант № 11

В програмном модуле modMy.bas имеются строки:

```
Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 10: j = 8 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub
```

В модуле формы frmMy1.frm имеются строки:

```
Private i as Integer, j as Integer
Private Sub B()
... i = 4: j = 0 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 5: j = 6 ...
... k = k + j + i ...
End Sub
```

В модуле формы frmMy2.frm имеются строки:

```
Private Sub E()
... i = 7: j = 9 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub
```

Чему равно k, если проект выполнялся в последовательности:

Main -> A -> D -> A -> F -> E

Вариант № 12

В програмном модуле modMy.bas имеются строки:

Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 3: j = 8 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub

В модуле формы frmMy1.frm имеются строки:

Private i as Integer, j as Integer
Private Sub B()
... i = 5: j = 5 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 2: j = 0 ...
... k = k + j + i ...
End Sub

В модуле формы frmMy2.frm имеются строки:

Private Sub E()
... i = 6: j = 4 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub

Чему равно k, если проект выполнялся в последовательности:
Main -> E -> D -> F -> A -> A

Вариант № 13

В програмном модуле modMy.bas имеются строки:

Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 8: j = 8 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub

В модуле формы frmMy1.frm имеются строки:

Private i as Integer, j as Integer
Private Sub B()
... i = 6: j = 1 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 5: j = 2 ...
... k = k + j + i ...
End Sub

В модуле формы frmMy2.frm имеются строки:

Private Sub E()
... i = 4: j = 10 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub

Чему равно k, если проект выполнялся в последовательности:
Main -> C -> C -> F -> E -> C

Вариант № 14

В программном модуле modMy.bas имеются строки:

Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 6: j = 5 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub

В модуле формы frmMy1.frm имеются строки:

Private i as Integer, j as Integer
Private Sub B()
... i = 10: j = 8 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 9: j = 9 ...
... k = k + j + i ...
End Sub

В модуле формы frmMy2.frm имеются строки:

Private Sub E()
... i = 1: j = 3 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub

Чему равно k, если проект выполнялся в последовательности:

Main -> F -> F -> D -> A -> C

Вариант № 15

В программном модуле modMy.bas имеются строки:

Public i as Integer, j as Integer, k as Integer
Public Sub Main()
... k = 0: i = 1: j = 4 ...
... k = i + j ...
End Sub
Public Sub A()
... k = k + i + j ...
End Sub

В модуле формы frmMy1.frm имеются строки:

Private i as Integer, j as Integer
Private Sub B()
... i = 8: j = 10 ...
... k = k + j + i ...
End Sub
Private Sub C()
... k = k + j + i ...
End Sub
Private Sub D()
Dim i as Integer, j as Integer
... i = 5: j = 6 ...
... k = k + j + i ...
End Sub

В модуле формы frmMy2.frm имеются строки:

Private Sub E()
... i = 0: j = 7 ...
... k = k + i + j ...
End Sub
Private Sub F()
... k = k + j + i ...
End Sub

Чему равно k, если проект выполнялся в последовательности:

Main -> B -> C -> D -> F -> A

1.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы.
2. Общие сведения о теме лабораторной работы «Синтаксис программного кода. Типы данных. Область определения. Время жизни переменных».
3. Протокол ответов на вопросы, предложенные программой prLab2_2.exe.
4. Ваш вариант задания на выполнение лабораторной работы.
5. Последовательность выполнения процедур проекта.
6. Протокол работы проекта prLab2_2Template.vbp.
7. Описание назначения процедур и операторов проекта.

1.3 Контрольные вопросы

1. Что такое синтаксис программного кода?
2. Каков синтаксис записи оператора?
3. Что такое разделители строк?
4. Для чего необходимы комментарии в программе?
5. Перечислите известные Вам типы данных.
6. Каков синтаксис оператора объявления данных?
7. Что такое явное объявление данных?
8. Как осуществляется неявное объявление данных?
9. Назначение опции Option Explicit?
10. Что такое область определения переменной?
11. Какие области определения переменных Вы знаете.
12. Что такое область время жизни переменных?
13. Что такое статические переменные?
14. Что такое массив данных?
15. Статические массивы. Границы массивов.
16. Назначение опции Option Base?
17. Синтаксис оператора объявления массива.
18. Многомерные массивы.
19. Динамические массивы. Оператор ReDim.
20. В чем отличие операторов ReDim и ReDim [Preserve]?
21. Присвоение массивов в Visual Basic 6.0.
22. Типы данных, определяемые пользователем.

ЛАБОРАТОРНАЯ РАБОТА 2

ВЫРАЖЕНИЯ И ВСТРОЕННЫЕ ФУНКЦИИ VISUAL BASIC

Цель работы – Научиться писать программный код на языке Visual Basic 6.0. Создать проект на Visual Basic 6.0 для изучения работы с выражениями. Научиться записывать арифметические, строковые и логические выражения. Научиться использовать встроенные математические функции Visual Basic 6.0. Научиться использовать встроенные функции Visual Basic 6.0 для работы со строками.

2.1 Общие сведения

2.1.1 Синтаксис программного кода

2.1.1.1 Переменные

Переменная представляет собой зарезервированное место в оперативной памяти для временного хранения данных. Каждая переменная имеет собственное имя (идентификатор). После того как переменной присвоено значение, вы можете использовать ее в программе вместо самого значения.

Для того чтобы сделать ваши переменные более наглядными и простыми для чтения, рекомендуется давать им имена, имеющие определенное смысловое значение. Существует несколько правил задания имен переменных:

- имя переменной может содержать не более 255 символов;
- имя переменной может содержать любые буквы и цифры;
- первый символ в имени переменной должен быть буквой;
- в имени переменной должны отсутствовать пробелы;
- имя должно быть уникальным в пределах области видимости.

Список ограничений достаточно велик, чтобы знать его наизусть, но вам всегда поможет проверка синтаксиса программы, при выполнении которой будет указано на использование недопустимых имен.

Например, допустимы следующие имена переменных:

```
CurrentNum, Total, Date_of_birth
```

Следующие имена недопустимы:

```
1Time, $Total, Date of birth
```

В последнее время в наименованиях переменных рекомендуется использовать префиксы, отражающие тип переменной. При таком обозначении переменных повышается читабельность программы и снижается количество ошибок программирования. Префиксы отражают тип переменной и область ее действия. С конкретными значениями префиксов вы познакомитесь в разделах, посвященных типам данных и области действия переменной.

Прежде чем использовать переменную в программе, ей необходимо присвоить значение. Самый простой способ присвоения заключается в использовании оператора присвоения " = ", который имеет следующий синтаксис:

```
переменная = выражение
```

Аргумент переменная задает имя переменной, которой будет присвоено значение выражения, стоящего справа от знака равенства. Например:

```
sngFirst = 10
strLastname = "Иванов"
```

Справа от знака равенства может стоять не только константа, но и более сложное выражение. Например:

```
sngResult = sngFirst + 255
strName = "Иванов" & ": " & strTeam
```

Переменную типа Variant можно использовать для хранения всех типов данных и выполнять операции, не заботясь о типе данных, в них содержащихся. Необходимо только помнить о двух исключениях. Во-первых, выполнять арифметические операции или функции над переменной типа Variant можно только в том случае, если она содержит числовое значение. Во-вторых, конкатенацию строк следует осуществлять с помощью оператора "&" вместо оператора "+".

Переменные типа Variant могут содержать специальные значения Empty, Null или Error. До присвоения значения переменной типа Variant ее значение отлично от 0, пустой строки или значения Null и именуется Empty. Для определения значения Empty можно использовать функцию isEmpty:

```
If IsEmpty(x) Then x = 0
```

Переменная типа Variant имеет значение Empty только до первого присвоения ей какого-либо значения.

Переменная типа Variant может иметь значение Null, используемое для указания пустых данных в приложениях, работающих с базами данных.

С помощью функции isNull можно проверить, имеет ли переменная типа variant значение Null:

```
If IsNull(x) Then
y = Null
Else
y = 0
End If
```

Для присвоения переменной типа Variant значения Null можно использовать ключевое слово Null:

```
y = Null
```

Присвоение значения Null переменной, не имеющей типа Variant, приведет к ошибке. При работе с переменной типа Variant, имеющей значение Null, необходимо иметь в виду следующее:

• если в выражении содержится переменная, имеющая значение Null, то само выражение принимает значение Null;
• при использовании в качестве параметра функции переменной, имеющей значение Null, большинство функций возвращает значение Null.

Переменная типа Variant может принимать значение Error при возникновении ошибки в процедуре. Однако в этом случае на уровне приложения ошибка не обрабатывается, что позволяет программисту в зависимости от ее значения предпринимать определенные действия.

Переменная типа Variant поддерживает внутреннее представление хранящихся в ней данных. При присвоении значения переменной типа Variant Visual Basic применяет наиболее компактное представление этого значения. Например, если переменной типа Variant присвоено небольшое числовое значение без дробной части, то используется представление Integer, если хранится дробное число — используется внутреннее представление Double.

2.1.1.2 Константы

Константой называют элемент выражения, значение которого не изменяется в процессе выполнения программ. Приведем несколько примеров:

• 75.07 – числовая константа;
• 2.7E+6 – числовая константа (равна 2 700 000);
• "Ошибка доступа к базе данных" – символьная константа;
• #8/12/1999# – константа типа дата;
• False – логическая константа.

Visual Basic содержит огромное количество встроенных констант практически для всех возможных случаев: цвета, клавиши, сообщения и т. п. Встроенные константы имеют префикс vb. Для поиска констант определенной категории воспользуйтесь браузером объектов (рис. 2.1), который открывается при нажатии кнопки Object Browser на стандартной панели инструментов.

Рисунок 2.1 – Браузер объектов (Object Browser)

Объявление констант во многом аналогично объявлению переменных. Константы можно объявлять на уровне модуля или процедуры. Область их действия при этом определяется теми же правилами, что и для переменных.

Для объявления константы на уровне процедуры используется оператор Const, имеющий следующий синтаксис:

```
Const имяКонстанты [As типДанных] = выражение
```

Например:

```
Const strDBErrorMessage As String = "Ошибка доступа к базе данных"
```

При объявлении константы на уровне модуля можно дополнительно указать область ее действия. В этом случае оператор Const имеет следующий синтаксис:

```
[Public | Private] Const имяКонстанты [As типДанных] = выражение
```

В приведенном ниже примере константа strDBErrorMessage объявлена глобальной:

```
Public Const strDBErrorMessage As String = "Ошибка доступа к базе данных"
```

2.1.2 Числовые выражения

Числовые выражения строятся из числовых констант, переменных и функций, связанных математическими операторами.

2.1.2.1 Математические операторы

Математические операторы позволяют выполнять в программе действия над числами. В таблице 2.1 приведены арифметические операторы и выполняемые ими функции.

Таблица 2.1 – Математические операторы

Оператор	Выполняемая операция
+	Сложение
–	Вычитание
*	Умножение
/	Деление
\	Целочисленное деление
Mod	Остаток от деления нацело
/\	Возведение в степень

Математические операторы предназначены для создания выражений. Выражения могут содержать переменные, константы, функции, связанные более чем одним оператором. 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.

Вы можете изменить порядок вычисления в выражении, используя круглые скобки. Например, в формуле (8 – 5 * (6 – 2)) / (3 + 7) вначале выполняется операция 6 – 2, затем умножение, затем вычитание из 8 предыдущего результата, сложение 3 + 7 и, наконец, деление.

Используя скобки, следите за тем, чтобы количество открывающих и закрывающих скобок было равно.

Рассмотрим примеры использования операторов деления. Их в таблице три. Первый из них выполняет деление с плавающей точкой. Его оператор записывается в виде косой черты (/). Введите в окне Immediate Visual Basic следующее выражение:

```
Print 10/3
```

В результате в окно Immediate будет выведено число 3,33333333333333.

Далее в таблице 3.1 следует оператор целочисленного деления. Он записывается в виде обратной косой черты (\). Введите в окне Immediate

```
Print 10\3
```

В результате будет получено число 3.

Третий оператор деления — оператор Mod. Он выдает остаток, получающийся в результате деления. Введите в окне Immediate Visual Basic следующее выражение:

```
Print 10 mod 3
```

В результате в окно Immediate будет выведено число 1.

Последним в таблице 3.1 приведен оператор возведения в степень. Он имеет следующий синтаксис:

```
результат = число ^ показательСтепени
```

Если параметр показательСтепени больше 1, это означает, что число возводится в степень. Если он меньше 1, но больше 0, то из числа извлекается корень. При значении параметра меньше 0 вычисляется обратное значение дроби. Например:

```
2 ^ 3 = 8
```

Для примера запишем, как будет выглядеть на Visual Basic следующее выражение:

$$y = \frac{\dfrac{a-b}{\sqrt{a^2+b^2}}}{\dfrac{e^{-x}-\sin\left(\dfrac{\pi}{2}-2x\right)}{\ln(a+\sqrt{a^2+b^2})}}$$

```
y = (((a - b) / Sqr(a^2 + b^2)) / (exp(-x) - sin(pi /
2 - 2 * x)))/Log(a + Sqr((a^2 + b^2))
```

2.1.2.2 Встроенные математические функции

Встроенные математические функции (имеются в распоряжении любого программиста на Visual Basic) используются для построения математических выражений при обработке численных данных. Перечень и назначение встроенных математических функций приведен в таблице 2.2.

Таблица 2.2 - Встроенные математические функции

№ п.п.	Функция	Описание (x и значение функции – действительные числа)
1	Abs(x)	Абсолютная величина числа
2	Atn(x)	Арктангенс
3	Cos(x)	Косинус (аргумент в радианах)
4	Exp(x)	Экспонента
5	Fix(x)	Отбрасывание дробной части числа
6	Int(x)	Округление до ближайшего целого
7	Log(x)	Натуральный логарифм
8	Rnd(x)	Вырабатывает случайное число в диапазоне от 0 до 1
9	Sgn(x)	Знак числа. Результат: -1; 0; 1 в зависимости от знака числа
10	Sin(x)	Синус (аргумент в радианах)
11	Sqr(x)	Квадратный корень
12	Tan(x)	Тангенс (аргумент в радианах)

2.1.2.3 Производные математические функции

Другие математические функции могут быть построены (для использования производных математических функций они должны быть оформлены как соответствующие функции пользователя) с помощью приведенных в таблице 2.3.

Пример создания производной математической функции:

```
Public Function HSin(X As Double) As Double
HSin = (Exp(X) - Exp(-X)) / 2
End Function
```

2.1.3 Работа со строками

Для работы в Visual Basic со строками используется оператор объединения, называемый также оператором конкатенации, и встроенные функции.

Строковая константа – набор символов ASCII кода, заключенный в кавычки. Строковые переменные (тип данных String) чаще всего получают своё значение с помощью оператора присваивания. Оператор конкатенации (объединения) имеет следующий синтаксис:

```
Строка1 {+|&} Строка2
```

Таблица 2.3 - Производные математические функции

Функция	Эквивалентное выражение
Secant	Sec(X) = 1 / Cos(X)
Cosecant	Cosec(X) = 1 / Sin(X)
Cotangent	Cotan(X) = 1 / Tan(X)
Inverse Sine	Arcsin(X) = Atn(X / Sqr(-X * X + 1))
Inverse Cosine	Arccos(X) = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
Inverse Secant	Arcsec(X) = Atn(X / Sqr(X * X – 1)) + Sgn((X) – 1) * (2 * Atn(1))
Inverse Cosecant	Arccosec(X) = Atn(X / Sqr(X * X - 1)) + (Sgn(X) – 1) * (2 * Atn(1))
Inverse Cotangent	Arccotan(X) = Atn(X) + 2 * Atn(1)
Hyperbolic Sine	HSin(X) = (Exp(X) – Exp(-X)) / 2
Hyperbolic Cosine	HCos(X) = (Exp(X) + Exp(-X)) / 2
Hyperbolic Tangent	HTan(X) = (Exp(X) – Exp(-X)) / (Exp(X) + Exp(-X))
Hyperbolic Secant	HSec(X) = 2 / (Exp(X) + Exp(-X))
Hyperbolic Cosecant	HCosec(X) = 2 / (Exp(X) – Exp(-X))
Hyperbolic Cotangent	HCotan(X) = (Exp(X) + Exp(-X)) / (Exp(X) – Exp(-X))
Inverse Hyperbolic Sine	HArcsin(X) = Log(X + Sqr(X * X + 1))
Inverse Hyperbolic Cosine	HArccos(X) = Log(X + Sqr(X * X – 1))
Inverse Hyperbolic Tangent	HArctan(X) = Log((1 + X) / (1 – X)) / 2
Inverse Hyperbolic Secant	HArcsec(X) = Log((Sqr(-X * X + 1) + 1) / X)
Inverse Hyperbolic Cosecant	HArccosec(X) = Log((Sgn(X) * Sqr(X * X + 1) + 1) / X)
Inverse Hyperbolic Cotangent	HArccotan(X) = Log((X + 1) / (X – 1)) / 2
Logarithm to base N	LogN(X) = Log(X) / Log(N)

Если Строка1 и/или Строка2 являются переменными типа Variant, то следует использовать знак «&». Если строки являются строковыми переменными или константами, то знак «+» или «&».

В Visual Basic 6 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 (&).

В предыдущих версиях Visual Basic для объединения строк использовался символ "плюс" (+). В Visual Basic 6 этот оператор также поддерживается.

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
Print sName ' Возвращает "Иванов Иван Иванович"
```

Список наиболее часто используемых функции для работы со строками приведен в таблице 2.4.

Таблица 2.4 – Функции, предназначеные для работы со строками

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

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

```
Str(число)
```

Функция Val() преобразовывает символьную строку в численное значение. Синтаксис функции:

```
Val(символьноеВыражение)
```

При преобразовании строки символов в число учитываются все цифровые символы, расположенные в строке слева направо. Пробелы, находящиеся в начале и конце символьной строки, игнорируются. Пробелы внутри строки недопустимы. Если первый символ выражения не является цифрой, функция Val возвратит значение ноль.

Рассмотрим следующий пример. Зададим две символьные строки a и b следующего вида:

```
a = "10"
b = "12"
```

При сложении этих двух строк получается символьная строка:

```
a & b -> "1012"
```

Если сложить две символьные строки, предварительно преобразовав их в числа, и распечатать с помощью команды:

```
Print Val(a) + Val(b),
```

то получится число 22.

Функции LTrim, RTrim и Trim используются для удаления пробелов в символьной строке:

LTrim Удаляет пробелы, расположенные в начале символьной строки

RTrim Удаляет пробелы, расположенные в конце символьной строки

Trim Удаляет пробелы, расположенные в начале и в конце символьной строки

Пример использования этих функции приведен ниже:

```
cComment = " Удаление пробелов "
Print LTrim(cComment) ' Возвращает "Удаление пробелов "
Print RTrim(cComment) ' Возвращает " Удаление пробелов"
Print Trim(cComment) ' Возвращает "Удаление пробелов"
```

Вы можете выделить подстроку заданной символьной строки, используя функции Left, Right и Mid. Функции Left и Right выделяют строку, начиная с крайнего левого или крайнего правого символа, а функция Mid позволяет выбрать любую подстроку.

Синтаксис функций:

```
Left(выражение, числоСимволов)
Right(выражение, числоСимволов)
Mid(выражение, номерПозиции [, числоСимволов ])
```

Ниже приведены примеры использования этих функций и возвращаемые ими значения:

```
cComment = "Выделение подстроки"
Print Left(cComment, 3) ' Возвращает "Выд"
Print Right(cComment, 6) ' Возвращает "строки"
Print Mid(cComment, 11,3) ' Возвращает "под"
```

Функции UCase и LCase используются в Visual Basic для преобразования строчных символов в заглавные и заглавных в строчные. Помимо этого, в Visual Basic имеется функция StrConv, которая преобразовывает выражение к имени собственному, начинающемуся с заглавной буквы.

Функции UCase() и LCase() возвращают значения, имеющие тип Variant. Для того чтобы возвращаемое значение имело тип String, необходимо использовать функции UCase$() и LCase$().

Функция UCase преобразует все строчные буквы в символьной строке в заглавные. Синтаксис функции:

```
UCase (символьнаяСтрока)
```

Например:

```
cComment = "вывод"
```

```
Print UCase(cComment) ' Возвращает "ВЫВОД"
Print UCase$(cComment) ' Возвращает "ВЫВОД"
```

Функция LCase возвращает заданную символьную строку, в которой все заглавные буквы преобразованы в строчные. Синтаксис функции:

```
LCase(символьнаяСтрока)
```

Например:

```
cComment = "ВЫВОД"
Print LCase(cComment) ' Возвращает "вывод"
Print LCase$( cComment) ' Возвращает "вывод"
```

Функция StrConv преобразовывает выражение, написанное строчными или заглавными буквами, в имя собственное.

Например:

```
cComment = "Иванов иван Иванович"
Print StrConv(cComment, vbProperCase) ' Возвращает "Иванов Иван Иванович"
```

Аналогичный результат будет получен и в следующем случае:

```
cComment = "ИВАНОВ ИВАН ИВАНОВИЧ"
Print StrConv(cComment, vbProperCase) ' Возвращает "Иванов Иван Иванович"
```

Visual Basic содержит две функции, позволяющие осуществлять поиск символьной строки в другой: inStr() и inStrRev(). Эти функции отличаются тем, что inStr о осуществляет поиск с начала строки и до ее конца, а InStrRev() проводит поиск в обратном направлении, то есть от конца строки к началу.

Функция inStr() имеет следующий упрощенный синтаксис:

```
InStr (исходнаяСтрока, строкаПоиска)
```

В результате проведенного поиска функция возвращает число, указывающее номер позиции первого вхождения строки.

Рассмотрим такой пример:

```
Print InStr ("Сегодня прекрасная погода", "погода")
```

В результате будет возвращено число 20.

Более полный синтаксис функции inStr() имеет следующий вид:

```
InStr (началоПоиска, исходнаяСтрока, строкаПоиска)
```

2.1.4 Условные выражения

Основанием для принятия решений в управляющих конструкциях являются условные выражения, поэтому предварительно необходимо сказать несколько слов об этих выражениях и работе с ними.

Условные выражения - это такие выражения, которые возвращают одно из двух значений True (Истина) или False (Ложь). В условных выражениях используются операторы сравнения, приведенные в таблице 2.5.

Над условными выражениями можно выполнять действия логической математики (логические операции), а именно:

And (И) — возвращает значение True (Истина), если все участвующие в операции выражения имеют значение True. В остальных случаях возвращается значение False (Ложь);

Таблица 2.5 – Операторы сравнения для условных выражений

Оператор	Назначение
=	Равно
>	Больше
<	Меньше
<>	Не равно
> =	Больше или равно
< =	Меньше или равно

Or (ИЛИ) — возвращает значение True, если хотя бы одно из участвующих в операции выражений имеет значение True. В случае, когда все выражения имеют значение False, возвращается значение False;

Xor (Исключающее ИЛИ) — возвращает значение True (Истина), если только одно из участвующих в операции выражений имеет значение True. В остальных случаях возвращается значение False;

Not (НЕ) — операция отрицания. Возвращает обратное для значения выражения значение, то есть если выражение равно True, то возвращается False и наоборот, если значение выражения равно False, то возвращается значение True.

Синтаксис использования логических операций такой же, как и у арифметических операций. Например:

```
(выражение1 And выражение2 And выражение3) Or (выражение4 Xor выражение5)
```

Скобки в условных выражениях действуют так же, как и в арифметических, то есть первыми всегда выполняются операции в скобках.

Сложные выражения можно предварительно вычислить и хранить в логических переменных типа **Boolean**. Например, предыдущий код с использованием переменных можно представить следующим образом:

```
Dim bVar1 As Boolean
Dim bVar2 As Boolean
bVar1 = выражение1 And выражение2 And выражение3
bVar2 = (выражение4 Xor выражение5)
```

Итоговым будет следующее выражение:

```
bVar1 Or bVar2
```

2.2 Порядок выполнения работы

2.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу prLab2_3.exe
3. Ответить на вопросы, предложенные программой.
4. Сохранить в файле и/или распечатать на принтере протокол ответов.
5. Запустить на выполнение программу «Visual Basic 6.0».

6. Открыть в ней проект “prLab2_3Template.vbp”.

7. Дополните проект объявлениями переменных в модуле mdLab2_3Template в секции (General) (Declarations). Объявлять нужно как переменные, в которые будут засылаться значения вычисленных выражений (т.е. X, Y и Z), так и переменные, которым будут присваиваться значения, необходимые для вычисления выражений (т.е. α, ω, β, A и т.д.). Все переменные должны быть объявлены как глобальные.

Вы имеете право добавлять свои программные строки только между следующими строками – комментариями:

```
' Программный код Вашего варианта
' следует добавлять после этой строки
Оператор 1: Оператор 2: Rem
Операторы, добавляемые при выполнении работы
Оператор 3 и т.д.
' Ниже этой строки ничего не исправлять!!!!!!!
```

Для перехода между программными кодами модуля и форм используйте окно проекта Project.

8. В процедуре Main модуля mdLab2_3Template.bas измените в соответствии с вариантом задания количество и идентификаторы параметров, используемых при вызове функций для вычисления выражений.

9. Дополните модуль mdLab2_3Template.bas соответствующими функциями для вычисления математических (задание 1), логических (задание 2) и строковых (задание 4) функций Вашего варианта (см. пример выполнения задания). Для создания функций используйте кнопку панели инструментов «Добавить процедуру» (“Add Procedure”). При написании тела любой нужной Вам функции соблюдайте синтаксис языка Visual Basic.

10. Измените в модуле класса формы frmLab2_3Template строки процедуры обработки события нажатия кнопки «Выполнить» - cmdExecute_Click() строки вызова функций, вычисляющих значения математического, логического и строкового выражений.

11. Сохраните проект в своей папке.

12. Запустите проект на выполнение. Если все сделано правильно, а вы следовали указаниям, появляющимся в строке состояния, то в ListBox’е Вы получите протокол работы программы, который следует сохранить в файле (протокол работы проекта сохраняйте в свою папку).

13. Изучите процедуры проекта prLab2_3Template.vbp. Постарайтесь понять назначение тех или иных процедур и/или операторов.

14. Постройте, как указано в третьем пункте задания Вашего варианта, на плоскости XOY область, в которой логическое выражение имеет значение TRUE (без ЭВМ).

2.2.2 Порядок выполнения

1. Вычислить значение числового выражения для Y.

2. Вычислить значение логического выражения для Z.

3. Начертить на плоскости XOY область, в которой логическое выражение имеет значение TRUE (без ЭВМ).

4. Вычислить значение строкового выражения для строки X.

В вариантах заданий приняты следующие обозначения:

- в логических выражениях:

Символ	Обозначение
¬	Not
|	Or
∩	And

- в строковых выражениях

Символ	Обозначение
(m:)	Выделить m символов строки справа
(:m)	Выделить m символов строки слева
//	Объединить строки

В заготовку проекта prLab2_3Template.vbp внесите изменения в соответствии с вариантом Вашего задания так, как это указано в пункте 3.2.1.

2.2.3 Варианты заданий

Вариант № 1. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{\operatorname{cosec}(a-\pi)+\sin^2\alpha+e^{-a}-\sqrt{x-0.8}}{0.5\cdot10^{-2}a^3-\ln|a|+0.3\cdot10^{-1.5}x}$;

при a = 0.5; x = 3.4; α = 1.0.

2) Z = ¬(A|B|C)∩(X<Y)|(Y>0.5);

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) X > Y | X ≤ Y | Y ≤ 3;

4) X = B(4:)//A(:2)//B(:4);

при B = «SIONEXPR», A = «ES34».

Вариант № 2. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{\lg(x+1)^2+1.5x^3+\sqrt[3]{x+0.6\cdot10^3}}{x^5+1.25\cdot10^{-2}\sin\beta^2+\sqrt[3]{a+0.6}}$;

при a = 0.5; x = 3.4; β = 1.65;

2) Z = A|B∩C∩¬(X≤Y∩X>–0.15);

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) X·Y ≤ 0|X ≤ 3|Y ≤ 3;

4) X = φ(:2)// A(:3)//a(2:),

при φ = ”AUTISM”, A = “TOBY”, a = “ICARUS”.

Вариант № 3. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{\log_3(x+1)^3 - 1.5\cdot 10^{-3.5} x^2 + \sqrt[5]{\alpha + 6\cdot 10^{-5}} + \arcsin\beta}{e^a x^3 + 6.15\cdot 10^4 \cos^2\beta + \sqrt[4]{x/a + tg0.6}}$;

a = 0.5; x = 3.4; β = 1.65; α = 1.0;

2) Z = A ∩ ¬(B∩C∩A) ∩¬(3.5<X|Y<1.2);

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) Y > X | Y ≤ 2 | Y ≥ 8;

4) X = R(4:)//r(2:3)//λ,

при R = “AKAI”, r = “PHILIPS”, λ = “SONY”.

Вариант № 4. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{e^{3x} tg(x+\beta) - \lg(x^3 - a) + \sqrt[5]{a + x^3}}{x^4 + \sqrt{x+1} - \sin^2\beta + 0.9\cdot 10^4 + 0.9ax}$;

при a = 0.5; x = 3.4; β = 1.65;

2) Z = ¬A|B∩¬(1.8 ≤ X ∩ Y ≥ 0.5) ∩C;

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) $X^2 + Y^2 \le 1 \cap X \ge 0 \mid Y \ge 0$;

4) X = B(:2)//b(3:)// A(1:3),

при B = “EXPRE”, b = “SION”, A = “ARITHM”.

Вариант № 5. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{\cos 4\alpha - \lg^2(x+a) e^x x^3 - 1.7\cdot 10^5 a}{x(x^3+5) + x^{a^2} - a\sqrt{x - 0.5\cdot 10^{-1.5}}}$;

при a = 0.5; x = 3.4; α = 1.65;

2) Z = X > Y ∩ ¬A | (B ∩ C) ∩ X ≤ 1.8;

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) $X^2 + Y^2 \ge 1 \mid X\cdot Y > 0 \mid X \le -3$;

4) X = E(:2)//δ(1:3)//e(4:),

при E = “LOGIC”, δ = “CALE”, e = “XPRESS”.

Вариант № 6. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{\sin 3\alpha + (x^2+1)\cos^2\alpha \cdot x - 0.8\cdot 10^{-2a} e^x}{\sqrt[3]{x+1} - \lg(a^2 x) + 0.3(x^3 - a) + \arccos x}$;

при a = 0.5; x = 3.4; α = 1.65;

2) Z = Y < 3.5 ∩ B | C ∩ ¬(A ∩ B | X < 1.9);

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) $X^2 + Y^2 < 1 \cap X\cdot Y \le 0 \cap Y \ge 1$;

4) X = μ(3:)//B(2:3)//b,

при μ = «FINAN», b = «SIAL», B = «TIMES!».

Вариант № 7. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{\sin(\log_3(x+1)) + 1.5x^3 e^{-ax} + \sqrt[4]{x^2 + 4.5\cdot 10^{-5}}}{\sqrt{\cos x}\cdot x^5 - 6\cdot 10^{2.3} - \sqrt[5]{|a-x|} + \arccos\beta}$;

при $a = 0.5$; x = 3.4; $\beta = 1.65\cdot 10^{-4}$;

2) $Z = \neg(X \le 1.5 \cap Y > 9.4) \cap A \cap B \mid C$;

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) $Y \le 2\cdot X + 2 \cap X^2 + Y^2 \ge 4$;

4) X = B2(:2)//b2(3:)// A(1:3),

при b2 = «CHIRO», A = «FORD», B2 = «SIERRA».

Вариант № 8. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{\cos^2(\alpha-\pi) + \ln|2x+1|^2 + 4.5x^4 + 0.9\cdot 10^{-2}\cdot\sqrt[3]{\sin x + 9.6\cdot 10^{3.1}}}{e^{-x}x^2 - 1.9(\arcsin\beta^2 + \sqrt{2x+a})}$;

при $a = 0.5$; x = 3.4; $\alpha = 1.0$; $\beta = 1.65$;

2) $Z = X > Y \cap A \mid C \cap (\neg A \mid B \cap C) \mid A$;

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) $X^2 + Y^2 \le 1 \mid X\cdot Y \le 0 \mid X \le -0.5$;

4) X = α(2:3)//β (1:2)//γ (4:),

при γ = «FOTON», α = «PECI», β = «LONDON».

Вариант № 9. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{\sin(\pi - \beta/2) + \lg(2x + a^2) + x^2 + \sqrt[4]{|x - a^3|}}{e^{x+a} + x^3 + 5.4\cdot 10^{-4} + tg(x^2 + 0.5\cdot 10^{2.1})}$;

при $a = 0.5$; x = 3.4; $\beta = 1.65$;

2) $Z = A \cap B \mid X < Y \cap (X < 1.5 \cap B \mid C)$;

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) $Y \ge X^2 - 3 \mid Y \le X \cap X \ge 0$;

4) X = φ(:3)//ψ(:2)//ε(2),

при ψ = «PSI_P», ε = «EPSI», φ = «PHI_LE».

Вариант № 10. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{1.9\cdot 10^3 x - e^{ax} + \arcsin(\beta + x) - \lg x^2 + a}{\sqrt[3]{(ax - 1.72)^2} + 4.75\cdot 10^{1.2}(a - x) - \cos^2(\beta - x)}$;

при $a = 0.5$; x = 3.4; $\beta = -3.35$;

2) $Z = B \cap C \mid (X \ge Y) \cap (X \ge 5.5) \mid A \cap \neg B$;

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) $Y \ge -X \mid Y \ge X^2 - 2 \cap Y > 3$;

4) X = ω(2:3)// α(:3)//τ(3:),

при τ = «BORLA», α = «ND_P», ω = «ASCAL!».

Вариант № 11. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{e^{-x} + \ln|\arccos x| + arctgx - 1.2\cos^2 x}{\sqrt{ax}(a \cdot e^{ax} + 3.5 \cdot 10^{-5} ax) - \sqrt[3]{x + a^2 - 0.3 \cdot 10^{-1.5}} + \lg\beta}$;

при $a = 0.5$; x = 3.4; β = 1.65;

2) Z = A | B ∩ X ≤ 1.3 | Y ≥ 1.5 ∩ ¬(A ∩ B);

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) $X^2 + Y^2 \le 1 \cap X \ge 0 \mid Y \le 0$;

4) X = λ(2:4)//μ(:3)//ε,

при μ = «EPSIL», ε = «1_3_», λ = «LAMBDA».

Вариант № 12. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \sqrt{\left|\lg\left|\dfrac{\arccos(x - 3\beta) - (ax^6 + e^{ax})^2 + 0.3 \cdot 10^{-1.5} x}{7.3 \cdot 10^7 \cdot |x - a^5| - tg(a + x) + \sqrt{\sin^2 a + \pi} - \ln x}\right|\right|}$;

при $a = 0.5$; x = 3.4; β = 1.1;

2) Z = A | ¬C ∩ X < 4.2 | Y ≥ 3.5 ∩ ¬(B | C ∩ X > Y);

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) X·Y ≤ 0 | X ≤ Y | Y ≤ 1;

4) X = α(:3)//a(2:3)//R(5:)//r(1:3),

при α = «DOLL», a = «AR_FR», R = «ANK_MA», r = «sony».

Вариант № 13. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{\pi + arctg(x^2 + a^5) - \sqrt{1 + xa} + \arcsin(a - 0.5 \cdot 10^{-3})}{4.8 \cdot 10^{2.6} - \lg_3|x - a| + 7 \cdot e^{-x} + \ln^2 x - tg(ax)}$;

при $a = 0.5$; x = 3.4;

2) Z = C ∩ A | C | Y ≥ 2.5·10^{-5} ∩ ¬(A ∩ B | Y ≤ 1.3);

при A = True; B = True; C = False; Y = 2.5;

3) $X^2 + Y^2 \le 1 \cap X \ge 0 \mid Y \le 1$;

4) X = ξ(3:)//θ(:2)//β(:3),

при β = «BETAL», ξ = «KSI_LET», θ = «TETA».

Вариант № 14. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \dfrac{\ln^2(a + 2e^x) - 6.6 \cdot 10^6 \sqrt{a + x^2} - 3.51 \cdot 10^{6.1} a}{\sqrt[5]{|\arccos x|} + a^x - 1.5\log_5 a - \beta}$;

при $a = 0.5$; x = 3.4; β = 1.65;

2) Z = B | A | ¬C ∩ 3 < Y ∩ 5 > X ∩ ¬(A ∩ B);

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) $X^2 + Y^2 < 1 \mid X \cdot Y \le 0 \mid X \le 0 \mid Y > 1$;

4) X = C(3:)//D(:2)//d(5),

при C = «wars», D = «engine», d = «TURBIN».

Вариант № 15. Вычислить выражения № 1, 2 и 4. Построить область 3:

1) $Y = \sqrt{\left| \frac{\arcsin(x-3\beta)-(ax^3+e^{ax})^2+0.3\cdot 10^{-1.5}x}{1.3\cdot 10^5\cdot |x-a^5| - tg(a+x)+\sqrt{\sin^2 a+\pi}-\lg x} \right|}$;

при a = 0.5; x = 3.4; β = 1.1;

2) Z = C ∩ X < 4.2 | Y ≥ 3.5 ∩ ¬(B | C ∩ X > Y);

при A = True; B = True; C = False; X = 1.5; Y = 2.5;

3) X·Y ≤ 0 | X ≤ 3 | Y ≤ 3;

4) X = A(:3)//a(2:3)//R(5)//r(1:3),

при A = «FREE», a = «OF_CH», R = «PANASO», r = «sony».

2.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы.
2. Общие сведения о теме лабораторной и цели работы «Выражения и встроенные функции».
3. Протокол ответов на вопросы, предложенные программой prLab2_3.exe.
4. Ваш вариант задания на выполнение лабораторной работы.
5. Протокол работы проекта prLab2_3Template.vbp.
6. Разработанные Вами функции для вычисления математических, логических и строковых выражений.
7. Схематическое изображение области для третьего задания вашего варианта.

2.3 Контрольные вопросы

1. Что такое синтаксис программного кода?
2. Каков синтаксис оператора Visual Basic?
3. Что представляют собой переменные в Visual Basic?
4. Перечислите правила задания имен переменных.
5. Каковы особенности использования переменных типа Variant?
6. Каков смысл специальных значений Empty, Null или Error?
7. Что такое константы в Visual Basic?
8. Опишите синтаксис объявления константы.
9. Что такое числовые выражения?
10. Опишите порядок выполнения математических операций.
11. Где используются встроенные математические функции?
12. Какие операции используются при работе со строковыми выражениями?
13. Какие функции позволяют выделять подстроку?
14. Что Вы знаете о логических выражениях?

ЛАБОРАТОРНАЯ РОБОТА 3

ПРОЦЕДУРЫ И ФУНКЦИИ VISUAL BASIC

Цель работы – Научиться разрабатывать процедуры и функции на языке Visual Basic 6.0. Создать проект на Visual Basic 6.0, содержащий процедуры и функции. Научиться создавать процедуры обработки событий и процедуры и функции обработки информации. Научиться передавать процедурам аргументы и пользоваться значениями функций.

3.1 Общие сведения

3.1.1 Процедуры

В Visual Basic, как и во многих других языках программирования, весь программный код находится внутри процедур (подпрограмм), или функций. Процедуры, позволяют разбивать программные коды на небольшие логические блоки, которые, во-первых, легче отлаживать, а во-вторых, можно в свою очередь использовать при создании других процедур. В Visual Basic существуют следующие виды процедур:

- Sub
- Function
- Property

Собственно, процедура — это подпрограмма. Она начинается оператором Sub и заканчивается оператором End Sub, между которыми и помещается исполняемый код. Такие процедуры могут вызываться или самим Visual Basic (процедуры обработки событий), или другими процедурами. Под процедурами подразумевают последовательность объявлений и инструкций (операторов), объединенных для выполнения. В зависимости от назначения можно выделить процедуры обработки событий и процедуры общего назначения. Процедура Sub имеет следующий синтаксис:

```
{[Private] | [Public] | [Static]} Sub имяПроц (аргументы)
     операторы
End Sub
```

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

3.1.1.1 Процедуры обработки событий

Процедурами обработки событий являются процедуры, которые предназначены для обработки некоторых событий, связанных с элементами управления или с самой формой.

Например, различные действия пользователя с кнопкой CommandButton (Click, KeyDown, MouseMove и т.п.) вызывают соответствующие события. Обработка каждого из этих событий оформляется в виде процедуры. Программист, применяя одну или несколько таких про-

цедур обработки события, может определить реакцию приложения на конкретное действие пользователя.

Процедуру обработки события легко отличить и по ее имени, в котором обязательно присутствуют имена объекта и события, а также по состоянию рабочей среды: если вы находитесь в такой процедуре, то в поле списка (Object) окна кода указывается имя объекта, а в поле списка (Procedure) — имя события. Имя процедуры обработки события всегда составляется из имен объекта и события, разделенных символом подчеркивания «_».

Для события, связанного с формой, процедура Sub имеет следующий синтаксис:

```
Private Sub Form_имяСобытия (аргументы)
        операторы
End Sub
```

Как видно из синтаксиса, наименование процедуры обработки события для формы содержит слово Form, затем размещается символ подчеркивания «_» и имя события. Например, имя процедуры, выполняемой при загрузке формы, будет Form_Load, а процедуры, выполняемой при щелчке мыши на форме - Form_Click.

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

```
Private Sub имяЭлементаУпр_имяСобытия (аргументы)
        операторы
End Sub
```

Наименование процедуры обработки события для элемента управления формы содержит имя элемента управления, заданное в свойстве Name, затем следует символ подчеркивания «_» и имя события. Например, имя процедуры, выполняемой при щелчке мыши на кнопке управления, имеющей наименование cmdPrint, будет cmdPrint_Click.

Visual Basic облегчает формирование имен создаваемых процедур. Разработчику необходимо выполнить для этого следующие действия:

1. В окне Properties с помощью свойства Name (Имя) задать имя объекта, для которого создается процедура. Если имя не будет задано, то при создании процедуры Visual Basic использует имя, присваиваемое объекту по умолчанию при его размещении в форме. При последующем изменении наименования объекта необходимо будет изменить и имя процедуры.

2. В окне редактора кода из списка Object (Объект) выбрать объект, для которого создается процедура.

3. Из списка Procedure (Процедура) выбрать событие, обработка которого будет выполняться.

После выполнения указанных действий в области размещения процедур редактора кода будут размещены операторы Sub и End Sub с указанием наименования процедуры (рис. 3.1). Вам необходимо разместить

между этими операторами выполняемый при наступлении этого события программный код.

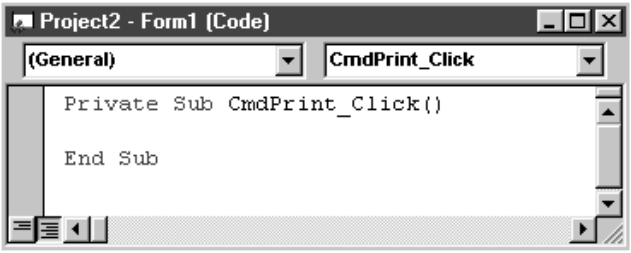

Рисунок 3.1 – Формирование процедуры средой Visual Basic

В случае, если после создания процедуры имя элемента управления будет изменено, а имя связанной с ним процедуры останется прежним, процедура станет общей.

3.1.1.2 Процедуры общего назначения

Основной отличительной чертой процедур общего назначения является то, что они не связаны ни с каким событием и их вызов разработчик осуществляет по своему усмотрению. Для создания такой процедуры достаточно ввести ключевое слово Sub и имя процедуры в окне кода (но не внутри другой процедуры или функции) и нажать клавишу [Enter]. После этого Visual Basic дополнит введенный код оператором конца процедуры End Sub самостоятельно.

Процедуры общего назначения относятся к секции (General). Так как процедура не связана ни с одним элементом управления, то поле списка (Object) окна кода вместо имени объекта содержит строку (General).

Процедуры общего назначения используются, как правило, для решения каких либо общих задач, например расчетов, которые необходимо выполнять в различных местах программы.

3.1.1.3 Закрытые процедуры

Закрытыми называют процедуры, доступные только внутри контейнера (формы, модуля, класса), в котором они содержатся. Все процедуры обработки событий объявляются по умолчанию как Private. Это значит, что такую процедуру можно вызывать только внутри этой формы. Общие процедуры формы или модуля класса также являются закрытыми. Они останутся закрытыми даже после того, как вы объявите их как Public.

Тем не менее, общую процедуру можно вызвать из другой формы, если только она не была объявлена как Private. Для этого следует указать перед ее вызовом имя формы, которой она принадлежит, например Form1.SomeProcedure, если процедура SomeProcedure находится на форме Form1.

3.1.1.4 Общие процедуры

Процедуры называются общими, если они могут быть вызваны процедурами другого контейнера. Это возможно только в том случае, если такая процедура содержится в модуле. Обычно, каждая процедура модуля может вызываться любой другой процедурой.

В модулях также полностью проявляется действие ключевых слов Private и Public. Чтобы объявить процедуру общей, в ее заголовке указывается ключевое слово Public. Это не обязательно, так как процедуры в модулях общедоступны по умолчанию. Если вызов процедуры другим контейнером нежелателен, это можно предотвратить с помощью ключевого слова Private.

3.1.1.5 Option Private Module

Выражение Option Private Module используется в модуле для указания того, что модуль является закрытым для других приложений. С опцией Option Private Module составные элементы модуля (переменные, процедуры, функции, пользовательские типы данных и пр.), не объявленные как Private, доступны другим модулям проекта, но не другим проектам или приложениям. В модуле выражение Option Private Module должно предшествовать всем процедурам.

3.1.1.6 Аргументы

Процедуры могут использовать аргументы, список которых (при необходимости с указанием типа), размещают в скобках после имени процедуры. Если вы сами пишете процедуру, то сами задаете аргументы. В процедурах событий набор аргументов зависит от события и не может быть изменен разработчиком:

```
Private Sub Form_MouseDown(Button As Integer, _
     Shift As Integer, X As Single, Y As Single)
End Sub
```

В общих процедурах количество и порядок используемых аргументов определяется разработчиком.

3.1.1.7 Вызов процедуры

Для вызова процедуры необходимо в командной строке записать ее имя, после чего выписать аргументы:

```
Имя_процедуры аргумент1, аргумент2,…
```

Второй способ заключается в использовании оператора CALL:

```
CALL Имя_процедуры (аргумент1, аргумент2, …)
```

При вызове процедуры модуля формы из другого модуля необходимо указывать ссылку на имя модуля формы, содержащего процедуру. Например, для вызова процедуры с именем Nameproc, находящейся в модуле формы Form1, оператор должен выглядеть так:

```
Call Form1.NameProc (аргумент1, ...аргументM)
```

3.1.2 Функции

Функция построена точно так же, как процедура. Однако есть одно отличие. Как и в математике, результатом работы функции является возвращаемое значение. Для определения функции используется ключевое слово Function. В конце процедуры вместо End Sub пишется End Function. В заголовке функции передаются аргументы. Visual Basic должен знать тип возвращаемого значения. За скобками объявления аргументов указывают тип данных возвращаемого значения. Синтаксис объявления функции имеет вид:

```
{[Private] | [Public] | [Static]} Function Имя_функции _
 (аргументы) As Тип_данных
        операторы
End Function
```

Имя функции используется в теле функции одновременно и как переменная:

```
Имя_функции = «выражение»
```

Это значит, что переменная с именем функции содержит возвращаемое значение, определяемое «выражение»'м.

Вызов функции несколько отличается от вызова процедуры:

```
Private Sub Commandl_Click()
Dim Tax As Currency
Tax = NDS(100, 0.15)
End Sub
```

В этой процедуре вызывается функция NDS. В качестве аргументов ей передаются значения 100 и 0.15. Возвращаемое значение присваивается переменной Tax.

Обратите внимание на различия при вызове функций и процедур. При вызове функций аргументы указываются в скобках. Возвращаемое значение либо является операндом какого-либо выражения, либо должно быть присвоено какой-нибудь переменной. В этом случае функция должна вызываться следующим образом:

```
Переменная = Имя_функции(Аргумент1, Аргумент2 ....)
```

В процедурах же аргументы не берутся в скобки. Нет также и возвращаемого значения:

```
Имя_процедуры Аргумент1, Аргумент2,...
```

Процедуры - это, как правило, маленькие подпрограммы, которые можно вызывать из других мест программы. Поэтому если в вашем приложении есть часто повторяющаяся задача, то следует создать процедуру, которая бы ее выполняла, а затем при необходимости просто вызывать ее - это может существенно сэкономить время написания программы. События всегда обрабатываются в процедурах. Функции ведут себя так же, как процедуры. Самое важное отличие - каждая функция всегда возвращает только одно значение. Кроме этого, процедуру Function можно вызывать так же, как процедуру Sub:

```
Call Square(Side)
```

```
Square Side
```

В этом случае Visual Basic игнорирует возвращаемое функцией значение.

3.1.3 Аргументы

В заголовке процедуры (функции) можно указывать тип данных для аргумента. В приведенном выше примере (см. Private Sub Form_MouseDown) аргументы Button и Shift имеют тип Integer, а X и Y - тип Single. В Visual Basic аргументы могут передаваться двумя способами: либо как ссылки (ByRef), либо как значение (ByVal).

Если аргумент передается как ссылка, то вызванная процедура получает физический адрес памяти передаваемой переменной. Различие между двумя видами передачи аргументов состоит в том, что при передаче аргумента как ссылки можно изменять значение этого аргумента. Так как вызываемая и вызывающая процедуры обращаются к одной и той же области памяти, значение переменной для них идентично. Для того чтобы передать аргумент как ссылку, следует перед аргументом указать ключевое слово ByRef. Однако поскольку по умолчанию аргументы в Visual Basic именно так и передаются, ByRef можно и опустить.

```
Private Sub Command1_Click()
A = 5: B = 5
SomeProcedure A, B
Print B 'результат: 25
End Sub
Sub SomeProcedure (ByRef First, Second)
Second = First * 5
End Sub
```

В данном примере переменные A и B передаются процедуре SomeProcedure по ссылке. В самой процедуре эти переменные фигурируют под именами First и Second соответственно. Значение переменной Second изменяется и затем может быть использовано в процедуре Command1_Click.

Таким образом, процедура может возвращать несколько значений. При вызове процедуры ей передаются аргументы, значения которых она может изменить. Если процедура не должна изменять аргументы, их следует передавать как значения.

Для передачи аргументов в качестве значений перед именем аргумента в заголовке процедуры следует указывать ключевое слово ByVal. В этом случае процедуре передается копия этого значения. При передаче аргументов в качестве значений ключевое слово ByVal должно указываться обязательно.

Ниже приведен пример объявления функции, в которой один из параметров передается по значению.

```
Private Sub Command1_Click()
A = 5: B = 5
```

```
SomeProcedure A, B
Print B 'результат: 5
End Sub
Sub SomeProcedure (ByRef First, ByVal Second)
Second = First * 5
End Sub
```

Для многих встроенных функций, операторов и методов Visual Basic обеспечивает возможность использования именованных аргументов для упрощения ввода их значений.

Обычно аргументы передают в последовательности, указанной в заголовке процедуры.

Использование именованных аргументов позволяет передавать часть или все аргументы в любом порядке. Для этого при вызове процедуры указывается имя аргумент и его значение, разделяемые специальным знаком — двоеточием со знаком равенства (:=), например: MyArgument := "someValue". Аргументы разграничиваются запятыми; порядок их следования значения не имеет:

```
SomeProcedure A, Second := 5
```

При вызове процедур можно передавать именованные и неименованные аргументы комбинированно. Однако следует помнить, что как только в списке передачи появляется именованный аргумент, все последующие аргументы должны передаваться также именованными.

Именованные аргументы поддерживают и многие другие функции Visual Basic, однако, именованные аргументы не поддерживаются методами объектов библиотеки Visual Basic. Они поддерживаются в Visual Basic for Applications (VBA) и методами доступа к данным библиотеки DAO.

Если при вызове процедуры указать не все аргументы, то последует сообщение об ошибке. Однако в процессе описания процедуры можно определить, что не все аргументы указываются при вызове. Такие аргументы называются необязательными.

Для того чтобы аргумент стал необязательным, перед именем аргумента ставится ключевое слово Optional. После первого необязательного аргумента все последующие должны быть также необязательными.

```
'Если аргументы функции объявлены следующим образом:
Function MyFunc(Sty As String, _
 Optional Arg1 As Integer = 5, Optional Arg2 = "Dol-ly")
Dim RetVal
'Функцию можно вызывать одним из способов:
RetVal = MyFunc("Hello", 2, "World")
```

В теле функции аргументы имеют значения: Sty = "Hello"; Arg1 = 2; Arg2 = "World".

```
RetVal = MyFunc("Test", , 5)
```

В теле функции аргументы имеют значения: Sty = "Test"; Arg1 = 5 (используется значение по умолчанию, см. объявление функции); Arg2 = 5

```
'Первый и второй аргументы используются как
'именованные, третий - опущен.
RetVal = MyFunc(Sty := "Hello ", Arg1 := 7)
```

В теле функции аргументы имеют значения: Sty = "Hello"; Arg1 = 7; Arg2 = "Dolly" (используется значение по умолчанию, см. объявление функции)

Функция IsMissing позволяет проверить, передан или нет аргумент типа Variant. Если некоторые необязательные аргументы не были переданы процедуре, они инициализируются значениями по умолчанию соответствующего типа данных.

Функции могут возвращать массивы. Таким образом, вы можете возвратить массив целых чисел без дополнительного преобразования его в строку, затем из строки. При создании функции, возвращающей массив, вы должны задать тип данных этого массива. Ниже приведен пример функции, возвращающей массив значений:

```
Public Function Имя_функции (аргументы) As Тип_данных ()
Dim V() As Тип_данных
' Объявление массива в теле функции
…
ReDim V(нижняя_граница To верхняя_граница)
'Переобъявление массива V()
…
V(k) = …
' Присвоение значений элементам массива V()
…
Имя_функции = V
' Присвоение массива V() выходному значению функции
End Function
```

Использование этой функции может иметь следующий вид (переменная – объявлена как динамический массив):

```
A = Имя_функции(аргументы)()
```

После этого присвоения, массив A() будет иметь размерность, определенную при выполнении функции для массива V(), т.е.

```
(нижняя_граница To верхняя_граница).
```

3.2 Порядок выполнения работы

3.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу prLab2_7_CONTROL.exe
3. Ответить на вопросы, предложенные программой.
4. Сохранить в файле протокол ответов. Протокол работы программы prLab2_7_CONTROL.exe должен быть помещен в отчет о лабораторной работе.
5. Запустить на выполнение программу «Visual Basic 6.0».

6. Разработать проект для выполнения заданий работы. Проект должен содержать несколько форм и модуль для обработки информации.

7. Запустить проект на выполнение. При необходимости исправьте ошибки.

8. Введите исходные данные каждого из заданий и получите результаты работы соответствующей части проекта. Перепишите результаты или сохраните их в файле.

9. Составьте отчет по лабораторной работе.

3.2.2 Порядок выполнения

1. Разработать проект для реализации алгоритма, приведенного в вариантах заданий.

2. Проект должен содержать несколько форм (для каждой части задания своя форма и общая форма, из которой производится вызов форм для ввода исходных данных каждой из частей задания) и программный модуль для обработки информации.

Для вызова соответствующей формы используйте процедуру обработки какого-либо из событий какого-либо элемента управления (Control), в которой должна быть строка:

```
ИмяФормы.Show
```

3. Формы должны содержать необходимые элементы управления для реализации алгоритма (Текстовые окна для ввода исходных данных и вывода результатов работы проекта, а также различные кнопки для выполнения тех или иных необходимых действий и/или операций).

4. Программный модуль должен содержать процедуры, выполняющие необходимые действия, обусловленные вариантом задания.

5. При старте проекта выбрать соответствующую часть задания, вызвать соответствующую форму проекта, ввести необходимые данные и провести их обработку.

6. Исходные данные и результаты их обработки для каждого из заданий необходимо занести в отчет.

3.2.3 Варианты заданий

1.2.3.1 Разработать процедуры для типовых алгоритмов

Общее задание для всех:

Для массивов случайных чисел, приведенных в варианте задания, записать типовые алгоритмы обработки информации. Для каждой части задания разработать форму и написать процедуру обработки события (студент должен самостоятельно выбрать элемент управления и событие), в которой:

a) считываются исходные данные из соответствующих текстовых окон в переменные(массивы);

b) эти данные затем передаются для обработки в соответствующую процедуру модуля обработки информации;

с) результаты обработки выводятся в другие текстовые окна на этой же форме.

Вариант №1
1. Найти минимум в одномерном массиве:
(3.1; 4.4; -7.6; 1.8)
2. Найти максимум в двумерном массиве:
(-5.3; -5.2; -5.6; -5.1
-0.7; -3.3; 6.7; 8.3
-8.5; 7.8; -4.6; 1
2.9; 9.1; 9.1; -8.1)
3. Упорядочить по убыванию одномерный массив:
(-4.2; 4.5; 8.8; -7.3)

Вариант №2
1. Найти максимум в одномерном массиве:
(-4.1; 3.8; 5.4; -2.2)
2. Найти минимум в двумерном массиве:
(7; -4; 7.1; -6.3
-7.7; 6; 5.1; 0.1
-3; 4.7; -4.6; -4.7
-0.5; -1.4; -5.2; -0.8)
3. Упорядочить по возрастанию одномерный массив:
(6.8; -3.1; -0.4; 1.9)

Вариант №3
1. Найти минимум в одномерном массиве:
(-3.1; -8; 7.2; -1.6)
2. Найти максимум в двумерном массиве:
(-8.1; -9; -1.8; 1.6
-0.4; 9.2; 3.5; 8.9
-8.2; -5.1; 5.1; -6
-7.7; -7.4; -0.3; -7)
3. Упорядочить по убыванию одномерный массив:
(-5.9; -8.7; -1; 6.6)

Вариант №4
1. Найти максимум в одномерном массиве:
(5.1; 3.1; 1.5; 2.4)
2. Найти минимум в двумерном массиве:
(-5.9; -5.1; 5.7; 6.9
0.1; -5.9; 0.9; 8
2.5; -4.9; -2.2; 3.2
5; 8.3; 0.8; 9.8)
3. Упорядочить по возрастанию одномерный массив:
(2.2; 0; 6.7; 0.4)

Вариант №5
1. Найти максимум в одномерном массиве:
(1.6; -9.5; 6.6; -5.9)
2. Найти минимум в двумерном массиве:
(4.6; 1.1; -2.2; 3.3
-1.5; 1.1; -1.8; 5.9
4.4; 7; -6; 2.7
6.6; 7.7; -5; -4.3)
3. Упорядочить по возрастанию одномерный массив:
(-2; 5.2; 5.9; 0)

Вариант №6
1. Найти минимум в одномерном массиве:
(-4.5; 0.4; 8.8; -7.9)
2. Найти максимум в двумерном массиве:
(3.5; 8.2; -3.9; -3.9
-0.5; -2.7; 3.5; -5.5
6.1; -3.2; -8; -6.3
0.9; 0.4; 1.3; 9)
3. Упорядочить по возрастанию одномерный массив:
(-4.4; 5.9; 4.7; -2.9)

Вариант №7
1. Найти максимум в одномерном массиве:
(5.2; 7.7; -5.4; -0.1)
2. Найти минимум в двумерном массиве:
(-1.9; -6.7; -7; -3.5
1.9; 6.4; 4.4; -5.4
-4.5; 7.3; 1.6; 7.7
6; -9.4; -7.5; -0.8)
3. Упорядочить по возрастанию одномерный массив:
(-6.4; 0.5; 1.3; 7.1)

Вариант №8
1. Найти минимум в одномерном массиве:
(7.5; -5.5; 1.8; 8.9)
2. Найти максимум в двумерном массиве:
(9.3; 1.2; -5.3; 0.5
3.7; -4.4; 4.5; -1.8
-5.6; -1.6; -2.9; 3.2
-3.7; 5.8; -0.3; 4.5)
3. Упорядочить по возрастанию одномерный массив:
(-6.7; 7.1; 0.3; 5)

Вариант №9
1. Найти максимум в одномерном массиве:
(-5.4; -1.5; 8.1; -6.1)
2. Найти минимум в двумерном массиве:
(1.6; -6.1; 9.2; -4
-8.7; -1.8; 2.8; 1.1
3; 5.5; -3.2; -3.9
-6.5; -6.3; -0.4; -8.8)
3. Упорядочить по возрастанию одномерный массив:
(-4.7; -9.2; 8.4; 1)

Вариант №10
1. Найти максимум в одномерном массиве:
(-7.2; 2.2; -2.1; -7)
2. Найти минимум в двумерном массиве:
(-8; -5.8; -8.3; -9.5
9.2; 1; -0.4; 6
8.9; 7.4; 8.7; -2.2
0.8; -5; 1.2; 0.6)
3. Упорядочить по убыванию одномерный массив:
(9.7; -2.3; 2.7; -5.2)

Вариант №11
1. Найти максимум в одномерном массиве:
(4.7; -6.8; -8.3; 8.7)
2. Найти минимум в двумерном массиве:
(9.6; -8.2; -5.7; -6.3
-6.7; 4.8; 7.5; 0.7
-3.7; 8.5; 3.7; -4
4.7; 5.4; -9.1; 5.9)
3. Упорядочить по возрастанию одномерный массив:
(6.1; -2; 4.9; 5)

Вариант №12
1. Найти минимум в одномерном массиве:
(0.5; 6.8; -7; 3)
2. Найти максимум в двумерном массиве:
(6; 6.6; -0.8; 2.8
-3.3; -5.5; 1.7; 1.5
6; -4.8; -1.5; 0.2
8.3; -2.6; 9.6; -0.8)
3. Упорядочить по убыванию одномерный массив:
(-8; 9.2; 3.2; -2.7)

Вариант №13
1. Найти максимум в одномерном массиве:
(9.1; -0.5; 6.3; -9.2)
2. Найти минимум в двумерном массиве:
(2.4; -2.2; 4.1; -3.2
-2.3; -2.3; -6.9; 8.9
8.7; 6; 7.4; -8
6.5; -3; 6.7; 3.2)
3. Упорядочить по возрастанию одномерный массив:
(2.4; 4.6; 4.5; 4.5)

Вариант №14
1. Найти максимум в одномерном массиве:
(7.4; 5.4; 3.1; 3.5)
2. Найти минимум в двумерном массиве:
(-0.6; -1.7; -3.1; -6.1
7.8; -0.6; -1.5; -0.9
7.4; 0.3; 0.8; 0.6
1; 8.8; -7.5; -0.3)
3. Упорядочить по возрастанию одномерный массив:
(5.4; 5.8; -1.7; 9.4)

Вариант №15
1. Найти минимум в одномерном массиве:
(1.3; 7.6; -5.4; 5.1)
2. Найти максимум в двумерном массиве:
(-4.1; 8.8; 8.9; -9.7
5.5; -9.7; -1.7; 6.6
8; -7.2; 6; 2.8
6.1; 4.2; -0.4; 3.1)
3. Упорядочить по возрастанию одномерный массив:
(6.5; -7.4; -5.2; -9.3)

1.2.3.2 Разработать процедуры для выполнения заданий

Общее задание для всех:

При выполнении следующих заданий используйте при необходимости процедуры, разработанные в первой части лабораторной работы.

Вариант № 1. Использовать для вычисления площади треугольника по его сторонам по формулу Герона $S = \sqrt{p(p-a)(p-b)(p-c)}$, где *a, b, c* – длины сторон треугольника, *p* – его полупериметр. Перед вычислением площади определить, составляют ли отрезки a, b, c треугольник (из трех отрезков можно построить треугольник, если сумма любых двух из них больше третьего). Определить площадь S_i для a_i, b_i, c_i – элементов заданных массивов A = (3, 1, 5, 4.1), B = (0.8, 1.7, 3.4, 5.2), C = (2.1, 3.5, 6.2, 10), i = 1,…,4.

Вариант № 2. Вычислить $Z = 1 - S/2$, где $S = \sum_{n=0}^{\infty} 2^n (a_n^2 - b_n^2)$, $a_n = ½(a_{n-1} + b_{n-1})$, $b_n = \sqrt{a_{n-1} \cdot b_{n-1}}$. Значения a_0 и b_0 заданы. Вычисления закончить при выполнении условия $a_n^2 - b_n^2 < Q$. Дано: $a_0 = 3$, $b_0 = 2$, $Q = 10^{-5}$.

Вариант № 3. Дана точка $M(x_0,y_0)$ и массив точек $A(x_i,y_i)$, где $x_i = m \cdot \mathbf{cos}(i)$, $y_i = k \cdot \mathbf{sin}(i)$, $i = 1,\ldots,n$. Найти точку A_i, ближайшую к точке M. Дано: $n = 20$, $m = 4.5$, $k = –1.5$, $x_0 = –0.11$; $y_0 = 0.34$.

Вариант № 4. Вычислить коэффициент корреляции, выражающий степень зависимости между упорядоченными наборами чисел {x}и {y} по формуле $\rho = \frac{S_{11} \cdot S_{22} - S_{21} \cdot S_{12}}{\sqrt{(S_{11} \cdot S_{31} - S_{21}^2)(S_{11} \cdot S_{13} - S_{12}^2)}}$; $S_{kj} = \sum_{i=1}^{n} x_i^k y_i^j$; $k = 1,2,3$; $j = 1,2,3$. Дано: n = 7, x = (1.1, 1.3, 1.5, 1.8, 2.7, 3, 4.2, 5.1), y = (–0.2, –0.1, 1.5, 2.7, 3, 4.2, 5.1).

Вариант № 5. Вычислить $S = \sum_{k=2}^{n+1} a_k$, где $a_k = \frac{1}{(2k)!} \sum_{i=1}^{k-1} x_i^2$, k = 2,…, n+1. Дано: n = 6, x = (0.1, 0.5, 0.6, 1.5, 2.3, 2.8).

Вариант № 6. Составить процедуру нахождения вектора P(n), элементы которого равны производным функции Y(x) в точках x_1, x_2, …, x_n, вычисленным по формулам численного дифференцирования: $P_1 = y'(x)_{x_1} = y'_1 = \frac{-3y_1 + 4y_2 - y_3}{2(x_2 - x_1)}$ $P_n = y'(x)_{x_n} = y'_n = \frac{y_{n-2} - 4y_{n-1} + y_n}{2(x_n - x_{n-1})}$;

$P_i = y'(x)_{x_i} = y'_i = \frac{y_{i+1} - y_{i-1}}{2(x_{i+1} - x_i)}$; i = 2, …, n – 1. Функция Y(x) задана табличо, т.е. значениям аргумента x_1, x_2, …, x_n соответствуют значения функции y_1, y_2, …, y_n. Исходные данные: n = 6, x = (0, 0.5, 1, 1.5, 2, 2.5), y = (0, 0.125, 1, 3.375, 8, 15.625).

Вариант № 7. Вычислить угол φ между векторами $A = \{a_i\}$ и $B = \{b_i\}$, i = 1, …, n по формуле $\cos\varphi = \sum_{i=1}^{n} a_i b_i \Big/ \sqrt{\sum_{i=1}^{n} a_i^2 \sum_{i=1}^{n} b_i^2}$. Дано: n = 8, B = {–24.6, 23, 80, –22, –1, –13.5, –42, –22}, A = {4, 3.2, 0.5, 0.4, –0.2, 11, –6, 31}.

Вариант № 8. Вычислить $Y = f(0.523) + f(1.1) - 2 \cdot f^2(-0.23)$. Значение $f(x)$ вычислять с точностью до $\varepsilon = 10^{-5}$ по формуле $f(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots + (-1)^{n+1} \frac{x^{2n+1}}{(2n+1)!}$.

Вариант № 9. Составить процедуру для вычисления вектора Y(n), где $Y_i = 0.26 \ln(1 + \sqrt{i})$; $i = 1, \ldots, n$, и присвоения переменной Z значения TRUE, если $S = (X_1 Y_n)^2 + (X_2 Y_{n-1})^2 + \ldots + (X_n Y_1)^2$ принадлежит отрезку [0, 1] и значение FALSE – в противном случае. Дано: n = 10, X = (0.6, 1.8, 1.2, 0.4, –0.8, 0.6, 3.1, –0.4, –0.2, 1.3).

Вариант № 10. В массиве $A = \{a_i\}$, $i = 1, \ldots, n$ элемент a_k вставить на место a_m а элементы с a_m-го по a_{k-1}-й сдвинуть на одну позицию вправо. Дано: A = (1.1, 2.5, 3.4, –0.8, 3.7, 4.5), n = 6, k = 4, m = 1.

Вариант № 11. Вычислить расстояние между векторами X = (–1.5, –0.1, 1.3, 2.9, 3.5, 4.8) и Y = (–2.7, –2.1, –0.5, 4.1, 5.6, 8.5), если известно, что расстояние между векторами A и B размером n определяется по формуле $d = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$.

Вариант № 12. Вычислить вектор $Z = \{z_i\}$, являющийся произведением матрицы $A = \{a_{ij}\}$ на вектор $X = \{x_j\}$, элементы которого вычисляются по формуле $x_j = \begin{cases} k \cdot \sin j, & j \le k, \\ \cos j, & k < j \le n, \end{cases}$ $j = 1, \ldots, n$. Каждую компоненту вектора Z определяют по формуле $z_i = \sum_{j=1}^{n} a_{ij} x_i$, $i = 1, \ldots, n$. Дано: $k = 2$, $n = 3$, $A = \begin{bmatrix} -6.2 & 3.5 & 4.1 \\ 0.8 & 1.5 & 2.1 \\ 1 & 2 & 3 \end{bmatrix}$.

Вариант № 13. Вычислить дисперсию массива случайных величин $D = \frac{1}{n} \sum_{i=1}^{n} (x_i - M)^2$, где $M = \frac{1}{n} \sum_{i=1}^{n} x_i$ – математическое ожидание, при n = 8, x = (–2.5, 0.9, –5.2, 3.4, 3.3, 6.1, –1.9, 5.3).

Вариант № 14. Выделить из вектора P(n) вектор R(m) ($m \le n$) по правилу: компонента вектора P является компонентой вектора R, если квадратное уравнение $x^2 - 2P_i x + q = 0$ имеет вещественные корни. Дано: n = 7, q = 4, P = (2.6, 3.3, 1.8, 5.6, 0.5, –2.8, –4.2).

Вариант № 15. Вычислить $J = h\sum_{i=1}^{n} y_i$, где n = 20; x_0 = 0; x_n = 5; h = $(x_n - x_0)/n$; $y(x) = e^{-ax}\cos^2 x$; a = –0.25; y_i – значение y(x) в точке $x_i = x_{i-1} + h$, i = 1,…,n.

1.2.3.3 Разработать процедуры для выполнения заданий

Общее задание для всех:

При выполнении следующих заданий используйте процедуры, разработанные в первой части лабораторной работы.

Вариант № 1. Построить матрицу A(n, n) с компонентами $a_{ij} = \sin(i + j)x$, i, j = 1, …, n, (подпрограмма 1), а затем преобразовать эту матрицу путем перестановки в каждой строке наибольшего по абсолютной величине элемента (первый, если их несколько) с диагональным (подпрограмма 2). Вывести исходную и преобразованную матрицы. Дано: n = 3, x = 3.5.

Вариант № 2. Для массива A(n, n) определить S – номер столбца, содержащего максимальный элемент массива (подпрограмма 1) и R – наименьшее из положительных отношений a_{in}/a_{is} (подпрограмма 2), i = 1, …, n. $A = \begin{bmatrix} -5 & 6 & 3 & -1 \\ 4 & 11 & 15 & 2 \\ 17 & 21 & 12 & -1 \\ -1 & 2 & 5 & -3 \end{bmatrix}$, n = 4.

Вариант № 3. Составить процедуру вычисления матрицы $C = \{c_{ij}\}$ по матрицам $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$, каждый элемент которой определяется по формуле $c_{ij} = a_{ij} - z \cdot b_{ij}$; i, j = 1, …, n (подпрограмма 1). Упорядочить по возрастанию абсолютных величин столбцы массива C (подпрограмма 2). Дано n = 3, $z = 1.15 \cdot 10^{-2}$, $A = \begin{bmatrix} 1 & 2 & 3 \\ 0.5 & 1 & 2 \\ 4 & 1.5 & 3.7 \end{bmatrix}$, $B = \begin{bmatrix} 7 & 9 & 15 \\ 8 & 6 & 2.5 \\ 4 & 4.7 & 17 \end{bmatrix}$.

Вариант № 4. Составить процедуру вычисления матрицы C(m, q), являющейся произведением матриц A(m, n) и B(n, q) (подпрограмма 1). Элементы матрицы C вычисляются так: $C_{kj} = \sum_{i=1}^{n} A_{ki} \cdot B_{ij}$, k = 1, …, m, j = 1, …, q. Заменить все элементы матрицы C ниже главной диагонали на 1 (подпрограмма 2). Исходные данные: m = 3, n = 4, q = 3, $A = \begin{bmatrix} 3 & 0 & 1 & 2 \\ 6.1 & 0 & 2 & 4 \\ 9 & 0 & 3 & 1.6 \end{bmatrix}$, $B = \begin{bmatrix} 1 & -1 & -2 \\ 3 & 2.1 & 4 \\ 2 & 4 & 1 \\ -2 & 0 & 3 \end{bmatrix}$.

Вариант № 5. Вычислить вектор D(n), равный произведению матрицы A(n, m) на вектор B(m), причем $D_i = \sum_{k=1}^{m} A_{ik} B_k$, i = 1, …, n. (подпрограмма 1). Элементы массива D от первого до максимального заменить их индексами (подпрограмма 2). Исходные данные: n = 3, m = 4, $B = \begin{bmatrix} 1 \\ 3 \\ 2 \\ -1.1 \end{bmatrix}$, $A = \begin{bmatrix} 3 & 8 & 5 & 1.1 \\ 3 & -1 & 3.1 & 1.5 \\ -6 & 0.5 & 3 & 1 \end{bmatrix}$.

Вариант № 6. Создать процедуру вычисления матрицы C(n, n), являющейся суммой матриц A(n, n) и B(n, n). Элементы C определяются по формуле: $C_{ij} = A_{ij} + B_{ij}$; i, j = 1, …, n (подпрограмма 1). Матрица A задана, а элементы матрицы B находятся по формуле $B_{ij} = \begin{cases} A_{ij}, & A_{ij} \geq 0; \\ 1, & A_{ij} < 0. \end{cases}$ Определить номер столбца матрицы C, с максимальной суммой элементов. Дано: n = 4, $A = \begin{bmatrix} -1 & 2 & 3 & -5 \\ 0 & 4 & -1.1 & 2 \\ -6 & 8 & 12 & 1.1 \\ -7 & 3.1 & 2.5 & -10 \end{bmatrix}$.

Вариант № 7. Составить процедуру для вычисления вектора B(m), равного P-й строке матрицы A(n, m) (подпрограмма 1) и вектора C(n), равного Q-му столбцу матрицы A (подпрограмма 2). Дано: n = 3, m = 4, P = 2, Q = 2, $A = \begin{bmatrix} 3 & 5 & -7 & 1 \\ 2 & 8 & 10 & 15 \\ -6 & -1 & 9 & 4.1 \end{bmatrix}$.

Вариант № 8. Компоненты заданного вектора $A = \{a_i\}$, i = 1, …, n расположить в порядке убывания (подпрограмма 1). Создать вектор $B = \{b_i\}$, i = 1, …, n+1 из компонентов вектора A и переменной C, сохранив порядок убывания (подпрограмма 2). Дано: n = 9, A = {9.6, –7.4, 4.1, 4.0, 3.8, –2.5, 2.4, 2.2, 1.7}, C = 2.0.

Вариант № 9. Элементы вектора A(n) расположены в порядке возрастания по абсолютной величине. Составить процедуру получения вектора B(n) из вектора A(n) путем удаления K-го элемента и вставки числа C без нарушения возрастания элементов по абсолютному значению (подпрограмма 1). Определить сумму элементов массива B, находящихся до минимального его элемента (включая сам минимальный элемент) (подпрограмма 2). Дано: K = 6, C = –1.2, n = 10, A = (0.1, –0.4, 0.8, 1.5, –1.8, 4.6, 5.2, –8.9, 9.1, 12.6).

Вариант № 10. Значения компонент вектора X(n) вычислить по формуле: $X_i = 1.2 \cdot \sin(0.1 \cdot i) + 0.4 \cdot \cos(i - 1)$; i = 1, …, 10 (подпрограмма 1). Составить процедуру (подпрограмма 2) для 1) нахождения наибольшей из компонент X и ее номера k, 2) умножения на X_k всех компонент вектора X с четными номерами, предшествующих наибольшей компоненте. Остальные компоненты оставить без изменения.

Вариант № 11. Если аппроксимировать таблично заданную массивами $\{x_i\}$ и $\{y_i\}$, i = 1, …, n функцию методом наименьших квадратов полиномом m-й степени $P_m(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_m \cdot x^m$, то для оценки погрешности аппроксимации используется сумма квадратов разностей $S_m(a_0, a_1, \ldots, a_m) = \sum_{i=0}^{n} \left[P_m(x_i) - y_i \right]^2$ (подпрограмма 1). Дано: функция x = (–3, –1, 0, 1, 2, 3, 4), y = (2.9, 1.0, –0.2, –1.5, –0.4, 0.5, 2.0), n = 6 аппроксимирована полиномами первой $P_1(x) = a_0 + a_1 \cdot x$ и второй степени $P_2(x) = b_0 + b_1 \cdot x + b_2 \cdot x^2$: a = (0.766, –0.177); b = (–0.458, -0.454, 0.256). Составить подпрограмму 2 для нахождения $P_m(x_i)$. Оценить погрешности аппроксимации функции полиномами первой и второй степени.

Вариант № 12. Методом итераций найти корень ξ уравнения $f(x) = 0$ на интервале изоляции корня $[a, b]$ (подпрограмма 1). Метод заключается в приведении уравнения $f(x) = 0$ к виду $x = \varphi(x)$ и последовательном приближении к корню: $x_{i+1} = \varphi(x_i)$ до тех пор, пока не выполнится условие $|x_i - x_{i+1}| < \varepsilon$. За корень принимается $\xi = x_{i+1}$. Выбираем начальное приближение: $x_0 = (a + b)/2$. Нахождение значения функции $\varphi(x)$ оформить в виде подпрограммы 2. Дано: $\varepsilon = 10^{-3}$, $f(x) = x^3 + 4x - 1$.

Вариант № 13. Определить приближенно значение функции Y, заданной таблично (т.е. массивами точек $\{x_k\}$ и $\{y_k\}$, k = 0, 1, …, n), в точках, заданных массивом $\{X_i\}$, i = 1, …, m, используя для этого интерполяционную формулу Лагранжа $P_n(x_i) = \sum_{k=0}^{n} \omega_k(x_i) \cdot f(x_k)$; i = 1, …, m, где многочлен $\omega_k(x) = \dfrac{(x - x_0)(x - x_1)\ldots(x - x_{k-1})(x - x_{k+1})\ldots(x - x_n)}{(x_k - x_0)(x_k - x_1)\ldots(x_k - x_{k-1})(x_k - x_{k+1})\ldots(x_k - x_n)}$.
Построение формулы Лагранжа оформить в виде подпрограммы 1, а массив искомый Y(m) получить в подпрограмме 2. Дано: x = (1.1, 1.5, 2.0, 2.6), y = (0.0953, 0.4055, 0.6931, 0.9555), n = 3, X = (1.1, 1.3, 1.5, 1.75, 2.0, 2.3, 2.6), m = 7.

Вариант № 14. Составить процедуру для вычисления матрицы B(n, n) (подпрограмма 1), равной произведению числа λ на матрицу A(n, n): $B_{ij} = \lambda \cdot A_{ij}$; i, j = 1, …, n, где λ – максимальный элемент заданного вектора Z(m). Найти произведение элементов массива B, стоящих выше главной

диагонали (подпрограмма 2). Дано: n = 3, m = 5, Z = (11, –5, –1, 0.2, –3), $A = \begin{bmatrix} 2 & -3.1 & 5.6 \\ 1.2 & 4 & 9 \\ 5.5 & 1.2 & -3.8 \end{bmatrix}$.

Вариант № 15. Преобразовать вектор A(n), выстроив его элементы в порядке возрастания абсолютных величин его компонент (подпрограмма 1) и определить среднее арифметическое S значений элементов вектора A (подпрограмма 2). Дано: n = 10, A = (–1.5, 0, 0.1, 2.1, 1.8, –0.25, 3, 1.7, 5.1, 4.5).

3.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы
2. Основные сведения о теме работы
3. Алгоритмы соответствующих частей разработанного проекта.
4. Перечень форм и элементов управления на них.
5. Процедуры обработки данных.
6. Исходные данные и результаты их обработки для каждого из заданий.

3.3 Контрольные вопросы

1. Описать структуру программного кода проекта.
2. Описать отличия процедуры обработки события от процедуры общего назначения.
3. Дать описание отличия имени процедуры обработки события от процедуры общего назначения.
4. Что такое закрытые процедуры?
5. Дать описание назначения опции Option Private Module.
6. Что такое аргументы процедуры или функции?
7. В каких процедурах набор аргументов не может быть изменен?
8. Описать способы вызова процедур.
9. Можно ли вызвать на выполнение процедуру обработки события, не вызывая самого события?
10. В чем заключается принципиальное отличие межу процедурами и функциями?
11. Что такое возвращаемое значение?
12. Как производится вызов функции?
13. Каковы способы передачи аргументов процедурам и функциям?
14. В чем отличие в передаче аргументов с опциями ByVal и ByRef.
15. Что такое именованные аргументы?
16. Что означает ключевое слово Optional перед именем аргумента при описании процедуры?

СПИСОК ЛИТЕРАТУРЫ

1. Сайлер Б., Споттс Дж. Использование Visual Basic 6. Классическое издание. - М.: Вильямс, 2007. – 832 с.

2. Сафронов И. Visual Basic в задачах и примерах. - СПб.: БХВ-Петербург, 2008. – 400 с.

3. Эпплман Д. Win32 API и Visual Basic. Для профессионалов. - СПб.: Питер, 2001. – 1120 с.

4. Сергеев В. Visual Basic 6.0. Наиболее полное руководство для профессиональной работы в среде Visual Basic 6.0. – СПб.: БХВ-Петербург, 2004. – 992 с.

5. Balena F. Programming Microsoft Visual Basic 6.0. – USA: Microsoft Press, 1999. – 1312 p.

6. Halvorson M. Microsoft Visual Basic Professional 6.0 Step by Step. – USA: Microsoft Press, 1998. – 672 p.

7. Holzner S. Visual Basic 6 Black Book: The Only Book You'll Need on Visual Basic. – USA: Coriolis Group Books, 1998. – 700 p.

8. Microsoft Visual Basic 6. Шаг за шагом: Практ. пособ. / Пер. с англ. – М.: Изд. ЭКОМ., Изд. 2-е, исправленное, 2003. – 432 с.

СОДЕРЖАНИЕ

Учебное издание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам
«Основы программирования на Visual Basic»
из раздела «Программирование в среде Visual Basic. Интегрированная среда разработки»
дисциплины «Информатика»
для студентов направления подготовки
6.050801 «Микро- и наноэлектроника»

Составители: ШКАЛЕТО Владимир Иванович
ЗАЙЦЕВ Роман Валентинович

Ответственный за выпуск М.В. Кириченко

Редактор

План 2013 р.

Підписано до друку _____. Формат 60×84 1/16. Папір друк. №2.

Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 2,5.

Обл.-вид. 3,0. Тираж 50 прим. Зам. № _____. Ціна договірна

Видавничий центр НТУ “ХПІ”. 61002, Харків, вул. Фрунзе, 21.
Свідоцтво про державну реєстрацію ДК № 116 від 10.07.2000 р.

Друкарня НТУ “ХПІ”. 61002, Харків, вул. Фрунзе, 21.