## Тема 4.4

## Программирование алгоритмов разветвляющихся структур

### 4.4.1.Программирование на основе событий

Программные проекты, которые были рассмотрены в предыдущих темах, перед тем как выполнить какие-либо вычисления (действия), ожидали реакции пользователя приложения, и только затем обра­батывали эти действия. Эта методология известна как ***программирование на основе событий****.* При создании программного проекта, со­здается группа объектов, которые знают, как реагировать на попытки пользователя приложения взаимодействовать с ними, а затем команды программного проекта обрабатывают ввод с помощью процедур событий, связанных с этими объектами.

События также могут поступать из самого компьютера. Напри­мер, программа может получать уведомление о том, что получено сооб­щение электронной почты, или о том, что по системным часам прошел некоторый период времени. Эти события генерирует компьютер, а не пользователь. Независимо от того, как генерируется событие, VBреагирует на него, вызывая процедуру события, которая ассоциируется с объектом, распознающим это событие. До сих пор мы, в основном, имели дело с событиями **Click***.* Одна­ко объекты VB также могут реагировать на множество других событий.

Привязка к событиям в VB означает, что большинство вы­числений, производимых программой, выполняется в процедурах собы­тий.

Каждый объект в VB имеет предопределенный набор событий, на кото­рые он может реагировать. Список этих событий можно увидеть, если в ***Имя класса*** в верхней части окна **Редактора кода** выделить имя объекта, а за­тем щелкнуть на стрелке ***Имя метода.*** Необходимо отметить, что события визуально иден­тифицируются по пиктограмме в форме молнии, добавляемой VS к их именам. Можно написать процедуру события для любого из этих событий и, если это событие в программе произойдет, VBвыполнит процедуру со­бытия, ассоциированную с ним. Например, объект списка поддерживает более 60 событий, в том числе **Click, DoubleClick, DragDrop, DragOver, GotFocus*.*** Обычно в приложениях не потребуется писать программный код более чем для трех или четырех из этих событий, но при создании элементов ин­терфейса следует знать, что есть такой большой выбор.

Однако, наряду с внешними событиями по отношению к программному коду процедур пользователя, необходимо иметь программные средства для сравнения переменных, свойств и значений, а также те средства, которые в соот­ветствии с результатом этого сравнения могут выполнить один или несколько операторов. Такими средствами в VBявляются операторы языка программирования – оператор ***условного перехода.***

### 4.4.2. Средства программирования разветвляющихся структур

***Разветвляющимися алгоритмическими струк­турами*** называются такие структуры, в которых порядок выполнения блоков операторов определяется значе­ниями логических выражений. Разветвляющийся алгоритм может состоять из нескольких ветвей, каждая из которых может содержать любую, сколь угодно сложную, алгоритми­ческую структуру. В процессе работы разветвляющегося алгоритма в первую очередь вычисляются ***логиче­ские выражения* L** (частным случаем логического выражения является ***логическая переменная или константа***). Если результат вычисления логического выражения принимает значение **True** (Истина), то выполняется часть алгоритма, расположенная по ветви «Да», если значение – **False** (Ложь), то – по ветви Нет».

В VBопределен логический тип данных **Boolean.** Например, описание переменной **L:**

|  |
| --- |
| **Dim L As Boolean** |

Такая переменная **L** может иметь одно из двух значений: **True** или **False.** Причем при преобразовании числовых типов в логический тип значение 0 преобразуется в **False,** а все другие значения в 1**.** Например,

|  |
| --- |
| **Dim f As Double = 6.8**  **L = CBool(f)** *'в результате L = True*  **f=0**  **L = CBool(f)** *'в результате L = False* |

***Логическое выражение*** может быть ***простым*** или ***сложным***.

***Простое логическое выражение*** – это два арифметических или строковых выражения, связанных операцией отношения (***сравнения***).

***Операции отношения (сравнения)*** приведены в табл. 4.4.2-1.

Таблица 4.4.2-1

|  |  |
| --- | --- |
| **Операции отношения** | **Значение** |
| **=** | *Равно* |
| **<** | *Меньше, чем* |
| **>** | *Больше, чем* |
| **<=** | *Меньше или равно* |
| **>=** | *Больше или равно* |
| **<>** | *Не равно* |

В таблице 4.4.2-2 приведены простые логических выражения.

Таблица 4.4.2-2

|  |  |
| --- | --- |
| **Логическое выражение** | **Результат** |
| **5 < 10** | **True**, так как **5** меньше **10** |
| **5 > 10** | **False,** так как **5** не больше **10** |
| **7 <> 3** | **True**, так как **7** не равно **3** |
| **X > 45** | **True**, если **X** больше **45**; в противном случае **False** |
| **TextBox1.Text="Stop"** | **True,** если в первом текстовом поле содержится слово **"Stop"**; в противном случае **False** |
| **5 - 1 > 1 + 2** | **True,** так как первой вычисляется разность **5-1 (4)**,  затем вычисляется сумма **1+2 (3)** и наконец, проверяется условие **4 > 3**, т.е. **4** больше **3** |
| **X >= y + z** | **True,** если **X** больше или равно сумме значений переменных **y** и **z**; в противном случае **False** |

В отличие от арифметических операций, все ***операции сравнения имеют одинаковый приоритет***, т.е., при наличии в логическом выражении нескольких операций сравнения проверяться условия будут в порядке их записи слева направо.

Кроме того, приоритет операций отношения ниже, чем приоритет арифметических операций, поэтому операции сравнения в табл. 4.4.2-3 выполняются после арифметических операций.

Например, выражение **Not 7 < 9** всегда прини­мает значение **False**, а   
**4 > 3 And 7 < 9** принимает значение **True**.

В табл.4.4.2-4 приведено описание логических операций (таблица истинности). Здесь **T** (**True)** «Истина», а **F** (**False**) – «Ложь».

Разветвляющиеся алгоритмические структуры (рис. В.3-5–В.3-8) в языках программирования реализуются ***операторами условного перехода.***

В VBдля реализации разветвляющихся алгоритмических структур используются три типа операторов: ***однострочный* If**, ***блочный***  **If** и ***оператор выбора* Select Case**.

Таблица 4.4.2-3

|  |  |
| --- | --- |
| **Логические**  **операции** | **Значение** |
| **Not** | Преобразует значение **True** в **False** и наоборот *–* **False** в **True** |
| **And** | Для того чтобы сложное условие было истинным, истинными должны быть все входящие в его состав условия |
| **Or** | Сложное условие истинно, если истинно хотя бы одно из составляющих его условий |
| **Xor** | Для того чтобы сложное условие было истинным, истинным должно быть только одно из его составляющих |

Таблица 4.4.2-4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Значения  операндов** | | **Not x** | **X And y** | **X Or y** | **x Xor y** |
| X | Y |
| T | T | F | T | T | F |
| T | F | F | F | T | T |
| F | T | T | F | T | T |
| F | F | T | F | F | F |

***Однострочный оператор* If** имеет следующий формат:

|  |
| --- |
| **If L Then** *оператор(ы****)***[**Else** *оператор(ы)*] |

 Порядок выполнения конструкции следующий: если значение логического выражения **L** равно **True**, то выполняется оператор (или операторы, отделенные друг от друга двоеточием), стоящий после **Then**, в противном случае выполняются операторы, стоящие после **Else**. В операторе **If** конструкция **Else** может от­сутствовать (в формате она взята в квадрат­ные скобки), такое разветвление называется ***усеченным***.

Особенностью ***однострочного оператора* If** является то, что он должен располагаться в одной строке программы.

Приведем примеры записи однострочного оператора **If**:

|  |
| --- |
| **If x > 0 Then y = Sqrt(x) Else y = 1**  **If d=>10 Then epsilon = 0.001**  **If lambda < 1 Or beta = 3 Then lambda = beta : x = 0**  **If x < 10 Then y = 1 Else If x < 100 Then y = 2 Else y = 4** |

Если запись оператора не умещается в пределах экрана, а также при программировании вложенных разветвлений, используется ***многострочный оператор*** (***блочный оператор***) **If**. Он имеет следующий формат:

|  |
| --- |
| **If  L1 Then   [** *блок\_операторов\_1* **] [ElseIf L2 Then  [** *блок\_операторов\_2* **]  . . . Else**  **[***блок\_операторов\_n***] ] End If** |

***Порядок и правила выполнения конструкции многострочого оператора* If** следующий:

* если значение **L1** равно **True**, то выполняется блок\_операторов\_1 и далее управление передается оператору, следующему после **End If**.
* если значение **L1** равно **False**, то проверяется значение следующего выражения и так далее.
* после выполнения любого блока оператора **If** управление передается оператору, следующему после **End If**.
* оператор If может содержать несколько конструкций типа ElseIf.
* блок операторов, следующий за словом Else, выполняется в том случае, если последовательные проверки всех выражений выше имеют значение **False**.
* конструкции ElseIf и Else могут отсут­ствовать, что и отмечается в синтаксисе оператора квадратными скобками.
* в многострочном операторе If слово Then должно быть последним в строке, а слово Else – единственным.
* слово ElseIf записывается слитно, а слова **End If** – через пробел.

Примеры блочных операторов **If:**

|  |  |
| --- | --- |
| **If x>0 Then**  **y =Sqrt(x)**  **z=1+y**  **p=4**  **End If** | **If x < 10 Then**  **y=1**  **Else**  **y=2**  **z=4**  **End If** |

Считается хорошим тоном выделять структуру операторов ***отступами***, как это сделано в приведенных выше примерах. Это сделает программы «читабельными», т.е. в них будет легче разо­браться.

**Пример 4.4.2-1. Написать процедуры ввода/вывода, которые будут использоваться в алгоритмах разветвляющихся структур.**

Требуемыми процедурами являются процедуры **vvodDbl1()** и  **ViodDbl1()** ввода и вывода данных типа **Double** в текстовые поля, процедуры **vvodSng3()** и **vivodSng3()** ввода и вывода данных типа **Single** в текстовые поля и процедуры **vvodInt4( )** и **vivodInt4( )** ввода и вывода данных типа **Integer** в текстовые поля.

Процедуры **vvodDbl1(), vivodDbl1(), vvodSng3()** и **vivodSng3()** приведены в Теме 4.4.3, а процедуры **vvodInt4()** и **vivodInt4()** представлены на рис. 4.4.2-1 и 4.4.2-2.

|  |
| --- |
| *'процедура-Function ввода данных типа Integer*  *'из текстового поля TextBox*  **Function vvodInt4(ByVal T As TextBox) As Integer**  **Return CInt(T.Text)**  **End Function** |

*Рис. 4.4.2-1. Процедура ввода данных типа* **Integer** *из поля* **TextBox**

|  |
| --- |
| ***'Процедура-Sub вывода данных типа Integer в TextBox***  **Sub vivodInt4(ByVal Z As Integer, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub** |

*Рис. 4.4.2-2. Процедура вывода данных типа* **Integer в TextBox**

**Пример 4.4.2-2. Дополнить проект Пример 3.4-3 из Темы 3, предусмотрев в нем проверку исходных данных.**

По результату проверки исходных данных программа должна выдавать предупреждающее сообщение, если вводятся значения сторон треугольника, при которых треугольник не существует.

|  |
| --- |
| **Sub PsPr(ByVal A As Single, ByVal B As Single, ByVal C As Single)**  **Dim P, S As Single**  **If (A + B > C) And (B + C > A) And (C + A > B) Then**  **PS(A, B, C, P, S):vivodSng3(P, TextBox4):vivodSng3(S,TextBox5)**  **Else**  **MsgBox("Ошибка!" & vbNewLine &, \_**  **"Сумма двух сторон треугольника" \_**  **&" должна быть больше третьей.")**  **TextBox1.Text = ""** *'Очистка текстовых поле*й  **TextBox2.Text = ""** *'Очистка текстовых полей*  **TextBox3.Text = ""** *'Очистка текстовых полей*  **TextBox1.Focus()** *'В фокусе оказывается*  *'текстовое поле для(ввода)значения стороны А*  *'(в нем будет мигать курсор),*  *'после чего можно повторить ввод*  **End If**  **End Sub** |

*Рис. 4.4.2-3. Процедура* **PsPR()****Проекта 3.4-3** *проверкой исходных данных*

Для того чтобы можно было построить треугольник, необходимо и достаточно выполнения следующих неравенств (**A, B,** **С** – длины сторон треугольника, **&** – операция логического умножения):   
**(А + В > С) & (А + C > В) & (В + C > А)**.

Открыть проект **Пример 3.4-3** и внесите изменения в процедуру,   
**Sub PSPr()** (рис. 4.4.2-3). В окне функции **MsgBox()** текст сообщения выводится в две строки. Обратить внимание, что для перевода на новую строку используется текстовая константа **vbNewLine**.

***Рассмотрим конструкцию*** **Select Case.**

Конструкция **Select Case** аналогична конструкции **If...Then**, но она более эффективна в тех ситуациях, когда ветвление за­висит от одной ***ключевой переменной****.*

Оператор **Select Case** позволяет осуществить разветвление по любому фиксированному числу ветвей. Он имеет следующий формат:

|  |
| --- |
| **Select  Case** *выражение*  **Case**  *тест1*  [*блок\_операторов*1]  [**Case** *тест2*  [*блок\_операторов2*] ]  . . .  [**Case Else**  [*блок\_операторовn*] ]  **End Select** |

Здесь *выражение* – арифметическое или строковое, а *тестi* может иметь следующую форму:

|  |
| --- |
| *выражение* **[,***выражение . . .*]  *выражение***To** *выражение*  **Is** *операция\_отношения**выражение* |

Например, **Case** могут содержать следующие тесты:

|  |
| --- |
| **Case Is  > 3** *' проверка на превышение значения 3;*  **Case  x^2** *' проверка на равенство значению выражения;*  **Case  5 To 7** *' проверка на принадлежность отрезку [5,7];*  **Case  1 To 4,  7  To 9,  11, 13,  Is > Max** |

В последнем примере через запятую перечисляются несколько тестов, при этом результат проверки выражения считается ис­тинным, если выполняется, хотя бы один тест.

Порядок выполнения оператора **Select Case** следующий: если значение *выражения* удовлетворяет *тесту,* расположенному после конструк­ции **Case**, то выполняется данный *блок\_операторов*, а затем управление передается оператору, следующему после конструкции **End Select**. В противном случае проверяется следующий *тест* и так далее. Все конструкции **Select Case** рабо­тают аналогично. *Блок\_операторов*, расположенный после **Case Else**, выполняется в том случае, если последовательные проверки всех *тестов* выше дали значе­ние **False** (т.е. ни один тест не выполнился).

Оператор имеет следующие особенности: *тест1*, *тест2* и так далее должны быть последними в строке, а конструкция **Case Else** должна быть единственной в строке.

Следующий пример показывает, как работает конструкция **Select Case**

|  |
| --- |
| **Dim Age As Integer= 18**  **Select Case Age**  **Case 16**  **Label1.Text="Теперь Вы можете водить машину!"**  **Case 18**  **Label1.Text="Теперь Вы можете участвовать в выборах!"**  **Case 21**  **Label1.Text="Теперь Вы можете еще кой-чего."**  **Case 65**  **Label1.Text="Пора уходить на пенсию."**  **End Select** |

исполь­зуется в программе для вывода нужного сообщения о возрасте и куль­турных вехах в жизни человека. Если переменная **Age** содержит значе­ние 18, то в свойство Text объекта надписи записывается строка «Теперь Вы можете участвовать в выборах!».

Также конструкция **Select Case** поддерживает оператор **Case Else***,* который может использоваться для отображения сообщений, если ни одно из предшествующих значений не совпадает со значением переменной **Аgе***.*

|  |
| --- |
| **Dim Age As Integer**  **Age = 25**  **Select Case Age**  **Case 16**  **Label1.Text = "Теперь Вы можете водить машину!"**  **Case 18**  **Label1.Text = "Теперь Вы можете участвовать в выборах!"**  **Case 21**  **Label1.Text = "Теперь Вы можете еще кое-чего"**  **Case 65**  **Label1.Text = "Пора уходить на пенсию"**  **Case Else**  **Label1.Text = "У Вас замечательный возраст! Наслаждайтесь!"**  **End Select** |

Чтобы включить в конструкцию **Select Case** диапазон проверяемых зна­чений, можно использовать ***операции отношения*** =, *<>,* >, <, >= и <=. Чтобы использовать опе­рации отношения, требуется включить в оператор ключевое слово **Is** или ключевое слово **То***,* определяющие производимое сравнение. Ключевое слово **Is** говорит компилятору о том, что требуется сравнить переменную с выражением, указанным после ключевого слова **Is***.* Ключевое слово **То** определяет диапазон значений. Следующая конструкция использует **Is, То** и несколько операций отношения для того, чтобы проверить значе­ние переменной **Age** и отобразить одно из пяти сообщений:

|  |
| --- |
| **Select Case Age**  **Case Is< 13**  **Label1.Text=" Наслаждайтесь детством!"**  **Case 13 To 19**  **Label1.Text =" Наслаждайтесь юностью!"**  **Case 21**  **Label1.Text =" Теперь Вы можете еще кое-чего"**  **Case Is> 100**  **Label1.Text =" Здорово выглядите!"**  **Case Else**  **Label1.Text =" Отличный возраст."**  **End Select** |

Если значение переменной **Аgе** меньше 13, отображается сообщение «Наслаждайтесь детством!». Для возраста от 13 до 19 лет отображается сообщение «Наслаждайтесь юностью!» и так далее.

Условная конструкция **Select Case** обычно гораздо яснее, чем конст­рукция **If...Then**, и более эффективна, в случае если в зависимости от значения или свойства перемен­ной следует образовать три или более ветвей программы. Однако, когда делается два или менее сравнений или, когда используется несколько различных значений, то лучше использовать условную конструкцию **If...Then.**

В следующем примере вы увидите, как использовать структуру **Select Case** для обработки ввода из поля списка. Для получения ввода будут использоваться свойства **ListBox1.Text и ListBox1.Selected.IndexChanged***,* а затем для отображения приветствия на одном из четы­рех языков будет использоваться конструкция **Select Case**.

**Пример 4.4.2-3. Создать проект, использующий конструкции Select Case для обработки и выбора данных из списка.**

1. Выполнить команду ***Создание проекта*** элемента Главного меню **Файл.**
2. На экране появится диалоговое окно Создание проекта.
3. Создать новый проект с именем **Пример 4.2-3**.

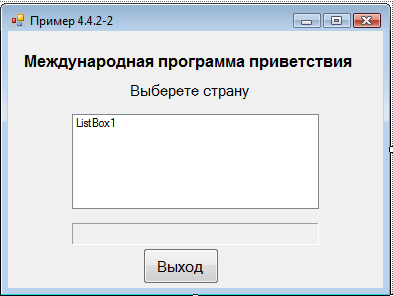
В окне **Конструктор формы** появится пустая форма.

1. В окне **Панель элементов** щелкнуть на элементе управления **Label1**, а затем ввести в верхней части формы надпись, которая будет отображать название программы.
2. Используя элемент управления **Label**, создать второй объект надписи ниже первого. Эта надпись будет использоваться как заголовок списка.
3. Щелкнуть в окне Панель элементов на элементе управления **ListBox** и ниже второй надписи отобразить поле списка.
4. Используя элемент управления **Label**, ниже поля списка ввести еще одну небольшую надпись для отображения вывода программы.
5. Используя элемент управления Button создать в нижней части формы небольшую кнопку.
6. Открыть окно **Свойства**, а затем для только что созданных объектов установить свойства, показанные в табл. 4.4.2-5.

Таблица 4.4.2-5

|  |  |  |
| --- | --- | --- |
| **Объект** | **Свойство** | **Значение** |
| **Form1** | Text | *Пример 4.2-3* |
| **Label1** | Font | *Times New Roman, Bold (Жирный), 12-point (12 пунктов)* |
| Name | *Label1* |
| Text | *Международная программа приветствия* |
| **Label2** | Name | *Label2* |
| Text | *Выберите страну* |
| **Label3** | AutoSize | *False* |
| BorderStyle | *Fixed3D* |
| ForeColor | *Red(Красный)* |
| Name | *Label3* |
| Text | *(пусто)* |
| **ListBox1** | Name | *ListBox1* |
| **Button1** | Name | *Button1* |
| Text | *(Выход)* |

После окончания установки свойств форма примет следующий вид (рис. 4.4.2-8).



*Рис. 4.4.2-8. Форма* ***Примера.4.2-3*** *с установленными свойствами*

1. Дважды щелкнуть мышью на форме.

В окне Редактор кода появится процедура события **Form1\_Load(…)**.

1. Для инициализации списка необходимо ввести следующий код:

|  |
| --- |
| **ListBox1.ltems.Add("England") : ListBox1.ltems.Add("Germany")**  **ListBox1.ltems.Add("Mexico") : ListBox1.ltems.Add("ltaly")** |

1. Для добавления элементов в список на форме эти строки используют

метод **Add()** объекта списка **ListBox**.

1. Чтобы переключиться обратно в окно **Конструктора формы**

необходимо щелкнуть на вкладке **Form1.vb[Конструктора]** **Основного**

**окна**, а затем дважды щелк­нуть мышью на объекте списка **ListBox1**

формы, чтобы отредактировать его процедуру события.

В окне **Редактор кода** появится процедура события

ListBox1.SelectedIndexChanged(…).

1. Для обработки выбора, сделанного пользователем в списке, ввести   
    следующие строки:

|  |
| --- |
| **Label3.Text = ListBox1.Text**  **Select Case ListBox1.Selectedlndex**  **Case 0**  **Label3.Text ="Hello, programmer"**  **Case 1**  **Label3.Text ="Hallo, programmierer"**  **Case 2**  **Label3.Text ="Hola, programador"**  **Case 3**  **Label3.Text ="Ciao, programmatore"**  **End Select** |

Первая строка копирует имя выбранного элемента списка в свойство Text третьей надписи формы. Наиболее важным свойством, использованным в этом опе­раторе, является свойство ListBox1.Text, которое содержит текст элемента, выбранного в списке. Остальные операторы являются час­тью условной конструкции **Select Case**. Эта конструкция в качестве проверяемой переменной использует свойство   
ListBox1.SelectedIndex, сравнивая его с несколькими значениями. Свойство SelectedIndex всегда содержит номер выбранного элемента списка; самый пер­вый элемент имеет номер 0 (ноль), второй элемент имеет номер 1, следующий – 2 и так далее. Используя SelectedIndex, конструкция **Select Case** может быстро определить, какой выбор сделал пользова­тель, и отобразить на форме правильное приветствие.

1. Снова отобразить форму и дважды щелкнуть мышью на кнопке **Выход**.

В окне **Редактор кода** появится процедура события **Button1\_Click(…)**.

В этой процедуре события ввести оператор **End** (рис. 4.4.2-9).

|  |
| --- |
| **Public Class Form1**  **Private Sub Form1\_Load(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) HandlesMyBase.Load**  **ListBox1.Items.Add("England") : ListBox1.Items.Add("Germany")**  **ListBox1.Items.Add("Mexico") : ListBox1.Items.Add("Italy")**  **End Sub**  **Private Sub Button1\_Click(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) Handles Button1.Click**  **End**  **End Sub**  **Private Sub ListBox1\_SelectedIndexChanged(ByVal sender \_**  **As System.Object, ByVal e As System.EventArgs) \_**  **Handles ListBox1.SelectedIndexChanged**  **Label4.4.Text = ListBox1.Text**  **Select Case ListBox1.SelectedIndex**  **Case 0**  **Label4.4.Text = "Hello, programmer"**  **Case 1**  **Label4.4.Text = "Hallo, programmierer"**  **Case 2**  **Label4.4.Text = "Hola, programador"**  **Case 3**  **Label4.4.Text = "Ciao, programmatore"**  **End Select**  **End Sub**  **End Class** |

*Рис. 4.4.2-9. Программный код проекта* **Пример 4.2-3**

1. Чтобы сохранить изменения, щелкнуть на кнопке ***Сохранить все*** на стандартной панели инструментов. В качестве места сохранения указать соответствующую папку.
2. Чтобы запустить программу, щелкнуть на кнопке ***Начать отладку***,

расположенной на стандартной панели инструментов.

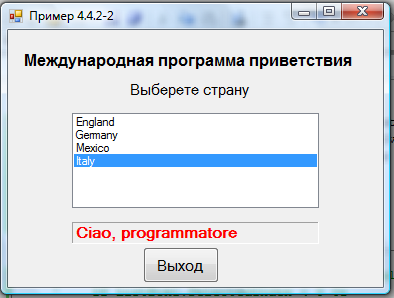
1. В списке ***Выберите страну*** щелкнуть на каждом из названий стран

по очереди.

1. Для каждой из перечисленных стран программа отображает   
    приветствие.

Иллюстрация на рис. 4.4.2-10 показывает при­ветствие для Италии.

1. Чтобы остановить программу, щелкните на кнопке ***Выход***.



*Рис. 4.4.2-10. Результат работы проекта* **Пример 4.2-3***,   
показывающий при­ветствие для Италии*

### 4.4.3. Базовые алгоритмы

### разветвляющихся структур

### и примеры их программирование

***Алгоритмы разветвляющихся структур*** позволяют описывать широкий класс задач:

* Вычисление значений сложных функций, зависящих от условий.
* Нахождение максимума и минимума среди нескольких переменных.
* Выбора различных путей вычислительного процессаи т.д**.**

В примерах 4.4.3-1 – 4.4.3-6 описаны алгоритмы и процедуры наиболее часто встречающихся задач, включающие базовые алгоритмы разветвляющих структур.

Ввиду сложности понимания и оформления *процедур*-**Sub** и *процедур*-**Function** необходимо сделать несколько замечаний.

Если процедура не содержит ни входных, ни выходных параметров (является автономной функционально полной функцией), то она может быть оформлена как *процедура*-**Sub** и как *процедура*-**Function**.

Если процедура содержит несколько входных параметров (с атрибутом **ByVal**) и несколько выходных параметров (с атрибутом **ByRef**), два и более, то эту процедуру следует оформлять как *процедуру*-**Sub**.

Если процедура содержит несколько входных параметров (с атрибутом **ByVal**) и один выходной параметр (с атрибутом **ByRef)**, то она может быть оформлена и как *процедура*-**Sub**, и как *процедура*-**Function**.

Этими замечаниями воспользуемся при написании процедур следующих примеров.

**Пример 4.4.3-1.Написать процедуру вычисления значения функции:**

  1, если x = 0;

y(x)=

Sin(x), если х > 0 или х < 0, где x – вводимое значение.

Схема алгоритма и программный код процедуры приведены на рис. 4.4.3-1. Процедуры **vvodDbl2()** и **vivodDbl2()** взяты из 4.3.4-2.

|  |  |
| --- | --- |
|  | **Function Pr431() As Double**  **Dim x As Double**  **Dim y As Double**  **x = vvodDbl2("Ввод x = ",TextBox1)**  **If x = 0 Then y = 1 Else y = Sin(x)**  **vivodDbl2(у, " у= ",TextBox2)**  **Return y**  **End Function** |

*Рис.4.4.3-1. Схема алгоритма и программный код процедуры* **Pr431()**

*проекта* **Пример 4.3-1**

*Процедура*-**Function Pr431()** может быть вызвана, как на рис. 4.4.3-2.

|  |
| --- |
| **Dim YY As Double**  **YY=Pr431()** |

*Рис.4.4.3-2. Пример вызова процедуры* **Pr431()**

*проекта* **Пример 4.3-1**

|  |  |
| --- | --- |
|  | **Function Pr433(ByVal x As Double*)* As Double**  **Dim y As Double**  **If x = 0 Then y = 1 Else y = Sin(x)**  **Return y**  **End Function** |

*Рис.4.4.3-3. Схема алгоритма и программный код процедуры* **Pr433()***проекта* **Пример 4.3-1**

Если процедура-**Function** не будет содержать ввода и вывода данных, то она будет выглядеть, как на рис. 4.4.3-3. В этом случае необходимо использование параметра функции для передачи значения **x** в вызываемую *процедуру*- **Function** **Pr433()**.

*Процедура*-**Function Pr433()** может быть вызвана из любой другой процедуры или из модуля формы, например, как на рис. 4.4.3-4.

|  |
| --- |
| **Dim YY, x As Double**  **x = vvodDbl2("Ввод x=", TextBox1)**  **YY = Pr433( x )**  **vivodDbl2(YY," YY", TextBox2)** |

*Рис.4.4.3-4. Пример вызова процедуры* **Pr433()**

*проекта* **Пример 4.3-1**

Оформим решение примера как *процедуру*-**Sub**. Программный код процедуры-**Sub** и соответствующее обращение к ней приведены на рис. 4.4.3-5 и рис. 4.4.3-6 соответственно. При создании процедуры типа **Sub** необходимо ввести еще один параметр **yy**, который передает в процедуру значение параметра по ссылке (**ByRef**), так как именно через этот параметр происходит возврат результата работы процедуры-**Sub** в вызывающую процедуру. Здесь **xx** и **yy** – фактические параметры, на которые заменяются формальные параметры **x** и **y** при работе (вызове) процедуры **Pr435( ).**

|  |
| --- |
| **Sub Pr435(ByVal x As Double, ByRef y As Double)**  **If x = 0 Then y = 1 Else y = Sin(x)**  **End Sub** |

*Рис.4.4.3-5. Схема алгоритма и программный код процедуры* **Pr435()***проекта* **Примера 4.3-1**

|  |
| --- |
| **Dim XX, YY As Double**  **XX=vvodDbl2("Ввод x=", TextBox1)**  **Pr435( XX, YY)**  **vivodDbl2(YY,"Значение функции YY", TextBox2)** |

*Рис. 4.4.3-6. Схема алгоритма и программный код процедуры* **Pr435()***проекта* **Пример 4.3-1**

**Пример 4.4.3-2. Написать процедуру, вычисляющую x**

Sin(a2) + b, если, а> b;

x =

a – b, в противном случае.

Данную задачу можно решить двумя способами. На рис 4.4.3-7 при­ведена схема алгоритма решения задачи, в которой использовано стан­дартное разветвление, а на рис.4.4.3-8 – усеченное разветвление. Второй способ в данном случае предпочтительнее, поскольку упрощает про­граммную реализацию алгоритма. Усеченное разветвление реко­мендуется использовать в тех случаях, когда выбор из двух возможных ветвей производится последовательно и неоднократно.

|  |  |
| --- | --- |
|  | **Function Pr437(ByRef a As Double \_**  **ByVal b As Double) As Double**  **Dim x As Double**  **If a > b Then**  **x = Sin(a^2) + b**  **Else**  **x = a – b**  **End If**  **Return x**  **End Function** |

*Рис. 4.4.3-7. Схема алгоритма и программный код процедуры* **Pr437()***проекта* **Пример 4.3-2**

|  |  |
| --- | --- |
|  | **Function Pr438(ByVal a As Double\_**  **ByVal b As Double)As Double**  **Dim x As Double**  **x = a – b**  **If a > b Then x = Sin(a^2) + b**  **Return x**  **End Function** |

*Рис. 4.4.3-8. Схема алгоритма и программный код процедуры* **Pr438()***проекта* **Пример 4.3-2**

*Процедура*-**Function Pr437()**или **Pr438()** может быть вызвана, например, как на рис. 4.4.3-9.

|  |
| --- |
| **Dim XX, aa, bb As Double**  **aa= vvodDbl2("Ввод a=", TextBox1)**  **bb= vvodDbl2("Ввод b=", TextBox2)**  **XX=Pr438(aa, bb )**  **vivodDbl2(XX,"Значение функции ", TextBox3)** |

*Рис. 4.4.3-9. Пример вызова процедуры* **Pr438()**

*проекта* **Пример 4.3-2**

**Пример 4.4.3-3. Написать процедуру-Function, которая присваивает переменной f наибольшее из значений двух переменных x и y.**

|  |  |
| --- | --- |
|  | **Function Pr4310(ByVal x As Double\_**  **ByVal y As Double) As Double**  **Dim f As Double**  **If x > y Then f = x Else f = y**  **Return f**  **End Function** |

*Рис. 4.4.3-10. Схема алгоритма и программный код процедуры* **Pr4310()** *проекта* **Пример 4.3-3**

Алгоритмы и программы решения задачи вы­бора наибольшего из двух значений приведены на рис.4.4.3-11 и 4.4.3-12.

|  |  |
| --- | --- |
|  | **Function Pr4311(ByVal x As Double\_**  **ByVal y As Double) As Double**  **Dim f As Double**  **f = x**  **If  y > f   Then  f = y**  **Return f**  **End Function** |

*Рис. 4.4.3-11. Схема алгоритма и программный код процедуры* **Pr4311()** *проекта* **Пример 4.3-3**

В первом способе (рис. 4.4.3-10) используется стандартное разветвление, а во втором (рис.4.4.3-11) – усеченное.

Процедура-**Function Pr4310()** (или **Pr4311())** может быть вызвана, например, как на рис. 4.4.3-12.

***Обратите внимание, что:***

**If x > y Then f=x Else f=y** *соответствует функции f = max{x, y}, а*

**If x < y Then f=x Else f=y** с*оответствует функции f = min{x, y },*

где **x** и **y** любые арифметические выражения.

|  |
| --- |
| **Dim ff As Double**  **Dim xx, yy As Double**  **xx= vvodDbl2("Ввод x=", TextBox1)**  **yy= vvodDbl2("Ввод y=", TextBox2)**  **ff=Pr4311(xx, yy)**  **vivodDbl2(ff, "max{x, y }= ", TextBox3)** |

*Рис. 4.4.3-12. Пример вызова процедуры* **Pr4311()**

*проекта* **Пример 4.3-3**

**Пример 4.4.3-4. Написать процедуру-Function, которая вычисляет r = min{ a+b, c2,  }.**

|  |  |
| --- | --- |
|  | **Function Pr4313(ByVal a As Double\_**  **ByVal b As Double\_**  **ByVal c As Double \_**  **ByVal d As Double) As Double**  **Dim min As Double**  **min = a + b**  **If c^2 < min Then min = c^2**  **If Sqrt(d) < min Then min = Sqrt(d)**  **Return min**  **End Function** |

*Рис. 4.4.3-13. Схема алгоритма и программный код процедуры* **Pr4313()** *проекта* **Пример 4.3-4**

Решение задач выбора наибольшего (наименьшего) из значений переменных или выражений числом более двух сводится к последовательному применению усеченных разветвлений.

Алгоритм, реализующий Пример 4.4.3-4 (рис.4.4.3-13), относится к числу ***базовых*** алгоритмов выбора наименьшего из нескольких значений. Основой алгоритма является усеченное разветвление. Здесь первоначально переменной **min** присваивается значение первого из выражений. Истинное значение наименьшего значения определяется путем последовательного сравнения со всеми остальными значениями выражений заданной последовательности.

Процедура-**Function Pr4313()** может быть вызвана, как на   
рис. 4.4.3-14.

|  |
| --- |
| **Dim rr As Double**  **Dim aa, bb, cc, dd As Double**  **aa = vvodDbl2("Ввод a=", TextBox1)**  **bb = vvodDbl2("Ввод b=", TextBox2)**  **cc = vvodDbl2("Ввод c=", TextBox3)**  **dd = vvodDbl2("Ввод d=", TextBox4)**  **rr = Pr4313(aa, bb, cc, dd )**  **ivodDbl2(rr,"Значение min = ", TextBox5)** |

*Рис. 4.4.3-14. Пример вызова процедуры* **Pr4313()** *проекта* **Пример 4.3-4**

**Пример 4.4.3-5. Написать процедуру-Function, которая вычисляет значение функции y(x) в соответствии со следую­щим правилом:**

ex, если x<=-1;

y(x)= 3, если x>1;

2, если –1<х≤1.

|  |  |
| --- | --- |
|  | **Function Pr4315(ByVal x As Double)\_**  **As Double**  **Dim y As Double**  **If  x <= −1  Then**  **y = Exp(x)  ElseIf  x > 1 Then**  **y=3  Else  y=2  End If**  **Return y**  **End Function** |

*Рис. 4.4.3-15. Схема алгоритма и программный код процедуры* **Pr4315()**

*Процедура*-**Function Pr4315()** может быть вызвана, например, как на рис. 4.4.3-16.

|  |
| --- |
| **Dim YY As Double**  **Dim x As Double**  **x= vvodDbl2("Ввод x=", TextBox1)**  **YY=Pr4315( x )**  **vivodDbl2(YY,"Значение функции ", TextBox2)** |

*Рис. 4.4.3-16. Пример вызова процедуры* **Pr4315()**

*проекта* **Пример 4.3-5**

**Пример 4.4.3-6. Даны действительные числа x0, y0. Определить принадлежит ли точка с координатами (x0, y0) геометрической фигуре: ромб с вершинами в точках (0, 1), (1, 0), (0, -1), (-1, 0).**

На этапе формализации задачи, прежде всего, построим заданную геометрическую фигуру (рис. 4.4.3-17).



Рис. 4.4.3-17. Ромб с вершинами в точках (0, 1), (1, 0), (0, -1), (-1, 0)

Известно, что **ax+by=c** – это уравнение прямой, а **ax+by≤c** и **ax+by≥c** – это два множества, одно из которых лежит над прямой (включая точки этой прямой), а другое под этой прямой. Для того чтобы выяснить, какое из неравенств надо взять, необходимо подставить координаты любой точки   
M(x1, y1) в это неравенство (лучше взять точку M(0, 0), чтобы упростить вычисления).

Если ее координаты удовлетворяют этому неравенству, значит, эта точка M принадлежит указанному множеству, в противном случае – нет. Запишем уравнения прямых **AB, BC, CD, DA**:

**AB**: x-y=-1; **BC**: x+y=1; **CD**: x-y=1; **DA**: x+y=-1.

Таким образом, для того чтобы точка M0(x0, y0) принадлежала замкнутой заштрихованной области, необходимо выполнить следующие условия:

**.**

Разобьем решение задачи на несколько отдельных этапов и создадим процедуры пользователя, реализующие эти задачи:

1. Процедуру-подпрограмму **Resh()**, которая возвращает строку с ответом на вопрос, принадлежит ли точка с передаваемыми через параметры координаты заданной геометрической фигуры.
2. Процедуру-функцию **vvod()** для ввода действительного числа из **TextBox**.
3. Процедуру-подпрограмму **vivod()** для вывода строки в **TextBox**.
4. Событийную процедуру, которая должна содержать вызовы всех разработанных процедур.
5. Программный код решаемой задачи представлен на рис. 4.4.3-18.

|  |
| --- |
| **Option Strict On**  **Option Explicit On**  **Imports System.Math**  **Public Class Form1**  *'Функция ввода значения вещественного типа из TextBox*  **Function vvod(ByVal T As TextBox) As Double**  **Return CDbl(T.Text)**  **End Function**  **'Процедура вывода результата в TextBox**  **Sub vivod(ByVal Z As String, ByRef T As TextBox)**  **T.Text = Z**  **End Sub**  *' Процедура решения задачи*  **Sub Resh(ByVal x0 As Double, ByVal y0 As Double, \_**  **ByRef z As String)**  **If Abs(x0 - y0) <= 1 And Abs(x0 + y0) <= 1 Then**  **z = "Точка принадлежит фигуре"**  **Else**  **z = "Точка не принадлежит фигуре"**  **End If**  **End Sub**    **Private Sub Button1\_Click(sender As Object, e As EventArgs) \_**  **Handles Button1.Click**  **Dim x, y As Double**  **Dim z As String**  **x = vvod(TextBox1) : y = vvod(TextBox2)**  **Resh(x, y, z) : vivod(z, TextBox3)**  **End Sub**  **End Class** |

*Рис. 4.4.3-18. Программный код проекта* ***Пример 4.3-6***

**Пример 4.4.3-7. Задано число n. Если n – натуральное нечетное число, меньше 10, то целочисленной переменной f присвоить 1, если n – натуральное четное число, то переменной f присвоить 2, если n – отрицательное, то присвоить f значение -1, если значение n находится внутри отрезка [10;1000], то f присвоить 10, иначе f присвоить 0.**

|  |
| --- |
| **Function Pr4319(ByVal n As Integer) As Integer**  **Dim f As Integer**  **Select Case n**  **Case 1, 3, 5, 7, 9**  **f =1**  **Case 2, 4, 6, 8**  **f =2**  **Case Is <0**  **f =-1**  **Case 10 TO 1000**  **f =10**  **Case Else**  **f =0**  **End Select**  **Return f**  **End Function** |

*Рис.4.4.3-19. Программный код процедуры* **Pr4319()**

*Проекта* **Примера 4.4.3-7**

Программный код решения ***Примера 4.4.3-7*** с использованием оператора **Select Case** приведены на рис.4.4.3-19.

Процедура-**Function Pr4319()** может быть вызвана из любой другой процедуры или из модуля формы, например, как на рис. 4.4.3-20.

|  |
| --- |
| **Dim ff As Integer**  **Dim nn As Integer**  **nn= vvodInt4(TextBox1)**  **ff=Pr4319(nn)**  **vivodInt4(ff, TextBox2))** |

*Рис. 4.4.3-20. Пример вызова процедуры* **Pr4319()**

*Проекта* **Примера 4.3-7**

### 4.4.4. Тестовые задания

1. К условным операторам относятся
   1. If...Then...Else
   2. For...Next
   3. While...Wend
   4. Select Case
   5. Do Until...Loop
2. Оператор **If-Then** – это оператор
3. стандартного разветвления
4. усеченного разветвления
5. вложенного разветвления
6. блочного разветвления
7. Оператор **If-Then-Else** – это оператор
8. усеченного разветвления
9. блочного разветвления
10. стандартного разветвления
11. вложенного разветвления
12. В операторе условного перехода между словами **If-Then** может быть
13. арифметическое или логическое выражение
14. логическое выражение
15. оператор присваивания
16. любой оператор
17. Разветвляющийся алгоритм может быть
18. стандартным, вложенным, усеченным, регулярным
19. стандартным, вложенным, усеченным
20. стандартным, вложенным, усеченным, итерационным
21. нет верного ответа
22. В операторе условного перехода **If** обязательно должно присутствовать слово
23. End If
24. Else
25. Then
26. ThenElse
27. Ошибочно записан оператор
28. If a>a Then c=1 : d =5
29. If a>a Then c=1 : d =5
30. If a+b Then c=1 : d=5
31. If a>B Then c=1 Else d=5
32. Логические переменные не могут принимать значения
33. истина
34. ложь
35. да
36. нет
37. может быть
38. В операторе условного перехода **If** после слова **Then** может быть
39. любой оператор Visual Basic
40. оператор присваивания
41. арифметическое или логическое выражение
42. нет верного ответа
43. Переменным x, y, z будут присвоены значения в результате выполнения фрагмента программы

|  |
| --- |
| **x = 66 : y = 58**  **If x > y Then z = x - y**  **If x <= y Then z = x – y** |

1. 58 58 124
2. 66 58 08
3. 58 58 0
4. Переменной **z** будет присвоено значение ### в результате выполнения фрагмента программы

|  |
| --- |
| **a = 15 : b = 10 : c = 25**  **If a > b And a < c Then**  **z = a**  **Else**  **z = c**  **End If** |

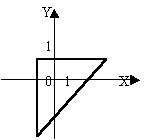
1. Последовательность операторов, в результате выполнения, которых будет найдено наибольшее значение переменных **X** и **Y**
2. МАХ = Abs( Х-Y) / 2 + (X + Y) / 2
3. If X > Y Then MAX = X
4. If X < Y Then MAX = Y
5. If X > Y Then MAX = X Else MAX = Y
6. MAX = ABS(X - Y) / 2 - (X + Y) / 2
7. Условное выражение **Not((X<=10) Or (X>= 20))** может использоваться для проверки
8. принадлежит ли число интервалу (10;20)
9. выходит ли число за пределы интервала (10;20)
10. принадлежит ли число интервалу [10;20]
11. выходит ли число за пределы интервала [10;20]
12. Для вычисления значения переменной у по формуле

**** необходима конструкция

1. If X < 2 Then Y = Abs(X) : Y = X^2
2. If X < 2 Then Y = X^2 : Y = Abs(X)
3. If X < 2 Then Y = Abs(X) Else Y = X ^ 2
4. If X < 2 Then Y = X^2 Else Y = Abs(X)
5. Логическое условие попадания точки (х, у) в заштрихованную область

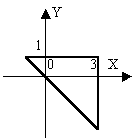
можно записать как #

1. ((x-2)\*\*2+y\*y<=4) Or (y>x/2)
2. (x-2)\*\*2+y\*y<=4 Or y>x/2
3. Логическое условие, описывающее ограниченную область,

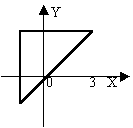


можно записать

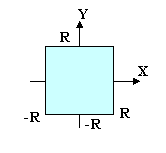
1. (x>-1) Or (y<1) Or (x-y<2)
2. (x>-1) And (y<1) And (x-y<2)
3. (x>-1) Or (y<1) Or (x+y>2)
4. (x>-1) Or (y<1) And (x+y>2)
5. (x>-1) And (y<1) Or (x+y>2)
6. Логическое условие, описывающее ограниченную область,



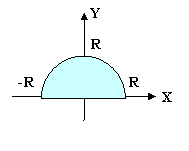
1. (x<3) And (y<1) And (x+y>0)
2. (x<3) Or (y<1) Or (y<x)
3. (x<3) Or (y<1) Or и (x+y>0)
4. (x<3) Or (y<1) And (x-y>0)
5. (x<3) Or (y<1) Or (x-y<0)
6. Логическое условие, описывающее ограниченную область,



1. (x >-1) And (y<3) And (x<y)
2. (x >-1) Or (y<3) Or (x>y)
3. (x >-1) Or (y<3) Or (x<y)
4. (x >-1) Or (y<3) And (x>y)
5. (x >-1) Or (x>y) And (y<3)
6. Логическое условие, описывающее ограниченную область,



1. Sqrt(x\*x+y\*y)<R Or y>0
2. Sqrt(x\*x+y\*y)<R Or y<0
3. Abs(x)<R Or y>0 Or y<R
4. Abs(x)<R Or y<0 Or y>-R
5. Abs(x)<R Or Abs(y)<R
6. Логическое условие, описывающее ограниченную область,



1. Abs(x)<R Or y<0 Or y>-R
2. Sqrt(x\*x+y\*y)<R Or y<0
3. Abs(x)<R Or y>0 Or y<R
4. Sqrt(x\*x+y\*y)<R Or y>0
5. Abs(x)<R Or Abs(y)<R

### 4.4.5. Лабораторная работа по теме *«Программирование алгоритмов разветвляющихся структур»*

**Цель** данной работы состоит в получении навыков разработки проектов, использующих алгоритмы разветвляющихся структур, в изучении стандартных алгоритмов разветвляющихся структур, использовании оператора **Select Case**.

### Вопросы, подлежащие изучению

1. Стандартные алгоритмы разветвляющихся структур, синтаксис операторов.
2. Программирование структур разветвления.
3. Оптимальные алгоритмы нахождения минимального и максимального значения.
4. Использование оператора выбора **Select Case**.

### Общее задание на разработку проект

1. ***Изучите вопросы программирования алгоритмов разветвляющихся структур (Тема 4).***
2. ***Создайте приложение*** *с именем* ***Проект-4.4.***
3. ***Выберите вариант задания*** *из табл. 4.4.5-1.*
4. ***Проведите формализацию*** *поставленной задачи.*
5. ***Разработайте графический интерфейс*** *пользователя.*
6. ***Составьте схемы алгоритмов*** *решения поставленной задачи.*
7. ***Напишите программный код*** *процедур пользователя в соответствии со схемами алгоритмов. Обмен данными между процедурами должен осуществляться через параметры, без использования глобальных переменных. Событийная процедура должна содержать только операторы вызова пользовательских (общих) процедур.*
8. ***Докажите правильность полученных результатов*** *на заранее разработанных тестах для всех ветвей программы.*

### Варианты индивидуальных заданий

Таблица 4.4.5-1

|  |  |
| --- | --- |
| **№** | **Задача** |
| **1)** | e= |
| **2)** | z= |
| **3)** | z= |
| **4)** | z= |
| **5)** | g= |
| **6)** | f= |
| **7)** | d= |
| **8)** | z= |
| **9)** | z= |
| **10)** | d= |
| **11)** |  |
| **12)** | p= |
| **13)** | r= |
| **14)** | r= |
| **15)** | s= |
| **16)** | t= |
| **17)** | u= |
| **18)** | f= |
| **19)** |  |
| **20)** | v= |
| **21)** | h= |
| **22)** | z= |
| **23)** |  |
| **24)** | l= |
| **25)** | Z= |
| **26)** | z= |
| **27)** | z= |
| **28)** |  |
| **29)** |  |
| **30)** | y= |

### Содержание отчёта

### Тема и название лабораторной работы.

1. Фамилия, имя студента, номер группы, номер варианта.
2. Задание на разработку проекта.
3. Формализация и уточнение задания.
4. Элементы, разрабатываемого проекта:
5. графический интерфейс пользователя;
6. таблица свойств объектов;
7. схема алгоритма решаемой задачи;
8. программный код проекта.
9. Результаты выполнения проектов.
10. Доказательство правильности работы программы.

### Пример выполнения задания

### Тема и название лабораторной работы:

Программирование алгоритмов разветвляющихся структур.

Вычисление условных выражений.

### Фамилия, имя студента, номер группы, номер варианта:

Иванов И., БИН1405, вариант 13.

1. **Задание на разработку проекта:**

Создайте проект с именем ***Проект-4*** для вычисления сложной функции с условием r=f(x, y, z):



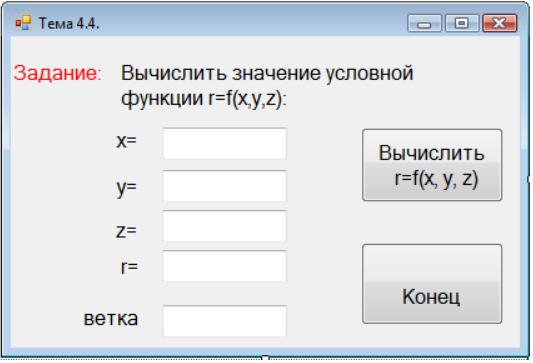
1. **Формализация и уточнение задания:**

Алгоритм решения данной задачи представляет собой комбинацию вычисления сложной функции с условием и выбора наименьшего (наибольшего) из нескольких значений. Он использует все виды разветвлений, и программируется с использованием как однострочных, так и блочных операторов **If.**

Данная задача решается с помощью процедуры **Razv()**, которая, получив в качестве входных параметров аргументы **x, y, z**, возвращает вычисленное значение **r** и номер ветки разветвляющейся функции **n**. Процедура **vivod()** предназначена для вывода в текстовое поле полученного значения функции **r** вещественного типа (**Double**) и для вывода номера **n** ветки разветвления целого типа (**Integer**), т.е. имеет аргументы разного типа, поэтому она является перегружаемой и объявлена как **Overloads**.

1. **Элементы, разрабатываемого проекта:**
2. **Графический интерфейс пользователя:**

Форма приложения может иметь такой же вид, как на рис. 4.4.5-1.



*Рис. 4.4.5-1. Форма проекта* ***Проект 4-1:   
Вычисление*** условных выражений.

1. **Таблица свойств объектов:**

Установите и сведите в таб. 4.4.5-2 свойства всех объектов.

Таблица 4.4.5-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Объект** Таблица П.4-2 **Объект** | **Имя объекта** | **Свойство** | **Значение свойства** |
| **Форма** | **Form1** | Name | *Form1* |
| Text | *ЛР по теме:Разветвляющиеся Разветвляющиеся структуры* |
| **Метка** | **Label1** | Text | *Вычислить значение r=f(x,y,z)* |
| ImageAlign | *MiddleCenter* |
| **Метка** | **Label2** | Text | *X=* |
| ImageAlign | *MiddleCenter* |
| **Метка** | **Label3** | Text | *Y=* |
| **Метка** | **Label4** | Text | *Z=* |
| **Метка** | **Label5** | Text | *R=* |
| **Текстовое поле** | **TextBox1** | Name | *TextBox1* |
| **Текстовое поле** | **TextBox2** | Name | *TextBox2* |
| **Текстовое поле** | **TextBox3** | Name | *TextBox3* |
| **Текстовое поле** | **TextBox4** | Name | *TextBox4* |
| **Кнопка** | **Button1** | Name | *Button1* |
| Text | *Вычислить r=f(x, y, z)* |
| **Кнопка** | **Button2** | Name | *Button2* |
| Text | *Конец* |

1. **Схема алгоритма процедуры:**

Схема алгоритма процедуры **Razv()**  представлена на рис. 4.4.5-2.

|  |
| --- |
|  |

*Рис. 4.4.5-2. Схемы алгоритмов процедур* **Razv()**  проекта ***Проект 4-1:***

Вычисление условных выражений

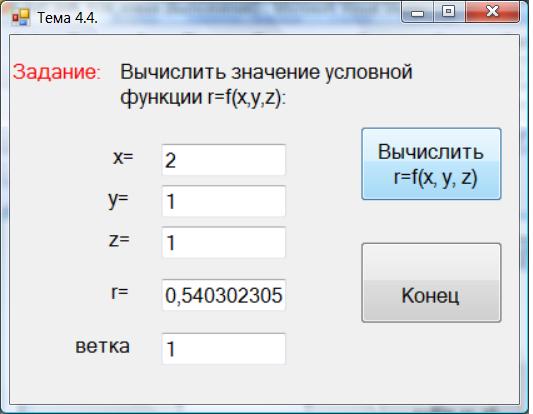
1. Программный код проекта приведен на рис. 4.4.5-3.

|  |
| --- |
| **Imports System.Math**  **Public Class Form1**  *'Функция ввода в TextBox*  **Function vvod(ByVal T As TextBox) As Double**  **Return Val(T.Text)**  **End Function**  *'Перегружаемая процедура вывода вещест. результата в TextBox*  **Overloads Sub vivod(ByVal Z As Double, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub**  *'Перегружаемая процедура вывода целого результата в TextBox*  **Overloads Sub vivod(ByVal Z As Integer, ByVal T As TextBox)**  **T.Text = CStr(Z)**  **End Sub**    **Sub Razv(ByVal x As Double, ByVal y As Double, \_**  **ByVal z As Double, ByRef r As Double,ByRef n As Integer)**  **Dim r1, r2 As Double**  **If y < x Then**  **r1 = Sin(y) : r2 = Cos(z)**  **r = x ^ 2**  **If r1 < r Then**  **r = r1**  **End If**  **If r2 < r Then**  **r = r2**  **End If**  **n = 1** *'1 ветвь разветвл. ф-ции*  **ElseIf y <= z Then**  **r1 = Log(z ^ 2) : r2 = Exp(x + y)**  **If r1 > r2 Then**  **r = r1**  **Else**  **r = r2**  **End If**  **n = 2** *'2 ветвь разветвл. ф-ции*  **Else**  **r = x + y + z**  **n = 3** *'3 ветвь разветвл. ф-ции*  **End If**  **End Sub**  **Private Sub Button1\_Click(sender As Object,e As EventArgs) \_**  **Handles Button1.Click**  **Dim x, y, z, r As Double**  **Dim n As Integer**  **x = vvod(TextBox1) : y = vvod(TextBox2) :z = vvod(TextBox3)**  **Razv(x, y, z, r, n)**  **vivod(r, TextBox4) : vivod(n, TextBox5)**  **End Sub**  **Private Sub Button2\_Click(sender As Object,e As EventArgs) \_**  **Handles Button2.Click**  **End**  **End Sub**  **End Class** |

*Рис. П.1.4-3. Программный код проекта* ***Проект 4-1:***

Вычисление условных выражений.

1. **Результат выполнения проекта приведен на рис. 4.4.5-4.**



*Рис. 4.4.5-4. Результаты выполнения проекта* ***Проект 4-1:***

Вычисление условных выражений

1. **Доказательство правильности результата**

Разработаны с тестовые исходные данные для контрольного решения, проверяющие все ветви процедуры функции, и получены результаты:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исходные данные | | | Результат | № ветви |
| x=2 | y=1 | z=1 | r= 0,540302305 | n=1 |
| x=1 | y=2 | z=3 | r**=** 20,08553692 | n=2 |
| x=1 | y=3 | z=2 | r= 6 | n=3 |

При тестовых данных результаты ручного расчета и вычисления на компьютере совпадают.

### Контрольные вопросы

1. Что такое разветвляющаяся структура?
2. Какие имеются типы разветвляющихся структур?
3. Когда необходимо вкладывать одну структуру стандартного разветвления в другую структуру стандартного разветвления?
4. Можно ли вложить структуру стандартного разветвления в структуру усеченного разветвления?
5. Какие операции отношения используются в VB?
6. Какие логические операции используются в VB?
7. Как описываются логические переменные?
8. Какие логические константы известны?
9. Что такое простое условие?
10. Что такое сложное условие?
11. Каковы правила вычисления логических выражений?
12. Чем отличается строчный оператор If от блочного оператора If?
13. В каких случаях необходимо использовать оператор If, а в каких оператор Select Case?
14. Каковы правила записи оператора If?
15. Каковы правила записи оператора Select Case?
16. Какие операторы относятся к условным операторам?
17. Как называется оператор If-Then?
18. Как называется оператор If-Then-Else?
19. Для чего предназначен оператор If?
20. Что может быть в операторе условного перехода между словами If-Then?
21. Какими могут быть разветвляющиеся алгоритмы?
22. Какими операторами реализуются разветвляющиеся алгоритмические структуры?
23. Какие необязательные ключевые слова в однострочном операторе If?
24. Какие слова должны обязательно присутствовать в операторе условного перехода If?
25. Какова последовательность операторов, в результате выполнения, которых будет найдено наибольшее значение двух переменных?