Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра экономической математики, информатики и статистики (ЭМИС)

УСЛОВНЫЕ ОПЕРАТОРЫ И ОПЕРАТОРЫ ЦИКЛОВ

Отчет по практической работе по дисциплине «Информационные технологии»

Студент группы 549
 Баулин С.К.
«__» _____ 2020 г.
Старший преподаватель кафедры ЭМИС
_____ _____ Афанасьева И. Г.
«__» _____ 2020 г.

Томск 2020

Практическая работа № 7

Условные операторы и операторы циклов

Цель работы

Повторить основы программирования линейных и разветвляющихся вычислительных процессов.

Теоретический материал, для освоения темы

Управляющие структуры VBA. If . . . Then, If . . . Then . . . Else, Select Case

Управляющие структуры позволяют управлять последовательностью выполнения программы. Без операторов управления все операторы программы будут выполняться слева направо и сверху вниз. Однако иногда требуется многократно выполнять некоторый набор инструкций автоматически, либо решить задачу по-другому в зависимости от значения переменных или параметров, заданных пользователем во время выполнения. Для этого служат конструкции управления и циклы.

VBA поддерживает следующие конструкции принятия решений:

```
If . . . Then
If . . . Then . . . Else
Select Case
```

Конструкция If . . . Then

Конструкция **If . . . Then** применяется, когда необходимо выполнить один или группу операторов в зависимости от некоторого условия. Синтаксис этой конструкции позволяет задавать ее в одной строке или в нескольких строках программы:

```
If условие Then выражение
```

```
If условие Then
выражение
End If
```

Обычно условие является простым сравнением, но оно может быть любым выражением с вычисляемым значением. Это значение интерпретируется как **False** (Ложь), если оно нулевое, а любое ненулевое рассматривается как **True** (Истина). Если условие истинно, то выполняются все выражения, стоящие после ключевого слова **Then**. Для условного выполнения одного оператора можно использовать как синтаксис для одной строки, так и синтаксис для нескольких строк (блоковую конструкцию).

Заметим, что синтаксис оператора **If . . . Then** для одной строки не использует оператор **End If**. Чтобы выполнить последовательность операторов, если условие истинно, следует использовать блоковую конструкцию **If . . . Then . . . End If**.

Если условие ложно, то операторы после ключевого слова **Then** не выполняется, а управление передается на следующую строку (или строку после оператора **End If** в блочной конструкции).

Конструкция If . . . Then . . . Else определяет несколько блоков операторов, один из которых будет выполняться в зависимости от условия:

```
If условие1 Then
выражение1
ElseIf условие2 Then
выражение2
. . .
Else
выражение-n
End If
```

При выполнении сначала проверяется **условие1**. Если оно ложно, VBA проверяет следующее **условие2** и т. д., пока не найдет истинного условия. Найдя его, VBA выполняет соответствующий блок операторов и затем передает управление инструкции, следующей за оператором **End if.** В данную конструкцию можно включить блок оператора **Else**, который VBA выполняет, если не выполнено ни одно из условий.

Конструкция **If . . . Then . . . ElseIf** в действительности всего лишь специальный случай конструкции **If . . . Then . . . Else**. Заметим, что в данной конструкции может быть любое число блоков **ElseIf**, или даже ни одного. Блок **Else** можно включать независимо от присутствия или, наоборот, отсутствия блоков **ElseIf**.

Рассмотрим пример вычисления функции

```
Private Sub UserForm_Initialize()
 Dim q, w, e As String
If CheckBox2 = True Then
q = InputBox("Введите название фирмы")
w = InputBox("Введите название тура")
e = InputBox("Введите название страны")
End If
Range("A8") = q
Range("C8") = w
Range("E8") = e
End Sub
```

Заметим, что можно добавить любое число блоков **Elself** в конструкцию **If . . . Then**. Однако количество блоков **Elself** может стать настолько большим, что конструкция **If . . . Then** станет очень громоздкой и неудобной. В подобной

ситуации следует применять другую конструкцию принятия решения – **Select Case**.

Конструкция Select Case

Конструкция **Select Case** является альтернативой конструкции **If . . . Then . . . Else** в случае выполнения блока, состоящего из большого набора операторов. Конструкция **Select Case** предоставляет возможность, похожую на возможность конструкции **If . . . Then . . . Else**, но в отличие от нее она делает код более читаемым при наличии нескольких вариантов выбора.

Конструкция **Select Case** работает с единственным проверяемым выражением, которое вычисляется один раз при входе в эту конструкцию. Затем VBA сравнивает полученный результат со значениями, задаваемыми в операторах **Case** конструкции. Если найдено совпадение, выполняется блок операторов, ассоциированный с оператором **Case**:

```
Select Case проверяемое_выражение
[Case список_выражений1
[блок_операторов1]]
[Case список_выражений2
[блок_операторов2]]
. . .
[Case Else
[блок_операторовn]]
End Select
```

Рассмотрим пример вычисления функции

```
Private Sub UserForm_Initialize()
Dim nR As Integer
Dim nC As Integer
Dim RC As String
```

```
nR = ActiveCell.Row
nC = ActiveCell.Column
RC = nR & nC
If Not Check(nR, nC) Then Exit Sub
ActiveCell = "0"
With WRK
  Select Case RC
    Case "23": .Cells(2, 2) = "X"
    Case "34": .Cells(2, 4) = "X"
    Case "32": .Cells(4, 2) = "X"
    Case "43": .Cells(4, 4) = "X"
    Case "22": .Cells(2, 4) = "X"
    Case "24": .Cells(4, 4) = "X"
    Case "42": .Cells(4, 4) = "X"
    Case "44": .Cells(4, 2) = "X"
  End Select
End With
End Sub
```

Обратим внимание на то, что здесь используется RC ссылка, для удобства выбора диапазона ячеек, а также заметим, что конструкция **Select Case** вычисляет выражение только один раз при входе в нее, а в конструкции **If . . . Then . . . Else** вычисляются различные выражения для каждого оператора **Elself**. Конструкцию **If . . . Then . . . Else** можно заменить конструкцией **Select Case**, только если оператор **If** и каждый оператор **Elself** вычисляют одно и то же выражение.

Операторы циклов. Вложенные циклы

Операторы циклов

Циклы позволяют выполнить одну или несколько строк кода несколько раз. VBA поддерживает следующие циклы:

```
For...Next
For Each...Next
Do... Loop
```

Конструкция **For . . . Next.** Когда число повторений известно заранее, используют цикл **For . . . Next**. В цикле **For** используется переменная, называемая переменной цикла или счетчиком цикла, которая увеличивается или уменьшается на заданную величину при каждом повторении цикла. Синтаксис этой конструкции следующий:

```
For counter = start To end [Step increment]
операторы
Next [counter]
```

Параметры counter (счетчик), start (начало цикла), end (конец цикла) и increment (приращение) являются числовыми.

Примечание. Параметр increment может быть как положительным, так и отрицательным. Если он положителен, параметр start должен быть меньше или равен параметру end, иначе цикл не будет выполняться. Если параметр increment отрицателен, то параметр start должен быть больше или равен значению параметра end, чтобы выполнялось тело цикла. Если параметр **Step** не задан, то значение параметра increment по умолчанию равно 1.

VBA выполняет цикл For в следующей последовательности:

1. Устанавливает значение переменной цикла counter в значение start.

2. Сравнивает значение переменной цикла counter и значение параметра end. Если переменная counter больше, VBA завершает выполнение цикла. (Если значение параметра increment отрицательно, то VBA прекращает

выполнение цикла при условии, что значение переменной цикла counter меньше значения параметра end.)

3. Выполняет операторы тела цикла statements.

4. Увеличивает значение переменной цикла counter на 1 или на величину значения параметра increment, если он задан.

Повторяет шаги со 2 по 4.

Конструкция Do...Loop

Цикл **Do** применяется для выполнения блока операторов неограниченное число раз. Существует несколько разновидностей конструкции **Do . . . Loop**, но каждая из них вычисляет выражение-**условие**, чтобы определить момент выхода из цикла. Как и в случае конструкции **If . . . Then условие** должно быть величиной или выражением, принимающими значение **False** (нуль) или **True** (не нуль).

В следующей конструкции **Do . . . Loop** операторы выполняются до тех пор, пока значением условия является **True** (Истина):

```
Do While условие
операторы
Loop
```

Выполняя этот цикл, VBA сначала проверяет условие. Если условие ложно (**False**), он пропускает все операторы цикла. Если оно истинно (**True**), VBA выполняет операторы цикла, снова возвращается к оператору **Do While** и снова проверяет условие.

Следовательно, цикл, представленный данной конструкцией, может выполняться любое число раз, пока значением условия является не нуль или **True** (Истина). Отметим, что операторы тела цикла не выполняются ни разу, если при первой проверке условия оно оказывается ложным (**False**).

Другая разновидность конструкции **Do . . . Loop** сначала выполняет операторы тела цикла, а затем проверяет условие после каждого выполнения. Эта разновидность гарантирует, что операторы тела цикла выполнятся по крайней мере один раз:

```
Do
операторы
Loop While условие
```

Две другие разновидности конструкции цикла аналогичны предыдущим, за исключением того, что цикл выполняется, пока условие ложно (**False**):

Цикл не выполняется вообще или выполняется много раз:

```
Do Until условие
операторы Loop
```

Цикл выполняется по крайней мере один раз:

```
Do
операторы
Loop Until условие
```

Вложенные циклы

Можно помещать структуры управления внутрь других структур управления (например, блок **If . . . Then** внутрь цикла **For . . . Next**). Говорят, что структура управления, помещенная внутрь другой структуры управления, является вложенной.

При вводе/выводе элементов двумерного массива на рабочий лист Microsoft Excel удобно применять пользовательские процедуры ввода/вывода:

```
Sub readcell (i As Integer, j As Integer, val As Variant)
```

```
    val = Лист1.Cells(i, j).Value
    End Sub
    Sub outcell (i As Integer, j As Integer, val As
Variant)
    Лист1.Cells(i, j).Value = val
    End Sub
```

где i – номер строки, j – номер столбца рабочего листа.

Задания на практическую работу

Задание 1. Найти минимальный и максимальный элементы массива из 10 элементов, заполненного случайными значениями, и поменять их местами.

Задание 2. Microsoft Excel. Составить таблицу начисления заработной платы работникам ООО «Воронья слободка».

Ф.И.О.	Тарифный разряд	% выполнения плана	Тарифная ставка, руб.	Заработная плата с премией, руб.
Пряхин Н.П.	3	102	?	?
Суховейко А.Д.	2	98	?	?
Лоханкин В.А.	1	114	?	?
Пферд Л.Ф.	1	100	?	?
Севрюгов Л.А.	3	100	?	?
Гигиенишвили Г.С.	2	94	?	?
Птибурдуков А.И.	3	100	?	?

Примечание 1. Тарифная ставка определяется в зависимости от разряда: 1-й разряд – 4000 руб.; 2-й разряд – 6500 руб.; 3-й разряд – 8000 руб. Тарифные ставки оформить отдельной таблицей.

Примечание 2. Размер премиальных определяется в зависимости от выполнения плана:

ниже 100 % – премия не начисляется;

100 % – премия 20 % от тарифной ставки;

101…110 % – премия 30 %;

111…115 % – премия 40 %.

Задание 3. Составить программу, переводящую числовое значение (до сотен включительно) в строковое. Например: 132 – "сто тридцать два".

Ход работы

Задание 1. Окно VBA с кодом и результат работы макроса представлены на рисунках 1-2.

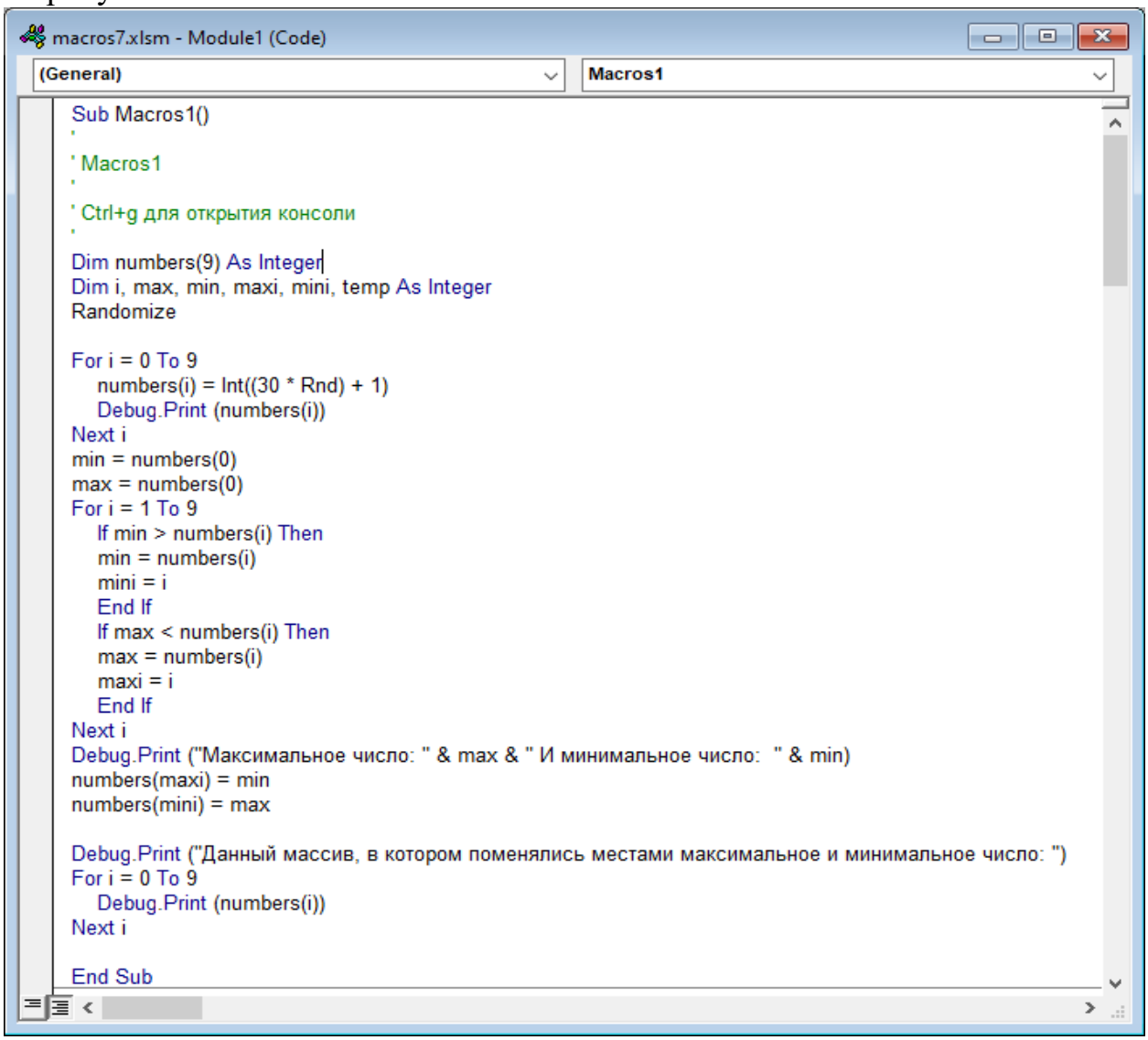

Рисунок 1 – Скриншот кода макроса задания 1

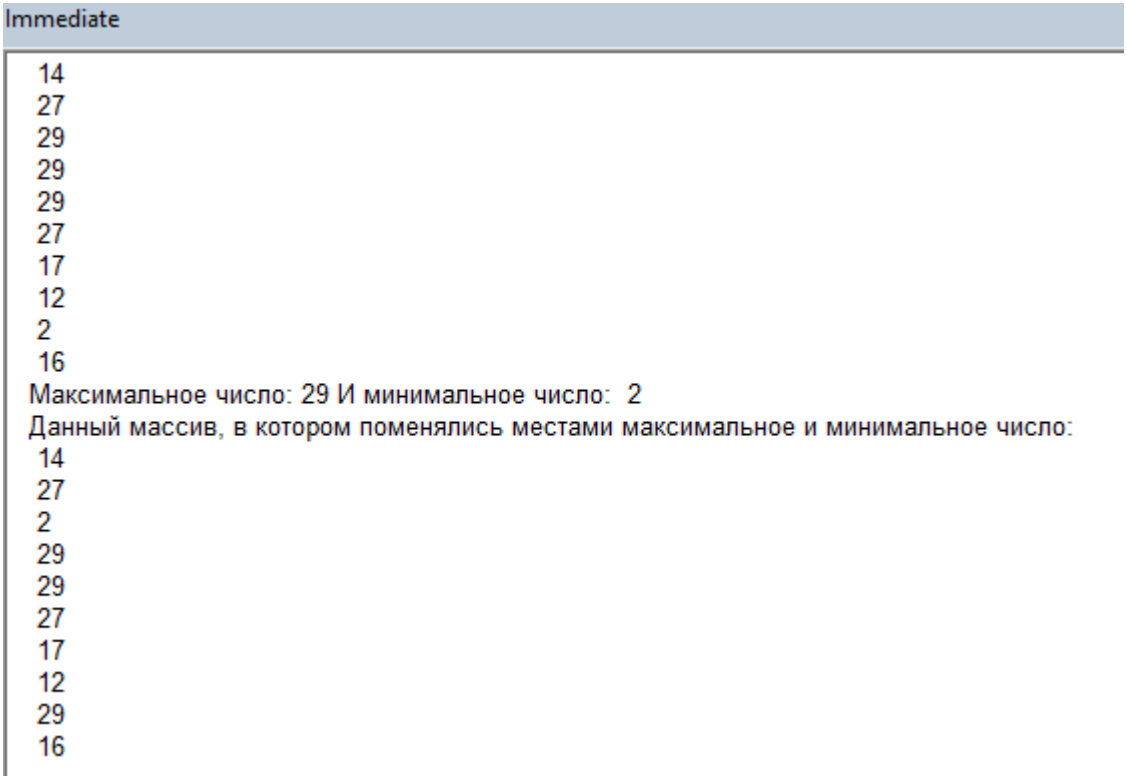

Рисунок 2 – Скриншот результата работы макроса задания 1

Задание 2. Окно VBA с кодом макроса представлен на рисунке 3.

Результат работы макроса представлен на рисунке 4.

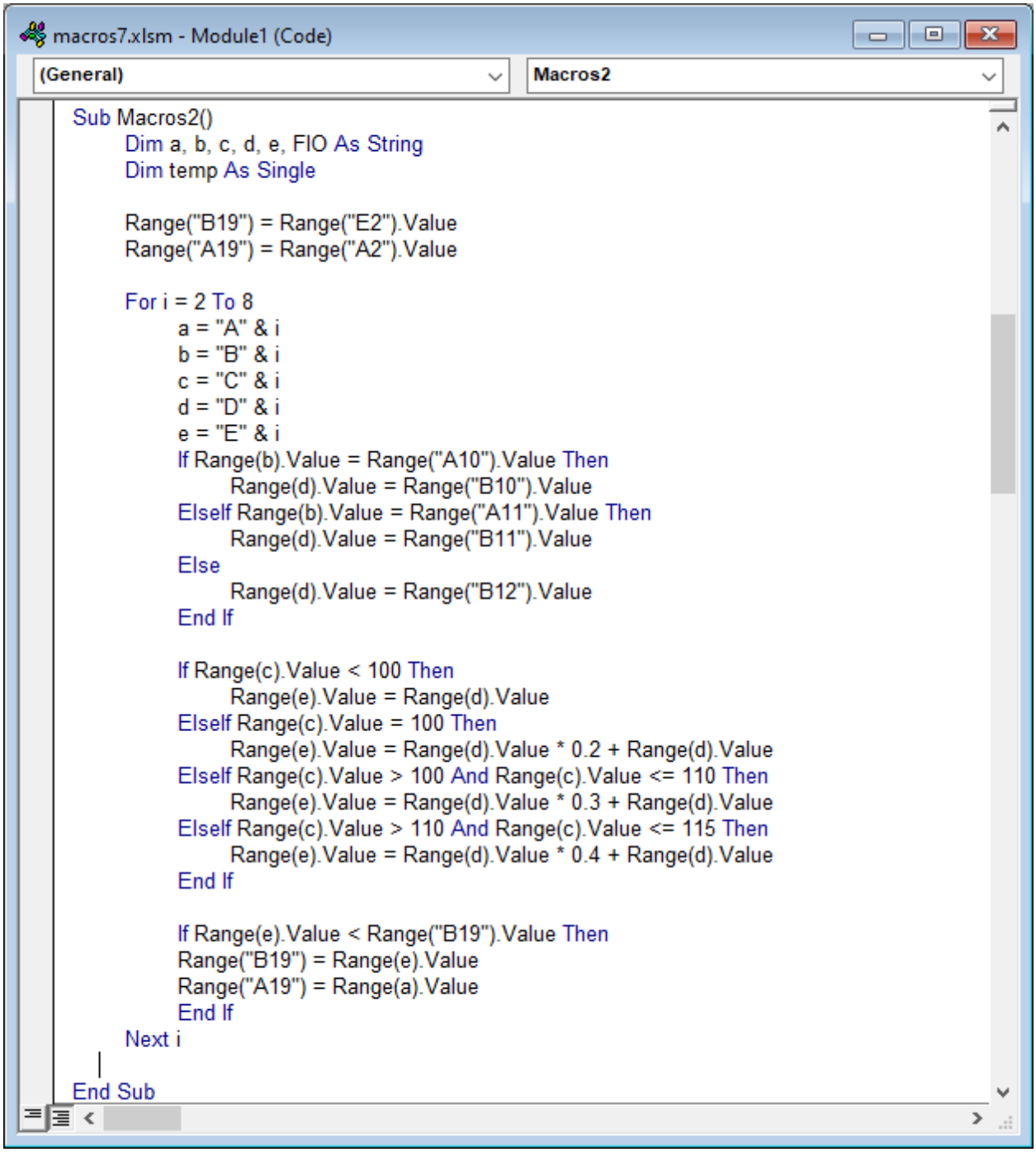

Рисунок 3 – Скриншот кода макроса задания 2

Ф.И.О	Тарифный разряд	% выполнения плана	Тарифная ставка, руб	Заработная плата с премий, руб
Пряхин Н.П.	3	102	8000	10400
Суховейко А.Д.	2	98	6500	6500
Лоханкин В.А.	1	114	4000	5600
Пферд Л.Ф.	3	110	8000	10400
Севрюгов Л.А.	1	100	4000	4800
Гигиенишвили Г.С.	2	94	6500	6500
Птибурдуков А.И.	3	100	8000	9600
1	4000			
2	6500			
3	8000			
<100	0			
100,00%	20			
101-110	30			
111-115	40			

Рисунок 4 – Скриншот результата работы макроса задания 2

Задание 3. Окно VBA с кодом макроса представлен на рисунках 5- 7. Диалоговое окно ввода показано на рисунке 8. Результат работы макроса представлен на рисунке 9.

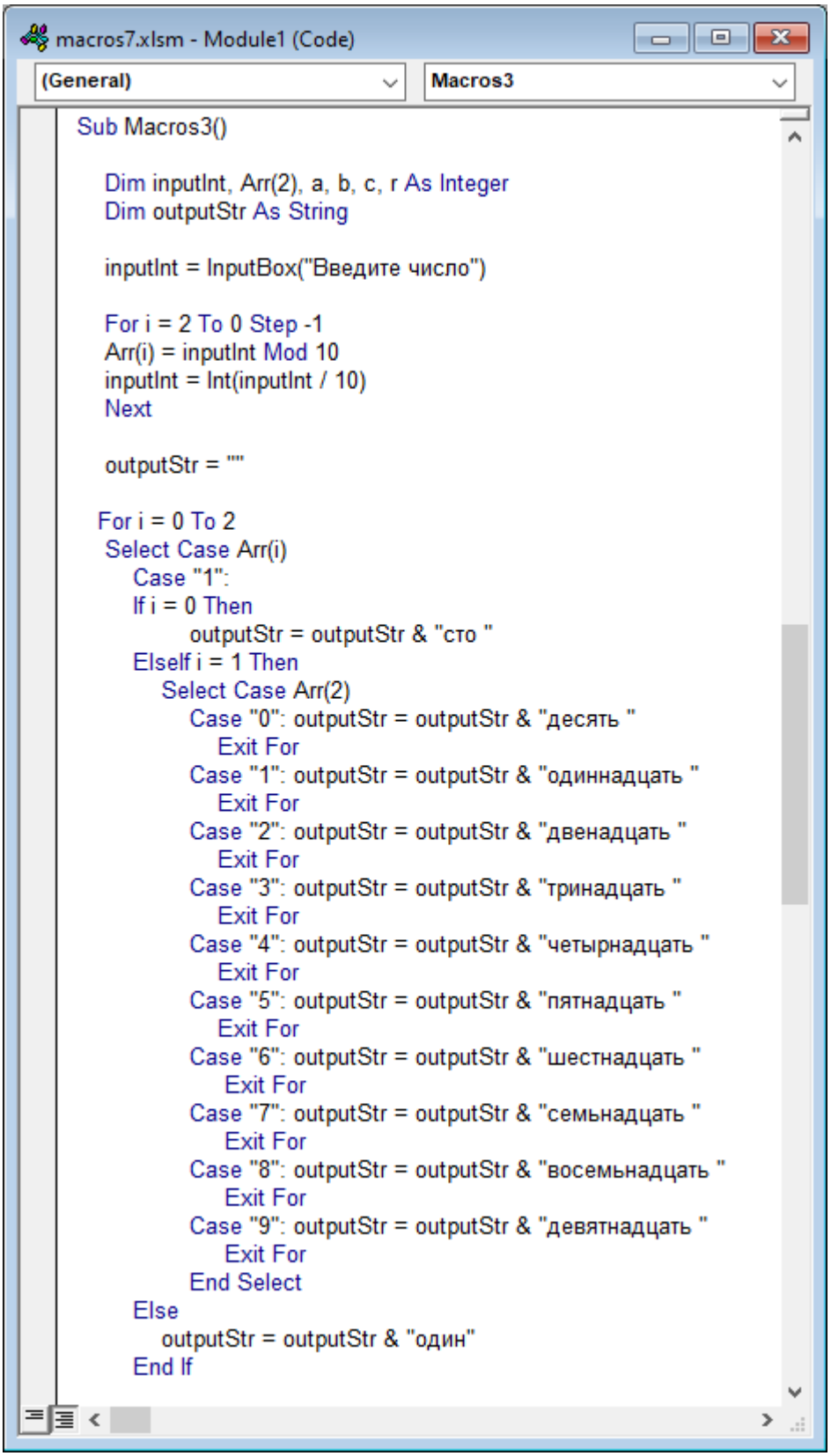

Рисунок 5 – Скриншот начала кода макроса задания 3

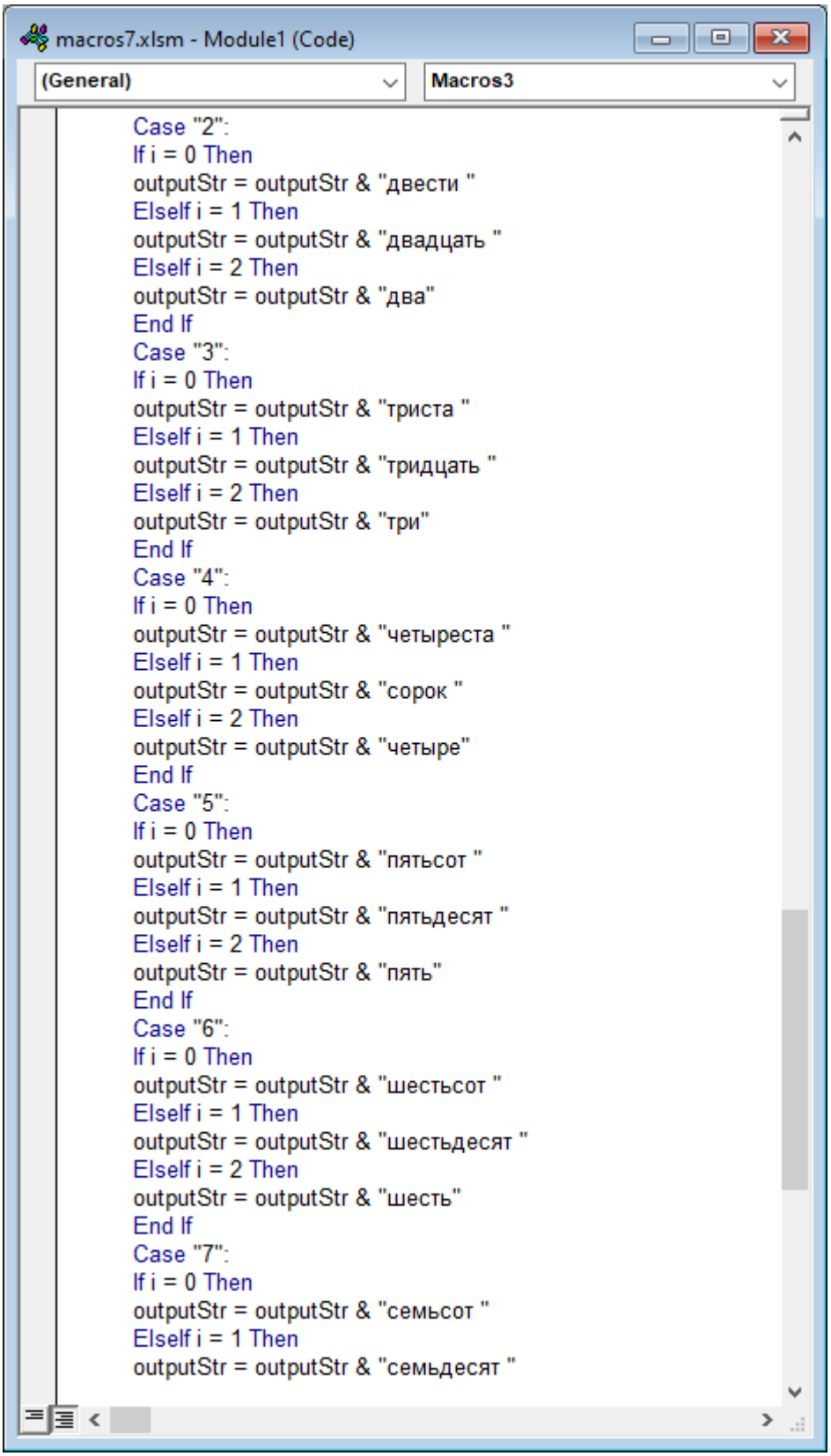

Рисунок 6 – Скриншот продолжения кода макроса задания 3

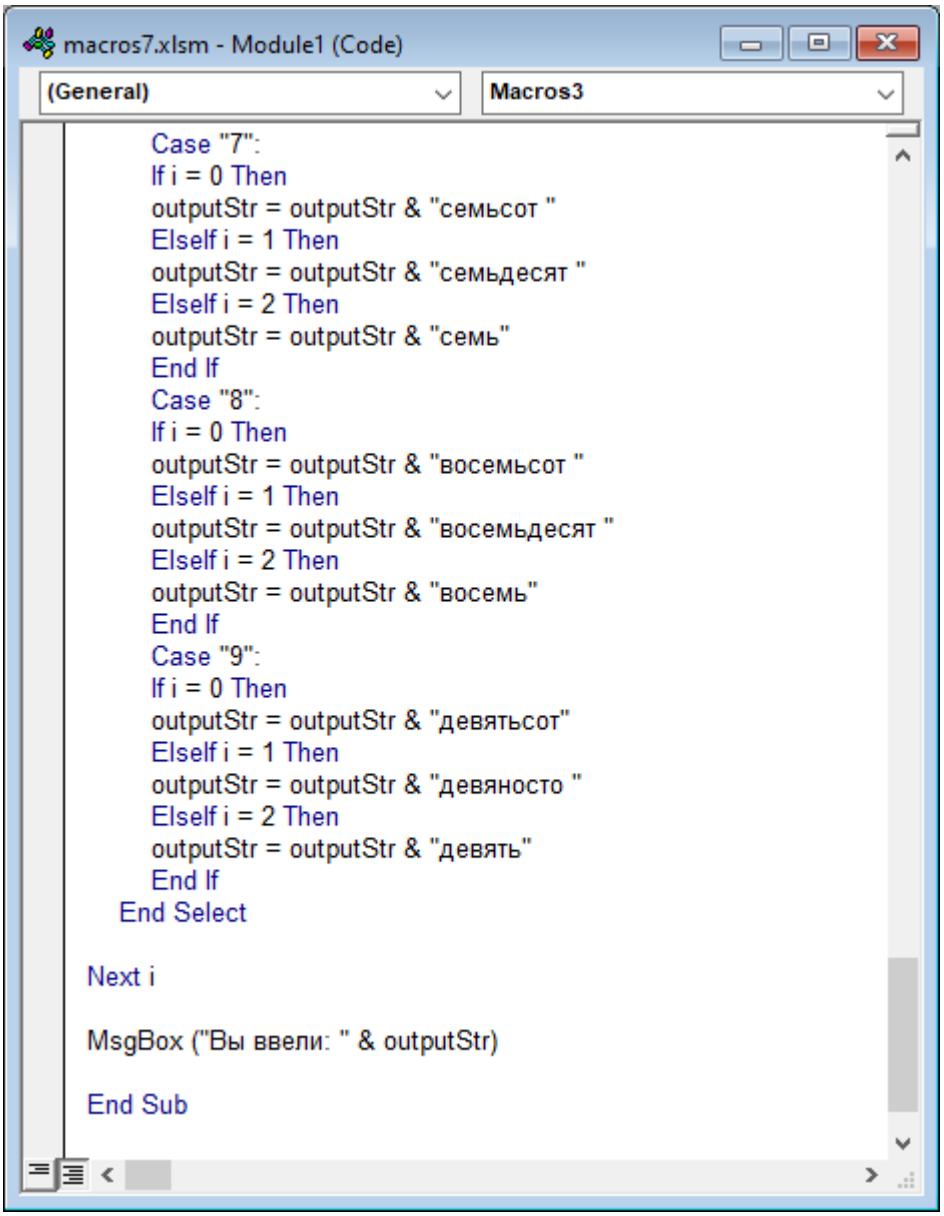

Рисунок 7 – Скриншот окончания кода макроса задания 3

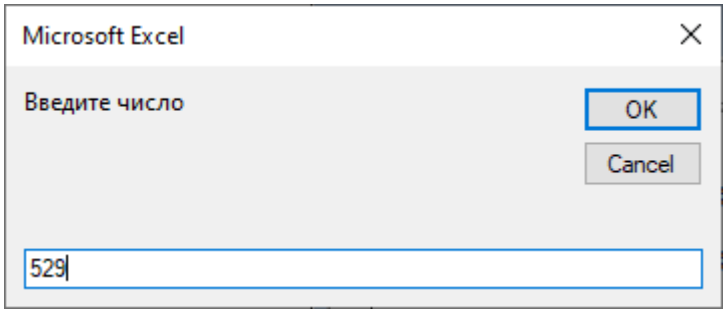

Рисунок 8 – Скриншот диалогового окна ввода

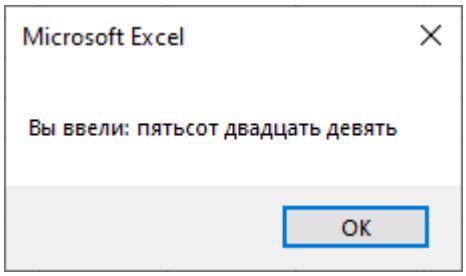

Рисунок 9 – Скриншот результата работы макроса задания 3

Вывод: Изучены основы программирования линейных и разветвляющихся вычислительных процессов.