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)

ТИПЫ ДАННЫХ И ПЕРЕМЕННЫЕ В VISUAL BASIC

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 № 6

Типы данных и переменные в visual basic

Цель работы

Приобрести навыки программирования с использованием встроенных функций ввода/вывода.

Теоретический материал, для освоения темы

Представление данных в памяти

Данные – величины, обрабатываемые программой. Имеется три основных вида данных: **константы, переменные и массивы**.

Константы – это данные, которые зафиксированы в тексте программы и не изменяются в процессе ее выполнения. Константы представляются в виде лексем, изображающих фиксированные числовые, логические, символьные или строковые значения.

Числовые константы могут быть целыми, вещественными (с фиксированной или плавающей точкой) и перечислимыми.

Целые константы могут быть десятичными, восьмеричными и шестнадцатеричными. Десятичная целая константа определена как последовательность десятичных чисел, начинающаяся не с нуля, если это не число нуль. Восьмеричные константы в Visual Basic for Application начинаются с префикса &O и содержат числа от 0 до 7. Шестнадцатеричные числа начинаются с префикса &H и содержат числа от 0 до 9 и латинские буквы от A до F.

Примеры констант: 123, &O247, &H1F.

Вещественные константы записываются в десятичной системе счисления и в общем случае содержат целую часть (десятичная целая константа), десятичную точку, дробную часть (десятичная целая константа),

признак (символ) экспоненты E и показатель десятичной степени (десятичная целая константа, возможно со знаком).

Примеры констант: 123.456, 3.402823E38.

Перечислимые константы – это набор обычных целочисленных констант. Перечисляемый набор может содержать конечный набор уникальных целых значений, каждое из которых имеет особый смысл в текущем контексте. Перечисляемые наборы являются удобным инструментом, обеспечивающим выбор из ограниченного набора параметров. Например, если пользователь должен выбрать цвет из списка, то можно установить соответствие: черный = 0, белый = 1 и т.д.

Логические (булевы) константы могут иметь лишь одно из двух значений: **да** (истина, **TRUE**), **нет** (ложь, **FALSE**).

Символьные и строковые константы. В отличие от большинства языков программирования, где существуют отдельно символьные (содержащие один символ алфавита) и строковые (массив символов) константы, в VBA существуют только строковые, имеющие два типа значений:

- Строки переменной длины, которые могут содержать до приблизительно 2 миллиардов (2^31) символов.
- Строки постоянной длины, которые могут содержать от 1 до приблизительно 64K (2^16) символов.

Примеры строковых констант: "abcde", "информатика", "" (пустая строка).

Типы данных

Тип данных определяет, каким образом биты данных, представляющие конкретное значение, хранятся в памяти ПК. В каждом языке программирования имеется свой фиксированный набор базовых типов

данных. Некоторые языки позволяют создание дополнительных (пользовательских) типов данных. В VBA имеются следующие типы данных:

Тип данных	**Размер**	**Диапазон значений**
Byte (байт)	1 байт	От 0 до 255.
Boolean (логический)	2 байт	True или False.
Integer (целое)	2 байт	От –32 768 до 32 767
Long (длинное целое)	4 байт	От –2 147 483 648 до 2 147 483 647
Single (с плавающей точкой обычной точности)	4 байт	От –3,402823E38 до –1,401298E-45 для отрицательных значений; от 1,401298E-45 до 3,402823E38 для положительных значений
Double (с плавающей точкой двойной точности)	8 байт	От –1,79769313486232E308 до –4,94065645841247E-324 для отрицательных значений; от 4,94065645841247E-324 до 1,79769313486232E308 для положительных значений
Currency (денежный)	8 байт	От –922 337 203 685 477,5808 до 922 337 203 685 477,5807

Тип данных	Размер	Диапазон значений
Decimal (масштабируемое целое)	14 байт	+/-79 228 162 514 264 337 593 543 950 335 без дробной части; +/-7,9228162514264337593543950335 с 28 знаками справа от запятой; минимальное ненулевое значение имеет вид +/-0,0000000000000000000000000001
Date (даты и время)	8 байт	От 1 января 100 г. до 31 декабря 9999 г.
Object (объект)	4 байт	Любой указатель объекта
String (строка переменной длины)	10 байт + длина строки	От 0 до приблизительно 2 миллиардов
String (строка постоянной длины)	Длина строки	От 1 до приблизительно 65 400
Variant (числовые подтипы)	16 байт	Любое числовое значение вплоть до границ диапазона для типа Double
Variant (строковые подтипы)	22 байт + длина строки	Как для строки (String) переменной длины
Тип данных, определяемый пользователем (с помощью	Объем определяется элементами	Диапазон каждого элемента определяется его типом данных

Тип данных	Размер	Диапазон значений
ключевого слова **Type**)		

Некоторые характерные для VBA типы данных

Byte – Массивы данного типа служат для хранения двоичных данных, например, изображений. Использование данного типа предохраняет двоичные данные во время преобразования формата.

Boolean – для хранения логических (булевых) значений. По умолчанию значением булевской переменной является **False** – ложь.

Currency – для хранения чисел с дробной частью до четырех цифр и целой частью до 15 цифр, то есть данных с фиксированной десятичной точкой, удобных для денежных вычислений. Числа с плавающей десятичной точкой (Single, Double) имеют больший диапазон значений, но могут приводить к ошибкам округления.

Date – используется для хранения как даты, так и времени в виде чисел с плавающей точкой. Дата может находиться в диапазоне от 1 января 100 года до 31 декабря 9999 года, а время в интервале от 0:00:00 до 23:59:59.Значения даты могут быть представлены в любом распознаваемом формате и должны ограничиваться знаками "#". Например: `#01/01/2011#`, `#01-01-2011#`.

Decimal – в версии 5.0 поддерживается использование типа данных Decimal только в пределах типа Variant, т.е. невозможно описать переменную с типом Decimal. Пользователь, однако, имеет возможность создать переменную типа Variant с подтипом Decimal с помощью функции CDec.

Object – поскольку VBA является объектно-ориентированным языком, в нем можно манипулировать различными объектами, адрес расположения которых в памяти (указатели) имеют этот тип.

String – по умолчанию данные строкового типа имеют переменную длину и могут удлиняться или укорачиваться. Однако такие строки занимают на 10байт памяти больше, поэтому можно объявить строки фиксированной длины, явно указав количество символов. Если количество символов будет меньше объявленного, то свободные места заполняются пробелами, при попытке занесения большего количества символов лишние отбрасываются.

Variant – может быть использован для хранения данных всех базовых типов без выполнения преобразования (приведения) типов. Применение данного типа позволяет выполнять операции, не обращая внимания на тип данных, которые они содержат. Удобен для объявления переменных, тип которых заранее неизвестен. Переменные этого типа могут содержать специальные значения: Empty, Null, Error.

Идентификаторы, переменные, массивы

Имена (идентификаторы) – употребляются для обозначения объектов программы (переменных, массивов, процедур и др.). В VBA имена констант, переменных и процедур должны удовлетворять следующим требованиям:

- должны начинаться с буквы;
- не могут содержать точки и символов объявления типа;
- не могут быть длиннее 255 символов. Длина имен объектов не должна превышать 40 символов.
- не могут быть ключевыми словами (именами операций, операторов, встроенных функций).

Переменные представляют собой зарезервированное место в памяти ПК для хранения значения. Переменные обозначаются именами – словами, используемыми для ссылки на значение, которое содержит переменная, и характеризуются типом, определяющим вид данных, которые можно хранить в переменной. Переменные могут изменять свои значения в ходе выполнения программы. По умолчанию переменные имеют тип данных Объявить переменную – значит заранее сообщить программе о ее существовании. Объявление переменной производится специальным оператором. Одновременно с объявлением переменной после ее имени можно записать ключевое слово **As**, после которого задается тип переменной.

Операции, выражения, операторы

Операции. В VBA существуют следующие типы операций:

- **арифметические** операции, используемые для выполнения математических вычислений: **^, *, /, \, Mod, +, -.** Здесь \ - Возвращает результат целого деления двух чисел, **Mod –** возвращает остаток при целом делении двух чисел (значение по модулю).
- операции **сравнения**, используемые для выполнения операций сравнения
- **< , > , <=, >= , = , <> ;**
- **логические** операции, используемые для выполнения логических операций

And – Возвращает результат конъюнкции (логического И) для двух выражений с операциями сравнения, либо выполняет поразрядное сравнение двух числовых выражений:

0	0	0

0	1	0
1	0	0
1	1	1

Eqv – Используется для проверки логической эквивалентности двух выражений с операциями сравнения, либо выполняет поразрядное сравнение двух числовых выражений:

0	0	1
0	1	0
1	0	0
1	1	1

Imp – Выполняет операцию логической импликации для двух выражений с операциями сравнения, либо выполняет поразрядное сравнение двух числовых выражений:

0	0	1
0	1	1

1	0	0
1	1	1

Not – Выполняет над выражением операцию логического отрицания, а также поразрядное изменение значений каждого разряда переменной:

0	1
1	0

Or – Выполняет операцию логического ИЛИ (сложения) для двух выражений:

0	0	0
0	1	1
1	0	1
1	1	1

Xor – Выполняет операцию исключающего ИЛИ для двух выражений:

0	0	0

0	1	1
1	0	1
1	1	0

- операция **конкатенации** символьных значений друг с другом с образованием одной длинной строки:

& – используется для слияния двух строковых выражений.

Выражения – предназначаются для выполнения необходимых вычислений, состоят из констант, переменных, функций (например, exp(*x*)), объединенных знаками операций.

Выражения записываются в виде **линейных последовательностей символов** (без подстрочных и надстрочных символов и т.д.), что позволяет вводить их в компьютер, последовательно нажимая на соответствующие клавиши клавиатуры.

Различают выражения **арифметические, логические и строковые.**

- **Арифметические выражения служат для определения одного числового значения**. Например, (1+sin(*x*))/2. Значение этого выражения при *x*=0 равно 0.5, а при *x*=π/2 – единице.

- **Логические выражения описывают некоторые условия, которые могут удовлетворяться или не удовлетворяться**. Таким образом, логическое выражение может принимать только два значения – **"истина"** или "**ложь"** (**да** или **нет**). Рассмотрим в качестве примера логическое выражение *x***x* + *y***y* < *r***r*, определяющее принадлежность точки с координатами (*x*,*y*)

внутренней области круга радиусом r с центром в начале координат. При *x*=1, *y*=1, *r*=2 значение этого выражения – **"истина"**, а при *x*=2, *y*=2, *r*=1 – **"ложь"**.

- **Значения строковых выражений – тексты**. В них могут входить литерные константы, литерные переменные и литерные функции, разделенные знаком операции сцепки. Например, A & B означает присоединение строки B к концу строки A. Если A = *"куст "*, а B = *"зеленый"*, то значение выражения A&B есть "*куст зеленый"*.

Операторы (команды). Оператор – это наиболее крупное и содержательное понятие языка: **каждый оператор представляет собой законченную фразу языка и определяет некоторый вполне законченный этап обработки данных.** В состав операторов входят:

- ключевые слова;
- данные;
- выражения и т.д.

Операторы подразделяются на исполняемые и неисполняемые. **Неисполняемые** операторы предназначены для описания данных и структуры программы, а **исполняемые** – для выполнения различных действий (например, оператор присваивания, операторы ввода и вывода, условный оператор, операторы цикла, оператор процедуры и др.).

Операторы описания

Объявление переменной производится одним из операторов **Dim**, **Static, Private, Public,** за которым следует имя переменной и необязательная часть с ключевым словом **As**, после которого задается тип переменной, например **Dim *name* [As type]**. Оператор **Public** используется только вне модуля, в его общей части и делает описываемую переменную доступной из всех процедур всех модулей проекта. Оператор **Private** служит для объявления переменной

уровня модуля, доступной только процедурам данного модуля. Можно использовать также оператор **Dim**, но применение **Private** предпочтительнее как противоположное **Public**.

Переменные могут быть объявлены внутри процедуры операторами **Dim** или **Static**. Такие переменные называют также локальными, поскольку доступны только в той процедуре, в которой они объявлены. Данное свойство (область видимости) позволяет использовать одинаковые имена переменных в разных процедурах, не опасаясь конфликтов или случайных изменений значений переменных. Время жизни локальных переменных, объявленных с помощью оператора **Dim** равно времени работы процедуры и по ее окончании значения таких переменных теряются.

Переменные, объявленные с помощью оператора **Static**, сохраняют свои значения в течение всего времени выполнения приложения. При повторном входе в процедуру, где описана такая переменная, ее значение сохраняется.

Операторы **Public** и **Private** можно применять при описании констант и процедур, что позволяет указать их область видимости.

Операторы присваивания

Инструкция **Let** присваивает значение выражения переменной или свойству:

[Let] имяПеременной = выражение

Явное использование ключевого слова **Let** необязательно.

Значение выражения может быть присвоено переменной, только если оно имеет совместимый с этой переменной тип данных. Невозможно присвоить строковое выражение числовой переменной или числовое выражение строковой переменной. Такая попытка приведет к ошибке во время компиляции.

Для ввода значений переменных в программу применяют функцию **InputBox(сообщение [, заголовок] [, значение по умолчанию] [, координата x] [, координата y])**. Эта функция отображает диалоговое окно, содержащее окно ввода, кнопки ОК и Отмена, сообщение (подсказку для ввода) и (необязательно) заголовок окна, значение, вводимое по умолчанию, координаты окна по горизонтали и вертикали в твиках. Заметим, что функция **InputBox** всегда (даже при нажатии кнопки Отмена) возвращает значение строкового типа, поэтому вызов ее должен иметь вид:

name = InputBox("Введи адрес ячейки", "Ввод", "a1", 100, 200)

Для отображения значений переменных в режиме нормальной работы необходимо ввести в тело программы вызов функции **MsgBox (сообщение, [кнопки, заголовок]).** Эта функция отображает диалоговое окно, содержащее сообщение длиной до 1024 символов, в которое с помощью операции конкатенации можно включить значение переменных, а также (необязательно) кнопки для реакции на отображения окна (по умолчанию только кнопка ОК) и заголовок окна (строковое выражение). Пример:

MsgBox "Значение val=" & val

Заметим, что VBA не имеет встроенных функций ввода/вывода в документ. Поэтому для вывода значений выражений и переменных в активный документ приходится создавать пользовательские процедуры.

Функции обработки строк

В VBA имеется несколько функций по работе со строковыми выражениями.

Функции **Asc(строка)** и **Chr(код)** позволяют получить ASCII-код начальной буквы строки и, наоборот, по ASCII-коду получить соответствующую букву. Для функции **Chr()** код может принимать значения

от 0 до 255. Значения от 0 до 31 соответствуют управляющим кодам (например, **Chr(13)** вернёт символ перехода на новую строку). Для обозначения некоторых часто употребляемых клавиш в VBA имеются встроенные константы, например, для клавиши <**Enter**> – **vbCr**, для клавиши <**Tab**> – **vbTab**, для клавиши <**Backspace**> – **vbBack**.

Примеры:

```
debug.Print Asc("F")
 70
debug.Print Chr(97)
a
Msgbox "Этот текст расположен" & vbCr & "в две строки".
```

Функция **Len(строка)** определяет длину строки.

Пример:

```
debug.Print Len("http://www.tusur.ru/")
 20
```

Функции **Left(строка, количество)** и **Right(строка, количество)** возвращают подстроки, состоящие из заданного количества соответственно первых и последних символов данной строки. Функция **Mid(строка, позиция, количество)** возвращает подстроку, содержащую заданное количество символов, начиная с указанной позиции.

Пример:

```
debug.Print Left("http://www.tusur.ru", 4)
http
debug.Print Right("http://www.tusur.ru", 2)
ru
debug.Print Mid("http://www.tusur.ru", 12, 5)
```

```
tusur
```

Функции **InStr([старт], строка1, строка2, [сравнение])** и **InStrRev(строка1, сторка2, [старт], [сравнение])** возвращают позицию первого вхождения строки2 в строку1, начиная соответственно с ее начала и с конца. Если вхождения нет, то возвращается 0. Параметр сравнение указывает способ сравнения строк, значения: 0 или **vbBinaryCompare** для двоичного сравнения, 1 или **vbTextCompare** для посимвольного сравнения без учета регистра.

```
debug.Print InStr(1, "Microsoft Office", "office", vbTextCompare)
 11
debug.Print InStr(1, "Microsoft Office", "office", vbBinaryCompare)
 0
```

Функции обработки даты и времени

В VBA имеется несколько функций даты и времени, которые позволяют производить самые разнообразные действия от определения текущей даты до сложения нескольких дат.

Функции **Date**, **Time** и **Now** возвращают значения типа **Variant (Date)**, содержащие соответственно текущие дату, время и одновременно дату и время.

Пример:

```
debug.Print Date
20.10.2011
debug.Print Time
13:18:02
```

```
debug.Print Now
20.10.2011 13:18:02
```

Функции **Hour(время)**, **Minute(время)**, **Second(время)** и **Day(дата)**, **Month(дата)**, **Year(дата)** возвращают значения типа **Variant (Integer)**, являющиеся целыми числами, которые представляют собой соответственно час, минуту, секунду и день, месяц, год в значении даты.

Пример:

```
debug.Print Hour(Now), Minute(Now), Second(Now)
 13            18            55
debug.Print Day(Now), Month(Now), Year(Now)
 20            10            2011
```

Функция **DatePart(интервал, дата [, день_недели, неделя_года]** возвращает значение типа **Variant (Integer)**, содержащее указанную часть даты. Параметр интервал – строковое значение, обозначающее какой временной интервал должен быть найден, допустимые значения – "yyyy" (год), "q" (квартал), "m" (месяц), "y" (день года), "d" (день месяца), "w" (день недели), "ww" (неделя), "h" (час), "m" (минута), "s" (секунда); параметр дата – дата, часть которой необходимо найти; параметры день_недели и неделя_года – необязательные параметры, указывающие первый день недели и первую неделю года.

Пример:

```
debug.Print DatePart("w", #01-01-2011#)
 7
```

Функция **DateDiff(интервал, дата1, дата2 [, день_недели, неделя_года]** возвращает значение типа **Variant (Long)**, содержащее временной интервал между двумя датами. Параметр интервал – строковое значение, обозначающее какой временной интервал необходимо различать,

допустимые значения – "yyyy" (год), "q" (квартал), "m" (месяц), "y" (день года), "d" (день месяца), "w" (день недели), "ww" (неделя), "h" (час), "m" (минута), "s" (секунда); параметры дата1 и дата2 – две даты, разность между которыми необходимо найти; параметры день_недели и неделя_года – необязательные параметры, указывающие первый день недели и первую неделю года.

Пример:

```
debug.Print DateDiff("yyyy", #01-10-2000#, #01-09-2011#)
 11
```

Функция **DateAdd(интервал, количество, дата)** возвращает значение типа **Variant (Date)**, содержащее дату, которой добавлено указанное количество времени. Параметр интервал – строковое значение, обозначающее какой временной интервал необходимо добавить, допустимые значения – "yyyy" (год), "q" (квартал), "m" (месяц), "y" (день года), "d" (день месяца), "w" (день недели), "ww" (неделя), "h" (час), "m" (минута), "s" (секунда); параметр количество – число, обозначающее количество времени, которое необходимо добавить к параметру дата.

Пример:

```
debug.Print DateAdd("yyyy", 5, #01-10-2000#)
10.01.2005
```

Некоторые функции проверки типов

Функции проверки типов необходимы в то случае, когда нужно определить, является ли значение переменной значением необходимого типа.

Функция	Проверяемый тип
IsArray	если переменная является массивом – возвращает True, иначе – False
IsDate	если переменная содержит дату – возвращает True, иначе – False
IsEmpty	была ли переменная инициализирована – возвращает True, иначе – False
IsNull	если переменная содержит какое-либо значение – возвращает True, иначе – False
IsNumeric	если переменная является числом – возвращает True, иначе – False

Функции преобразования типов

Функция	Возвращаемый тип
CBool	Boolean
CByte	Byte

CCur	Currency
CDate	Date
CDbl	Double
CDec	Decimal
CInt	Integer
CLng	Long
CSng	Single
CVar	Variant
CStr	String

Форматирование значений разных типов

Функция **Format(выражение [, формат, день_недели, неделя_года])** возвращает значение типа **Variant (String)**, содержащее выражение, отформатированное согласно заданному формату; параметры **день_недели** и **неделя_года** – необязательные параметры, указывающие первый день недели и первую неделю года.

Ниже перечислены некоторые форматы для представления значений.

Формат	Описание
General Number	Число без разделителя тысяч
Currency	Число в денежном формате, используя настройки операционной системы
Fixed	Число, у которого отображается хотя бы одна цифра слева и две справа от десятичного символа
Standard	Число, у которого отображается хотя бы одна цифра слева, две справа от десятичного символа и разделитель тысяч
Percent	Число в процентном формате с двумя цифрами справа от десятичного символа
Scientific	Число в формате с плавающей точкой
Yes / No	No, если число равно 0, Yes в противном случае
True / False	False, если число равно 0, True в противном случае

Формат	Описание
On / Off	Off, если число равно 0, On в противном случае
General Date	Отображает дату или время
Long Date	Дата в полном формате
Medium Date	Дата в обычном формате
Short Date	Дата в сокращенном формате
Long Time	Время с часами, минутами и секундами
Medium Time	Время с часами и минутами в 12-часовом формате
Short Time	Время с часами и минутами в 24-часовом формате

Примеры:

```
debug.Print Format(456789.0123, "General Number")
456789,0123
debug.Print Format(456789.0123, "Currency")
456 789,01р.
debug.Print Format(456789.0123, "Fixed")
456789,01
```

```
debug.Print Format(456789.0123, "Standard")
456 789,01
debug.Print Format(456789.0123, "Scientific")
4,57E+05
debug.Print Format(#01-01-2011#, "General Date")
01.01.2011
debug.Print Format(#01-01-2011#, "Long Date")
1 Январь 2011 г.
debug.Print Format(#01-01-2011#, "Medium Date")
01-янв-11
debug.Print Format(#01-01-2011#, "Short Date")
01.01.2011
debug.Print Format(#13:50:50#, "Long Time")
13:50:50
debug.Print Format(#13:50:50#, "Medium Time")
01:50
debug.Print Format(#13:50:50#, "Short Time")
13:50
```

Если же нужного формата нет, можно настроить вид отображения выводимого значения при помощи пользовательских форматов с использованием специальных символов "0", "#", "%", ",", ":", "/", ("E+", "E–", "e+", "e–"), ("d", "m", "s"), ("h", "m", "s").

Примеры:

```
debug.Print Format(125 / 2, "###.###")
62,5
debug.Print Format(125 / 2, "000.000")
062,500
debug.Print Format(125 / 2, "#.##e+##")
```

```
6,25e+1
debug.Print Format(#01-01-2011#, "dd/mm/yy")
01.01.11
debug.Print Format(#13:50:50#, "hh:mm:ss")
13:50:50
```

Для форматирования чисел в VBA имеется отдельная функция **FormatNumber(число [, число_знаков, ведущий_ноль, отрицательные, группировать])**, где **число** – то число, которое нужно отформатировать; **число_знаков** – параметр, задающий число знаков после десятичного символа (значения – vbTrue, vbFalse, vbUseDefault); **ведущий_ноль** – параметр, указывающий надо ли отображать нулевую целую часть (значения – vbTrue, vbFalse, vbUseDefault); **отрицательные** – параметр, указывающий, надо ли отображать отрицательные значения в скобках (значения – vbTrue, vbFalse, vbUseDefault); **группировать** – параметр, указывающий, надо ли группировать цифры (значения – vbTrue, vbFalse, vbUseDefault).

Примеры:

```
debug.Print FormatNumber(sin(5), 4)
-0,9589
debug.Print FormatNumber(sin(5), 4, vbFalse)
-,9589
debug.Print FormatNumber(sin(5), 4, vbFalse, vbTrue)
(,9589)
```

Для форматирования процентов в VBA имеется отдельная функция **FormatPercent**, которая имеет такой же синтаксис, как и **FormatNumber**.

Для форматирования денежных значений в VBA имеется отдельная функция **FormatCurrency**, которая имеет такой же синтаксис, как и **FormatNumber**.

Для форматирования значений даты и времени в VBA имеется отдельная функция **FormatDateTime(дата[, формат])**, которая имеет такой же синтаксис, как и **FormatNumber**. где дата – параметр, задающий дату, которую необходимо отформатировать; формат – необязательный параметр, указывающий нужное форматирования (значения – vbGeneralDate, vbLongDate, vbShortDate, vbLongTime, vbShortTime).

Отладка, использование среды для отладки программ

Поскольку идеальных программистов не существует, в большинстве систем разработки программ имеются инструменты, с помощью которых можно решить проблемы, возникающие в процессе программирования. В VBA также есть средства, которые позволяют либо исключить ошибки при разработке, либо задать обработку ошибок при выполнении программ.

Отладка программ – это проверка и внесение исправлений в программу при её разработке. Отладка позволяет идентифицировать ошибки, допущенные при программировании (синтаксические – ошибки в выражениях и именах, и логические – в логике работы программы).

Обработка ошибок – это задание реакции на ошибки, которые возникают при выполнении программы. Их причиной могут быть как ошибки программиста, так и внешние факторы – отсутствие нужных файлов, отказы аппаратуры, неправильные действия пользователя.

Типы ошибок. Ошибки в программе делятся на три категории:

Ошибки компиляции – возникают, когда компилятор не может интерпретировать введённый текст. Некоторые ошибки компиляции обнаруживаются при вводе, а другие – перед выполнением программы. Такие ошибки легко определить и исправить, поскольку VBA выявляет их автоматически, а сами ошибки очевидны.

Примечание. VBA автоматически компилирует программу каждый раз при запуске на выполнение после внесения изменений. Можно также запустить компиляцию командой *Отладка – Компилировать*.

Ошибки выполнения – возникают при выполнении программы после успешной компиляции. Их причиной обычно является отсутствие данных или неправильная информация, введённая пользователем. Такие ошибки идентифицируются VBA с указанием инструкции, при выполнении которой произошла ошибка. Для исправления таких ошибок обычно приходится выводить значения переменных или другие данные, которые влияют на успешное выполнение программы.

Логические ошибки трудно заметить и устранить. Они не приводят к прекращению компиляции или выполнения, однако являются причиной того, что программа не выдаёт желаемых результатов. Выявление таких ошибок производят путём тщательной проверки с помощью средств отладки VBA.

Средства отладки. В VBA имеется большое количество средств, предназначенных для отладки программ. К ним относятся: использование оператора **Option Explicit**, пошаговое выполнение программы, работа в режиме прерывания, использование точек останова, вывод значений переменных.

Использование Option Explicit. Данный оператор описания требует явного задания переменных в программах. При его использовании возникает ошибка компиляции при неправильном написании имени переменной или использовании неописанной переменной. Кроме того, явное описание переменных позволяет обойтись без использования типа данных Variant и связанных с его использованием ошибок при неявном приведении типов данных.

Пошаговое выполнение программы. Этот режим служит для локализации ошибок в теле программы. Для его запуска используются команды меню, клавиатура или панель инструментов *Отладка*, отображаемая командой *Вид* - *Панели инструментов* - *Отладка*. В меню и на панели инструментов имеются четыре команды (кнопки) для выполнения программы в пошаговом режиме.

- Команда *Отладка – Шаг с заходом* (клавиша <**F8**>) позволяет выполнить одну строку программы и перейти к следующей. Если следующая строка – вызов процедуры, то происходит переход к первому выполняемому оператору этой процедуры.
- Команда *Отладка – Шаг с обходом* (клавиши <**Shift+F8**>) также выполняет одну строку программы, но если строкой является вызов процедуры, то она выполняется как одна инструкция. Данная команда используется, если известно, что эта процедура работает правильно.
- Команда *Отладка – Шаг с выходом* (клавиши <**Ctrl+Shift+F8**>) заканчивает выполнение текущей процедуры и останавливается на следующей после вызова текущей процедуры инструкции в вызывающей подпрограмме.
- Команда *Отладка – Выполнить до текущей позиции* (клавиши <**Ctrl+F8**>) выполняет программу от текущей до выбранной инструкции. Перед выбором данной команды требуется установить курсор в окне модуля на требуемую позицию.

Работа в режиме прерывания. Переход в данный режим выполняется:

- При нажатии кнопки *Отладка* в окне сообщения об ошибке выполнения.
- При прерывании работы программы нажатием клавиш <**Ctrl+Break**>. Текущая строка программы выделяется в окне модуля.
- По достижении точки останова.

- По достижении оператора Stop.
- При пошаговом выполнении программы.

В режиме прерывания можно:

- Вывести значение переменной.
- Вычислить выражение в окне отладки.
- Сбросить программу
- Выполнить программу в пошаговом режиме.
- Продолжить выполнение программы.

Для выхода из режима прерывания используется команда *Запуск – Сброс*.

Использование точек останова. *Точка останова* – это строка в процедуре, на которой приостанавливается выполнение программы. Все команды, находящиеся выше точки останова, выполняются с обычной скоростью, а по достижении контрольной точки программа переходит в режим прерывания. Затем можно отлаживать процедуру в пошаговом режиме, либо использовать различные способы вывода значений переменных. Кроме того, имеется возможность остановить выполнение или сбросить процедуру командами меню *Запуск* или кнопками панели инструментов *Отладка*. В одном проекте можно задать несколько точек останова, причём в различных процедурах.

Чтобы установить или снять точку останова, используется команда *Отладка – Точка останова* или клавиша <**F9**>, либо кнопка *Точка останова* панели инструментов *Отладка*. Можно также установить или снять точку останова, щелкнув левой кнопкой мыши на полосе индикатора против требуемой строки. Точка останова отмечается коричневой жирной точкой на полосе индикатора, а сама строка выделяется коричневым цветом.

Чтобы быстро удалить все точки останова открытого проекта, используется команда *Отладка – Снять все точки останова* или комбинация клавиш <**Ctrl+Shift+F9**>. Для данной команды не предусмотрена кнопка на панели инструментов *Отладка*.

Вывод значений переменных. При наличии тестового примера вывод значений переменных позволяет сравнить ожидаемые и полученные значения переменных. Для отображения значений переменных в режиме прерывания необходимо:

- При установленном флажке *Подсказки значений переменных* в окне *Сервис – Параметры* достаточно переместить указатель мыши на требуемую переменную для отображения имени и значения переменной во всплывающей подсказке.
- Выбрать команду *Отладка – Контрольное значение* (нажать клавиши <**Shift+F9**>) для вывода диалогового окна *Контрольное значение*. При этом курсор должен находиться возле переменной, значение которой надо контролировать. В окне *Контрольное значение* отображается контекст (имя модуля и процедуры), выделенное выражение (переменная) и кнопки *Добавить* и *Отмена*. При нажатии кнопки *Добавить* откроется окно *Контрольные значения*, содержащее имена переменных (выражения), их значения, тип данных и контекст.
- Для добавления других контрольных значений используется команда *Отладка – Добавить контрольное значение*.
- Выбрать команду *Вид – Окно локальных переменных*. Откроется окно *Локальные переменные*, в котором в режиме прерывания отображаются имена, значения и типы всех переменных модуля.

Выбрать команду *Вид – Окно отладки*. В нем немедленно выполняется введённая в него инструкция, обычно операция отображения значения выражения вида Print имя, или операция присваивания значения переменной.

Задания на практическую работу

Задание 1. Составить программу, которая переводит Дюймы в метры (1 дюйм = 2,54 см). Исходные данные вводятся с клавиатуры, результат выводится на экран с необходимым форматированием выводимых значений.

Задание 2. Составить программу, присваивающую все вычисляемые значения строковой переменной через соответствующие функции. Переменную добавить в *Контрольное значение* и отслеживать её изменение. Необходимо вычислить количество следующих значений между датой своего рождения и текущей датой: количество дней, недель, месяцев, кварталов, лет. Выполнить программу в пошаговом режиме. В отчете представить скриншоты (5-6) порядка выполнения программы в пошаговом режиме.

Ход работы

Задание 1. Окно VBA с кодом программы показаны на рисунке 1. Диалоговое окно с вводом и результат работы макроса показаны на рисунках 2-3

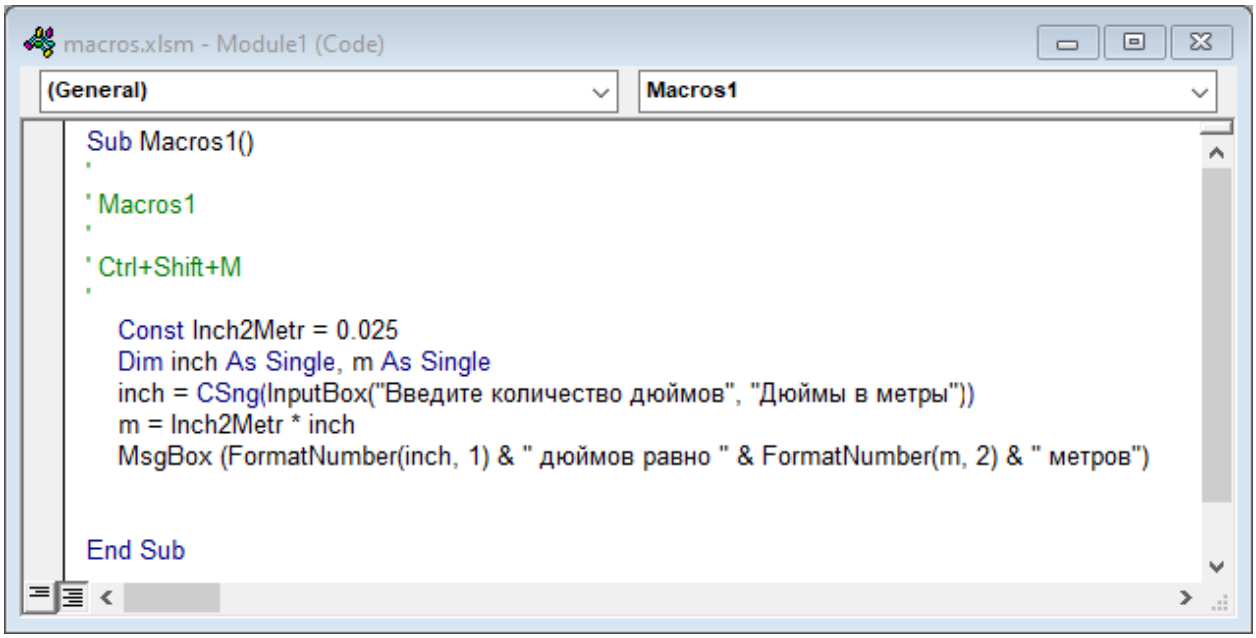

Рисунок 1 – Скриншот кода программы задания 1

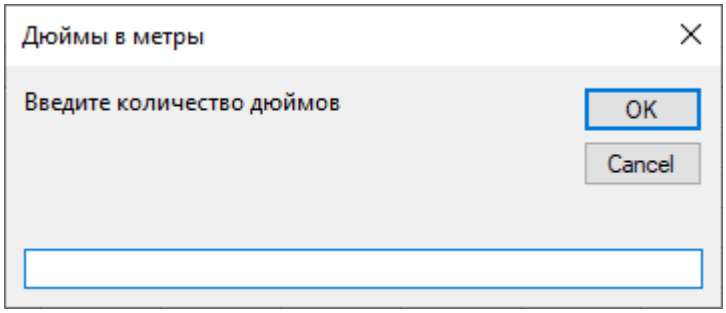

Рисунок 2 – Скриншот диалогового окна задания 1

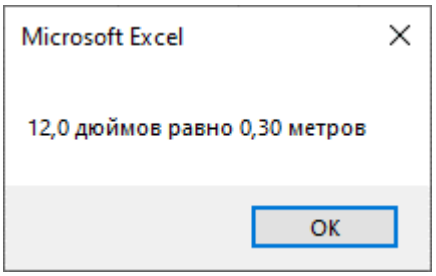

Рисунок 3 – Скриншот результата работы макроса задания 1

Задание 2. Окно VBA с кодом программы показаны на рисунке4. Диалоговое окно с вводом, ход работы отладчика и результат работы макроса показаны на рисунках 5-10

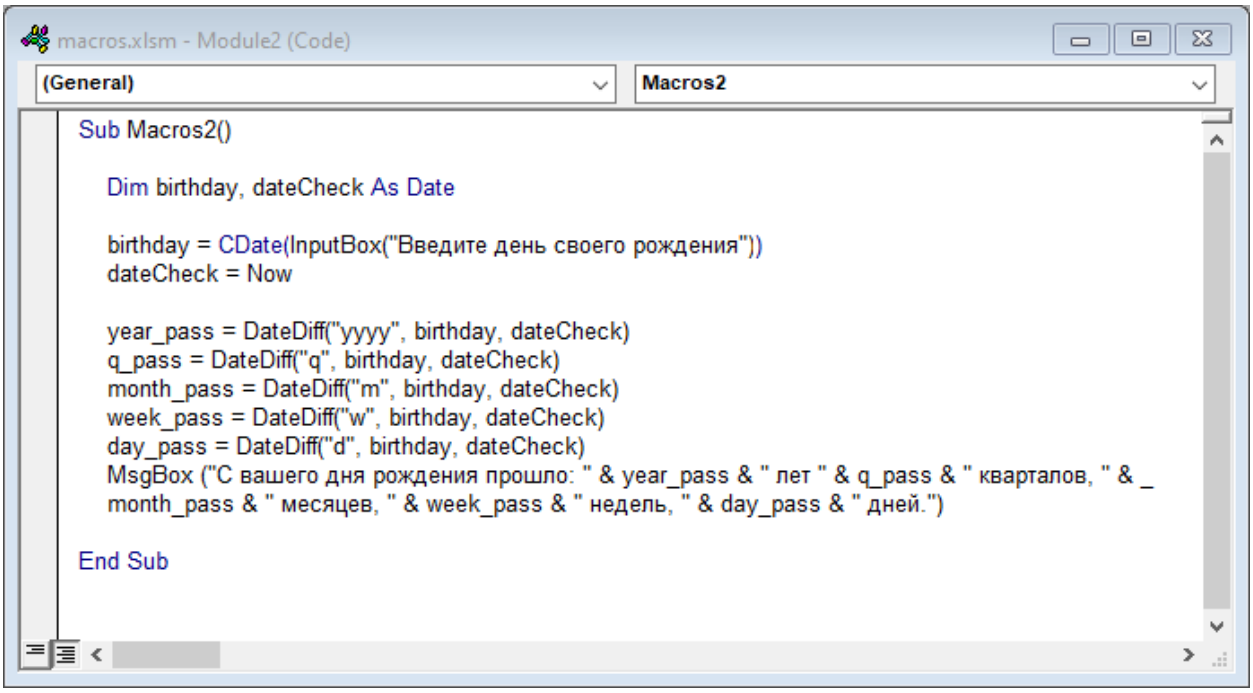

Рисунок 4 – Скриншот кода программы задания 2

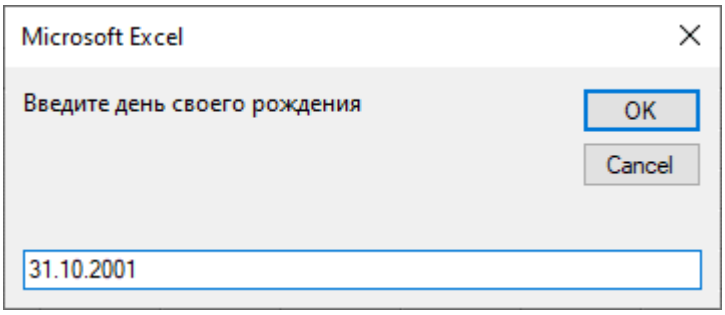

Рисунок 5 – Скриншот диалогового окна для ввода

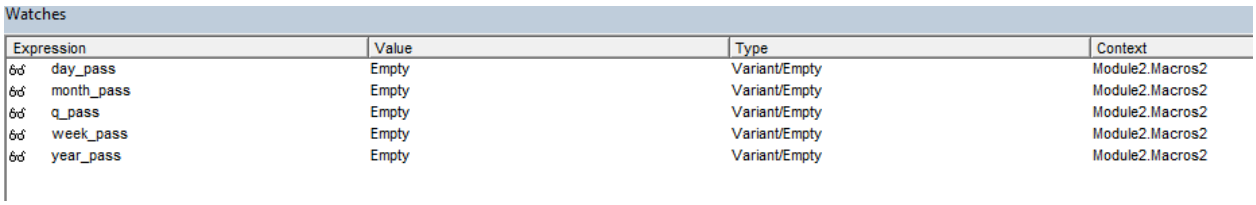

Рисунок 6 – Скриншот инициализации переменных в контрольных значениях

Рисунок 7 – Скриншот с изменением переменной year_pass

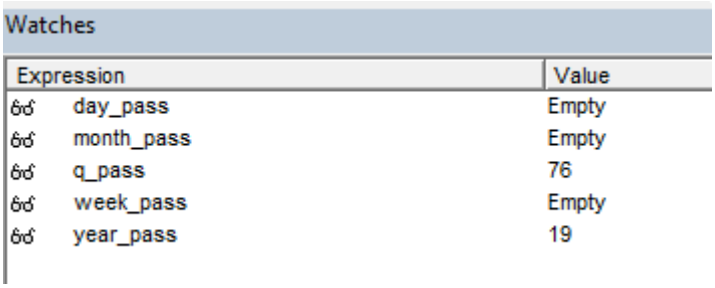

Рисунок 8 – Скриншот с изменением переменной q_pass

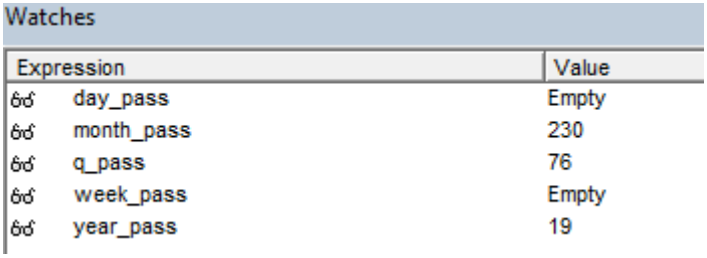

Рисунок 9 – Скриншот с изменением переменной month_pass

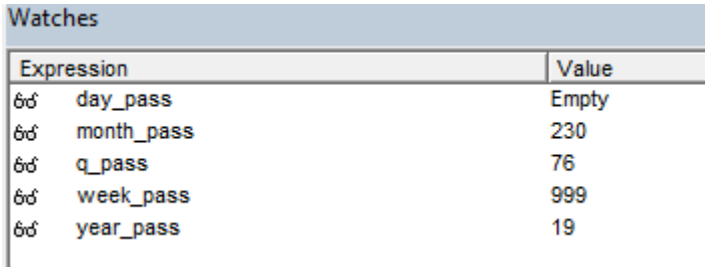

Рисунок 10 – Скриншот с изменением переменной week_pass

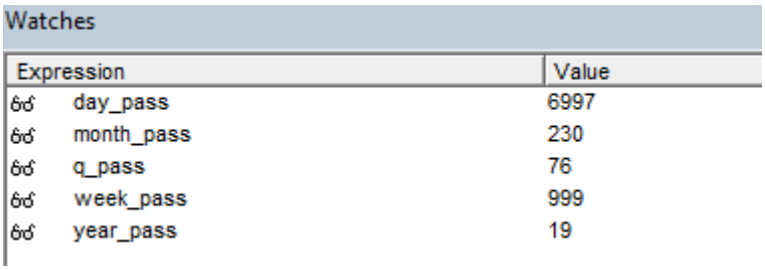

Рисунок 11 – Скриншот с изменением переменной day_pass

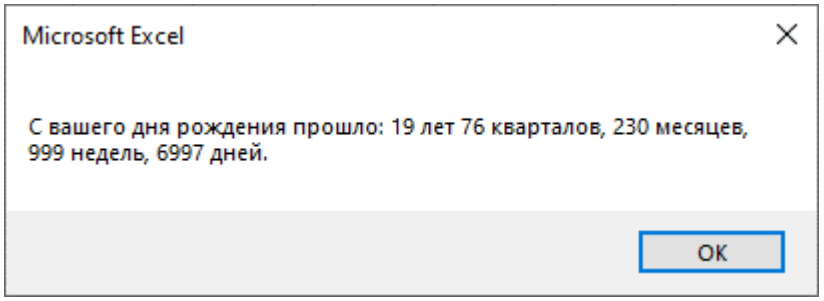

Рисунок 12 – Скриншот результата работы макроса задания 2

Вывод: приобретены навыки программирования с использованием встроенных функций ввода/вывода.