ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ
А.Г. Степанов

ВЫСОКОУРОВНЕВЫЕ МЕТОДЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

Методические указания
для выполнения лабораторных работ

Санкт-Петербург
2006

УДК 681.3

Степанов А.Г. Высокоуровневые методы информатики и программирования // Методические указания для выполнения лабораторных работ. СПб. 2006.

Приводится цикл лабораторных работ по дисциплине «Высокоуровневые методы информатики и программирования». Рассматривается элементарное взаимодействие Excel и VBA и собственно программирование на VBA Выделены вопросы, касающиеся стандартных классов и объектов при взаимодействии Excel и VBA и программирования с использованием библиотечных классов VBA, связанных с Excel. Методические указания предназначены для студентов, обучающихся по специальности 351400 «Прикладная информатика в экономике», и могут быть использованы в составе дисциплины «Информатика» при подготовке по экономическим специальностям, для которых изучение языков программирования высокого уровня является обязательным.

Рецензент: доцент кафедры информационных систем и технологий Международного банковского института кандидат педагогических наук Титова Ю. Ф.

Содержание

Элементарное взаимодействие Excel и VBA

Лабораторная работа №1

Разработка пользовательской таблицы средствами процессора Excel, создание и выполнение макросов Excel

Методические указания

Мы предполагаем, что у вас уже есть начальные знания по работе с табличным процессором Excel. Мы считаем, что вы знакомы с понятиями *рабочей книги*, *рабочего листа* Excel и что вы умеете ими пользоваться (создавать, удалять переименовывать, вставлять и т.п.). Мы исходим из того, что вы знакомы с относительным и абсолютным способами адресации *ячеек* рабочего листа Excel, умеете задавать и осознанно выбирать *формат* ячейки, знакомы со способами ее оформления (*шрифт*, *фон*, *рамки*). Мы считаем, что вы умеете *программировать формулы* в Excel и пользоваться *встроенными функциями* Excel. Наконец, мы предполагаем, что вы в состоянии придумать собственную пользовательскую таблицу, данные в которой организованы по строкам и столбцам, имеют вполне определенный практический смысл и требуют некой обработки, в частности, вычислений. Вы также в состоянии набрать ее на рабочем листе и задать форматы ячеек **ФОРМАТ, Ячейки…**, в том числе тип данных (вкладка *Число*), выполнить *Выравнивание* в ячейке, задать *Шрифт*, сделать обрамление ячейки (вкладка *Граница*), сделать заливку ячеек (вкладка *Вид*).

Термином *макрос* обычно называют файл, хранящий последовательность действий, заданных пользователем системы. Каждый макрос должен иметь собственное имя. С помощью макроса можно автоматизировать типовые технологические этапы при работе с системой. Если макрос создан, то после его запуска хранящаяся в нем последовательность действий (команд) будет автоматически исполнена. По своей сути макрос представляет собой программу и может быть создан автоматически в специальном режиме работы программной системы (в том числе и *Excel*) или как результат программирования в терминах языка системы. Если пользователь владеет языком задания макроса, то созданный любым способом макрос может

быть подвергнут редактированию с целью изменения его возможностей или устранения ошибок. В пакете Microsoft Office таким языком является язык VBA.

При работе с Excel, как, впрочем, и с другими программами пакета Microsoft Office, для создания макроса легче всего использовать автоматический режим его создания, вызываемый из главного меню системы командами **СЕРВИС, Макрос**. При первоначальном запуске системы макросы отсутствуют, поэтому диалоговое окно <<Макрос>>, вызываемое пунктом **Макросы…** показывает пустой список. Пункт меню **Безопасность…** открывает дополнительное меню, позволяющее задавать уровень безопасности при использовании макросов. Известен ряд компьютерных вирусов, маскирующихся под макросы, в связи с чем разработчиками Excel предпринят ряд дополнительных мер защиты. Так, например, может быть задан высокий, средний и низкий уровни безопасности при работе с макросами (по умолчанию средний и рекомендуемый уровень безопасности). Если он используется, то при загрузке файла с диска система попросит разрешение на подключение макросов к программе. Если такое разрешение будет дано, то макрос будет доступен в загружаемой таблице. Пункты меню **Редактор Visual Basic** и **Редактор сценариев** вызывают соответствующие программы (они должны быть установлены на компьютер отдельно с инсталляционных дискет и подключены к операционной системе).

Если в меню **СЕРВИС, Макрос** выбрать пункт **Начать запись…**, то откроется диалоговое окно, позволяющее задать имя макроса и, при желании, комбинацию клавиш, с помощью которой он также может вызван в обход пункта меню **Макросы…**. По умолчанию система предлагает стандартное имя **Макрос#**. Во избежание недоразумений старайтесь задавать собственные имена макросов, отличные от стандартных. Начиная с этого момента все действия с рабочей книгой дополнительно записываются в файл макроса. Остановить запись макроса можно кнопкой **Остановить запись** дополнительно открывшейся панели инструментов или через аналогичный пункт главное меню **СЕРВИС, Макрос**. Записанный макрос может быть сохранен в текущей рабочей книге и тогда он доступен в ней и других книгах в том случае, когда она открыта или в личной книге макросов. В последнем случае он может быть доступен в любой открытой книге.

Удалить макрос, созданный в текущей рабочей книге, можно кнопкой <Удалить> диалогового окна <<Макросы>>. Если макрос создан в личной книге макросов, то для его удаления потребуются более сложные действия, о которых будет расска-

зано позднее. Поэтому старайтесь в первое время не пользоваться макросами личной книги.

Если макрос создан в личной книге макросов, то для его удаления необходимо запустить **Редактор Visual Basic**. В запустившейся оболочке надо открыть окно проектов командами **VIEW, Project Explorer** (если оно не открылось автоматически). После этого надо раскрыть содержимое проекта VBAProject (PERSONAL.XLS) и раскрыть ветвь Modules. В ответ на эти действия откроется список модулей проекта. Активируя каждый модуль двойным щелчком, просматривается его содержимое в окне редактора VBA. После того, как интересующий макрос найден, его текст выделяется в окне и удаляется. При необходимости можно удалить весь модуль, щелкнув его правой клавишей мышки, и воспользовавшись пунктом открывшегося меню, например, <Remove Module1>.

Необходимо принять во внимание существование двух возможных типов записи ссылок на ячейки в Excel: A1 и R1C1. По умолчанию при программировании формул используется стиль A1, для которого адрес каждой ячейки представляет собой строку символов, содержащую имя столбца и номер строки. Использование этого стиля позволяют организовать относительную и абсолютную адресацию к ячейкам таблицы (за счет введения в строку символа $). Тем не менее, при записи макросов Excel использует тип ссылки R1C1. В обозначении типа присутствуют первые буквы английских слов Row (строка) и Column (колонка). В первую очередь обратите внимание на то, что, в отличие от типа A1, при использовании типа ссылок R1C1 сначала записывается строка, а потом столбец. При использовании абсолютной адресации после символов R и C указывается собственно номер строки и столбца. Так, например, ячейка B3 имеет адрес R3C2. При использовании относительной адресации в стиле R1C1 после обозначения строки или колонки в квадратных скобках указывается смещение по отношению к текущей ячейке. Так, например, если данные находятся в ячейке B3, а ссылка на нее программируется в ячейке A5, то в формуле она запишется как R[-2]C[1]. Эта запись может интерпретироваться как обращение к ячейке, находящейся на две строки выше и одну колонку правее текущей. Соответственно запись R[2]C[-1] означает обращение к ячейке на две строки ниже и одну колонку левее (по отношению к активной ячейке A5 такая ячейка не существует).

Пример 1. Рассмотрим таблицу, показанную на рис. 1. В ней необходимо рассчитать сумму подоходного налога (с учетом используемой ставки налога), сумму к выдаче для каждого сотрудника, а также общие суммы уплачиваемых налогов

и выплаченной заработной платы. Записывался макрос с именем Расчет_заработной_платы. Текст макроса имеет вид:

```
Sub Расчет_заработной_платы()
' Расчет _заработной_платы Макрос
' Макрос записан 01.12.2005 (Администратор)
    Range("C2").Select
    ActiveCell.FormulaR1C1 = "=RC[-1]*R7C3"
    Range("D2").Select
    ActiveCell.FormulaR1C1 = "=RC[-2]-RC[-1]"
    Range("C2:D2").Select
    Selection.AutoFill Destination:=Range("C2:D5"), Type:=xlFillDefault
    Range("C6").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
    Range("D6").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
End Sub
```

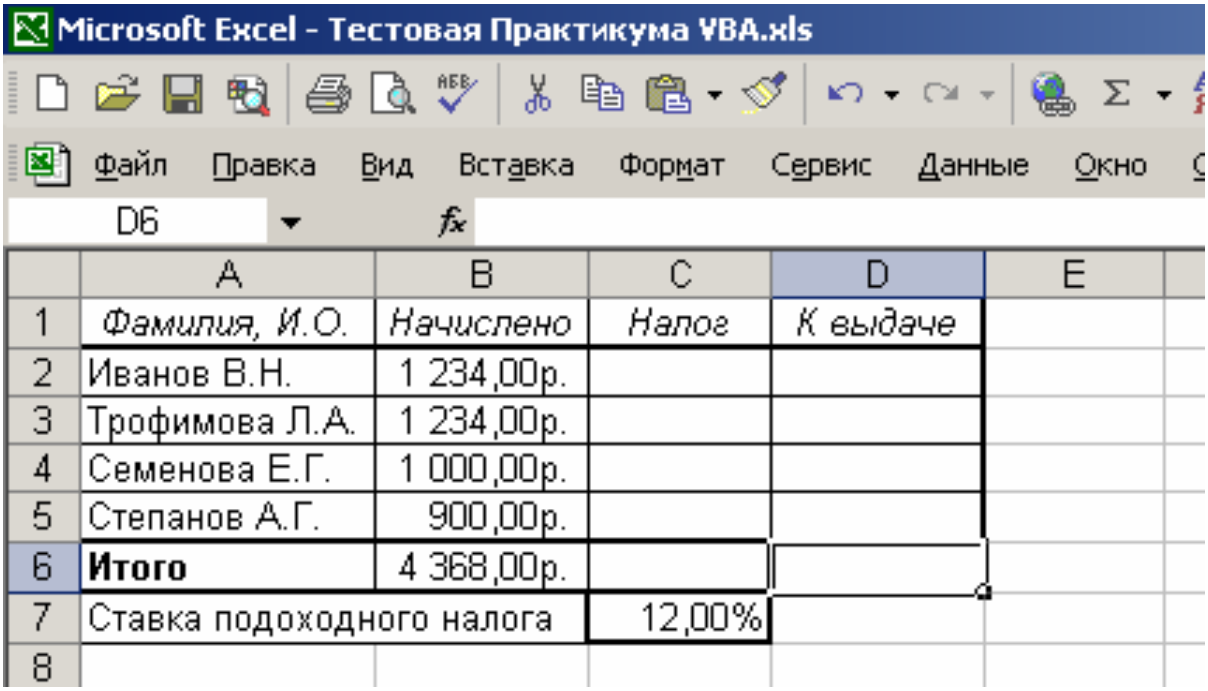


Рис. 1. Пример таблицы

В рассматриваемом примере первый оператор представляет собой заголовок процедуры. Имя процедуры совпадает с именем макроса. Следующие шесть строчек созданы системой в виде автоматически вставляемого комментария.

Первый исполняемый оператор программы Range("C2").Select создан системой в виде выражения, которое содержит в терминологии VBA свойство Range в сочетании с методом Select. Обратите внимание на то, что свойство имеет записанный в круглых скобках аргумент в виде строки символов и отделяется от метода точкой. В нашем примере аргумент свойства представляет собой ссылку на ячейку в стиле A1, с которой началось программирование макроса.

С помощью Help-системы разберитесь с назначением свойства Range. Для этого установите в окне модуля маркер на текст Range и нажмите клавишу F1. Если вы испытываете затруднения с чтением текста на английском языке, который используется Help-системой, воспользуйтесь дополнительной русскоязычной литера-

турой, посвященной описанию языка VBA. В этом случае удобно составлять собственное описание встречающихся англоязычных терминов и хранить его в удобном месте (например, в виде отдельного файла Excel).

Аналогично изучите назначение метода Select.

Фактически анализируемая строка программы представляет собой набор действий по активизации ячейки С3 рабочего листа Excel. Система всегда одинаково интерпретирует действия пользователя Excel, поэтому в случае затруднений с анализом результатов ее работы удобно создать новый дополнительный макрос как результат конкретного короткого действия и изучить его содержимое. Наконец, в особо сложных случаях можно скопировать текст созданного макроса, изменить его имя и запустить его из Excel для того, чтобы увидеть результат действий интересующего вас оператора.

Продолжите изучение операторов созданного макроса и убедитесь в том, что вы понимаете смысл и результат действия каждого оператора. Так следующий оператор рассматриваемого примера заносит в активную ячейку формулу для вычисления величины подоходного налога. В формуле используется стиль ссылок R1C1, причем ее первый операнд задан в относительной адресации, а второй в абсолютной.

Два следующих оператора программы задают другую активную ячейку и заносят в нее формулу для вычисления суммы к выдаче.

Следующий оператор программы выделяет диапазон ячеек листа Excel, после чего выделенные ячейки копируются во все содержащие фамилии сотрудников строчки таблицы.

Для расчета суммы уплачиваемых налогов делается активной предназначенная для этого ячейка рабочего листа и в нее заносится формула, содержащая функцию суммирования данных выделенных ячеек. Система использовала относительную адресацию в формате R1C1. Аналогичная операция проводится и с ячейкой, предназначенной для хранения общей суммы к выдаче.

Задание

Согласуйте с преподавателем выбранный вами вариант задания (табл. 1). Предполагается, что в рамках одной учебной группы варианты заданий не повторяются. Разработайте и заполните таблицу и запрограммируйте в ней необходимые вычисления. При необходимости воспользуйтесь функциями. Убедитесь в правильности вычислений. Оформите таблицу, задайте шрифты, границы и т.п. В качестве примера будет рассматриваться таблица, предназначенная для расчета налогов и определения суммы заработной платы.

Используя копию созданной таблицы, создайте и изучите макросы, позволяющие программировать вычисления в таблице.

Порядок выполнения работы

1. Создайте новую рабочую книгу Excel. Сделайте ее настройку:

- выполните команду **СЕРВИС, Параметры** и в диалоговом окне выберите вкладку *Общие*, установив следующие параметры:

Стиль ссылокR1C1: выключено. Листов в новой книге: 3. Стандартный шрифт: Arial Cyr, размер 10. Выберите рабочий каталог для сохранения новых книг. Введите имя пользователя.

- выберите вкладку *Вид*, установив флажки следующих параметров:

Отображать: область задач при запуске, строку формул, строку состояния, окна на панели задач. Примечания: только индикатор. Объекты: отображать. Параметры окна: заголовки строк и столбцов, горизонтальная полоса прокрутки, символы структуры, вертикальная полоса прокрутки, сетка, нулевые значения, ярлычки листов.

- выберите вкладку *Вычисления*, установив флажки следующих параметров:

Вычисления: автоматически. Параметры книги: обновлять удаленные ссылки, сохранять значения внешних связей.

2. Переименуйте рабочий лист, выполнив следующие действия:

- установите указатель мыши на вкладку с именем листа (Лист 1) и вызовите контекстное меню, щелкнув правой клавишей мыши;
- выберите в текстовом меню параметр **Переименовать**;
- введите в диалоговом меню новое имя листа, придуманное вами.

3. Сохраните созданную рабочую книгу с новым, придуманным вами именем, выполнив команду **ФАЙЛ, Сохранить как…**
4. Создайте шаблон придуманной вами пользовательской таблицы.
5. Задайте наименования полей головки таблицы. При необходимости укажите в них единицы измерения.
6. Заполните таблицу данными и запрограммируйте в ней необходимые вычисления. Убедитесь в правильности вычислений.

7. Скопируйте созданную таблицу на новый рабочий лист. Удалите в ней все формулы.
8. В меню **СЕРВИС, Макрос выберите пункт Начать запись…**. Задайте имя макроса.
9. Повторно запрограммируйте формулы таблицы Excel и остановите запись макроса.
10. Командой **Сервис, Макрос, Редактор Visual Basic** запустите редактор Visual Basic. В окне проектов (**Project-VBAProject**) (рис. 2) раскройте содержимое проекта **VBAProject (PERSONAL.XLS)** и ветвь **Modules**. В ее составе должен быть один (например, Module1) или несколько модулей. Дважды щелкните левой клавишей мышки по имени модуля. В ответ в правом верхнем окне должен появится его текст. Просмотрите содержимое модулей и найдите записанный вами макрос.
11. Изучите текст макроса.
12. Удалите формулы из таблицы рабочего листа Excel и выполните макрос командой **Сервис, Макрос, Макросы**. Убедитесь, что в результате его работы содержимое таблицы восстанавливается.
13. Снова удалите формулы из таблицы рабочего листа Excel. Перейдите в окно VBA, установите маркер на первом операторе макроса. Выберите пункт **Run, Run Sub/User Form** и запустите модуль на выполнение. Перейдите в таблицу Excel и убедитесь, что в результате работы макроса формулы в ней восстановились.
14. Окончательно оформите созданную таблицу для представления ее в отчетной документации. Воспользуйтесь возможностями задания шрифтов, границ, заливок. Обеспечьте компактность отображения таблицы за счет минимизации ширины строк и столбцов в соответствии с имеющимися данными.

Контрольные вопросы

1. В каком случае используется стиль ссылок Excel :A1, а в каком R1C1?
2. В чем разница абсолютной и относительной адресации ссылок в Excel?
3. Когда целесообразно использовать абсолютную адресацию в Excel?

4. Если написать макрос вручную, то какие обязательные операторы он должен содержать?

5. Каково назначение свойства Range?

6. Каково назначение метода Select?

7. Как вызвать систему помощи и получить справку по конкретному выражению макроса?

8. Как можно запустить макрос на выполнение?

9. Что такое построчный комментарий и как он оформляется?

10. Какие существуют возможности для оформления внешнего вида таблицы Excel перед ее публикацией в отчетной документации?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, формулировку задания, пример созданной таблицы, содержание ее программирования и текст созданного вами макроса с включенными в него построчными комментариями действий системы, созданными вам как результат анализа текста макроса. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Вариант титульного листа отчета, какой он был в 2004-2005 учебном году, приведен в приложении А. С действующим вариантом титульного листа можно ознакомиться на http://standarts.guap.ru

Таблица 1. Варианты заданий для выполнения лабораторной работы

Номер варианта	Вид таблицы
1	Ведомость складских остатков (наименование, цена, количество, отпускная цена, оптовая скидка)
2	Ведомость операций квартплаты (плательщик, вид услуги, полный тариф, начислено, льгота, пени, к оплате, задолженность, оплачено).
3	Ведомость операций оплаты за электроэнергию (плательщик, начальное показание, конечное показание, израсходовано, полный тариф, начислено, льгота, пени, к оплате, задолженность, оплачено)
4	Журнал учета выполнения лабораторных работ (фамилия и инициалы студента, названия лабораторных работ, для каждой работы дата, оценка защиты и рейтинг, средний балл, итоговый рейтинг, дата получения зачета).
5	Журнал учета экзаменационных оценок, перечень дисциплин, для каждой дисциплины дата, оценка, рейтинг по итогам семестра и сессии, общий рейтинг, средний балл).
6	Расписание занятий преподавателей кафедры (фамилии преподавателей, должность, ученое звание, ученая степень, для каждого дня нечетной и четной недели и каждой учебной пары название или код дисциплины, вид занятия, номера учебных групп, номер аудитории, объем учебной нагрузки).
7	Индивидуальная выписка для преподавателя по проведенным занятиям для представления на оплату (дата проведения, время проведения, номер аудитории, номера групп, вид занятия, источник финансирования (государственный бюджет или договор на оплату образовательных услуг), количество часов, количество оплачиваемых часов, часовая ставка, сумма к оплате).

Номер варианта	Вид таблицы
8	Ведомость командировок (фамилия, город, страна, цель поездки, источник финансирования, дата убытия, дата прибытия, срок командировки, стоимость проезда туда, стоимость проезда обратно, суточные, сумма затрат).
9	Ведомость операций туристического агентства (фамилия, страна, город, вид транспорта туда, вид транспорта обратно, транспортные расходы туда, транспортные расходы обратно, отель, стоимость проживания в сутки, дата заезда, дата убытия, срок проживания, затраты на проживание, общие затраты).
10	Ведомость операций риэлтерского агентства (адрес, район, метро, этаж, жилая площадь, количество комнат, вспомогательная площадь, удобства, стоимость квадратного метра, цена помещения, затраты на ремонт и переоборудование помещения, общая стоимость).
11	Ведомость операций обменного пункта валюты (валюта прихода, сумма прихода, курс к рублю, комиссия вид валюты, курс валюты комиссии к рублю, комиссия в рублях, валюта расхода, сумма расхода, курс к рублю).
12	Ведомость операций авиакассы (фамилия, направление, рейс, дата вылета, время вылета, тариф авиакомпании, валюта тарифа, тариф в рублях, аэропортовский сбор пункта отправления, валюта сбора пункта отправления, сумма в рублях, аэропортовский сбор пункта прибытия, валюта сбора пункта прибытия, сумма в рублях, стоимость трансфера, валюта трансфера, сумма трансфера в рублях, комиссия кассы, валюта кассы, комиссия в рублях, общая сумма операции).
13	Ведомость продаж универсама (вид товара, единица измерения, имеющееся количество, цена складская, цена отпускная, объем продажи в единицах измерения, остаток, стоимость продажи, скидка, льгота, сумма к оплате, вид оплаты, комиссия банка).
14	Смета затрат на ремонт (номер операции, операция, материалы, единица измерения, цена, стоимость, нормочасы, тариф, зарплата, наценка, стоимость, скидка, к оплате).
15	Ведомость операций телефонной компании (абонент, тарифный план, вид операции, тариф, время, цена операции, наценка, стоимость, скидка, льгота, к оплате).
16	Ведомость комплектации изделия (наименование комплектующего, количество, цена, количество на складе, стоимость складского остатка, затраты, наценка, стоимость).
17	Таблица футбольного чемпионата (команда, страна, город, игр, побед, ничьих, поражений, технических поражений, забито голов, пропущено голов, очков).
18	Ведомость операций типографии (автор, название, издательство, машинописных страниц, печатных листов, рисунков, таблиц, тираж, тип бумаги, цена печатного листа, цена печати, тип переплета, цена переплета, затраты на материалы, затраты на амортизацию оборудования, заработная плата, накладные расходы, стоимость).
19	Список трудов (номер, название, место опубликования, дата опубликования, вид публикации, номер страницы начала, номер страницы конца, всего страниц, формат страницы, машинописных листов, печатных листов, соавторы, доля автора, машинописных страниц автора, печатных листов автора).
20	Ведомость операций отделения связи (адрес назначения, адрес отправителя, вид отправления, вес отправления, тариф, дата отправления, упаковка, цена упаковки, страховка, общая цена отправления).
21	Ведомость операций страхового агентства (фамилия страхуемого, объект страхования, вид страхования, дата страхования, дата начала действия страховки, дата окончания действия страховки, срок страхования, тариф, цена полиса, скидка, льгота, к оплате).
22	Ведомость операций библиотеки (автор, название, издательство, год издания, объем, цена, дата выдачи, контрольная дата возврата, планируемый срок пользования, фактическая дата возврата, фактический срок пользования, ставка штрафных санкций, штраф).
23	Ведомость операций фотоателье (фамилия заказчика, вид операции, дата заказа, дата исполнения, общее время исполнения, тариф, срочность, количество, стоимость, скидки, льготы).
24	Ведомость банковских операций (фамилия, дата, вид операции, валюта операции, сумма операции, сумма операции в рублях, комиссия операции, валюта комиссии операции, комиссия операции в рублях).
25	Ведомость операций диспетчерской такси (клиент, адрес подачи машины, адрес назначения, дата поездки, время начала поездки, время окончания поездки, километраж, тип машины, расчетное время выполнения заказа, время на подачу машины, тариф, стоимость, скидка, льгота, к оплате).
26	Ведомость судейства соревнований по фигурному катанию (участник, город, страна, вид программы, оценки судей, каждая из которых включает оценку за технику исполнения, оценку за художественное впечатление, место в общем зачете, итоговая оценка за технику исполнения, итоговая оценка за художественное впечатление, суммарное место в общем зачете).

Номер варианта	Вид таблицы
27	Ведомость операций автомобильной стоянки (регистрационный номер автомобиля, марка (модель), дата постановки, время постановки, планируемая дата освобождения, планируемое время освобождения, планируемое время стоянки, тариф, расчетная цена, фактическая дата освобождения, фактическое время освобождения, к оплате, возврат или доплата).
28	Учебная нагрузка преподавателей кафедры (семестр, специальность, группа, дисциплина, преподаватель, должность, звание, степень, ставка по бюджету, ставка по договору на оплату образовательных услуг, лекции, практические занятия, лабораторные работы, курсовое проектирование, контрольные работы, дипломное проектирование, участие в ГАК, рецензирование, практика, итого часов, итого рублей. Необходимо предусмотреть разделение учебной нагрузки на часы государственного бюджета и договора на оплату образовательных услуг.)
29	Ведомость операций стоматологической поликлиники (фамилия пациента, фамилия врача, дата приема, назначенная дата повторного приема, операция, материалы тариф, оплата труда тариф, стоимость услуги, скидка, льгота, сумма к оплате).
30	Предложенная студентом (содержание полей таблицы необходимо согласовать с преподавателем)

Лабораторная работа №2

Отладка и выполнение программы в среде VBA

Методические указания

Мы уже просматривали макрос, созданный в процессе выполнения предыдущей лабораторной работы средствами VBA. Рассмотрим назначение интегрированной среды разработки приложений VBA (рис. 2) более подробно.

Верхние строки представляют собой главное меню программы и набор пиктограмм часто используемых операций. Как обычно, этот набор может настраиваться в зависимости от потребностей пользователя. Ниже в левой части экрана находится уже знакомое нам окно проектов **Project-VBAProject**. Содержимое выбранного в этом окне модуля Module2 раскрыто в окне редактора кодов, находящемся правее. На рисунке зафиксирован момент отладки программы, поэтому одна из строк кода выделена специальным цветовым маркером. Его положение указывает на следующий выполняемый оператор программы. Обратите внимание на окно локальных переменных **Locals** в нижней части экрана. В нем отображается содержимое ячеек памяти. В выполняемом макросе нет собственных переменных, поэтому ветвь Module2 в настоящий момент пустая. Окно тестирования **Immediate** позволяет изменять значения переменных программы в момент ее выполнения и даже вводить дополнительные операторы. На настоящий момент оно также пустое. Остальные элементы (кнопки управления, полосы прокрутки, раскрывающиеся списки и т.п.) являются

стандартными для Windows и должны быть вам хорошо знакомы. Наконец дополнительно в левой нижней части экрана командой **View, Properties Window** открыто окно свойств.

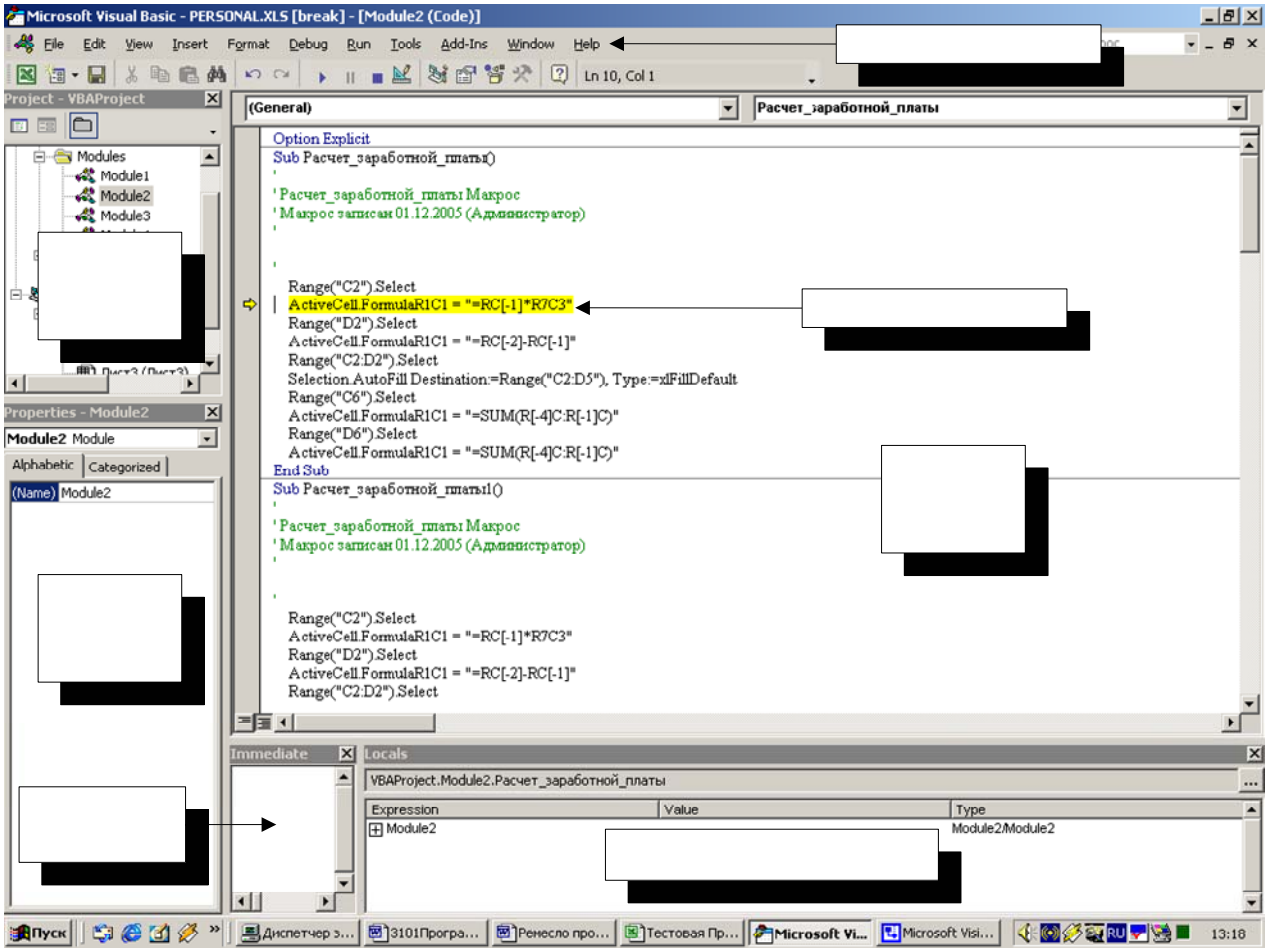


Рис. 2. Интегрированная среда разработки VBA

Готовая программа, которой, в частности, является созданный нами макрос, может быть запущена и выполнена с использованием интегрированной среды VBA. Мы можем запустить конкретный модуль, предварительно указав его маркером в окне редактора кодов. Для этого достаточно выбрать пункт **Run, Run Sub/User Form** главного меню. Операторы программы будут выполняться автоматически один за другим, а после завершения ее выполнения мы можем только контролировать результаты ее работы. Этот режим мы уже использовали при выполнении предыдущей лабораторной работы. Очень часто, особенно при создании новой программы, разработчика интересуют не только итоговые, но и промежуточные результаты ее выполнения. Для этого удобно воспользоваться режимом отладки программы (пункт **Debug** главного меню). Выбор строки **Step Into** позволяет выполнить текущий опе-

Окно проектов

ратор программы. Если программа еще не запущена, то в ответ на выбор этого пункта активируется маркер отладчика (рис. 2). Дальнейший выбор **Step Into** приведет к выполнению следующей строки программы. Результаты выполнения могут, например, проявится в виде изменения переменных программы и быть проконтролированы в окне локальных переменных **Locals** (если такие переменные существуют) или непосредственно на листе Excel, если выполняется макрос, созданный средствами Excel.

Примечание. Меню отладчика предлагает еще две возможности пошагового выполнения программы: **Step Over** и **Step Out**. Они представляют интерес при работе с программами, содержащими вызываемые функции или процедуры. Режим **Step Over** позволяет автоматически выполнить вызываемую функцию (процедуру) и перейти к следующему оператору основной программы, а режим **Step Out** закончит выполнение текущей функции или процедуры. Способы их использования будут обсуждены в лабораторной работе, посвященной функциям и процедурам.

В качестве дополнительных возможностей отладчика отметим следующее. Можно автоматически выполнять операторы программы до оператора, на котором установлен курсор в окне редактора кодов. Этот режим вызывается строкой **Run To Cursor** пункта **Debug** главного меню. Программа запускается в автоматическом режиме из своего текущего состояния, а после остановки на отмеченном курсором операторе он выделится маркером отладчика. В текст программы можно вводить так называемые точки останова (строка **Toggle Breakpoint**). В окне редактора кодов такой оператор помечается специальным маркером. После запуска программы любыми средствами она автоматически выполняется до точки останова. Кроме этого, существует возможность наблюдать значения избранных вами переменных программы в окне наблюдаемых выражений Watches (на рис. 2 это окно не показано). Если вы захотите воспользоваться этим режимом, то командой **View/Watch Window** главного меню включите его, выберите соответствующую строку пункта **Debug** главного меню и задайте наблюдаемое выражение в открывшемся меню.

Задание

Используйте согласованный с преподавателем вариант задания (табл. 1), выполненную на его основе таблицу Excel и созданный вами макрос. Научитесь выполнять макрос в режиме отладчика и проверьте правильность его работы.

Порядок выполнения работы

1. Откройте созданную вами рабочую книгу Excel. Скопируйте свою таблицу на новый лист. Удалите в ней все формулы.
2. Проверьте работоспособность ранее созданного макроса, для чего воспользуйтесь командой **Сервис, Макрос, Макросы**. Убедитесь в правильности вычислений.
3. Командой **Сервис, Макрос, Редактор Visual Basic** запустите редактор Visual Basic. Сделайте настройку интегрированной среды разработки VBA:

- выполните команду **Tools, Options** и в диалоговом окне выберите вкладку *Editor*, установив следующие параметры:

Code Settings: Auto Syntax Check, Require Variable Declaration, Auto List members, Auto Quick Info, Auto Data Tips - включено. Window Settings: Drag-and-Drop Text Editing, Default to Full Module View, Procedure Separator - включено. Auto Indent - включено. Tab - 4.

В этом случае редактор автоматически находит синтаксические ошибки в программе, любые переменные программы могут использоваться только после их явного предварительного объявления, разрешается автоматический вывод списка свойств и методов, разрешается автоматический вывод информации о функции, на экране отображается значение переменной, на которую установлен указатель мыши. Кроме этого, разрешено перетаскивание фрагментов программы мышью, в окне редактора кодов отображаются все процедуры текущего модуля, а между текстами процедур и функций модуля присутствует разделительная линия. Наконец, В программе автоматически устанавливаются отступы в тексте, а при нажатии клавиши Tab в текст вставляется 4 пробела.

- Выберите вкладки *Editor Format*, *General* и *Docking*. Изучите их содержимое. Изменять их настройку, задаваемую по умолчанию, не рекомендуется.

4. Снова удалите формулы из таблицы. Выполните созданный вами макрос в пошаговом режиме. Для этого установите маркер в окне редактора кодов в тексте созданного вами макроса. Выполните команду **Debug, Step Into** и убедитесь, что маркер отладчика активировался на заголовке выполняемого мак-

роса. Последовательно используя команду **Debug, Step Into** и переключаясь на рабочий лист Excel, контролируйте процесс заполнения таблицы. Убедитесь в правильности вычислений.

5. Снова удалите формулы из таблицы. Установите маркер в окне редактора кодов в тексте созданного вами макроса на один из выполняемых операторов программы. Выполните команду **Debug, Run To Cursor** и убедитесь, что маркер отладчика активировался на выбранном вами операторе. Проверьте, что в результате выполнения начального фрагмента программы заполнилась соответствующая им часть таблицы Excel. Убедитесь в дальнейшей работоспособности программы за счет выполнения команд **Debug, Step Into**. Снова установите маркер на один из следующих операторов программы и убедитесь в правильности выполнения команды **Debug, Run To Cursor**.
6. Снова удалите формулы из таблицы. Установите маркер в окне редактора кодов в тексте созданного вами макроса на один из выполняемых операторов программы. Командой **Debug, Toggle Breakpoint** задайте точку останова. Выполните команду **Run, Run Sub/User Form** и убедитесь, что маркер отладчика активировался на выбранном вами операторе. Убедитесь в дальнейшей работоспособности программы за счет выполнения команд **Debug, Step Into** и **Debug, Run To Cursor**.
7. Снова удалите формулы из таблицы. Установите маркер в окне редактора кодов в тексте созданного вами макроса на точку останова. Повторно выполните команду **Debug, Toggle Breakpoint** и удалите точку останова. Убедитесь в правильности выполнения программы, запустив ее одним из возможных способов. Проверьте возможность задания нескольких точек останова в программе.

Контрольные вопросы

1. Каково назначение окна локальных переменных?
2. Каково назначение окна редактора кодов?
3. Как выглядит маркер отладчика?
4. Каково назначение окна проектов?
5. Как выполнить программу по шагам?

6. Какие команды существуют для выполнения программы и в чем их отличие?
7. Что такое точка останова?
8. Как при выполнении программы по шагам можно автоматически выполнить ее определенную часть?
9. Как запустить программу на выполнение из Excel?
10. Как можно наблюдать результаты выполнения программы в пошаговом режиме в таблице Excel?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист и описание последовательности ваших действий с отладчиком, а также результаты выполнения программы. Дайте письменные ответы на контрольные вопросы. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Лабораторная работа №3

Обмен данными между Excel и VBA

Методические указания

Процессор ЭВМ манипулирует с данными, имеющимися в памяти машины. В зависимости от конкретной задачи эти данные могут принимать самые разнообразные значения, но они должны быть занесены в стандартные физические ячейки памяти, размер которых определяется конструкцией конкретного устройства. Поскольку для хранения различных данных может потребоваться различный объем памяти, используется метод последовательного размещения их в памяти. В этом случае одна единица данных может занимать одну или несколько последовательных физических ячеек памяти машины. Адресация к данным производится по адресу первой ячейки, но при этом общее число используемых ячеек должно быть точно известно. Так как только программист в состоянии предусмотреть возможные значения данных, используемых в программах, вопрос о распределении памяти для хранения информации ложится на его плечи. Конкретная организация памяти в задаче осуществляется за счет выбора программистом конкретного типа данных для хранения его информации.

Тип данных - способ внутреннего представления данных в памяти машины, учитывающий метод их кодирования в одной или нескольких ячейках памяти и предусматривающий возможности их расшифровки или преобразования.

Первые языки программирования содержали всего два типа данных – целые (Integer) и дробные (Real или Float или Single). С развитием языков программирования и расширением круга решаемых задач число используемых типов данных непрерывно росло. Так, для обеспечения требуемой точности и диапазона вычислений были введены соответственно для целых и дробных чисел типы Long и Double. Потребность в хранении текстовой информации привела к появлению типа данных Byte (в терминологии VBA), позволяющего наиболее экономно расходовать память ЭВМ (современные таблицы кодировки символов используют диапазон кодов от 0 до 255). Задачи, связанные с анализом и составлением текстовых сообщений, стали поддерживаться типом данных String. Для обеспечения возможностей ссылки на различные участки памяти был предложен специальный тип данных, называемый указателем (Object). В случае использования указателя в памяти хранится адрес ячейки памяти, содержащей интересующие нас данные или коды программы. Кроме этого, программисту предоставили возможность самому создавать интересующий его тип данных. Необходимость выполнения вычислений с датами и временем породила свой специальный тип данных Date. Особые условия выполнения вычислений с деньгами заставили добавить в перечень специальный тип Currency. Наконец, для упрощения начального ввода данных в клетки электронных таблиц Excel был разработан специальный тип данных Variant, позволяющий автоматически распознавать и обрабатывать числа и строки.

Готовясь к написанию программы, программист обязан задуматься над вопросом: какие значения могут принимать данные его программы? Ответив на этот вопрос, программист подбирает удобный ему тип данных из числа стандартных или создает свой. При этом приходится принимать во внимание следующее обстоятельство: использование стандартных типов данных существенно упрощает процесс создания программы, поскольку в языке программирования заложены возможности действий с этими данными и их преобразование из типа в тип. Мы предполагаем, что вы знакомы с типами данных Excel, задаваемыми ячейкам командой **Формат, Ячейки** вкладка **Число** окно **Числовые форматы**. В языке VBA существуют типы данных, приведенные в табл. 2

Пример 2. В программе, предназначенной для расчета начисления заработной платы рис. 1. для хранения номера в списке (если он будет добавлен в таблицу) можно выбрать тип данных Integer, для хранения фамилий сотрудников тип данных String. Ставка заработной платы и величина начисленного налога может быть описана типом данных Currency, а ставка налога типом данных Single. Кроме этого, например можно создать свой тип данных (Type), в который входят фамилия, начисленная сумма, сумма уплачиваемого налога и сумма к выдаче как самостоятельная единица хранимых в памяти данных.

Поскольку физически данные программы оказываются содержимым конкретных ячеек памяти машины, для их отыскания достаточно знать адрес первой ячейки, связанной с данными, и по типу данных определить общее число используемых для хранения элементарных ячеек. Такой подход имел место на самой ранней стадии программирования и оказался крайне неудобным из-за отсутствия наглядности в записи программы. Действительно, если память современной машины содержит несколько десятков, а то и сотен миллионов ячеек памяти, то обращение к ним по номерам было бы крайне неразумным. Уже первые трансляторы использовали прием, основанный на использовании так называемых идентификаторов.

Идентификатором называется символическое имя ячейки памяти. Каждый язык программирования содержит свои правила составления таких имен, общим является то, что программист вправе сам придумать имя, что позволяет ему сохранить в нем смысловое значение. В языке VBA имеются следующие ограничения на имена:

- Длина имени не должна превышать 255 символов.
- Имя должно начинаться с буквы.
- Имя не может содержать точек и символов %, &, !, #, @, $.
- Буквы рассматриваются инвариантно по отношению к регистру, то есть имя Аа и аА есть одно и то же имя.
- Совпадения имен идентификаторов с так называемыми ключевыми словами не допускается.

Ключевые слова - набор специальных слов, написанных символами латыни и имеющих определенный смысл с тоски зрения конструкций языка программирования. Ключевыми словами обозначаются, в частности, операторы языка и встроенные функции языка.

Пример 3. Возможные варианты идентификаторов языка VBA: I, j, Name, Переменная, Результат_вычислений. Еще варианты записи идентификаторов: A%, B&, C!, D#, E@, F$. В этом случае символы %, &, !, #, @, $ не входят в состав идентификатора и используются в качестве специального признака типа данных (смотри табл. 2).

Программист может вводить переменные в текст программы на VBA по мере их необходимости, применяя явное или неявное (по умолчанию) их объявление. В

последнем случае переменная просто начинает использоваться в тексте. При первом ее появлении компилятор (интерпретатор) заносит новое имя в таблицу и закрепляет за ним определенный адрес и тип данных (в VBA – Variant).

Хотя возможность объявления переменных по умолчанию предусмотрена разработчиками языка, она представляется крайне нежелательной. Текст программы сам по себе представляет документ, в котором содержится исчерпывающая информация о ее работе, в том числе и о типах используемых данных. Введение переменных по умолчанию приводит к затруднениям при изучении программы и, как следствие, к ошибкам. Поэтому рекомендуется всегда явно определять переменные с помощью оператора Dim с указанием типа и задавать специальный режим принудительного объявления переменных программы помещенной в начале текста модуля инструкцией Option Explicit.

Пример 4. Явное объявление переменной:

```
Dim I As Integer, Name, j As Integer, Переменная As Integer, GGG As Integer
```

Обратите внимание на то, что если вы не указываете явно тип переменной, то по умолчанию она имеет тип Variant. Так, в рассмотренном выше примере такой тип имеет переменная Name.

Примечание. Интегрированная среда разработки VBA в окне редактора кодов предлагает в качестве сервиса возможность выбора одного из существующих типов данных из автоматически раскрывающегося списка. Так, после набора ключевого слова Dim, указания идентификатора переменной и набора ключевого слова As автоматически открывается список возможных значений (в данном случае типов данных). Перемещение по списку может осуществляться с помощью маркеров или путем ввода символов с клавиатуры. После того, как требуемое значение в списке установлено, оно может быть перенесено в текст программы клавишей Tab или в результате двойного клика клавишей мышки. Этой возможностью удобно пользоваться для избежания грамматических ошибок при наборе текста программы.

Рассмотренные выше примеры объявления переменных предусматривали создание одиночных констант или переменных, обращение к которым осуществляется только по имени. Практика программирования широко использует переменные, обращение к которым ведется как по имени, так и по номеру. В этом случае можно говорить о создании переменных табличного типа, когда обращение к данным ведется по имени и номеру (индексу) внутри этого имени. Такие переменные обычно называются массивами. Массив - последовательно упорядоченные в памяти данные одного типа.

Таблица 2.Типы данных языка VBA

Тип данных	Размер (байт)	Служебный символ	Диапазон значений

Тип данных	Размер (байт)	Служебный символ	Диапазон значений
Byte (байт)	1		От 0 до 255
Boolean (логический)	2		True или False
Integer (целые)	2	%	От -32768 до 32767
Long (длинное целое)	4	&	От -2147483648 до 2147483647
Single (плавающее обычной точности)	4	!	От -3,402823E38 до -1,401298E-45 и от 1,401298E-45 до 3,402823E38
Double (плавающее двойной точности)	8	#	От -1,79769313486231E308 до -4,94065645841247E-324 и от 4,94065645841247E-324 до 1,79769313486231E308
Currency (денежный)	8	@	От -922337203685477,5808 до 922337203685477,5808
Decimal (масштабируемое целое)	14		+/-79228162514264337593539503З5 и 28 знаков после запятой. Минимальное отличное от нуля значение имеет вид +/-0,0000000000000000000000000001
Date (время и дата)	8		От 1 января 100 г. до 31 декабря 9999 г.
Object (объект)	4		Любой указатель объекта
String (строка переменной длины)	10+ длина строки		От 0 до приблизительно двух миллиардов
String (строка постоянной длины)	Длина строки	$	От 1 до 65400
Variant (числовые подтипы)	16		От -1,79769313486232E308 до -4,94065645841247E-324 и от 4,94065645841247E-324 до 1,79769313486232E308
Variant (строковые подтипы)	22+ длина строки		От 0 до приблизительно двух миллиардов
Type (определяемый пользователем)	Определяется элементами типа		Диапазон каждого элемента определяется его типом данных

Если мы хотим воспользоваться массивом, то мы должны его явно объявить. В дополнение к предыдущему объявление предусматривает еще и задание диапазона изменения номера элемента массива. По умолчанию считается, что минимальное значение номера (базовый индекс) равно нулю, а максимальное задается при объявлении. При необходимости (например, из соображений совместимости со старыми версиями языка) значение базового индекса 0 может быть изменено на 1 инструкцией Option Base 1. Другим способом изменения базового индекса конкретного массива является использование в объявлении ключевого слова To. Минимальное значение индекса указывается до слова To, а максимальное после. В некоторых случаях, например для хранения таблиц, оказывается удобным использовать двумерные массивы. В этом случае при объявлении через запятую описываются оба индекса. Возможно и применение массивов и большей размерности.

Пример 5. Объявления массивов:

```
Dim YY(25)
```

Объявляется одномерный массив из 26 элементов. Начальный (базовый) индекс принят по умолчанию равным 0.

Dim ZZ(3,10) As Single

Объявляется двумерный массив ZZ типа Single, первый индекс которого меняется в диапазоне от 0 до 3, а второй в диапазоне от 0 до 10.

Dim SS(-3 To 3,1 To 10) As Integer

Переопределение базовых индексов с помощью явного указания нижних и верхних границ номеров элементов массива с использованием ключевого слова To.

Для обращения к ячейке памяти или элементу массива достаточно в тексте программы использовать соответствующий идентификатор (в случае массива с номером элемента, указанным в скобках). Важной особенностью систем программирования является то обстоятельство, что в качестве номера элемента массива может выступать не только константа, но и другая переменная, заданная своим идентификатором. Заметим, что недостатком рассмотренного приема является относительно высокая вероятность возникновения ошибки программирования связанной с выходом индекса (номера элемента) за границы массива. Программная среда VBA автоматически локализует такую ситуацию, выдавая соответствующее диагностическое сообщение.

Пример 6. Обращение к элементу массива в тексте программы с явным указанием номеров элементов: SS(-2,5).

Если переменная Name содержит число –2, а ячейка Переменная число 5, то обращение SS(Name, Переменная) полностью эквивалентно предыдущему.

Если в процессе предыдущих вычислений переменная Name примет значение –4, а мы попытаемся выполнить SS(Name, Переменная), то произойдет обращение к несуществующему элементу массива и возникнет ошибка выхода индекса за границы массива.

Массивы удобно использовать при программировании однотипных действий с ячейками памяти. В качестве примера рассмотрим задачу расчета начисления заработной платы (рис. 1). Поскольку исходные данные и результаты промежуточных вычислений должны храниться в памяти ЭВМ, в процессе программирования решения задачи на VBA приходится использовать идентификаторы. Заметим, что обычный идентификатор в этом случае не очень удобен. Действительно, хотя возможно введение в текст программы обычной переменной вида Налог_Трофимова_Л_А, создаваемая программа может быть в этом случае использована только для расчетов налога, уплачиваемого именно Л.А. Трофимовой. Если мы хотим запрограммировать вычисления для другого лица, то нам придется вводить другой идентификатор. Подобные действия ведут к изменению текста исходной программы и крайне нежелательны на практике. Конечно, мы можем ввести идентификаторы обычных перемен-

ных вида Налог_запись_2, однако и в этом случае мы должны будем индивидуально описать последовательность манипуляций с ячейками памяти для каждого сотрудника, включенного в список. Для нашего примера это вполне возможно, но реальный список может состоять, например, из 100 фамилий.

Кроме всего прочего, каждый раз при изменении количества сотрудников мы должны корректировать объявления переменных и, возможно, делать добавления в текст программы. Программирование существенно упростится, если ввести в рассмотрение массивы данных, имеющие смысл Начислено(1 To 4), Налог(1 To 4), К_выдаче(1 To 4) и рассматривать их элементы с одинаковыми номерами как записи, относящиеся к сотруднику, имеющему соответствующий идентификационный номер. На первый взгляд этот способ ничем существенным не отличается от использования идентификаторов одиночных переменных с номерами, однако если вспомнить, что существует возможность обращения к элементу массива с использованием идентификатора другой переменной, то можно рассматриваемую задачу попытаться описать и в общем виде.

Пример 7. В общем виде выражение для вычисления величины суммы к выдаче для каждого сотрудника может быть записано как:

К_выдаче(i) = Начислено(i) – Налог(i)

Здесь символом = обозначена операция присваивания результата вычислений в правой части оператора ячейке, указанной в левой части. Во время выполнения этой операции старое содержимое ячейки К_выдаче(i) теряется и она получает новое значение. В то же время символ – есть символ операции вычитания.

Если организовать повторения вычислений по этой формуле столько раз, сколько сотрудников имеется в списке для последовательно изменяющихся значений индекса i, то рассматриваемая задача может быть решена заметно проще, чем в случае объявления одиночных переменных.

Иногда приходится создавать массивы, размер которых невозможно определить на этапе компиляции программы. В нашем примере нам может быть неизвестно общее число сотрудников, для которых должна быть начислена зарплата. Конечно, можно объявить массивы с запасом, так, чтобы номер максимального элемента массива был заведомо большим максимально возможного числа сотрудников, допустим 100 человек. Однако такой прием приводит к нерациональному распределению памяти. Альтернативой является метод динамического объявления размера массива. В этом случае конкретный размер массива вычисляется в процессе выполнения программы и память для хранения данных отводится тоже во время выполнения. Чтобы воспользоваться этим методом, необходимо первоначально объявить массив без

указания его размеров, а затем воспользоваться инструкцией ReDim. Менять границы изменения индекса массива можно сколь угодно много раз. Если массив больше не требуется в программе, память, занимаемая им, может быть освобождена с помощью инструкции Erase Начислено.

Пример 8.

```
Dim Начислено() As Currency, i As Integer
i = 10
ReDim Начислено(1 To i)
```

Массив Начислено() первоначально был объявлен как массив неопределенной длины. Инструкция ReDim изменила массив, причем память под него была отведена в момент выполнения программы.

Очень часто при программировании возникает необходимость создания новых типов данных, вид которых определяется конкретной задачей. Так, например, программируя задачу, представленную на рис. 1, обратим внимание на то обстоятельство, что информация, размещенная в этой таблице, имеет одинаковую структуру по строкам. Более того, даже программируя соответствующую колонку таблицы в виде массива, программист обязан следить за тем, чтобы номера элементов разных массивов, относящихся к одному сотруднику, не отличались бы один от другого. Из соображений надежности программирования оказывается удобным рассматривать все, относящееся к одному сотруднику, в виде целой неделимой записи, содержащей соответственно фамилию, начисленную сумму, рассчитанный налог и сумму к выдаче. На самом деле речь идет о создании нового типа данных, определенного пользователем и включающего в себя относящиеся к записи поля. Структура данных - объединение под одним именем различных компонентов с индивидуальными именами и типами, называемых членами структуры.

Признаком структуры данных, как правило, является символ точки в ее идентификаторе, причем имя структуры записывается до точки, а имя ее компонента (члена) после точки. В языке VBA структуры данных можно создавать на основе типов данных, определяемым пользователем. Задание типа данных только описывает структуру, информация о которой размещается в общей области программы VBA. Для ее непосредственного объявления и резервирования ячеек памяти под хранение данных требуется явно объявить переменную в конкретном модуле.

Пример 9. Создание пользовательского типа данных, представляющего собой одну строку записи рис. 1.

```
Type Запись_Ведомости
  Фамилия_И_О As String
  Начислено_Ведомость As Currency
  Налог_Ведомость As Currency
```

```
  К_выдаче_Ведомость As Currency
End Type
```

Объявление переменной:

```
Dim Запись1 As Запись_Ведомости
```

Запись значений в элементы структуры с использованием оператора присваивания:

```
Запись1.Фамилия_И_О = "Иванов В.Н."
Запись1.Начислено_Ведомость = 1234
Запись1.Налог_Ведомость = Запись1.Начислено_Ведомость * 0.12
Запись1.К_выдаче_Ведомость=Запись1.Начислено_Ведомость-Запись1.Налог_Ведомость
```

Здесь символом * обозначена операция умножения.

Объявление массива структур:

```
Dim Ведомость(1 To 4) As Запись_Ведомости
```

Соответствующие обращения к элементам массива и членам структуры будут иметь вид:

```
Ведомость(1).Фамилия_И_О = "Иванов В.Н."
Ведомость(1).Начислено_Ведомость = 1234
Ведомость(2).Фамилия_И_О = "Трофимова Л.А.
Ведомость(2).Начислено_Ведомость = 1234
```

Примечание. Интегрированная среда разработки VBA в окне редактора кодов предлагает в качестве сервиса возможность конкретного выбора типа данных, определенных пользователем, из автоматически раскрывающегося списка. Если структура данных ранее была объявлена и выполнена компиляция проекта, после набора символа точки автоматически открывается список возможных имен полей структуры. Этой возможностью удобно пользоваться для избежания синтаксических ошибок при наборе текста программы.

Отдельную проблему представляет прямая и обратная передача данных из таблицы Excel в ячейки памяти, объявленные в программе, написанной на VBA. Автоматически созданный макрос непосредственно манипулирует с ячейками таблицы, используя стили ссылки на ячейки в Excel: A1 и R1C1. Конечно, такой прием может быть использован и в рабочей программе, однако в этом случае ее модификация и использование существенно затруднены. Гораздо предпочтительнее использовать свойство Cells() стандартного объекта Excel Range. Сам объект представляет собой ячейку, столбец, строку или выделенный диапазон листа Excel. Свойство Cells() позволяет непосредственно обратиться к объекту Excel по номеру строки и колонки. Поскольку это свойство установлено по умолчанию для рабочего листа Excel, то его можно использовать без дополнительных указаний.

Свойство Cells() позволяет обратиться к ячейке рабочего листа задав номер строки и колонки. Если запись свойства стоит слева от символа равенства (оператор присваивания), то производится запись данных в ячейку таблицы, если справа, то считывание значения из ячейки таблицы. Кроме собственно записи данных свойство Cells() в сочетании со свойствами других объектов (Font, Color и т.п.) позволяет задавать параметры шрифта, его цвет, фон и так далее. Для изучения этих возможно-

стей целесообразно ознакомиться с описанием соответствующих свойств и объектов в литературе, воспользоваться Help-системой или, что вероятно проще всего, запустить режим записи макроса в Excel, выполнить, например, установку цвета и изучить текст полученного макроса.

Пример 10. Использование свойства Cells() для считывания данных в переменную VBA и возврата значения в Excel и установки нового цвета шрифта. Используется тот факт, что положение и количество ячеек в таблице рис. 1. известно. Дополнительно в программе используется символ комментария ' и комбинация символов «пробел»_ (_) для обозначения продолжения длинной строки

```
Sub Расчет_заработной_платы2()
Dim Начислено(1 To 4) As Currency, Налог(1 To 4) As Currency, _
   К_Выдаче(1 To 4) As Currency, i As Integer
   i = 1
   Начислено(i) = Cells(i + 1, 2)  'В первую ячейке массива Начислено записывается содержимое
'второй строки и второй колонки исходной таблицы Excel
    Cells(i + 1, 2).Font.ColorIndex = 7 'В ячейке устанавливается новый цвет шрифта
   Налог(i) = Начислено(i) * 0.12  'Рассчитывается значение налога и запоминается
'в соответствующей ячейке
   Cells(i + 1, 3) = Налог(i) 'Значение налога возвращается в таблицу Excel
   К_Выдаче(i) = Начислено(i) - Налог(i) 'Рассчитывается значение к выдаче
'и запоминается в соответствующей ячейке
   Cells(i + 1, 4) = К_Выдаче(i) 'Значение к выдаче возвращается в таблицу Excel
   i = i + 1 'Переход к следующей записи
   Начислено(i) = Cells(i + 1, 2)
   Cells(i + 1, 2).Font.ColorIndex = 7
   Налог(i) = Начислено(i) * 0.12
   Cells(i + 1, 3) = Налог(i)
   К_Выдаче(i) = Начислено(i) - Налог(i)
   Cells(i + 1, 4) = К_Выдаче(i)
   i = i + 1 'Переход к следующей записи
   Начислено(i) = Cells(i + 1, 2)
   Cells(i + 1, 2).Font.ColorIndex = 7
   Налог(i) = Начислено(i) * 0.12
   Cells(i + 1, 3) = Налог(i)
   К_Выдаче(i) = Начислено(i) - Налог(i)
   Cells(i + 1, 4) = К_Выдаче(i)
   i = i + 1 'Переход к следующей записи
   Начислено(i) = Cells(i + 1, 2)
   Cells(i + 1, 2).Font.ColorIndex = 7
   Налог(i) = Начислено(i) * 0.12
   Cells(i + 1, 3) = Налог(i)
   К_Выдаче(i) = Начислено(i) - Налог(i)
   Cells(i + 1, 4) = К_Выдаче(i)
End Sub
```

Задание

Используйте согласованный с преподавателем вариант задания (табл. 1), выполненную на его основе таблицу Excel и созданный вами макрос. Модифицируйте созданный вами макрос и напишите новую программу так, чтобы ее основные вы-

числения производились с переменными VBA. При этом исходные данные первоначально должны быть считаны из таблицы, а результаты вычислений возвращены в нее.

Порядок выполнения работы

1. Откройте созданную вами рабочую книгу Excel. Скопируйте свою таблицу на новый лист. Удалите в ней все формулы.
2. Определите и, при необходимости, задайте формат ячеек таблицы в соответствии с требованиями вашей задачи.
3. Скопируйте созданный вами макрос в окне редактора кода, удалите все внутренние операторы, оставив только его заголовок и последний оператор End Sub. Измените название процедуры (макроса).
4. Задайте режим обязательного объявления переменных, для чего выше заголовка вставьте строку Option explicit.
5. Напишите коды объявления внутренних переменных своей программы и задайтесь их типом данных. Прокомментируйте их в тексте программы.
6. Напишите коды программы считывания исходных данных из таблицы Excel во внутренние переменные программы VBA. Прокомментируйте их в тексте программы.
7. Напишите коды вычислений результирующих значений так, чтобы их результаты оказались во внутренних переменных программы VBA. Прокомментируйте их в тексте программы.
8. Напишите коды программы записи рассчитанных значений в соответствующие ячейки таблицы Excel. Прокомментируйте их в тексте программы.
9. Запустите созданную программу в режиме отладки командами **Debug, Step Into**. На каждом шаге выполнения контролируйте изменение внутренних переменных программы в окне локальных переменных Locals. Убедитесь в правильности выполнения расчетов. При выполнении фрагментов программы, обеспечивающих запись рассчитанных значений в ячейки Excel, дополнительно убедитесь в правильности выполнения этих действий.

10. Удалите в таблице Excel результаты вычислений и проверьте работоспособность программы в режиме **Debug, Run To Cursor**. Проконтролируйте значения в окне локальных переменных Locals. Введите точки останова командой **Debug, Toggle Breakpoint** и убедитесь в правильности работы программы.
11. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

Контрольные вопросы

1. Как связаны между собой типы данных Excel и VBA?
2. В чем необходимость использования данных типа Long и Double?
3. Как можно использовать тип данных Type?
4. Как явно объявить переменную в тексте программы?
5. Какие слова в языке программирования называются ключевыми?
6. В чем недостатки метода объявления переменной по умолчанию?
7. Как включить режим обязательного явного объявления переменных?
8. Чем массив отличается от обычной переменной?
9. Как можно использовать возможности динамического объявления размера массива?
10. Как можно использовать свойство Cells() для организации обмена данными между Excel и VBA?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист и текст написанной вами программы с построчным комментарием ее действий. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Программирование на VBA

Лабораторная работа №4

Операции и операторы VBA

Методические указания

Под программированием обычно понимают процесс составления упорядоченной последовательности действий, реализующей алгоритм[1] решения некой задачи. Язык описания этой последовательности может быть самым различным. Так, например, может быть создано описание алгоритма в виде конструкции обычного разговорного языка (построить, вычислить, перевезти, обеспечить). С другой стороны, алгоритм может быть сформулирован в терминах элементарных арифметических операций. Очевидно, что если предполагается выполнять программу на ЭВМ, то эта последовательность должна быть разработана с учетом возможностей компьютерной реализации. Размер элементарных действий алгоритма определяется возможностями их последующей обработки. Поскольку в общем случае программирование ЭВМ сводится к заданию последовательности команд ее процессора, то в степень детализации алгоритма может быть доведена до уровня команд процессора.

Документирование алгоритмов можно вести разными способами. На практике наибольшее распространение получило их графическое представление. При составлении рисунков алгоритмов программ необходимо пользоваться стандартными обозначениями. Некоторые из них приведены на рис. 3 и имеют следующий смысл:

- Процесс – действия, приводящие к изменению данных.
- Предопределенный процесс – выполнение ранее созданного алгоритма.
- Решения – действия, приводящие к изменению последовательности выполнения операторов программы.
- Ввод – вывод – действия по вводу – выводу информации на внешние устройства.
- Пуск – останов – точки начала и конца алгоритма.
- Соединитель – обозначение точек разрыва на линиях связи (например, для переноса линии на следующую страницу).

---

[1] Алгоритм – одно из основных понятий (категорий) математики, не обладающих формальным определением в терминах более простых понятий, а абстрагируемых непосредственно из опыта (Большая Советская Энциклопедия).

- Модификация – действия по изменению кодов ранее созданной программы.
- Класс, объект – соответственно обозначения класса и объекта.
- Связь, синхронизация – обозначения последовательности выполнения алгоритма и связей между объектами.

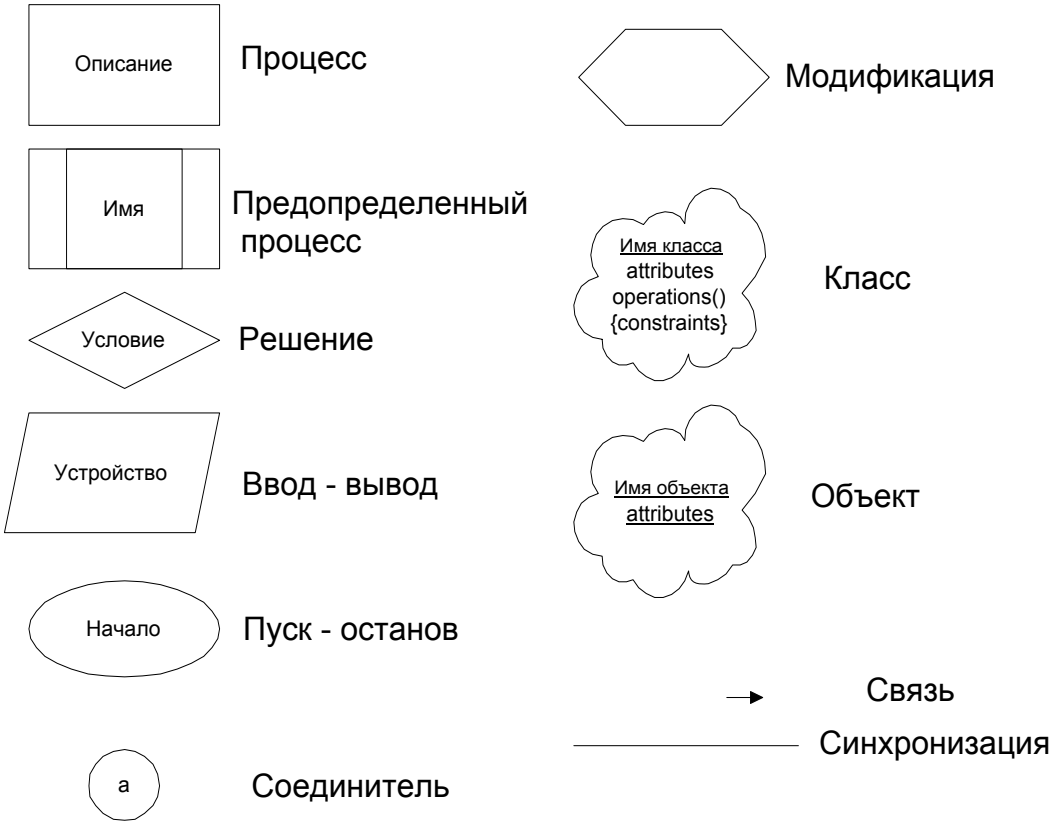


Рис. 3. Некоторые обозначения, используемые при записи алгоритмов

В первую очередь разработчиками языков программирования решалась задача снижения затрат труда, требуемого на подготовку программного обеспечения. Базовая система команд процессора позволяет обеспечить только самые минимальные потребности программиста в части обработки данных. Так, например, очень небольшое число существующих в мире процессоров имеют в своем составе команду деления чисел с плавающей точкой. На практике необходимые программистам операции эмулируются программным обеспечением. Это означает, что разработчиками компиляторов заранее созданы последовательности кодов команд конкретного процессора, позволяющие в конечном итоге получить желаемый результат, в частности, уже упомянутое деление чисел с плавающей точкой. Такие последовательности включаются в коды программы в результате компиляции определенного зарезервированного символа или группы символов, встретившихся в тексте программы.

Система команд процессора отражает текущее состояние микроэлектроники и строится исходя из принципов достаточности, технической реализуемости с учетом требуемого быстродействия. При программировании реальных задач ее возможностей оказывается явно недостаточно. Это приводит к необходимости создания неких более крупных конструкций, которые выполняют заранее определенные действия. Появление таких конструкций, которые получили название операций и операторов, связано с возникновением языков программирования высокого уровня. Опыт их применения привел к созданию набора типовых операций и операторов, состав и назначение которых постоянно совершенствуется с учетом потребностей практики. Существующий набор операций и операторов языков высокого уровня ориентирован на программиста (а не на процессор) и позволяет ему решить подавляющее большинство практических задач. Каждая операция и оператор выполняют вполне конкретные действия, связанные с изменением данных в памяти и (или) управлением последовательностью выполнения команд. Поэтому можно говорить о составлении алгоритмов в терминах операций и операторов языка, когда строго определено их назначение, а сама операция или оператор представляет собой элементарную функциональную единицу алгоритма.

Операция - инструкция языка программирования, которая однозначно обрабатывается компилятором в виде генерации стандартной последовательности кодов процессора. В английской версии Help-системы VBA они именуются как Operators. В качестве участников операции (операндов) могут выступать константы и переменные. Если в одном выражении используется несколько операций, то порядок их выполнения определяется приоритетом (чем меньше номер, тем выше приоритет и раньше выполняется операция). Если операции имеют одинаковый приоритет, то они выполняются слева направо. При необходимости, последовательность выполнения операций может регулироваться круглыми скобками (сначала выполняются действия в скобках). Хотя многие операции реализованы во всех языках программирования, их конкретный набор, а также обозначения в разных языках программирования разный, что во многом затрудняет одновременное использование различных языков программирования из-за повышения вероятности ошибки кодирования. Поэтому, несмотря на кажущуюся внешнюю простоту вопроса, требуется тщательный анализ существующих в конкретном языке программирования операций и их обозначений.

Арифметические операции языка VBA реализованы в виде стандартного общеупотребительного набора. Они могут быть выполнены с числовыми данными любых типов и используют привычные обозначения. К их числу относятся операции, представленные в табл. 3. Результатом выполнения арифметической операции является число, которое может использоваться в последующих операциях.

Таблица 3. Арифметические операции языка VBA.

Операции	Приоритет	Название	Пример записи	Если A=11и B=5, то результат
-	2	Смена знака	-A	-11
+	6	Сложение	A+B	16
-	6	Вычитание	A-B	6
*	3	Умножение	A*B	55
/	3	Деление	A/B	2.2
\	4	Целочисленное деление	A\B	2
Mod	5	Остаток от деления по модулю	A Mod B	1
^	1	Возведение в степень	A^B	161015

Операции сравнения позволяют установить результат сравнения двух операндов в терминах *истина (True)* или *ложь (False).* Реализованы операции, представленные в табл. 4. Обратим внимание на тот факт, что, в отличие от арифметических операций, результатом вычислений является логическое утверждение *истина* или *ложь*.

Логические операции реализуют алгебру Буля. В качестве операндов операции могут выступать константы или переменные. При анализе выполнения логических операций следует принимать во внимание то обстоятельство, что утверждение *ложь* в вычислительной технике обычно кодируется кодом 0, а *истина* кодом, отличным от нуля (например, 1, 2, и тому подобное). Очевидно, что, как и в случае операций сравнения, результатом вычислений является логическое утверждение *истина* или *ложь*. Список и правила выполнения логических операций представлены в табл. 5.

Язык VBA поддерживает две операции со строками – сцепление и сравнение (табл. 6). Операция сцепления строк позволяет создавать новую строку, состоящую из строк операндов. Результатом ее выполнения является строка символов. А вот операция сравнения строк возвращает значение *истина* в том случае, когда первый операнд удовлетворяет условиям, заданным в виде некоторого шаблона и представленного вторым операндом, и *ложь*, если условие, заданное шаблоном, не удовлетворяется. Сам шаблон представляет собой строку символов, имеющих оп-

ределенный смысл. Если в строке шаблона присутствует символ алфавита, то результат сравнения строк будет истина только в том случае, если на том же месте в исходной строке стоит такой же символ. При необходимости вместо одного символа можно указать диапазон. Наличие символа # в позиции строки шаблона означает, что в исходной строке на том же месте должна стоять цифра. Символ ? в шаблоне разрешает находиться на соответствующей позиции рабочей строки любому другому символу.

Таблица 4. Операции сравнения языка VBA.

Операции	Приоритет	Название	Пример записи	Если A=11и B=5, то результат
<	7	Меньше	A<B	False
>	7	Больше	A>B	True
<=	7	Меньше и равно	A<=B	False
>=	7	Больше и равно	A>=B	True
<>	7	Не равно	A<>B	True
=	7	Равно	A=B	False
Is		Сравнение со ссылкой на объекты	Dim A,B,C,D,E Set A=D Set B=D Set C=E F=A Is B F=A Is C	 True False

Понятие оператора во многом схоже с понятием операции. И в том, и в другом случае целью их использования является снижение трудозатрат на создание программного обеспечения. В обоих случаях в соответствующее место итоговой программы подставляется некая заранее созданная заготовка кодов процессора, реализующая конкретный оператор или операцию. Тем не менее, существует определенное отличие оператора от операции. Оно заключается в том, что оператор является законченной самостоятельной конструкцией программы, которая, в отличие от операции, может быть самостоятельно откомпилирована и выполнена. В то же время операция может многократно входить в состав одного оператора, и выполняется только в его составе.

В целом оператор – это конструкция более высокого уровня, чем операция. В его состав могут входить аргументы, константы, переменные, операции и другие операторы. Таким образом, оператор **-** самостоятельная конструкция языка программирования, которая может быть отдельно откомпилирована и выполнена в виде заранее определенной последовательности кодов процессора. В английской версии Help-системы VBA операторы именуются как Statements.

Таблица 5. Логические операции языка VBA.

Операции	Приоритет	Название	Пример записи	Если A=True B=True C=False D=False, то результат
And	10	Логическое умножение (и)	A And B A And C C And B C And D	True False False False
Or	11	Логическое сложение (или)	A Or B A Or C C Or B C Or D	True True True False
Xor	12	Исключающее или	A Xor B A Xor C C Xor B C Xor D	False True True False
Not	9	Отрицание	E=Not B E=Not D	False True
Imp	14	Импликация	A Imp B A Imp C C Imp B C Imp D	True False True True
Eqv	13	Эквивалентность	A Eqv B A Eqv C C Eqv B C Eqv D	True False False True

Таблица 6. Операции со строками языка VBA.

Операции	Приоритет	Название	Пример записи	Если A=”abc”,B=”123”, C=”a?*”,то результат
&		Сцепление строк	A&B	”abc123”
Like		Сравнение строк	A Like C	True

Язык VBA содержит весь базовый набор операторов классического языка Бейсик с добавлениями, учитывающими потребности объектно-ориентированного программирования. Операторы записываются на отдельных строчках и могут не нумероваться. Для размещения нескольких операторов на одной строке между ними необходимо поставить символ двоеточие (:). Этот же символ используется для обозначения меток. Для переноса продолжения оператора на следующую строку используется комбинация символов пробел знак подчеркивания (_). Нельзя разбивать переносом идентификаторы и строки. Допускается не более семи переносов строк одного оператора.

Оператор объявления переменных. Основные идеи, связанные с типами данных, были уже рассмотрены выше. Здесь приводится формальное описание оператора объявления переменных в языке VBA Dim, имеющего следующий синтаксис:

Dim [WithEvents] ИмяПеременной [(Индексы])] [As [New] Тип]

Примечание. В документации по программированию при описании синтаксиса языка принято следующее общепринятое соглашение: параметры и ключевые слова в квадратных

скобках не являются обязательными. При написании программ они могут опускаться, и тогда предполагается некоторое их значение по умолчанию.

Одним оператором Dim можно описывать несколько переменных. В качестве обозначений используются:

WithEvents – ключевое слово, указывающее, что ИмяПеременной является именем объектной переменной, используемым при отклике на события, генерируемые другими приложениями (внешняя ссылка).

Индексы – размерности массивов, задаваемые в формате
[Нижний To] Верхний

По умолчанию нижний индекс считается равным 0.

New – ключевое слово, указывающее возможность неявного создания объекта. Новый экземпляр объекта создается при первой ссылке на него.

Тип – тип переменной в соответствии с табл. 2.

Отметим, что оператор объявления переменных не создает исполняемых кодов программы, а только резервирует память машины для хранения данных. Как следствие, отсутствует обозначение такого оператора в алгоритме программы.

Оператор присваивания обеспечивает занесение информации в ячейки памяти, связанные с идентификатором и имеет символ равенства (=). Необходимо обратить внимание на то обстоятельство, что в отличие обычного равенства, которое выполняется всегда, оператор присваивания имеет динамические свойства (зависит от времени). При его выполнении результат вычислений правой части оператора заносится в ячейку памяти, указанную слева от знака равенства, число в которой имело одно значение до выполнения оператора и другое после его выполнения. Задавая последовательность операторов присваивания, мы можем программировать запись данных в ячейки памяти ЭВМ. Поэтому для его обозначения в алгоритме лучше всего подходит символ процесс (рис. 3). Синтаксис оператора имеет вид

[Let] Идентификатор = Выражение

Пример 11. Несколько операторов присваивания, производящих вычисления с использованием операций

```
j=1 ‘В ячейку j записывается 1
j=j+1 ‘Считывается число из ячейки j (там была 1), и к этому числу добавляется 1. ‘Результат (число 2) снова записывается в ячейку j
К_выдаче = Начисленно - Начисленно*Ставка_налога ‘Предполагается, что в ячейки
‘Начисленно, Ставка_налога заранее были занесены значения (например, предыдущими 'операторами присваивания). Извлекается число из ячейки Начисленно и запоминается.
‘Извлекается число из ячейки Ставка_налога. Выполняется операция умножения.
‘К числу, извлеченному из ячейки Начисленно добавляется вычисленное произведение.
‘Полученная сумма заносится в ячейку К_выдаче
```

Язык VBA предусматривает работу с так называемыми объектами. При работе такие переменные должны быть специально объявлены. Объектные переменные рассматриваются как указатели (адреса ячеек памяти) на объект. Для записи значе-

ния в указатель (ссылки на объектную переменную) используется ключевое слово Set. Синтаксис оператора в этом случае:

Set ОбъектнаяПеременная = [New] ОбъектноеВыражение

или

Set ОбъектнаяПеременная = Nothing

Примечание. Указателем называется ячейка памяти, предназначенная для хранения адреса другой ячейки памяти.

Ключевое слово New используется при создании нового экземпляра класса, а ключевое слово Nothing позволяет освободить системные ресурсы (память) от объекта, который в дальнейшем использоваться не будет.

Пользуясь оператором присваивания можно создавать так называемые линейные программы, выполняющие последовательные вычисления и запись данных в ячейки памяти. Как только выполнится последний такой оператор, закончится и программа в целом (смотри, например, пример 10).

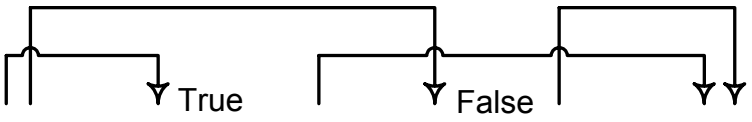
Оператор условия If Then Else EndIf предназначен для выбора последовательности выполнения других операторов программы. Такая ситуация возникает в том случае, когда, в зависимости от конкретной ситуации, требуется выполнить одну или другую ветви алгоритма. Сама ситуация задается в виде так называемого условия, в состав которого могут входить константы и идентификаторы, а также различные операции с ними. Выполнение оператора начинается с вычисления условия, которое может принимать значения истина или ложь. Синтаксис оператора имеет вид:

If Условие Then [Операторы1] [Else: Операторы2] Endif

Если Условие принимает значение True, выполняются операторы, размещенные в тексте программы после ключевого слова Then до ключевого слова Else, иначе выполняются операторы после ключевого слова Else до ключевого слова EndIf. Допускаются вложенные операторы If Then Else EndIf. Схема и алгоритм выполнения оператора представлены на рис. 4. Отметим, что Следующий оператор программы не имеет отношения к оператору If Then Else EndIf и является просто очередным оператором (либо может вообще отсутствовать, если оператор If Then Else EndIf является последним оператором программы).

Пример 12. Проверка допустимости выполнения операции деления с использованием оператора If Then Else EndIf.

```
A = 100: B = 10 'Двоеточие - признак записи второго оператора в одной строке
If B = 0 Then
  A = 1
Else: A = A / B  'Поскольку условие ложно, выполнится эта ветвь оператора
EndIf
```



If ***условие*** Then Операторы1 Else: Операторы2 Endif Следующий оператор программы

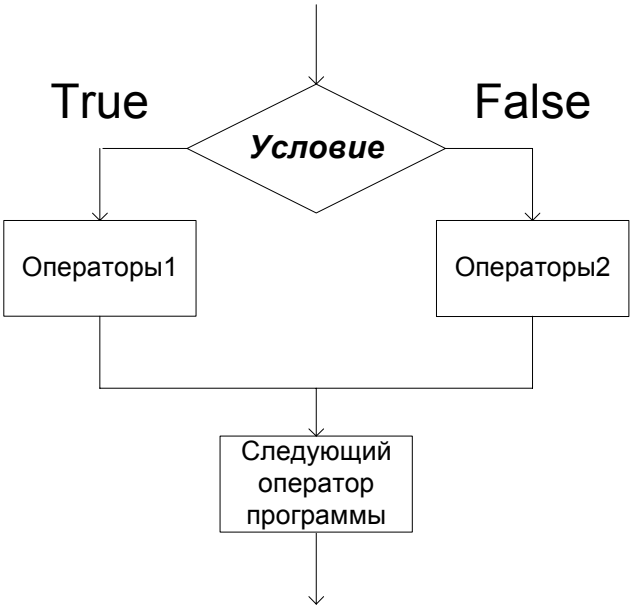


Рис. 4. Схема и алгоритм выполнения оператора If Then Else EndIf

Оператор ветвления Select Case End Select представляет собой более развитую конструкцию, чем оператор If Else EndIf. Он позволяет выбрать один из многих вариантов дальнейшего выполнения программы. Если аргументом оператора If Else EndIf является условие, принимающее только два возможных значения и обеспечивающее один из двух вариантов продолжения последовательности действий, то в качестве аргумента оператора Select Case End Select выступает выражение, результатом вычисления которого оказывается целое число или строка. При записи оператора Select Case End Select в следующих после заголовка строках программы в виде констант предусматриваются различные варианты значений этого числа или строки. Выполняясь, оператор последовательно проверяет все имеющиеся в его теле строки Case. Если в процессе выполнения программы реальное значение вычисленного выражения совпало с константой, записанной в одной из строк, то выполняется последовательность операторов, соответствующая этой строке, посте чего управление передается следующему после конструкции Select Case End Select оператору.

Синтаксис оператора Select Case End Select имеет вид:

```
Select Case Выражение
  [Case Значение1: [Операторы1]]
  [Case ЗначениеN: [ОператорыN]]
```

```
  [Case Else: [ОператорыElse]]
End Select
```

Возможен случай, когда реальное значение выражения не совпадет ни с одной константой, предусмотренной строками Case. В этом случае выполняются операторы, предусмотренные строкой Case Else. Схема и алгоритм выполнения оператора изображены на рис. 5. Как и в предыдущем случае, последний оператор на рис. 5, обозначенный как Следующий оператор программы не имеет отношения к конструкции Select Case End Select и представляет собой просто следующий оператор программы.

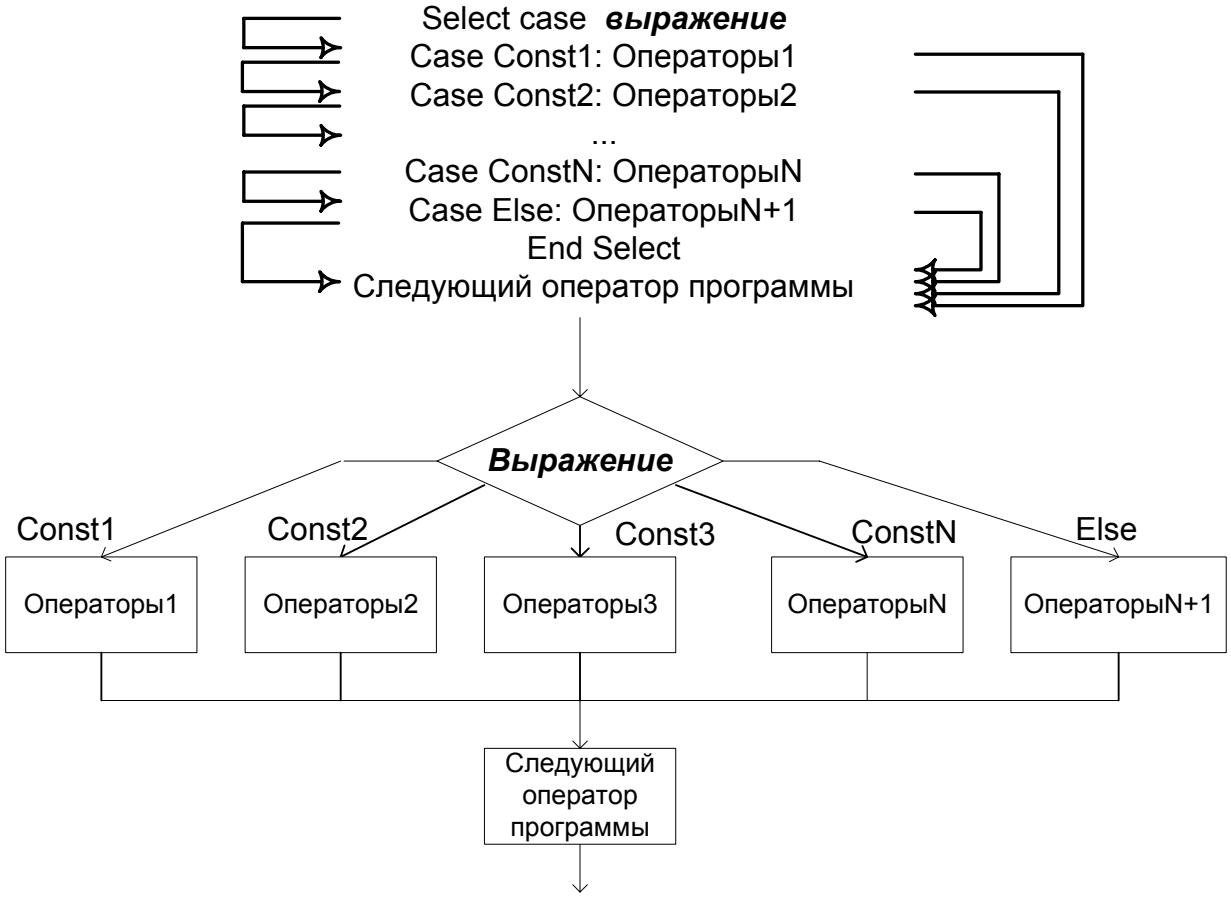


Рис. 5. Схема и алгоритм выполнения оператора Select Case End Select

Пример 13. Программа с использованием оператора Select Case End Select.

```
Dim День_недели As Integer, Rezult As String
День_недели = 2
Select Case День_недели
  Case 1: Rezult = "Прием заказов"
  Case 2: Rezult = "Выполнение заказа"
  Case 3: Rezult = "Выполнение заказа"
  Case 4: Rezult = "Выполнение заказа"
  Case 5: Rezult = "Выдача заказов"
  Case 6: Rezult = "Выходной день"
  Case 7: Rezult = "Выходной день"
```

```
  Case Else: Rezult = "Ошибка задания дня недели"
End Select
'В ячейке Rezult будет записана строка "Выполнение заказа"
```

Операторы цикла. При написании программ очень часто возникает необходимость многократного выполнения определенных операторов программы. По своему назначению операторы цикла предназначены как раз для решения именно этой задачи - организации автоматического повторения выполнения неких операторов или группы операторов. Как правило, можно сформулировать условие, до каких пор собственно должны повторяться действия. Это условие является параметром (аргументом) оператора цикла и называется условием продолжения цикла. В языке VBA имеется несколько операторов цикла, предназначенных для автоматизации создания повторяющихся действий в программе.

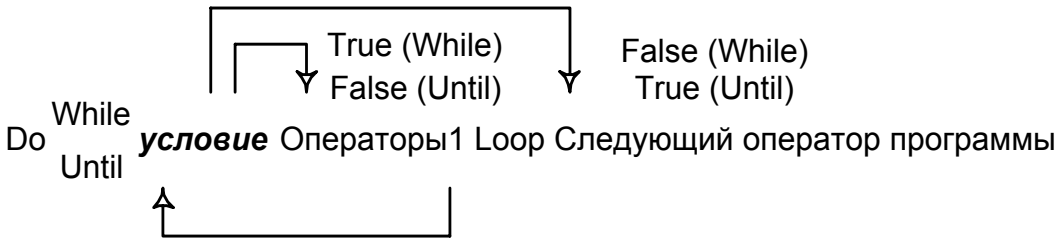
Операторы Do While Loop и Do Until Loop предполагают выполнение всех операторов, размещенных после заголовка Do While, до ограничителя тела цикла обязательного ключевого слова Loop, и в качестве аргумента содержит условие продолжения цикла. Оператор Do While Loop продолжает выполнение тела цикла, пока условие продолжения имеет значение *истина*. То же самое делает Do Until Loop – но пока условие продолжения цикла имеет значение *ложь*.

При необходимости, тело цикла может содержать оператор принудительного завершения цикла Exit Do. Его выполнение приводит к передаче управления на следующий после ключевого слова Loop оператор. Туда же будет передано управление, если условие цикла примет значение соответственно для оператора Do While Loop *ложь*, а для оператора Do Until Loop *истина*.

Примечание. В принципе включение в состав операторов VBA оператора Do Until Loop является избыточным. Аналогичный результат может быть получен при использовании оператора Do While Loop с инвертированным условием.

Операторы Do While Loop и Do Until Loop языка VBA – это операторы с предусловием. Термин «предусловие» означает, что проверка условия проводится до выполнения операторов тела цикла. Если условие продолжения цикла с самого начала не выполняется, то операторы тела цикла при выполнении программы будут пропущены. На рис. 6 представлена схема и алгоритм выполнения операторов Do While Loop и Do Until Loop с предусловием, причем логика их действий противоположна по отношению к значению условия. Синтаксис таких операторов имеет вид:

```
Do [While Условие]                    или    Do [Until Условие]
  [Операторы]                                  [Операторы]
[Exit Do]                                    [Exit Do]
  [Операторы]                                  [Операторы]
Loop                                         Loop
```



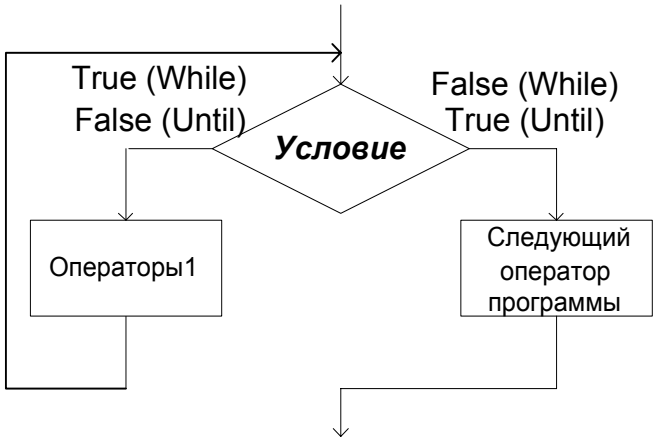


Рис. 6. Схема и алгоритм выполнения операторов Do While Loop и Do Until Loop с предусловием

Пример 14. Иллюстрация возможности сокращения числа операторов линейной программы из примера 10 за счет использования оператора цикла с предусловием. Очевидно, что для обработки списка, например, из 100 сотрудников в этой программе необходимо только поменять диапазоны массивов и условие в операторе Do While Loop. В то же время линейная программа в этом случае займет несколько страниц текста.

```
Sub Расчет_заработной_платы3()
Dim Начислено(1 To 4) As Currency, Налог(1 To 4) As Currency, _
К_Выдаче(1 To 4) As Currency, i As Integer
i = 1 'Задание начального номера массива
Do While (i <= 4)
   Начислено(i) = Cells(i + 1, 2)  'В первую ячейку массива Начислено записывается
   'содержимое второй строки и второй колонки исходной таблицы Excel
   Налог(i) = Начислено(i) * 0.12  'Рассчитывается значение налога и запоминается
   'в соответствующей ячейке
   Cells(i + 1, 3) = Налог(i) 'Значение налога возвращается в таблицу Excel
   К_Выдаче(i) = Начислено(i) - Налог(i) 'Рассчитывается значение к выдаче
   'и запоминается в соответствующей ячейке
```

```
    Cells(i + 1, 4) = К_Выдаче(i) 'Значение к выдаче возвращается в таблицу Excel
    i = i + 1 'Модификация счетчика строк
Loop
End Sub
```

Примечание. В языке VBA существует и более простая форма оператора While Wend по смыслу совпадающая с оператором Do While Loop с предусловием, но имеющая более простой синтаксис:

```
While Условие
  [Операторы]
Wend
```

Множество операторов Do языка VBA содержит пару операторов Do Loop While и Do Loop Until с постусловием. В отличие от операторов с предусловием, проверка условия в этом случае проводится после выполнения операторов тела цикла. Таким образом, при выполнении оператора с постусловием тело цикла обязательно выполнится один раз независимо от начального значения условия. Схема и алгоритм выполнения операторов Do Loop While и Do Loop Until с постусловием представлены на рис. 7. Синтаксис операторов, имеет вид:

```
Do                                  Do
  [Операторы]                         [Операторы]
[Exit Do]              или          [Exit Do]
  [Операторы]                         [Операторы]
Loop [While Условие]                Loop [Until Условие]
```

Рассмотренная в примере 1 программа практически без изменений может быть реализована и с помощью оператора с постусловием. Отличие программ проявится только тогда, когда первоначально заданное значение переменной i ('задание начального номера массива) будет больше 4. В этом случае программа с постусловием будет работать неправильно, поскольку действия с массивами выполнятся раньше проверки условия. Так как размеры массивов равны 4, то на этапе выполнения программы возникнет ошибка, связанная с выходом индекса за границы массива. Заметим, что в программировании существуют ситуации, когда независимо от значения условия продолжения цикла тело цикла должно быть выполнено один раз. В этих случаях использование оператора с постусловием оказывается предпочтительным.

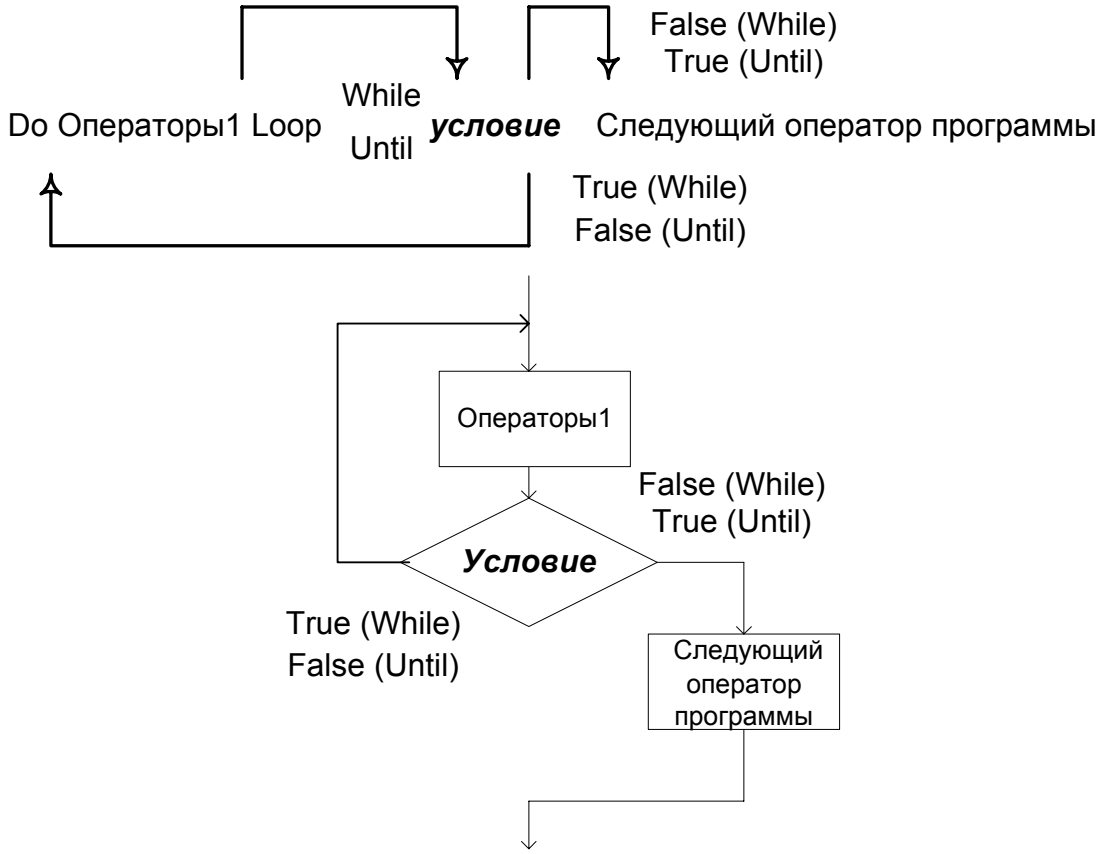


Рис. 7. Схема и алгоритм выполнения операторов Do Loop While и Do Loop Until с постусловием

В программе примера 1 в состав операторов тела цикла, обозначенного на рис. 6 и рис. 7 как Операторы1, пришлось включить оператор присваивания i=i+1. Его назначением является подсчет количества выполненных циклов. Кроме этого, программа содержала оператор, задающий начальное значение переменной i=1. Условие продолжения цикла было задано явно, поэтому цикл должен выполняться строго определенное количество раз. Для упрощения программирования подобных задач в состав операторов языка VBA включен специальный оператор For To Next. Он позволяет прямо в заголовке задать начальное значение аргументу цикла (инициализировать цикл), указать условие продолжения цикла после ключевого слова To, и автоматически модифицировать переменную цикла после завершения выполнения операторов цикла с шагом, заданным после ключевого слова Step. Заметим, что если шаг изменения аргумента цикла в заголовке не задан, то он предполагается равным 1. Схема и алгоритм выполнения оператора For To Next показаны на рис. 8, а описание синтаксиса имеет вид:

```
For Счетчик=Начало To Конец [Step Шаг]
  [Операторы]
[Exit For]
  [Операторы]
Next [Счетчик]
```

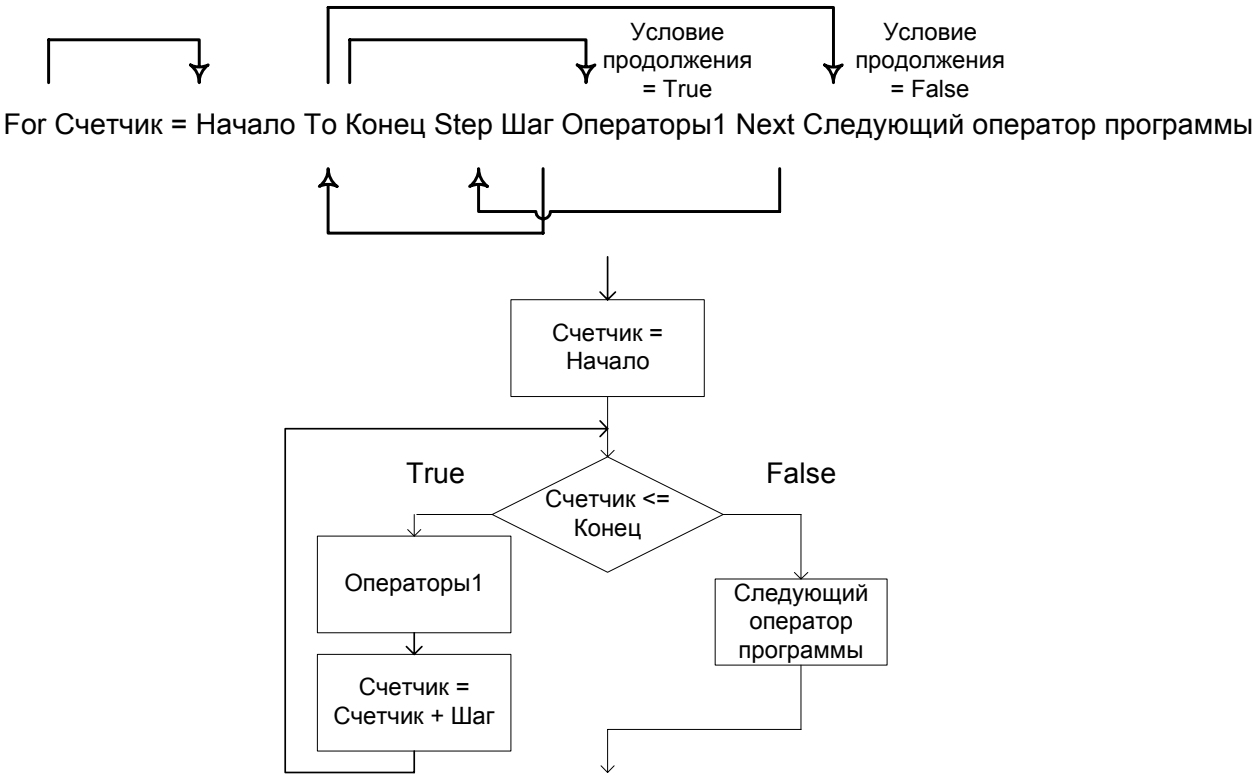


Рис. 8. Схема и алгоритм выполнения оператора For To Next

Пример 15. Программа с оператором For To Next, реализующая задачу примера 10.

```
Sub Расчет_заработной_платы4()
Dim Начислено(1 To 4) As Currency, Налог(1 To 4) As Currency, _
К_Выдаче(1 To 4) As Currency, i As Integer
For i = 1 To 4
   Начислено(i) = Cells(i + 1, 2)  'В первую ячейке массива Начислено записывается
   'содержимое второй строки и второй колонки исходной таблицы Excel
   Налог(i) = Начислено(i) * 0.12  'Рассчитывается значение налога и запоминается
   'в соответствующей ячейке
   Cells(i + 1, 3) = Налог(i) 'Значение налога возвращается в таблицу Excel
   К_Выдаче(i) = Начислено(i) - Налог(i) 'Рассчитывается значение к выдаче
   'и запоминается в соответствующей ячейке
   Cells(i + 1, 4) = К_Выдаче(i) 'Значение к выдаче возвращается в таблицу Excel
  Next i
End Sub
```

Наконец, для автоматизации работы с объектами предусмотрена еще одна разновидность оператора цикла For Each Next:

```
For Each Элемент In Группа
  [Операторы]
```

```
[Exit For]
[Операторы]
Next [Элемент]
```

Оператор For Each Next в этом случае повторяет выполнение операторов тела цикла для каждого элемента группы или семейства.

Пример 16. Пример программы, реализующей задачу рис. 1 использованием различных операторов языка VBA.

```
Sub Расчет_заработной_платы5()
Dim Начислено(1 To 4) As Currency, Налог(1 To 4) As Currency, _
К_Выдаче(1 To 4) As Currency, i As Integer
For i = 1 To 4
   Начислено(i) = Cells(i + 1, 2)  'В первую ячейке массива Начислено записывается
   'содержимое второй строки и второй колонки исходной таблицы Excel
   If Начислено(i) > 1000000 Then
      Начислено(i) = 1000000
      Cells(i + 1, 2).Font.ColorIndex = 4
   Else
   End If
   If Начислено(i) < 0 Then
      Начислено(i) = 0
      Cells(i + 1, 2).Font.ColorIndex = 3
   Else
   End If
   Налог(i) = Начислено(i) * 0.12  'Рассчитывается значение налога и запоминается
   'в соответствующей ячейке
   Cells(i + 1, 3) = Налог(i) 'Значение налога возвращается в таблицу Excel
   К_Выдаче(i) = Начислено(i) - Налог(i) 'Рассчитывается значение к выдаче
   'и запоминается в соответствующей ячейке
   Cells(i + 1, 4) = К_Выдаче(i) 'Значение к выдаче возвращается в таблицу Excel
  Next i
End Sub
```

Задание

Используйте согласованный с преподавателем вариант задания (табл. 1), выполненную на его основе таблицу Excel, созданный вами макрос и написанную программу вычислений в таблице с использованием переменных VBA. Модифицируйте созданную вами линейную программу так, чтобы в ее составе использовались операторы выбора последовательности вычислений, ветвления, цикла. При этом исходные данные первоначально должны быть считаны из таблицы Excel, а результаты вычислений возвращены в нее.

Порядок выполнения работы

1. Откройте созданную вами рабочую книгу Excel. Скопируйте свою таблицу на новый лист. Удалите в ней все формулы. Запустите интегрированную среду разработки VBA.
2. Скопируйте созданную вами программу в окне редактора кода, измените название программы.
3. Убедитесь в том, что строкой Option explicit задан режим обязательного объявления переменных.
4. Напишите (скорректируйте) коды объявления внутренних переменных своей программы и задайтесь их типом данных. Прокомментируйте их в тексте программы.
5. Введите в текст программы проверку считанных из таблицы Excel значений переменных. Для этого задайтесь диапазоном возможных значений переменных, например минимальное и максимальное значение, и в случае выхода переменной из указанного диапазона присвойте ей необходимое граничное значение за счет использования оператора выбора последовательности вычислений If Then Else EndIf.
6. Организуйте повторяющиеся вычисления в программе с помощью оператора Do While Loop и Do Until Loop с предусловием.
7. Модернизируйте цикл вычислений или введите дополнительный цикл с использованием оператора Do Loop While и Do Loop Until с постусловием.
8. Продемонстрируйте возможности работы оператора цикла For To Next.
9. Изменяя значения переменных в таблице Excel и используя отладчик, убедитесь в правильности выполнения программы.
10. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

Контрольные вопросы

1. Как задать последовательность выполнения операций?
2. Какие операции разрешены в языке VBA?

3. Чем оператор отличается от операции?

4. Каково назначение оператора присваивания? Как он обозначается в алгоритмах?

5. Для каких целей используется символ модификации в алгоритмах?

6. Чем отличается оператор If Then Else EndIf от оператора Select Case End Select?

7. В чем заключается необходимость применения операторов цикла в программировании?

8. В каких случаях целесообразно использовать операторы Do Loop While с постусловием?

9. В каких случаях целесообразно использовать оператор For To Next?

10. Как можно запрограммировать бесконечно выполняющийся цикл?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, рисунок алгоритма и текст написанной вами программы с построчным комментарием ее действий. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Лабораторная работа №5

Функции и процедуры. Создание пользовательской функции Excel

Методические указания

Составление программ для ЭВМ требует большого объема трудозатрат. Вполне очевидно, что один раз созданные и проверенные программы представляют собой самостоятельную ценность. Программисты стремятся использовать свои разработки в новых программных проектах и, как следствие, создают методы, позволяющие относительно несложно включать ранее разработанные коды в новые программные изделия. Достаточно быстро стало понятно, что обычное механическое копирование кодов в новую программу чревато серьезными ошибками. Для уменьшения вероятности появления ошибок при использовании ранее созданных программ в языках программирования высокого уровня была внедрена концепция так называемых функций и процедур. Для ее практической реализации потребовались существенные

доработки в системе команд процессора, результатом которых явилось появление специальных команд вызова функции (процедуры) и возврата в точку вызова.

Ключевая идея создания функций и процедур заключалась в обеспечении возможности многократного обращения к одной и той же последовательности кодов из разных мест программы. По своей сути термины функция и процедура в языках высокого уровня взаимозаменяемы. Отличие одного от другого сводится к непринципиальной разнице в способах их оформления в теле программы и, что более важно, в способах оформления вызова. Некоторые языки программирования, например Си, вообще рассматривают только функции. В языке VBA сохранились описатели Function для обозначения функций и Sub для обозначения процедур.

Функцией или процедурой называется самостоятельная программа, предназначенная для решения определенной задачи. Поскольку любая работоспособная программа попадает под это определение, можно сделать вывод, что функцией может быть любая последовательность кодов. На самом деле это действительно так. Написанная нами программа на языке высокого уровня, оформленная в соответствии с правилами языка, оттранслированная и запущенная на выполнение представляет собой функцию, запускаемую, например, операционной системой. В данном случае правила оформления такой программы представляют собой ничто иное, как правила оформления функций операционной системы. Самостоятельный интерес представляют собой правила оформления функций, написанных на языке высокого уровня и вызываемых из других программ, написанных также на том же или другом языке высокого уровня или на ассемблере. Наиболее строго определены правила создания функций в том случае, когда для написания вызывающей программы и собственно функции используется один и тот же язык программирования.

Практический смысл использования функций (процедур) в программировании определяется следующими обстоятельствами. Появляется возможность разбиения большой программы на отдельные составляющие, разработка которых гораздо проще разработки всей программы. Сокращается объем кодов программы за счет удаления повторяющихся действий и замены их вызовами. Повышается надежность программного обеспечения, поскольку программа использует уже многократно проверенные последовательности кодов. Все это, в конечном итоге, ведет к росту производительности труда программиста.

Изложенные соображения играют важную роль при проектировании сложных программ. Так, например, взявшись за выполнение задачи, программист разбивает ее на набор возможно менее связанных между собой функций или процедур, каждая из которых решает некую самостоятельную задачу. Подобный прием называется декомпозицией и широко используется на практике. Очевидно, что для любой более или менее сложной задачи можно найти чрезвычайно большое, если даже не бесконечное количество вариантов декомпозиции. Поэтому выбор конкретного варианта разбиения задачи во многом определяется используемой методологией ее проектирования, а также опытом и пристрастиями разработчика.

При изучении способов создания функций (процедур) следует принимать во внимание следующие моменты:

- Каждая функция (процедура) имеет имя. Это имя является идентификатором и должно быть тем или иным способом объявлено.
- Каждая функция (процедура) имеет свои коды, которые должны быть оформлены заданным языком программирования способом. Эти коды называются определением функции.
- Для решения задачи функция (процедура) может потребовать набор аргументов (исходные данные), которые передаются ей в момент вызова.
- Функция (процедур) может возвращать результаты своих вычислений (возвращаемые данные) в вызывающую программу. Возврат значений может, в частности, производиться через список аргументов.
- Каждая функция (процедура) должна быть вызвана по имени. Если вызов отсутствует, то функция выполняться не будет.

Имя функции (процедуры) рассматривается как ее идентификатор и составляется исходя из правил составления идентификаторов конкретного языка программирования.

Можно выделить два вида функций, которые используются в программах. С одной стороны, это функции, которые созданы программистом для решения своей собственной задачи. В этом случае программист создает коды необходимой ему функции и оформляет их в соответствии с правилами языка программирования. С другой стороны, в программах могут использоваться так называемые библиотечные функции. Обычно с их помощью выполняются некие часто встречающиеся действия: некоторые математические вычисления, операции проверки типов, преобразования форматов, обработки строк, работы со временем и датами и тому подобное. Полный список имеющихся в языке функций можно получить, воспользовавшись, например, системой помощи, которая стандартно вызывается нажатием клавиши F1 при запущенной системе программирования.

При использовании библиотечных функций, обращаясь к ним в своей программе, программист использует написанные другими неизвестными ему программистами коды для решения собственной задачи. Для обеспечения работы всей системы программисту необходимо позаботиться о подключении к программе кодов библиотечных функций. Обычно это происходит на этапе редактирования связей. Заметим, что в большинстве случаев программист не имеет доступа к исходным кодам библиотечных функций и пользуется только описанием их действий (назначением функции) и описанием списка аргументов.

Под определением функции или процедуры обычно понимают создаваемую программистом последовательность операторов программы, которая после трансляции может быть вызвана из другой программы. Если в программе используются встроенные процедуры или функции, то об их определении программисту беспокоиться не приходится. Их объектные коды заранее размещены в специальных библиотеках, используемых компилятором, и автоматически подключаются к итоговой программе. Задумываться об определениях программисту надо в том случае, когда он хочет создать собственные функции или процедуры.

Формально функция в VBA может быть описана так:

```
[Public или Private] [Static] Function Имя [(СписокАргументов)] [As Тип]
  [Операторы]
  [Имя=Выражение]
  [Exit Function]
  [Операторы]
  [Имя=Выражение]
End Function
```

Если указано ключевое слово Public (используется по умолчанию), процедура может быть вызвана из других процедур любых модулей. Ключевое слово Private означает, что процедура может быть вызвана только из того модуля, в котором она описана. Из соображений повышения надежности программирования рекомендуется, как правило, использовать ключ Private.

Установленный ключ Static означает, что локальные переменные процедуры сохраняют свои значения между вызовами и могут быть использованы в последующих вычислениях при следующем вызове процедуры (время жизни переменной). Из соображений повышения надежности программирования использовать ключ Static не рекомендуется

Имя функции - это обычный идентификатор языка VBA.

СписокАргументов представляет собой перечисление аргументов функции. Он имеет еще одно название: список формальных параметров. Функция может иметь один аргумент (формальный параметр) или несколько. Как и обычные переменные, формальные параметры имеют определенный тип. Их основным отличием от обычных переменных является то обстоятельство, что под их хранение не выделяется память машины, а сами они используются в определении функции только для указания последовательности действий с аргументами функции. Каждый элемент списка формальных параметров имеет следующий формат:

[Optional] [ByVal или ByRef] [ParamArray] ИмяПеременной[()] [As Тип] [=поУмолчанию]

Ключевое слово Optional означает, что элемент списка является необязательным аргументом и должен иметь тип Variant. Все последующие элементы списка должны иметь такой же ключ и тип. Необязательные аргументы могут отсутствовать в списке переменных функции при записи оператора ее вызова.

Ключ ByVal означает, что параметр передается по значению. Если задан этот ключ, то вызывающая программа может передать в функцию значение аргумента, однако изменить это значение функция никаким способом не может. Этот прием призван защитить данные вызывающей программы, и может использоваться как основной при односторонней передаче данных от вызывающей программы к функции.

Ключ ByRef (используется по умолчанию) указывает, что параметр передается по ссылке. Это означает, что функции известен физический адрес памяти формального параметра. При необходимости функция может произвести запись по этому адресу (например, оператором присваивания). Подобный прием оказывается удобным для возврата результатов вычислений функции в вызывающую программу через список формальных параметров в том случае, когда оказывается необходимым вернуть больше одного параметра. При использовании процедур это вообще единственный способ возврата результатов вычислений.

Ключевое слово ParamArray может быть использовано только с последним элементом списка формальных параметров и позволяет передавать динамически объявляемый массив.

Ключ Тип представляет собой тип передаваемого параметра (табл. 2.2), а значение поУмолчанию может использоваться только с ключом Optional и задает значение переменной.

После заголовка функции следует конечное число обычных операторов языка VBA, представляющих собой тело определения функции. Если в их состав входит

оператор объявления переменных Dim, то имеет место объявление собственных локальных переменных функции. Если в заголовке функции не указан ключ Static, то эти переменные не сохраняют свои значения между вызовами, и каждый раз значения в них должны записываться заново. Кроме операторов объявления, в состав тела определения могут входить операторы присваивания, цикла и другие. В качестве их аргументов могут выступать как локальные переменные, константы, так и формальные параметры. Последние выступают как полноправные участники любых операций и операторов с той лишь оговоркой, что свое конкретное значение они получат только в момент вызова.

Результатом работы функции является некое значение, например число, которое вычисляется в теле функции. Возвращаемое значение должно иметь некий тип, указанный в заголовке функции как As Тип, соответствующий типу возвращаемого функцией значения. Для указания того, что все-таки является результатом вычислений функции и должно быть возвращено в вызывающую программу, в определении функции записывается отдельный оператор присваивания. В его левой части указывается Имя функции (из ее заголовка), а в правой - возвращаемое значение.

Формальное описание процедуры в VBA похоже на формальное описание функции и имеет вид:

[Public или Private] [Static] Sub Имя [(СписокАргументов)]
  [Операторы]
  End Sub

Формат элементов списка формальных параметров процедуры аналогичен формату формальных параметров функции. Таким образом, кроме ключевых слов заголовка и окончания, единственным принципиальным отличием определения функции от определения процедуры является наличие ее в тексте определения функции оператора [Имя=Выражение], указывающего возвращаемое в точку вызова значение.

Встречаются задачи, в которых в точку вызова необходимо возвращать не одно, а несколько значений. Поскольку функция может вернуть в вызывающую программу через оператор присваивания только одно значение, похожие задачи приходится решать способом передачи данных через список формальных параметров по ссылке. Если формальный параметр описан с ключом ByRef, то это означает, что вызываемая функция получает в свое распоряжение не копию данных, а адрес ячейки памяти, в которой эти данные находятся. Как следствие, у вызываемой функ-

ции появляется возможность изменить содержимое ячейки памяти вызывающей программы. Для этого в определении функции оператором присваивания задаются необходимые значения формальному параметру. В момент вызова процедуры (функции) формальному параметру ставится в соответствие фактическая ячейка памяти вызывающей программы. Именно в ней и произойдут указанные в определении изменения.

Вызов процедуры в языке VBA производится из любого места основной (вызывающей) программы за счет включения в ее текст специального оператора вызова. Вызов процедуры записывается как отдельный оператор с использованием ключевого слова Call. После него должно стоять имя процедуры и список ее фактических параметров, записанный в круглых скобках. Под фактическими параметрами понимаются имена ячеек памяти, объявленных в вызывающей программе. Очевидно, что если процедуре должно быть передано некоторое значение в виде аргумента, то вызывающая программа предварительно должна занести это значение в свою ячейку с использованием, например, оператора присваивания. Далее эта ячейка должна быть указана на соответствующем месте в списке формальных параметров.

Примечание. Альтернативным и часто используемым вариантом вызова процедур в VBA является просто запись имени процедуры с перечислением ее аргументов (фактических параметров) без заключения их в круглые скобки.

В отличие от процедуры, функция возвращает некоторое значение в точку вызова. Поэтому вызов функции производится с оператором присваивания. В левой части оператора указывается имя переменной, куда должен быть записан результат вычислений функции, а в правой ее имя и в круглых скобках аргументы (фактические параметры). Фактические аргументы функции выбираются из числа ячеек вызывающей программы.

Имя функции (процедуры) заносится компилятором в таблицу идентификаторов при первом вызове или при компиляции ее кодов, оформленных в виде текста программы и соответствующих заголовков с окончаниями (Sub Имя ([Аргументы]) [Операторы тела функции] End Sub или Function Имя ([Аргументы]) As Тип [Операторы тела функции] End Function).

Пример 17. Пример программы, реализующей задачу рис. 1 использованием процедуры задания цвета шрифта в ячейке Excel.

```
Sub Расчет_заработной_платы6()
Dim Начислено(1 To 4) As Currency, Налог(1 To 4) As Currency, _
К_Выдаче(1 To 4) As Currency, i As Integer
For i = 1 To 4
```

```
    Начислено(i) = Cells(i + 1, 2)  'В первую ячейке массива Начислено записывается
    'содержимое второй строки и второй колонки исходной таблицы Excel
    If Начислено(i) > 1000000 Then
       Начислено(i) = 1000000
       Call Изменение_цвета_шрифта_в_ячейке(i + 1, 2, "Желтый") 'Вызов процедуры с
            ' ключевым словом Call. Параметры процедуры заключены в круглые скобки
    Else
    End If
    If Начислено(i) < 0 Then
       Начислено(i) = 0
       Изменение_цвета_шрифта_в_ячейке i + 1, 2, "Красный" 'Вызов процедуры без
            ' ключевого слова Call. Параметры процедуры в круглые скобки не заключаются
    Else
       Call Изменение_цвета_шрифта_в_ячейке(i + 1, 2, "Сброс")
    End If
    Налог(i) = Начислено(i) * 0.12  'Рассчитывается значение налога и запоминается
    'в соответствующей ячейке
    Cells(i + 1, 3) = Налог(i) 'Значение налога возвращается в таблицу Excel
    К_Выдаче(i) = Начислено(i) - Налог(i) 'Рассчитывается значение к выдаче
    'и запоминается в соответствующей ячейке
    Cells(i + 1, 4) = К_Выдаче(i) 'Значение к выдаче возвращается в таблицу Excel
  Next i
End Sub
Sub Изменение_цвета_шрифта_в_ячейке(Строка As Integer, Столбец As Integer, Цвет As String)
Dim C As Integer
   Select Case Цвет
      Case "Красный": C = 3
      Case "Желтый": C = 6
      Case "Зеленый": C = 10
      Case Else: C = 0 'Автоматический выбор (Авто)
   End Select
   Cells(Строка, Столбец).Font.ColorIndex = C
End Sub
```

Примечание. Интегрированная среда разработки VBA в окне редактора кодов предлагает в качестве сервиса возможность указания имени переменной и типа данных при наборе операторов вызова функции или процедуры. Если функция или процедура ранее была объявлена и была выполнена компиляция проекта, после набора ее имени всплывает перечень ее аргументов.

При программировании в Excel иногда возникает необходимость создания дополнительных по отношению к стандартному библиотечному набору функций. Такие функции могут быть запрограммированы в ячейках таблицы обычным для Excel способом (**Вставка, Функция…**). Далее программист выбирает нужную ему *Категорию* функции, в соответствии с которой произведена их классификация. Если создать в модуле проекта VBA функцию, имеющую атрибут Public, используемый по умолчанию, то эта функция появится в списке библиотечных функций Excel в категории *Определенные пользователем*. Этот прием оказывается удобным для введения в систему Excel дополнительных возможностей программирования нестандартных, но многократно используемых действий. Так с его помощью легко реализовать возмож-

ности операторов ветвления и цикла, которые в обычной конфигурации оказываются недоступными.

Пример 18. Пример создания пользовательской функции Excel на VBA.

```
'Функция, определенная пользователем
Public Function Расчет_налога(Начислено As Integer)
  Расчет_налога = Начислено * 0.12
End Function
```

Задание

Используйте согласованный с преподавателем вариант задания (табл. 1), выполненную на его основе таблицу Excel, написанную программу вычислений в таблице с использованием переменных VBA. Модифицируйте созданную вами программу так, чтобы перенести часть вычислений в процедуру или функцию. При этом исходные данные первоначально должны быть считаны из таблицы Excel, а результаты вычислений возвращены в нее.

Порядок выполнения работы

1. Откройте созданную вами рабочую книгу Excel. Скопируйте свою таблицу на новый лист. Удалите в ней все формулы. Запустите интегрированную среду разработки VBA.
2. Выберите фрагменты кодов вашей программы, которые будут реализованы в виде процедуры или функции.
3. Создайте определение процедуры на основе выбранного вами фрагмента. Определите перечень формальных параметров процедуры и задайте их тип.
4. Замените выбранные фрагменты кодов программы на вызовы процедуры. Объявите и задайте фактические параметры процедуры.
5. Создайте определение функции на основе выбранного вами фрагмента. Определите перечень формальных параметров функции и задайте их тип.
6. Замените выбранные фрагменты кодов программы на вызовы функции. Объявите и задайте фактические параметры функции.
7. Проверьте работоспособность всей программы в режиме отладчика с использованием команды **Step Into**.

8. Ознакомьтесь с принципом отладки программы командой **Step Over,** автоматически выполняющим вызываемую процедуру или функцию и командой **Step Out** автоматически завершающей выполнение текущей процедуры или функции.

9. Введите в процедуру или функцию проверку значений аргументов с помощью оператора If Then Else EndIf.

10. Воспользуйтесь оператором Select Case End Select в процедуре или функции для демонстрации возможностей его работы.

11. Отметьте в таблице Excel факт выхода значения данных за пределы диапазона. Для этого, например, измените цвет шрифта в ячейке.

12. Проверьте правильность комментариев с учетом возможных изменений в тексте программы и, при необходимости, измените и их.

13. Создайте функцию, определенную пользователем.

14. Перейдите на лист Excel. Выполните команду **Вставка, Функция…** Выберите из категории *Определенные пользователем* запрограммированную вами функцию. Задайте данные и убедитесь в правильности ее выполнения.

15. Проверьте работоспособность функции в режиме отладчика с использованием команды **Step Into**.

16. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

Контрольные вопросы

1. Чем отличается объявление функции от ее определения?
2. В чем заключается практический смысл использования функций или процедур?
3. Что такое список формальных параметров и чем формальные параметры отличаются от фактических?
4. В каких случаях формальный параметр целесообразно передавать по ссылке, а в каких по значению?
5. Как вызвать библиотечную функцию VBA?
6. Какой смысл имеет задание типа функции в ее определении?

7. Чем отличается вызов функции от вызова процедуры?
8. Как создать определенную пользователем функцию Excel?
9. Каковы особенности отладки программы, использующей функции или процедуры?
10. Как результаты работы функции или процедуры могут быть получены в вызывающей программе?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, рисунок алгоритма созданной вами программы включая алгоритм функции или процедуры, текст написанной вами процедуры (функции). Приведите алгоритм и текст созданной вами функции пользователя. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Лабораторная работа №6

Классы и объекты

Методические указания

Хотя применение функций и процедур существенно упрощает создание программ (повышает производительность труда программиста), их использование в сложных программных системах наталкивается на ряд принципиальных ограничений. По своей сути обычная функция (процедура) представляет собой так называемый автомат без памяти. Это означает, что ее реакция на входное воздействие однозначно определена в момент разработки и никак не зависит от текущей ситуации.

Некоторые языки программирования (в том числе и VBA) допускают использование так называемых глобальных переменных. Написанная с использованием таких переменных функция может иметь существенно больший набор реакций на входное воздействие, поскольку в этом случае отклик функции зависит не только от текущих аргументов, но и от состояния ее глобальных переменных. Поскольку эти переменные существуют все время работы программы, функция, входящая в состав программы, может использовать их, в частности, для сохранения результатов вычислений от вызова к вызову. Аналогично можно использовать и статические пере-

менные. Отметим, что в отличие от обычной функции, такая функция представляет собой автомат с памятью.

Исследования в области надежности программного обеспечения показали, что использование глобальных и статических переменных существенно увеличивает вероятность программной ошибки. Их основной причиной является возникающая неопределенность момента изменения состояния переменной, так как доступ к глобальной переменной имеет, в том числе и ошибочно, любая другая процедура или функция программы. Это обстоятельство в конечном итоге привело к появлению целой методологии программирования - так называемому структурному программированию. В его основе лежит концепция проектирования программы сверху вниз, модульное программирование и структурное кодирование. Предполагается, что модуль в структурном программировании представляет собой законченную конструкцию с одним входом и одним выходом, что, в частности, запрещает использование в нем глобальных переменных, которые по своей сути являются средством дополнительного воздействия на поведение модуля. Поэтому при проектировании программы сверху вниз заранее оговаривается перечень всех аргументов модулей, причем они обязательно передаются через список формальных параметров.

С другой стороны, глобальные и статические переменные позволяют существенно упростить межмодульные связи и сократить количество аргументов функции (процедуры). Поэтому стремление ряда руководителей административно внедрить методы структурного программирования наталкивалось на явное или скрытое сопротивление программистов, для которых подобные действия приводили, в конечном итоге, к усложнению межмодульных интерфейсов.

Попытки найти компромисс между потребностями практики программирования с одной стороны и требованиями обеспечения надежности программирования с другой привели к созданию специфических типов функций и процедур, называемых объектами. В отличие от обычных функций и процедур, объекты имеют переменные, значения которых сохраняются от обращения к обращению. В то же время доступ к этим переменным возможен только через сам объект за счет использования его собственных свойств и методов. Это обстоятельство существенно снижает вероятность ошибки программирования связанной с несанкционированным изменением значения глобальной или статической переменной.

Функции и процедуры в программировании создавались, в первую очередь, для обеспечения возможности их многократного вызова из различных точек программы. Поскольку реакция функции как автомата без памяти на одинаковое воздействие всегда одинакова, различные по функциональному назначению, но одинаковые по алгоритму фрагменты программы могут реализовываться одним и тем же программным кодом. Так, например, все функции печати имеют один и тот же алгоритм. Поэтому было бы заманчиво написать универсальную функцию печати, не зависящую от вида, типа и состояния устройства. С другой стороны, при печати данных на разные устройства, приходится принимать во внимание их текущие настройки, состояние и историю работы. Если эти данные брать не из аргументов функции и не из глобальных или статических переменных, то приходится создавать механизм их хранения. Одним из вариантов такого механизма является создание собственных функции для каждого устройства, имеющих одинаковый алгоритм работы и программный код, но разные для каждого устройства ячейки данных, хранящие информацию об их состоянии. В конечном итоге такие функции получили название объектов.

Объект – это комбинация кода и данных. Код объекта фактически представляет собой набор функций или процедур одинаковый для всех схожих объектов. С каждым объектом связывается свой набор данных, который может быть изменен средствами кода объекта. Этот набор появляется в памяти машины в момент создания объекта и исчезает вместе с его удалением.

Для реализации подобного подхода к программированию функций, систематизации объектов и стандартизации принципов работы с ними, в языки программирования было введено понятие класс. Класс - это некоторое множество объектов, имеющих общую структуру и поведение. Фактически класс содержит набор функций и процедур, описывающих свойства и поведение объектов. Этот набор хранится в единственном экземпляре в виде программного кода и используется всеми объектами. Кроме этого, класс содержит описание структуры данных каждого объекта.

Как было показано в примере 9, структура представляет собой специфический тип данных (в языке VBA тип данных, определяемый пользователем). Там же было отмечено, что для создания собственно переменной типа объявленной структуры, эта переменная должна быть явно описана в программе и иметь свое уникальное имя. Поскольку за каждым элементом такой переменной закреплены соответствую-

щие ячейки памяти, можно говорить, что она обладает неким состоянием, определяемым содержимым закрепленных за ней ячеек памяти, и идентичностью, определяемой именем переменной в программе.

При введении в языки программирования понятия объект к описанным уже характеристикам состояния и идентичности добавили характеристику поведения. Под поведением обычно понимают реакцию объекта на внешнее воздействие сводящуюся к изменению его состояния. В отличие от функции с глобальными и статическими переменными, возможность изменения состояния объекта существенно ограничивается и определяется заранее, что позволяет сохранить надежность программирования на разумном уровне. Поведение объекта описывается набором функций и процедур класса. Наконец, как и в случае структуры, характеристика идентичности представляет собой свойство объекта, отличающее его от других объектов. Это имя задается в момент создания объекта.

Модуль класса содержит коды общих для всех объектов функций и процедур и описание структуры данных объекта. Для выделения памяти под хранение переменных объекта необходимо выполнить набор действий по его созданию. Далее каждый объект использует общие для всего класса процедуры и функции, но оперирует с собственными данными, которые хранятся в памяти до удаления объекта.

В языке VBA для изменения состояния объекта пользуются так называемыми свойствами. Все объекты одного класса имеют одинаковый набор свойств. Конкретный набор свойств объекта, возможность их считывания и изменения определяется при создании класса в виде набора функций специального вида. Поведение объекта в языке VBA задается методами и событиями. По своей сути метод представляет собой обычную процедуру. Возможные методы для объекта также описываются на этапе создания класса.

Событие представляет собой действие, распознаваемое объектом, для которого можно запрограммировать отклик. Событие вызывается действиями пользователя (например, щелчок мышью) или генерируются системой (например, деление на ноль).

Создание класса в языке VBA представляет собой типовую последовательность действий. Сначала командой **Insert, Class Module** интегрированной системы отладки VBA создается так называемый модуль класса и ему присваивается имя, являющееся далее именем пользовательского класса. После этого описываются пе-

ременные класса. Обратите внимание на то, что при этом описании определяется только структура ячеек данных объектов класса. Сами переменные класса получат конкретные значения адресов в памяти машины только после того, когда на основе класса будут создаваться объекты, причем каждый объект будет иметь свой индивидуальный набор таких ячеек.

Затем определяется процедура инициализации класса Sub Class_Initialize(). Эта процедура выполняется каждый раз, когда создается новый объект и может быть использована, например, для задания начальных значений переменным класса, динамического переобъявления размеров массивов в соответствии с требованиями конкретной задачи, чтения файлов и т.п. Далее может быть создана процедура Sub Class_Terminate(). Она описывает действия, которые надо выполнить перед удалением объекта (например, печать результата). Если начальных или завершающих действий с объектом не требуется, то эти процедуры можно не создавать. Синтаксис определения процедур Sub Class_Initialize() и Sub Class_Terminate() имеет вид:

```
Private Sub Class_Initialize()
  [Операторы]
End Sub

Private Sub Class_Terminate()
  [Операторы]
End Sub
```

После этого создаются функции вида Property Get, Property Let и Property Set[2], позволяющие читать и задавать значения свойств переменных класса. Их формальное описание имеет вид:

```
[Public или Private] [Static] Property Get Имя [(СписокАргументов)] [As Тип]
  [Операторы]
  [Имя=Выражение]
  [Exit Function]
  [Операторы]
  [Имя=Выражение]
End Property

[Public или Private] [Static] Property Let Имя [(СписокАргументов)]
  [Операторы]
  [Имя=Выражение]
  [Exit Function]
  [Операторы]
End Property

[Public или Private] [Static] Property Set Имя [(СписокАргументов)]
```

[2] Функция Property Set позволяет устанавливать значения свойств объекта в составе класса.

```
  [Операторы]
  [Имя=Выражение]
  [Exit Function]
  [Операторы]
End Property
```

Наконец, создаются методы класса, которые оформляются в виде обычных процедур Sub.

```
[Public или Private] [Static] Sub Имя [(СписокАргументов)]
  [Операторы]
  End Sub
```

Примечание. Непосредственно запрограммировать события класса нельзя, однако свойства классов могут использовать события других стандартных объектов VBA.

Пример 19. Пример программы описания класса, реализующий задачу рис. 1. Метод класса основан на процедуре из примера 17. Дополнительно в класс введено свойство, позволяющее изменять количество строк таблицы.

```
Dim Фамилия() As String, Начислено() As Currency, Налог() As Currency, _
К_Выдаче() As Currency
'Переменные класса представляют собой набор динамически объявляемых массивов.
'Необходимость динамического объявления связана с неопределенностью числа строк
'таблицы конкретного объекта
Dim Размер_таблицы As Integer, Ставка_налога As Single

Private Sub Class_Initialize()
   Размер_таблицы = 4
   Ставка_налога = 0.12
   ReDim Фамилия(1 To Размер_таблицы)
   ReDim Начислено(1 To Размер_таблицы)
   ReDim Налог(1 To Размер_таблицы)
   ReDim К_Выдаче(1 To Размер_таблицы)
End Sub

Private Sub Class_Terminate()
'Действия не предусматриваются
End Sub

Sub Расчет_заработной_платы()
For i = 1 To Размер_таблицы
   Начислено(i) = Cells(i + 1, 2)  'В первую ячейке массива Начислено записывается
   'содержимое второй строки и второй колонки исходной таблицы Excel
   If Начислено(i) > 1000000 Then
      Начислено(i) = 1000000
      Call Изменение_цвета_шрифта_в_ячейке(i + 1, 2, "Желтый") 'Вызов процедуры с
' ключевым словом Call. Параметры процедуры заключены в круглые скобки
   Else
   End If
   If Начислено(i) < 0 Then
      Начислено(i) = 0
      Изменение_цвета_шрифта_в_ячейке i + 1, 2, "Красный" 'Вызов процедуры без
' ключевого слова Call. Параметры процедуры в круглые скобки не заключаются
   Else
      Call Изменение_цвета_шрифта_в_ячейке(i + 1, 2, "Сброс")
   End If
```

```
    Налог(i) = Начислено(i) * Ставка_налога   'Рассчитывается значение налога и запоминается
    'в соответствующей ячейке
    Cells(i + 1, 3) = Налог(i) 'Значение налога возвращается в таблицу Excel
    К_Выдаче(i) = Начислено(i) - Налог(i) 'Рассчитывается значение к выдаче
    'и запоминается в соответствующей ячейке
    Cells(i + 1, 4) = К_Выдаче(i) 'Значение к выдаче возвращается в таблицу Excel
   Next i
End Sub

Sub Изменение_цвета_шрифта_в_ячейке(Строка As Integer, Столбец As Integer, Цвет As String)
Dim C As Integer
   Select Case Цвет
      Case "Красный": C = 3
      Case "Желтый": C = 6
      Case "Зеленый": C = 10
      Case Else: C = 0 'Автоматический выбор (Авто)
   End Select
   Cells(Строка, Столбец).Font.ColorIndex = C
End Sub

Property Let Число_строк_таблицы(Размер As Integer)
Размер_таблицы = Размер
ReDim Фамилия(1 To Размер_таблицы)
ReDim Начислено(1 To Размер_таблицы)
ReDim Налог(1 To Размер_таблицы)
ReDim К_Выдаче(1 To Размер_таблицы)
End Property

Property Get Число_строк_таблицы() As Integer
Число_строк_таблицы = Размер_таблицы
End Property
```

Поскольку объекты как элементы языка программирования создавались на основе структур, процедур и функций, нотация, используемая для обращения к ним, сохранилась. Так символ точка, используемый в структурах для разделения общего имени переменной и ее составной части, используется при записи объектов для разделения его имени и свойства или метода. Как это принято в функциях, аргументы свойств объекта, если они требуются, заключаются в круглые скобки. При использовании методов аналогично правилам вызова процедур VBA список параметров следует после указания метода и в круглые скобки не заключается (смотри пример 17).

```
Объект.Свойство = Значение_свойства
Объект.Свойство1(параметр 1, параметр2, …, параметрN) = Значение_свойства
Значение_свойства = Объект.Свойство
Значение_свойства = Объект.Свойство1(параметр 1, параметр2, …, параметрN)
Объект.Метод
Объект.Метод1 параметр 1, параметр2, …, параметрN
```

Примечание. Интегрированная среда разработки VBA в окне редактора кодов предлагает в качестве сервиса возможность конкретного выбора имени свойства или метода,

допустимых для данного объекта, из автоматически раскрывающегося списка. Если свойства или методы ранее были включены в состав класса и выполнена компиляция проекта, после набора символа точки автоматически открывается список возможных имен. Этой возможностью необходимо пользоваться для избежания синтаксических и логических ошибок при наборе текста программы.

После того, как класс создан, можно на его основе создавать конкретные экземпляры, а также выполнять с ними различные действия. Для создания объекта на основе имеющегося класса необходимо выполнить следующую последовательность действий:

- объявить переменную оператором Dim и указать ее тип как имя используемого класса;
- создать объект оператором Set с именем ранее объявленной переменной, используя ключевое слово New и указание имени класса (синтаксис оператора Set смотри стр. 39).

После выполнения этих действий в памяти создается набор переменных, связанных с указанным объектом. С этого момента оказываются доступными свойства и методы класса в отношении созданного объекта. Аналогично можно создать еще несколько объектов используемого класса и выполнять с ними разнообразные действия. При этом значения переменных класса и, следовательно, свойств разных объектов могут быть различными, а при использовании одних и тех же методов будет получен разный результат.

Если работа с объектом завершена, то он может быть удален из памяти оператором Set с ключевым словом Nothing (смотри стр. 39). После его выполнения занимаемая объектом память освобождается. Кроме этого, все созданные объекты автоматически удаляются из памяти в момент завершения работы программы.

Пример 20. Программа использования класса, реализующая задачу рис. 1.

```
Sub Расчет_заработной_платы7()
'Объявление переменной с типом созданного класса
Dim Первый_объект As Ведомость_заработной_платы, i As Integer
'Создание объекта
Set Первый_объект = New Ведомость_заработной_платы
'Использование метода объекта
Первый_объект.Расчет_заработной_платы
'Использование свойства объекта. Задание нового числа строк таблицы
Первый_объект.Число_строк_таблицы = 3
'Чтение текущего числа строк таблицы
i = Первый_объект.Число_строк_таблицы()
'Удаление объекта
Set Первый_объект = Nothing
End Sub
```

Возможность программирования классов и создания на их основе необходимого количества однотипных объектов оказывает важное влияние на способ декомпозиции сложной программной системы при ее проектировании. Проектировщик создает описание некой сущности, являющейся предметом исследования и проектирования, в виде программной модели. Описание системы взаимодействия объектов между собой позволяет составлять совокупную модель описываемой сущности в виде множества взаимодействующих по определенным правилам объектов различных фрагментов сущности. В этом случае декомпозиция исходной задачи может рассматриваться как иерархия классов объектов с учетом их взаимодействия. Подобный прием называется объектной декомпозицией [0].

Задание

Используйте согласованный с преподавателем вариант задания (табл. 1), выполненную на его основе таблицу Excel, написанную программу вычислений в таблице. Создайте на ее основе класс, позволяющий производить требуемое количество таблиц и обработку данных в них.

Порядок выполнения работы

1. Откройте созданную вами рабочую книгу Excel. Скопируйте свою таблицу на новый лист. Удалите в ней все формулы. Запустите интегрированную среду разработки VBA.
2. Воспользовавшись командой **Insert, Class Module** создайте модуль класса. Если эти действия выполняются в первый раз, то в окне проекта появится папка Class Modules, а в ней запись Class1. Если в проекте классы уже создавались, то вставка нового модуля класса просто добавит запись в папку Class Modules.Далее командой **View, Properties Window** вызовите окно свойств создаваемого класса. В этом окне задайте имя класса вместо предлагаемого по умолчанию имени Class1. Выполните команду **Debug, Compile VBAProject**.
3. В окне редактора кодов скопируйте созданную вами в процессе выполнения предыдущей работы процедуру или функцию, предназначенную для вычислений в таблице, в модуль класса. Далее используйте ее в качестве основы для программирования свойств и методов создаваемого класса.

4. Определите перечень переменных класса. Скорее всего, в их числе будут внутренние переменные вашей процедуры и, возможно, некоторые дополнительные. Выделите из состава используемой процедуры (функции) объявления и перенесите их в начало модуля класса в виде самостоятельных строк. Добавьте необходимые дополнительные переменные. Обратите внимание на массивы переменных. Если они также являются переменными класса, их целесообразно объявлять как динамические массивы (пример 8). В этом случае надо предусмотреть переменную класса для хранения актуального размера массива, а также свойства, позволяющие задать ей новое значение и предусматривающее изменение размеров массивов (ReDim).

5. Создайте процедуру Sub Class_Initialize() и разработайте коды действий, выполняемых в момент создания объекта. В частности, в этой процедуре можно предусмотреть объявление динамических массивов в соответствии с неким начальным их размером.

6. Создайте процедуру Sub Class_Terminate() и разработайте коды действий, выполняемых в момент удаления объекта. В частном случае процедура может не выполнять никаких действий.

7. Создайте функции вида Property Get и Property Let, позволяющие читать и задавать значения переменных класса и выполнять на их основе обработку данных. Если в качестве основы программирования класса вы использовали функцию, то оформите ее в виде одной из процедур Property Get или Property Let.

8. Если для основы программирования класса вы использовали процедуру, то она будет являться одним из методов класса. При необходимости создайте еще методы класса, оформленные в виде обычных процедур Sub.

9. Напишите программу создания объекта на основе разработанного вами класса. Для этого объявите переменную с типом созданного вами класса. Далее создайте новый объект. Используйте методы класса. Используйте свойства класса.

10. Запустите созданную программу в режиме отладки командами **Debug, Step Into**. На каждом шаге выполнения контролируйте изменение внутренних переменных программы в окне локальных переменных **Locals**. Убедитесь в пра-

вильности выполнения расчетов. При выполнении фрагментов программы, обеспечивающих запись рассчитанных значений в ячейки Excel, дополнительно убедитесь в правильности выполнения этих действий.

11. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

Контрольные вопросы

1. Какие проблемы возникают при практическом использовании функций или процедур?
2. В чем заключаются основные идеи метода структурного программирования?
3. В чем отличие автомата с памятью от автомата без памяти?
4. Каковы преимущества и недостатки использования глобальных переменных в тексте программы?
5. Чем объекты отличаются от обычных функций и процедур?
6. Чем класс отличается от объекта?
7. Каким образом на этапе выполнения программы можно получить доступ к переменным класса?
8. Чем метод класса отличается от свойства класса?
9. Что надо сделать для создания объекта класса?
10. Что такое событие?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, алгоритм и текст написанной вами программы класса, а также алгоритм и текст программы, создающей объекты на основе созданного класса и демонстрирующей возможности работы с ними. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Лабораторная работа №7

Базовые операторы ввода-вывода VBA и работа с файлами

Методические указания

Данные в памяти ЭВМ хранятся в виде двоичных чисел. Единственное, что может сделать процессор с данными – это извлечь содержимое некой ячейки памяти, выполнить над ним некоторое заранее оговоренное и выбранное из перечня возможных действие и занести число (результат) назад в память в ту же или другую ячейку. Вполне естественным является следующий вопрос: каким способом числа попали в ячейку памяти первоначально?

Существует всего четыре варианта ответа. Во-первых, это данные могли остаться в ячейке памяти от предыдущей программы или, если программа загружается в память сразу после включения машины, в ячейке памяти осталась случайная комбинация установок триггеров, возникшая после подачи напряжения на ОЗУ. Часто такие данные называют мусором. Во-вторых, данные могут быть размещены в ячейке вместе с программой, то есть сама программа при компиляции предусматривает некое начальное значение в конкретной ячейке памяти. В-третьих, число могло попасть в ячейку в результате выполнения команды процессора на запись данных в ОЗУ, например, при выполнении оператора присваивания. Наконец, в четвертых, число могло быть занесено в ячейку памяти в результате выполнения команды ввода.

На первый взгляд существенных различий между двумя последними вариантами нет. Тем не менее, следует принимать во внимание следующее обстоятельство: в третьем варианте заносимое число является результатом вполне конкретных действий над данными, которые при необходимости могут быть повторены. В то же время, в четвертом варианте занесенное число представляет собой результат реального физического воздействия на устройство ввода в данный момент времени, которое может быть уникальным и никогда более не повторяющимся.

Вполне естественным было бы ожидать то, что любой язык программирования высокого уровня содержит в своем составе команды или операторы ввода вывода. Поскольку на первом этапе развития вычислительной техники способы подключения устройств ввода-вывода к процессору существенно рознились, языки программирования предусматривали отдельные операторы для вывода на печать, вывода на

дисплей, ввода с клавиатуры, файловой работы. В настоящее время произошла унификация подобного рода операторов. Так, в языке VBA сохранились несколько операторов, смысл которых представлен в табл. 7. Основным назначением базовых операторов ввода вывода VBA является работа с файлами. Файл представляет собой единицу хранения данных, имеющих конкретный смысл. Так, файл может быть программой (исполняемыми кодами), исходным текстом, документом, просто хранилищем записей. Физически файл хранится, как правило, на накопителях на магнитных дисках, хотя операционная система машины рассматривает любое внешнее устройство как приемник или источник файлов.

В VBA существует три способа организации данных в файле. Эти типы определяют и тип доступа к файлу. Различают:

- последовательные файлы, предназначенные для чтения и записи последовательных блоков символьных данных, представляющих собой последовательность кодов символов, включая служебные;
- файлы произвольного доступа, предназначенные для записи и чтения данных, структурированных как записи фиксированной длины;
- двоичные файлы, предназначенные для записи и чтения числовых данных произвольной длины и представляющие собой частный случай файла произвольного доступа с длиной записи 1 байт.

Перед началом работы программы с файлом он должен быть открыт инструкцией Open, которая задает имя открываемого файла (включая указание пути к нему). При открытии указывается номер открываемого канала системы. Дополнительно может быть задан тип файла (последовательного доступа, произвольного доступа, двоичный), ключ записи и состояние файла (для чтения, записи или добавления). Кроме этого для файла может быть указан его задаваемый размер в байтах. Номер свободного канала может быть определен предварительно с помощью инструкции FreeFile.

Пример 21. Открытие для чтения несуществующего файла. Выполнение этого фрагмента программы приведет к выдаче системой сообщения об ошибке “Файл не найден” и прекращению работы программы.

```
Open "test1. sss " For Input As #1
Close #1
```

Пример 22. Открытие для записи не существующего файла последовательного доступа. Выполнение этого фрагмента программы приведет к созданию в текущем каталоге нового файла test1.hhh. Поскольку вывод в файл не производился, размер созданного файла 0 байт.

```
Dim canal As Integer
'Определение номера свободного файлового канала ввода-вывода
canal = FreeFile()
Open "test1. sss " For Output As #canal
Close #canal
```

Если файл уже существует, то при открытии его в состоянии Output старый файл удалится, а новый запишется на его место. Если предполагается внесение изменений в уже существующий файл, он должен быть открыт в состоянии Append.

Пример 23. Программа записи файла исходных данных для начисления зарплаты в соответствии с рис. 1

```
Open "Зарплата.sss" For Output As #1
'Запись в файл
Print #1, "Иванов В.Н."
Print #1, 1234
Write #1, "Трофимова Л.А."
Write #1, 1234
Write #1, "Семенова Е.Г.", 1000, "Степанов А.Г.", 900
Close #1
```

Файлы с произвольным доступом позволяют обращаться к записи в файле по ее номеру. Такая возможность обеспечивается за счет создания регулярной структуры записей определенного формата, которую легко обеспечить, например, за счет типов данных, определяемых пользователем.

Пример 24. Предполагается, что создана структура (тип данных, определяемый пользователем) следующего вида:

```
Type Запись_файла
Фамилия_И_О As String
Начислено_Ведомость As Currency
End Type
```

Тогда программа работы с файлом произвольного доступа может содержать следующие операторы:

```
Dim record As Запись_файла
Dim nomber As Integer, j As Integer
Open "Зарплата1.rrr" For Random As #1
'Запись в файл произвольного доступа
record.Фамилия_И_О = "Иванов В.Н."
record.Начислено_Ведомость = 1234
Put #1, 1, record
'Чтение только что сделанной записи
Get #1, 1, record 'Считывается только что записанное значение из файла
record.Фамилия_И_О = "Трофимова Л.А."
'record.Начислено_Ведомость содержит считанное из файла число 1234
Put #1, 2, record
record.Фамилия_И_О = "Семенова Е.Г."
record.Начислено_Ведомость = 1000
'Запись в файл произвольного доступа по текущему номеру счетчика
Put #1, , record 'Действие с записью 3
j = Seek(1) 'Значение указателя записи равно 4
record.Фамилия_И_О = "Степанов А.Г."
record.Начислено_Ведомость = 900
Put #1, j, record
Close #1
```

Таблица 7. Операторы ввода вывода VBA[3]

Действие	Ключевые слова
Создать или редактировать файл	Open
Закрыть файл	Close, Reset
Управление форматами записи	Format, Print, Print #, Spc, Tab, Width #
Копирование файлов	FileCopy
Чтение свойств файлов	EOF, FileAttr, FileDateTime, FileLen, FreeFile, GetAttr, Loc, LOF, Seek
Определение размера файла	FileLen
Установка или чтение атрибутов файлов	FileAttr, GetAttr, SetAttr
Управление свойствами файлов	Dir, Kill, Lock, Unlock, Name
Чтение последовательных файлов	Input #, Line Input #
Запись в последовательный файл	Print #, Write #
Чтение файлов произвольного доступа или двоичных	Get
Запись в файл произвольного доступа или двоичный	Put
Задание номера записи файла произвольного доступа	Seek

Файл произвольного доступа открывается с указанием типа Random и по умолчанию доступен для чтения и записи. Явное указание режима только чтения или только записи обеспечивается за счет добавления ключевых слов Access Read или Access Write.

Для записи в файл произвольного доступа можно воспользоваться оператором Put. Его первый параметр есть номер канала, второй – номер записи в файле (может отсутствовать), третий – имя переменной, значение которой надо записать в файл. Чтение информации из файла может быть осуществлено оператором Get, параметры которого аналогичны.

В системе существует внутренний указатель текущего номера рабочей записи файла. Его начальное значение равно нулю. Каждый раз при выполнении операторов Put и Get значение указателя становится равным номеру записи, с которой выполнялся оператор, плюс единица. Если в операторах Put и Get номер записи явно не задан, то новое действие будет выполняться с записью, номер которой содержится в указателе. Текущее значение указателя номера рабочей записи может быть получе-

[3] Для детального ознакомления с операторами целесообразно воспользоваться справочной системой и приведенными в ней примерами их использования.

но оператором Seek. Двоичные файлы представляют собой разновидность файлов с произвольным доступом с элементарными записями размером в один байт, в связи с чем указатель текущего номера рабочей записи двоичного файла имеет смысл счетчика байтов. Программирование действий с записями двоичных файлов проводится так же, как и в случае работы с файлами произвольного доступа.

В процессе работы с файлом изменение его содержимого происходит в буфере операционной системы, а не на диске. Это обстоятельство приходится принимать во внимание, например, при отладке программы. Если при работе с отладчиком интегрированной системы отладки выполнить команду **Run, Reset**, то изменения файла будут потеряны. Непосредственное завершение работы с файлом и запись данных на диск происходит в момент выполнения инструкции Close. Кроме этого, запись информации на диск (закрытие файла) происходит в момент завершения работы всей пользовательской программы. Для снижения вероятности ошибок программирования целесообразно всегда после завершения действий с файлом принудительно закрывать его инструкцией Close.

Задание

Используйте согласованный с преподавателем вариант задания (табл. 1), выполненную на его основе таблицу Excel, написанную программу вычислений в таблице и созданный на ее основе класс, позволяющий производить требуемое количество таблиц и обработку данных в них. Научитесь работать с последовательными файлами и файлами произвольного доступов. Напишите программу, позволяющую сохранять исходные данные вашей таблицы в виде файлов последовательного или произвольного доступов. Включите ее в состав методов класса. Используйте сохраненные файлы в следующем сеансе работы с таблицей для восстановления в ней исходных данных.

Порядок выполнения работы

1. Напишите программу, открывающую для чтения не существующий файл последовательного доступа. Ознакомьтесь с диагностикой системы.

2. Напишите программу, открывающую для записи несуществующий файл. Внесите в текст комментарии действий программы. Убедитесь, что после завершения ее работы создался новый файл.

3. Напишите программу, открывающую для записи существующий файл. С помощью оператора Print # выполните запись в файл некой информации. Закройте файл. Внесите в текст комментарии действий программы

4. Объявите в программе некоторую переменную из числа исходных данных вашей таблицы, откройте ранее записанный файл, считайте в режиме отладки в объявленную переменную записанные в файл данные оператором Input #, контролируя считанные значения в окне локальных переменных.

5. Измените записываемую в файл информацию и при повторном считывании убедитесь в том, что информация в файле изменилась.

6. Откройте файл в режиме Append. Сделайте запись в файл новой информации и закройте файл.

7. Откройте файл только для чтения и убедитесь, что файл содержит как предыдущую, так и новую записи.

8. Создайте файл с произвольным доступом. Запишите в него информацию. Закройте файл.

9. Напишите программу чтения информации из файла произвольного доступа. Внесите в текст комментарии действий программы. Проконтролируйте правильность ее работы.

10. Выберите тип файла для хранения исходных данных вашей таблицы. Создайте его и запишите в файл все исходных данные вашей таблицы.

11. Удалите исходные данные из таблицы, считайте их из созданного вами файла заново и занесите их в таблицу. Проконтролируйте правильность считывания информации в вашу таблицу.

12. Измените содержимое исходных данных вашей таблицы и проконтролируйте изменение файла произвольного доступа.

13. Подключите созданную вами программу в качестве метода созданного вами класса.

14. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

Контрольные вопросы

1. Как организован последовательный файл?
2. Как организован файл произвольного доступа?
3. Что нужно сделать для того, чтобы начать работу с файлом?
4. В чем заключается отличие в обращении к элементу данных последовательного файла и файла произвольного доступа?
5. Как определить факт достижения конца файла?
6. Как создать файл?
7. Чем отличается формат оператора Print от формата оператора Put?
8. Как указать, что файл открывается только для чтения?
9. Как определить номер свободного канала для открытия файла?
10. Как закрыть файл и в какой момент данные оказываются на диске?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, рисунок алгоритма и текст написанной вами программы работы с файлами последовательного и произвольного доступа. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Лабораторная работа №8

Ввод с клавиатуры и вывод на экран в VBA

Методические указания

При программировании обмена с клавиатурой и дисплеем можно воспользоваться специальными встроенными функциями InputBox и MsgBox языка VBA, предназначенными именно для этой цели. В отличие от обычных файловых операторов ввода-вывода эти функции позволяют стандартно, то есть так, как это принято в большинстве программ Windows, оформлять действия по вводу и выводу. В некотором смысле эти функции можно рассматривать как дополнительные операторы ввода – вывода, работающие по специальным правилам.

Предназначенная для ввода данных с клавиатуры функция InputBox выводит на экран диалоговое окно, содержащее сообщение, и поле ввода, устанавливает

режим ожидания ввода текста пользователем или нажатия кнопки, а затем возвращает в программу значение типа String, содержащее текст, введенный в поле. Формат записи функции:

InputBox(prompt[,title][,default][,Xpos][,Ypos][,helpfile,context])

Здесь prompt – строковое выражение, которое будет отображаться как сообщение в диалоговом окне; title - сообщение, отображаемое в заголовке окна (если оно опущено, то отображается имя приложения); default – сообщение, которое будет выводиться в строке при запуске; Xpos и Ypos задают положение окна на экране; helpfile и context – соответственно имя файла и номер раздела справочной системы.

Функция MsgBox выводит на экран диалоговое окно с сообщением и ожидает нажатия кнопки пользователем. Значение нажатой кнопки возвращается как число типа Integer. Формальная запись функции, которая не анализирует вид нажатой кнопки и вызывается как процедура, выглядит так:

MsgBox prompt[,buttons][,title][, helpfile,context]

Переменная buttons позволяет задать количество и виды кнопок и информационные значки (см. табл. 8). Если необходимо задать и то, и другое, то в параметр buttons записывается сумма соответствующих констант.

Если генерируется несколько кнопок, то узнать, какая из них была нажата, можно в результате анализа возвращаемого функцией MsgBox значения (см. табл. 9). В этом случае она вызывается через оператор присваивания, а ее параметры заключаются в круглые скобки:

Rezult = MsgBox (prompt[,buttons][,title][, helpfile,context])

Одноименные с функцией InputBox аргументы имеют тот же самый смысл, а значение buttons определяется как сумма констант, задающих число и тип отображаемых кнопок, тип используемого значка и основную кнопку.

На рис 9 показан внешний вид диалоговых окон для ввода информации, создаваемых функцией InputBox. Если пользователь нажмет кнопку OK или клавишу Enter, то функция вернет строку, набранную пользователем в рабочем окне. При нажатии кнопки Cancel возвращается пустая строка.

Внешний вид диалоговых окон, создаваемых функцией MsgBox, показан на рис. 10.

Пример 25. Программа, задающая с клавиатуры первую и вторую записи в файле " Зарплата.hhh":

```
Open " Зарплата.hhh" For Output As #1
'Запрос данных с клавиатуры
Rezult = InputBox("Фамилия И.О. сотрудника", "Учебник")
```

```
Print #1, Rezult
'Второй запрос данных с клавиатуры
Rezult = InputBox("Ставка заработной платы", "Учебник")
Print #1, Val(Rezult)
Close #1
```

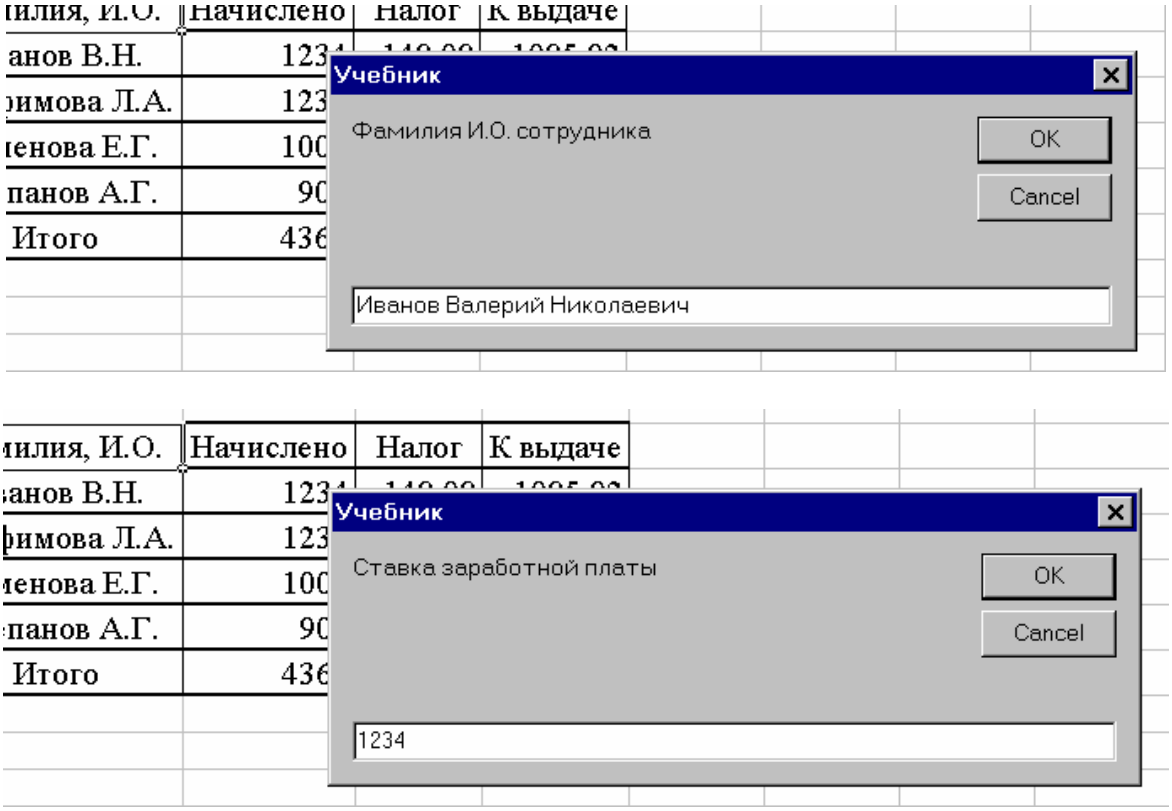


Рис. 9. Внешний вид окон, создаваемых функцией InputBox

Таблица 8. Константы аргумента buttons

Идентификатор константы	Значение	Пояснение
Количество и вид кнопок		
VbOKOnly	0	Только кнопка OK
VbOKCancel	1	Кнопки OK и Отмена
VbAbortRetryIgnore	2	Кнопки Стоп, Повтор, Пропустить
VbYesNoCancel	3	Кнопки Да, Нет, Отмена
VbYesNo	4	Кнопки Да и Нет
VbRetryCancel	5	Кнопки Повтор и Отмена
Информационные значки		
VbCritical	16	Ошибка
VbQuestion	32	Вопрос
VbExclamation	48	Утверждение

Идентификатор константы	Значение	Пояснение
VbInformation	64	Информация
Основная кнопка		
VbDefaultButton1	0	Кнопка 1
VbDefaultButton2	256	Кнопка 2
VbDefaultButton3	512	Кнопка 3
VbDefaultButton4	768	Кнопка 4

Таблица 9. Возвращаемые значения функции MsgBox

Идентификатор константы	Значение	Пояснение
VbOK	1	Нажато OK
VbCancel	2	Нажато Отмена
VbAbort	3	Нажато Прервать
VbRetry	4	Нажато Повторить
VbIgnore	5	Нажато Пропустить
VbYes	6	Нажато Да
VbNo	7	Нажато Нет

Пример 26. Чтение содержимое созданного файла:

```
Open " Зарплата.hhh" For Input As #1
Input #1, Rezult
'Выдача первой записи на экран
MsgBox "Фамилия И.О. сотрудника" & Chr(10) & Chr(13) & Rezult, _
vbOKOnly + vbExclamation, "Учебник"
Input #1, d
'Выдача второй записи на экран и обработка факта нажатия клавиш
j = MsgBox("Ставка заработной платы" & Chr(10) & Chr(13) & Str(d), _
vbYesNoCancel + vbExclamation, "Учебник")
Close #1
```

Ячейка j этой программы после ее выполнения содержит или число 6 (константа VbYes), если в процессе выполнения была нажата кнопка Да, или число 7 (константа VbNo), если была нажата кнопка Нет, или число 2 (константа vbCancel), если была нажата кнопка Отмена.

Задание

Используйте согласованный с преподавателем вариант задания (табл. 1), выполненную на его основе таблицу Excel, написанную программу вычислений в таблице и созданный на ее основе класс, включая методы работы с файлом. Изучите методы работы с функциями InputBox и MsgBox. Научитесь организовывать диалог с пользователем. Научитесь подключать созданные программы к пользовательскому классу.

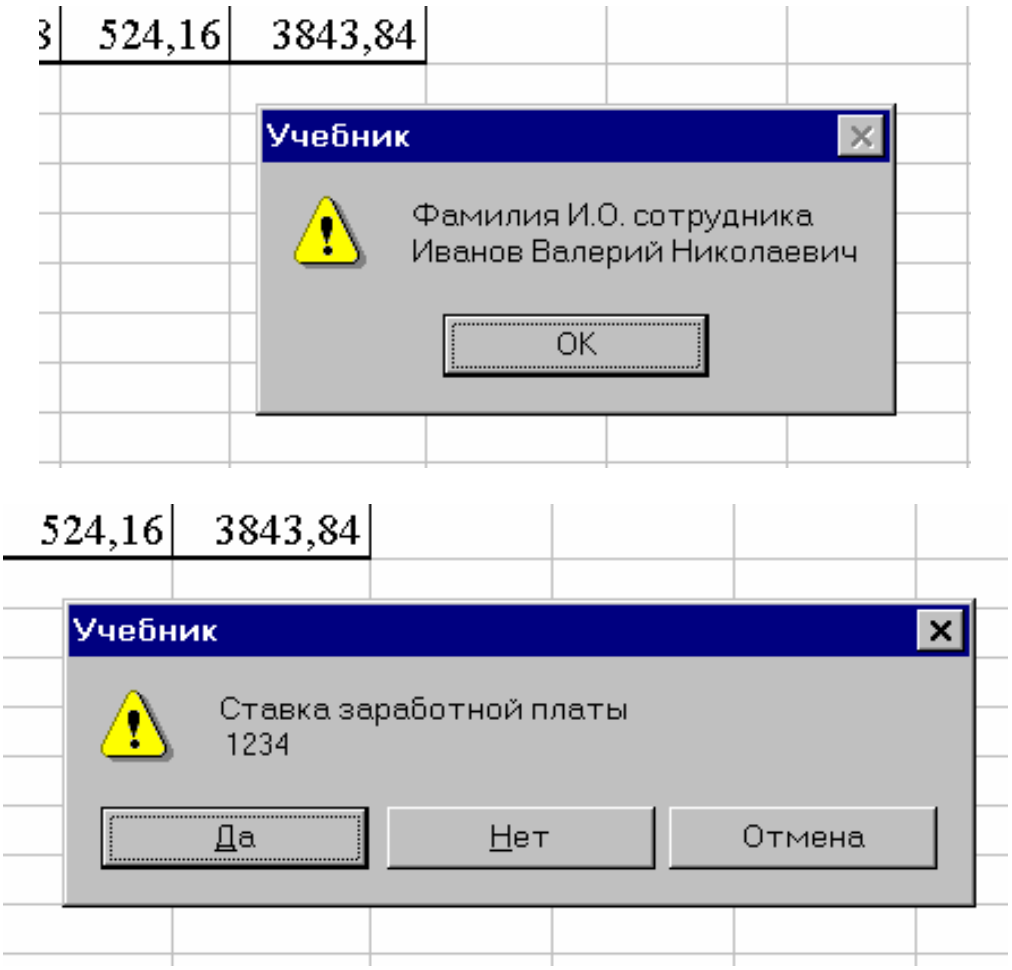


Рис. 10. Внешний вид сообщений, выдаваемых функцией MsgBox

Порядок выполнения работы

1. С использованием функций InputBox и MsgBox напишите программу диалога с пользователем, запрашивающего у него необходимость чтения старого или создания нового файла.

2. С использованием функций ввода с клавиатуры и вывода на экран напишите диалоговую программу, позволяющую вручную задавать имя требуемого файла.

3. Напишите с использованием функций InputBox и MsgBox программу, позволяющую вводить данные в файл на основе диалога. Введите данные в диалоге и создайте файл. Считайте данные из файла в таблицу и убедитесь в работоспособности программы.

4. Напишите программу, позволяющую задавать оператору вопрос о выборе способа ввода данных в файл (из таблицы или с помощью диалога).

5. Подключите программы к созданному вами классу. Модифицируйте программу инициализации Sub Class_Initialize() так, чтобы при создании объекта оператору задавался бы вопрос об источнике начальных данных (таблица или файл) и, если это файл, об имени файла.

6. Модифицируйте программу Sub Class_Terminate() своего класса так, чтобы перед удалением объекта оператор получал запрос о необходимости сохранения данных в файле и об имени этого файла, если он сохраняется.

7. Проверьте работоспособность созданных вами программ. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

Контрольные вопросы

1. В чем заключается отличие функций InputBox и MsgBox?
2. Каково назначение аргумента title?
3. Каково назначение аргумента prompt?
4. Каково назначение констант аргумента buttons функции MsgBox?
5. Что является результатом работы функции MsgBox?
6. Что является результатом работы функции InputBox?
7. Как функциями InputBox и MsgBox организовать доступ к файлу со стороны объекта?
8. Каков смысл возвращаемых значений функции MsgBox?
9. Как задать комбинацию кнопок на рабочей панели функции MsgBox?
10. Как определить нажатую на рабочей панели функции InputBox кнопку?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, алгоритм и текст написанной вами программы файлового ввода-вывода и текст модифицированных программ инициализации и удаления объекта класса. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Стандартные классы и объекты при взаимодействии Excel и VBA

Лабораторная работа №9

Элементы управления рабочего листа Excel

Методические указания

Программы, подготовленные с помощью VBA, предназначены для выполнения в многозадачной и многооконной операционной среде. Желательно, чтобы взаимодействие с этой средой осуществлялось в стандартном для нее виде. Учитывая то, что, во-первых, средства такого взаимодействия уже созданы, а, во-вторых, созданные средства предусматривают возможность их использования пользовательскими программами, целесообразно пользоваться уже существующими решениями, обеспечивающими взаимодействие оператора и ЭВМ. Реализация подобного взаимодействия существенно облегчена за счет предусмотренного разработчиками операционной среды объектного подхода к программированию.

Существующая библиотека классов и объектов чрезвычайно обширна, а ее детальное изучение может потребовать больших трудозатрат. Самое главное, детальное знание всей библиотеки обычно не требуется, поскольку пользователь, как правило, сталкивается с ограниченным кругом задач. Поэтому представляется целесообразным изучать не все существующие в системе стандартные классы и объекты, а только необходимые для решения конкретной задачи. При этом важно освоить технологию работы по внедрению, использованию и программированию объекта. В основу дальнейшего обучения положено обсуждение возможностей работы только с некоторыми стандартными классами и объектами, а освоение других классов можно осуществить самостоятельно по аналогии с рассмотренными на основе новой практической задачи.

Из всего множества классов и объектов операционной среды первоначально выделим встроенный в VBA класс элементов управления. Он используется для создания различных дополнительных компонентов рабочего листа Excel. В его состав входят классы более низкого иерархического уровня. Общим для всех классов явля-

ется наличие набора свойств, методов и возможность генерации и обработки различного рода событий.

Для внедрения объектов на рабочий лист Excel необходимо командой **Вид, Панели инструментов, Элементы управления** включить панель инструментов *Элементы управления*. В числе прочих на панели присутствует кнопка *Режим конструктора*. При ее нажатии можно создавать, видоизменять и задавать свойства объектов управления. Если кнопка *Режим конструктора* отпущена, то созданные элементы выполняют возложенные на них функции (рабочий режим). На рис. 11 представлен набор классов основных элементов управления, а также изображения кнопок панели *Элементы управления* с расшифровкой их названий.

Когда режим конструктора включен, можно внедрить на рабочий лист объект, относящийся к одному из классов. Для этого надо выбрать интересующий класс (например, Кнопка) нажатием соответствующей пиктограммы на панели инструментов. После этого надо установить маркер в том месте рабочего листа, где требуется установить новый объект, и щелкнуть левой кнопкой мыши. В ответ в составе листа Excel появится новый объект выбранного нами класса. При необходимости можно создавать несколько однотипных объектов, а система по умолчанию будет присваивать им после одинакового общего имени порядковые номера.

После установки объекта на лист можно осуществлять действия по редактированию его свойств. Существующий набор свойств объекта можно узнать нажатием кнопки *Свойства* на панели управления рис. 12. В ответ на экран генерируется таблица, которая содержит полный перечень свойств объекта. Он может быть упорядочен или по алфавиту, или по категориям. Меняя содержимое правых полей этой таблицы, можно задать новые значения свойств. Так, например, можно сделать новую надпись на объекте (Caption), новый идентификатор объекта в программе (Name) и так далее. Значения некоторых свойств (например, размер или положение кнопки) могут быть изменены и с помощью мышки прямо на листе Excel.

Набор методов, применимый к объекту, может быть получен, в частности, за счет всплывающего окна справки. Оно активируется в результате набора в тексте программы имени объекта и ввода символа точки (смотри примечание на стр. 65).

За каждым классом элементов управления закрепляется набор функций предназначенных для обработки различного рода событий, возникающих при работе с объектом. Для того, чтобы получить доступ к программам обработки событий, доста-

точно нажать кнопку *Исходный текст* на панели управления. В этом случае автоматически запускается VBA, на экране появляются окно структуры проекта и окно модуля кода, связанное с конкретными внедренными объектами. Обратите внимание на то, что окно модуля кода в структуре проекта связано с тем листом Excel, на который внедрен элемент управления.

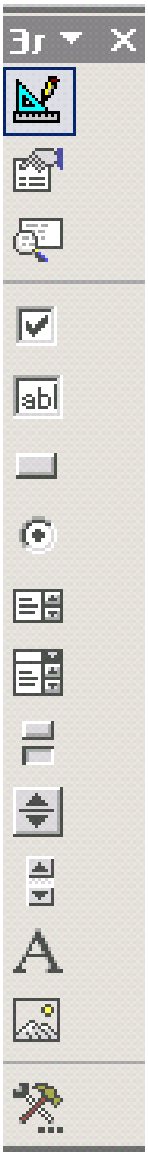


Рис. 11. Изображения кнопок управления панели Элементы управления, их названия и символические имена

После внедрения на рабочий лист кнопки окно модуля кода имеет вид рис. 13. Правая часть заголовка окна модуля кода содержит раскрывающийся список, в котором перечислены события, обрабатываемые объектом. Для создания заголовка программы обработки события достаточно щелкнуть его наименование в раскрытом ме-

ню. К числу стандартных событий, которые обычно есть у всех элементов, относятся щелчок (Click), двойной щелчок (DblClick), изменение (Change) и некоторые другие.

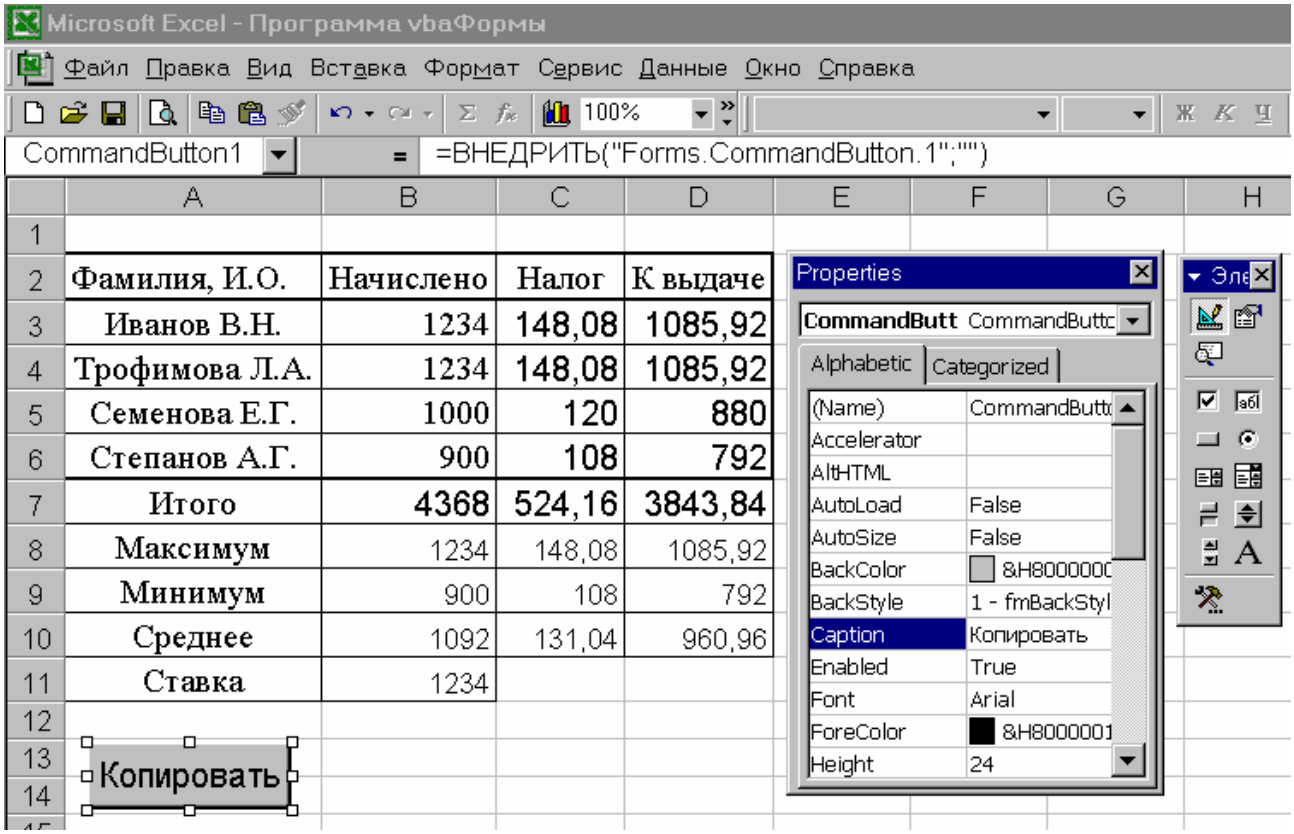


Рис. 12. Внешний вид рабочего листа с внедренной кнопкой и открытым окном свойств

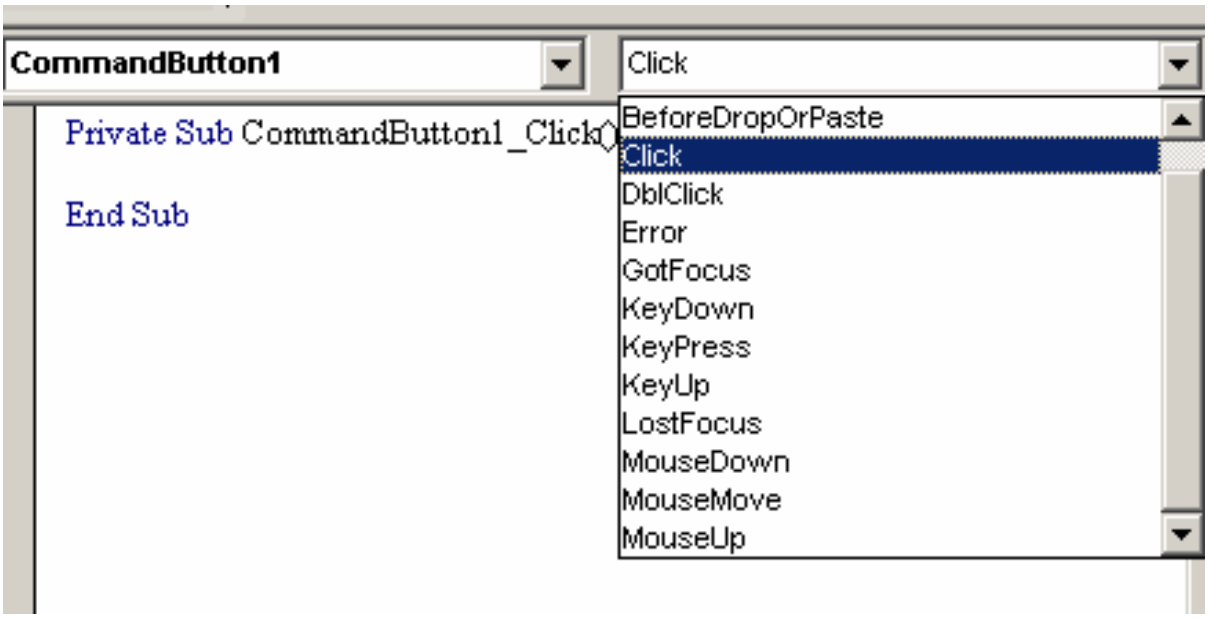


Рис. 13. Внешний вид окна модуля кода с раскрытым списком обрабатываемых событий

Интегрированная среда разработки создает заготовку процедуры обработки соответствующего события. Создавая программу обработки события, можно предусмотреть действия, которые должны быть выполнены в случае наступления соответствующего события. Если такая программа не создана, то соответствующее событие игнорируется.

Набор свойств и обрабатываемых событий одинаков для всех объектов одного класса. В тоже время объекты другого класса могут иметь свой набор свойств и обрабатываемых событий.

Рассмотрим подробнее некоторые классы, входящие в общий класс элементов управления. *Кнопка* (CommandButton) позволяет обрабатывать события, связанные с ее нажатием. Ее основная функция – генерация события Click, которое далее может быть обработано программным путем в результате выполнения соответствующей процедуры (рис. 13). Эта же процедура может быть вызвана и в некоторых других случаях.

Если кнопку можно только нажать, то *Флажок* (CheckBox) показывает, включено или выключено определенное условие, определяемое свойством Value и отображаемое на изображении флажка. Аналогичные возможности предоставляет элемент управления *Выключатель* (ToggleButton).

Пример 27. На рабочем листе Excel в режиме конструктора созданы два объекта: флажок (CheckBox1) и поле (TextBox1), а также набрана программа обработки события, возникающего после щелчка мышью по флажку и заключающаяся в проверке состояния флажка (свойство Value) и выдаче соответствующего сообщения в текстовом поле. Ее текст представлен ниже:

```
Private Sub CheckBox1_Click()
If CheckBox1.Value Then
   TextBox1.Value = "Флажок включен"
Else
   TextBox1.Value = "Флажок выключен"
End If
End Sub
```

В отличие от флажка и выключателя *Переключатель* (OptionButton) предоставляют пользователю выбор из двух или более возможностей. Они всегда работают как часть группы: выбор одного из них немедленно сбрасывает все другие переключатели группы. На каждом листе Excel можно разместить только одну группу переключателей.

Особенностью элементов *Список* (ListBox) и *Поле со списком* (ComboBox) является связанный с каждым экземпляром список хранимых значений. Именно этот список и отображается в окне. По умолчанию список имеет только один столбец, хо-

тя, при необходимости, свойством ColumnCount может быть задано несколько столбцов. Если количество элементов в списке превышает количество возможных отображаемых элементов, автоматически появляются полосы прокрутки.

Первоначально список пуст. Добавить в него элементы можно за счет использования свойства AddItem. Необходимо помнить, что добавленные элементы хранятся в списке до конца выполнения программы. Поэтому, если список многократно вызывается одинаковой процедурой, то записи в нем начинают повторяться. Для предотвращения этого необходимо перед активацией списка дополнительно очищать его методом Clear.

В процессе работы со списком пользователь выделяет одну из записей. Номер выбранной записи можно узнать, воспользовавшись свойством ListIndex. Оно доступно только во время выполнения программы. Это свойство устанавливает и возвращает текущий выбранный элемент. Если выбран первый элемент списка, значение свойства равно 0. Следующий элемент списка дает значение свойства 1, следующий 2 и так далее. Если не выбрано ни одно значение списка, то значение свойства -1. Отметим, что установка значения свойства ListIndex вызывает появление события Click для этого списка. Кроме этого, значение текущего выбранного элемента может быть получено с помощью свойства Text.

Еще одна возможность доступа к элементам списка реализуется через свойство List. При его использовании список рассматривается как массив, нижнее значение индексов которого равно 0.

Пример 28. Программа обработки события GotFocus с использованием свойств Clear, ColumnCount, AddItem, List таблицы рис. 12:

```
Private Sub ListBox1_GotFocus()
Dim Строка_списка As Integer, Колонка_списка _
 As Integer, Элементов As Integer, i As Integer
'Задание начального адреса размещения и количества элементов списка
   Строка_списка = 3
   Колонка_списка = 1
Элементов = 4
'Очистка списка и подготовка к его заполнению
   ListBox1.Clear
   ListBox1.ColumnCount = 2
'Заполнение списка тем
   For i = 0 To Элементов - 1
      ListBox1.AddItem (Cells (Строка_списка + i, Колонка_списка).Value)
      ListBox1.List(i, 1) = Cells(Строка_списка + i, Колонка_списка + 1).Value
   Next i
End Sub
```

Если запись списка выбрана, то дальнейшая работа программы может обеспечиваться за счет добавления к списку специальной кнопки для совместного использования. Процедура обработки факта нажатия этой кнопки может включать в себя считывание выбранного значения списка и последовательность дальнейших действий.

Пример 29. Программа обработки события Click:

```
Private Sub ListBox1_Click()
Dim AA As String, Выбор As Integer
'Определение номера выбранной строки списка
   Выбор = ListBox1.ListIndex
'Определение адреса активной ячейки. Адрес возвращается в формате A1
   AA = ActiveCell.Address
'Запись в активную ячейку значения из второй колонки списка
   Range(AA).Value = ListBox1.List(Выбор, 1)
End Sub
```

Элемент управления *Надпись* (Label) предназначен для отображения текста. Текущее значение текста может быть считано и, при необходимости, изменено свойством Caption.

Элемент управления *Поле* (Text Box) используются как для отображения, так и для ввода текста с клавиатуры в процессе выполнения программы. Содержимое поля может быть считано свойством Text. По умолчанию в текстовом поле отображается только одна строка без полос прокрутки. При необходимости на этапе разработки свойством MultiLine можно задать возможность работы с несколькими строками, а свойством ScrollBars включить полосы прокрутки.

Примечание. В категории *Другие элементы управления* меню рис. 11 присутствует еще ряд элементов, которые могут быть использованы для организации диалога. Среди них отметим *Элемент управления Календарь 10.0* предназначенный непосредственно для чтения и ввода дат.

Задание

Используйте согласованный с преподавателем вариант задания (табл. 1), выполненную на его основе таблицу Excel, написанную программу вычислений в таблице и созданный на ее основе класс, позволяющий производить требуемое количество таблиц и обработку данных в них. Научитесь внедрять на рабочий лист элементы управления и писать программы обработки возникающих с ними событий. Организуйте с их помощью ввод исходных данных в таблицу.

Порядок выполнения работы

1. Внедрите на рабочий лист кнопку, позволяющую при нажатии скопировать таблицу и ее исходные данные на новый лист. На этот лист внедрите кнопку, позволяющую вызвать процедуру вычисления данных в таблице. Далее в качестве рабочего будет использоваться этот лист.

2. Создайте на рабочем листе флажок. Напишите процедуру обработки факта нажатия флажка.

3. Создайте на рабочем листе выключатель. Напишите процедуру обработки факта нажатия выключателя.

4. Внедрите три переключателя на рабочий лист. Убедитесь, что созданная вами группа переключателей обеспечивает выбор только одного из возможных режимов. Напишите процедуру обработки состояния переключателей.

5. Внедрите на рабочий лист список и поле со списком. Напишите программу инициализации списков с учетом данных, имеющихся в таблице.

6. Внедрите на рабочий лист кнопку, обеспечивающую обработку информации, выбранной в списках, в результате ее нажатия.

7. Внедрите на рабочий лист надпись. Напишите процедуру изменения текста надписи, которая должна учитывать результат выбора в группе переключателей.

8. Внедрите на рабочий лист поле. Напишите процедуру считывания текущего содержания поля и отображения введенного текста в виде надписи.

9. Убедитесь в работоспособности созданных программ.

10. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

Контрольные вопросы

1. Что нужно сделать, чтобы установить элемент управления на рабочем листе Excel?
2. Как получить список свойств внедренного объекта?
3. Как посмотреть список событий внедренного объекта?
4. Как узнать список методов, который применим к объекту?

5. Чем элемент управления кнопка отличается от элемента управления флажок?

6. Чем элемент управления поле со списком отличается от элемента управления список?

7. Как добавить запись в элемент управления список?

8. Как узнать, какое значение было выбрано оператором при работе со списком?

9. Чем элемент управления поле отличается от элемента управления надпись?

10. Как перейти от программирования событий элемента управления непосредственно к его работе?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, алгоритмы и тексты написанных вами процедур обработки событий элементов управления (кнопка, флажок, выключатель, переключатели, список и поле со списком, надпись и поле). Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Лабораторная работа №10

Конструирование форм

Методические указания

Пользовательские формы (User Form) – это класс программируемых, в том числе визуальными и графическими средствами, операций ввода вывода, настраиваемых под конкретную задачу. На основе пользовательских форм можно создавать диалоговые окна разрабатываемых приложений. Грамотное проектирование пользовательских форм применительно к конкретной задаче позволяет полностью исключить ручные операции с рабочим листом Excel, что может оказаться весьма удобным, например, из соображений защиты информации, находящейся на рабочем листе, от ошибочных действий пользователя. Созданная пользовательская форма представляет собой класс, на основе которого может производиться генерация объектов, представляющих собой типовые для разрабатываемой системы панели управления.

Для того, чтобы создать новую пользовательскую форму, необходимо запустить пакет Excel, выбрать в нем пункт главного меню **Сервис, Макрос, Редактор Visual Basic** и перейти в редактор Visual Basic. Затем необходимо выбрать пункт

меню редактора VBA **Вставить User Form** и получить на экране заготовку пользовательской формы и панель элементов управления, которые могут быть в нее внедрены. С каждой формой связывается свое окно редактора кодов.

Если форма уже создана, то для обращения к ее графическому представлению надо щелкнуть ее имя в списке форм проекта. Переход к окнам свойств и кода формы может быть осуществлен, в частности, через меню, вплывающем при выборе формы и щелчке правой кнопкой мышки.

Конструирование формы осуществляется за счет внедрения в заготовку необходимых элементов управления и создания процедур обработки событий аналогично тому, как это делалось при внедрении элементов управления на рабочий лист Excel. Дополнительно необходимо принять во внимание то обстоятельство, что создаваемая форма представляет собой разновидность пользовательского класса со всеми вытекающими из этого следствиями. Например, в одной программе можно открыть несколько одинаковых форм (объектов), хранящих свои самостоятельные данные (состояние) и имеющих общие свойства.

При открытии формы в программе возникает событие ее инициализации. Для описания необходимых в этом случае действий служит процедура UserForm_Initialize. Соответственно при удалении формы выполняется процедура UserForm_Terminate. Скорее всего, эти процедуры придется дополнительно создать при программировании формы. В процессе их выполнения можно предусмотреть, например, заполнение полей элементов формы, начальную установку кнопок, флажков, переключателей и другие необходимые действия (например, считывание исходных данных из файла и запись результатов в файл).

Пример 30. Создадим пользовательскую форму для задачи расчета заработной платы (рис. 1),. Будем считать, что, из соображений надежности хранения данных, значения в колонку Начислено, имеющуюся на рабочем листе Excel, могут вноситься только с помощью специального меню, включающего в свой состав Список (List Box) для выбора собственно значения ставки заработной платы, задаваемой в отдельной таблице. Фамилия сотрудника, для которого задается выбранное значение ставки, выбирается с помощью имеющегося в составе формы меню, выполненного как Поле со списком (Combo Box). Операция начисления проводится в тот момент, когда нажата имеющаяся в форме Кнопка (CommandButton). Рабочее окно редактора кода VBA, окно свойств и окно проекта изображены на рис. 14. После создания формы UserForm1 ее свойство Заголовок (Caption) получило значение Панель управления, а соответствующее свойство внедренной кнопки значение Запись. Для обозначения окон использовались два элемента управления Надпись (Label), имеющие значения Заголовка (Caption) Выбор ставки и Выбор сотрудника.

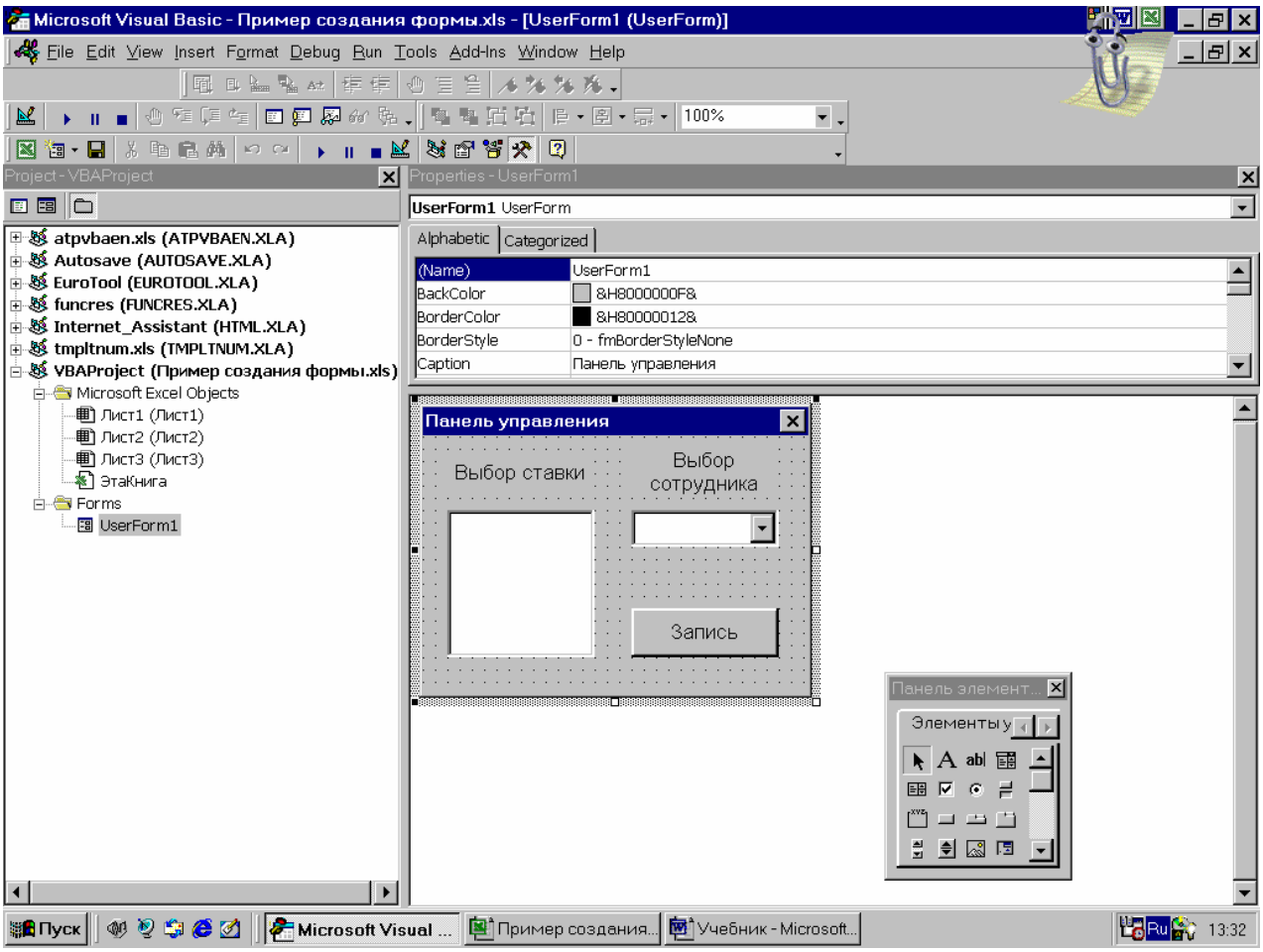


Рис. 14. Рабочее окно VBA с созданной формой

Программирование формы (создание модуля класса форм) сводится к объявлению данных (переменных класса), написанию процедур обработки событий в форме и ее составляющих, созданию свойств и методов формы. Программирование этих процедур осуществляется в окне редактора кодов.

Если необходимо, то в составе формы можно объявит собственные переменные, которые будут играть роль переменных класса. Для этого необходимо вставить соответствующие строчки в начало окна кодов.

Полный список обрабатываемых формой событий можно посмотреть в открывающихся меню в верхней части окна редактора кодов. Аналогично можно получить и список событий, связанных с конкретным элементом формы.

Существующие свойства формы могут быть дополнены функциями Property Get, Property Let и Property Set. Для этого их достаточно добавить в связанное с конкретной формой окно кода, Аналогичным приемом можно воспользоваться при создании новых методов формы.

Пример 31. Для задачи, рассмотренной в примере 30, предусмотрим включение пользовательской формы за счет нажатия специальной кнопки на рабочем листе. Разместим на рабочем листе Excel кнопку, нажатие на которую приводит к активизации формы, и назовем ее Форма. Предположим, что создаваемая нами форма имеет имя UserForm1. Процедура обработки нажатий на кнопку имеет вид:

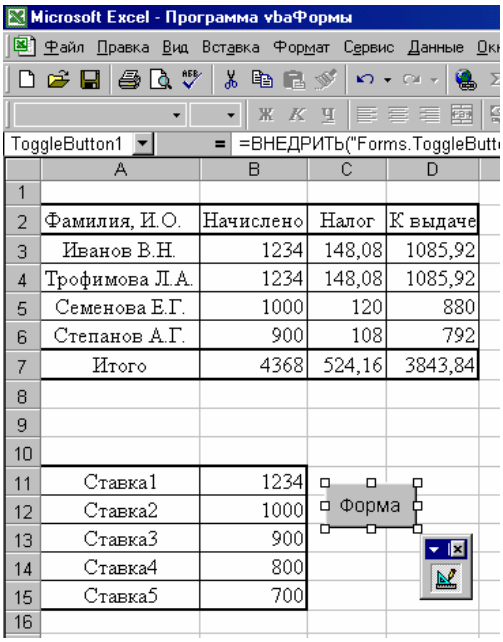
```
Private Sub ToggleButton1_Click()
If ToggleButton1.Value Then
UserForm1.Show
Else
UserForm1.Hide
End If
End Sub
```

Здесь используется свойство кнопки (ToggleButton) последовательно при нажатиях менять свое значение (Value) на значения *истина* или *ложь* и связанный с формой метод отображения (Show) и скрытия (Hide) пользовательской формы.

Пример 32. Предположим, что на рабочем листе Excel подготовлена и запрограммирована таблица рис. 15, предназначенная для расчета заработной платы в соответствии с рассматриваемым примером, а также таблица значений ставки оплаты труда. Создадим процедуру инициализации формы, которая будет выполняться каждый раз, как форма отображается на экране. Опишем начальные назначения переменных формы, которые будут выполнены во время выполнения этой процедуры. Допустим, что начальное размещение списка сотрудников на рабочем определено и не будет изменяться. Кроме этого известны количество сотрудников, которым начисляется заработная плата, а также определены положения на листе списка возможных используемых ставок и их количество.

```
'Инициализация формы
Private Sub UserForm_Activate()
'Объявление переменных с указанием признака %, соответствующего типу Integer
Dim Строка_списка%, Колонка_списка%, Элементов%, i%
'Задание начального адреса размещения поля со списком
   Строка_списка = 3
   Колонка_списка = 1
'Задание количества элементов списка
   Элементов = 4
'Очистка списка и подготовка к его заполнению
   ComboBox1.Clear
   ComboBox1.ColumnCount = 2
'Заполнение списка тем
   For i = 0 To Элементов - 1
ComboBox1.AddItem (Cells(Строка_списка + i, Колонка_списка).Value)
'Запоминание рабочего адреса
ComboBox1.List(i, 1) = Cells(Строка_списка + i, Колонка_списка + 1).Address
   Next i
'Задание начального адреса размещения списка
   Строка_списка = 11
   Колонка_списка = 1
'Задание количества элементов списка
   Элементов = 5
'Очистка списка и подготовка к его заполнению
   ListBox1.Clear
   ListBox1.ColumnCount = 2
'Заполнение списка тем
   For i = 0 To Элементов - 1
```

```
  ListBox1.AddItem (Cells(Строка_списка + i, Колонка_списка).Value)
  ListBox1.List(i, 1) = Cells(Строка_списка + i, Колонка_списка + 1).Value
   Next i
End Sub
```



	A	B	C	D
1				
2	Фамилия, И.О.	Начислено	Налог	К выдаче
3	Иванов В.Н.	1234	148,08	1085,92
4	Трофимова Л.А.	1234	148,08	1085,92
5	Семенова Е.Г.	1000	120	880
6	Степанов А.Г.	900	108	792
7	Итого	4368	524,16	3843,84
8				
9				
10				
11	Ставка1	1234		
12	Ставка2	1000		
13	Ставка3	900		
14	Ставка4	800		
15	Ставка5	700		
16				

Рис. 15. Исходный лист для программируемой задачи.

В программе используется свойство ColumnCount классов ComboBox и ListBox, позволяющее задавать количество полей в списках, а также методы Clear и AddItem, обеспечивающие начальную очистку и добавление строк в список. Во вторые колонки созданных списков с использованием свойства List занесены соответственно адреса ячеек, в которые будут записаны значения начисленной заработной платы и сами значения используемых ставок.

Пример 33. Работа с созданной формой заключается в выборе значения ставки и фамилии сотрудника, которому эта ставка назначается. Эти действия сводятся к работе с Полем со списком и Списком и обеспечиваются библиотечными функциями классов ComboBox и ListBox. Специальных действий, связанных с создаваемой формой, не требуется, поэтому никаких пользовательских процедур обработки событий в Поле со списком и Списке не создается. После того, как выбор сделан, должны быть произведены изменения на рабочем листе Excel. Командой на внесение изменений является нажатие кнопки Запись создаваемой формы. Если щелкнуть мышкой по этой кнопке, возникает событие Click, которое обрабатывается процедурой CommandButton1_Click, связанной с этой кнопкой.

```
Private Sub CommandButton1_Click()
Dim Запись As String 'Хранение адреса записи
Dim Выбор%, ВыборCombo%
'Определение номера выбранной строки списка
   Выбор = ListBox1.ListIndex
'Определение номера выбранной строки поля со списком
   ВыборCombo = ComboBox1.ListIndex
If ВыборCombo >= 0 Then
'Определение адреса  ячейки. Адрес возвращается в формате A1
'Проверка факта выбора
  If Выбор >= 0 Then
```

```
 'Нацеливание рабочей ячейки
   Запись = ComboBox1.List(ВыборCombo, 1)
   Else
   End If
Else
End If
'Запись в по адресу записи значения из второй колонки списка
   If Запись <> "" Then
'Адрес определен
   Range(Запись).Value = ListBox1.List(Выбор, 1)
   Else
   End If
End Sub
```

Процедура использует свойство ListIndex, позволяющее определить номер выбранного ранее элемента Поля со списком и Списка. Если в процессе работы с формой выбор в списках был сделан, то эти номера отличны от 0, что и используется для проверки возможности изменения содержимого ячеек рабочего листа Excel. Поскольку адрес ячейки, в которую заносится значение ставки, был ранее сохранен во второй колонке Поля со списком, а само значение ставки во второй колонке Списка, эти данные используются для внесения изменений на рабочий лист Excel.

Пример 34. Работа с созданной формой может проводиться как угодно долго до тех пор, пока она не будет выключена нажатием кнопки отмены в правом верхнем углу формы. В ответ на это запускается процедура прекращения работы с формой. В нашем случае эта процедура восстанавливает значение выключателя на рабочем листе.

```
Восстановление исходного значения выключателя на активном листе
Private Sub UserForm_Terminate()
   With ActiveSheet
      .ToggleButton1.Value = False
   End With
End Sub
```

Все внедренные на рабочий лист переключатели связаны между собой (см. стр. 45). Если при конструировании формы возникает необходимость создания нескольких независимых групп переключателей, то используются так называемые контейнеры. В этом случае связь между переключателями реализуется внутри контейнера. В качестве одного из вариантов создания контейнера является использование рамок (Frame).

Для того, чтобы воспользоваться рамкой, ее необходимо внедрить на рабочий лист. Соответствующая кнопка имеется на панели инструментов **Toolbox**, открывающейся при конструировании формы (рис. 14). Далее курсором указывается место размещения рамки на рабочем листе. После того, как рамка внедрена, можно разместить в ней переключатели. Для создания второй группы переключателей нужно внедрить в форму вторую рамку и установить в ней дополнительные переключатели.

На рис. 16 показан результат конструирования формы с двумя группами переключателей.

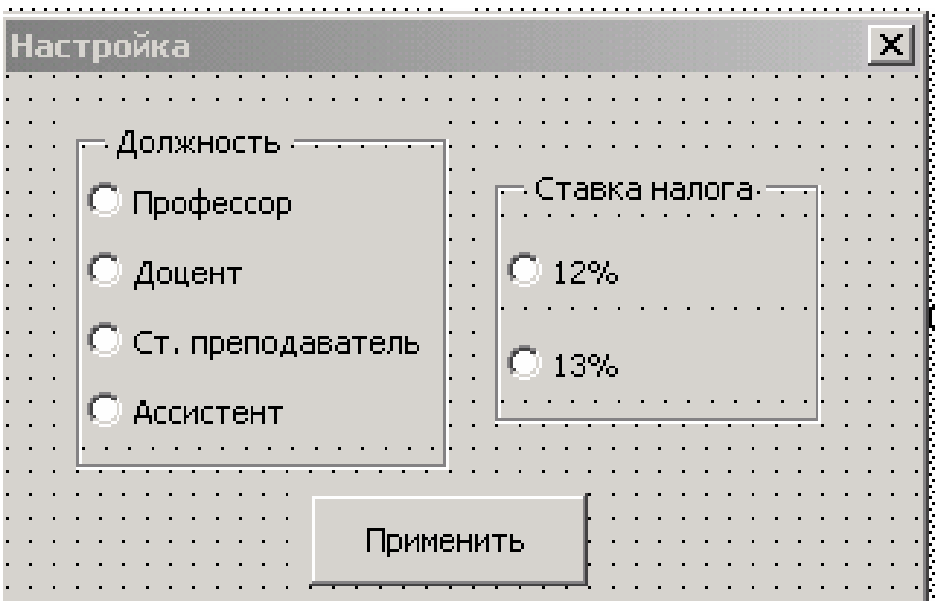


Рис. 16. Пример формы с двумя переключателями.

Пример 35. Программа обработки состояний групп переключателей, запускаемая после нажатия кнопки Применить.

```
'Внешние переменные, которые могут быть включены, например, в состав переменных класса
Dim Должность As Variant, Ставка_налога As Single
Private Sub CommandButton1_Click()
 'Обработка состояния переключателей рамки Должность
   If UserForm2.OptionButton1.Value <> 0 Then
      Должность = UserForm2.OptionButton1.Caption
   Else
   End If
   If UserForm2.OptionButton2.Value <> 0 Then
      Должность = UserForm2.OptionButton2.Caption
   Else
   End If
   If UserForm2.OptionButton3.Value <> 0 Then
      Должность = UserForm2.OptionButton3.Caption
   Else
   End If
   If UserForm2.OptionButton4.Value <> 0 Then
      Должность = UserForm2.OptionButton4.Caption
   Else
   End If
'Обработка состояния переключателей рамки Ставка налога
   If UserForm2.OptionButton5.Value <> 0 Then
      Ставка_налога = 0.12
   Else
   End If
   If UserForm2.OptionButton6.Value <> 0 Then
      Ставка_налога = 0.13
   Else
```

```
    End If
'К этому моменту переменные Должность и Ставка налога
'содержат значения, выбранные переключателями
    UserForm2.Hide
End Sub
```

Задание

Используйте согласованный с преподавателем вариант задания (табл. 1), выполненную на его основе таблицу Excel, написанные программы вычислений в таблице и созданный на ее основе класс, позволяющий производить требуемое количество таблиц и обработку данных в них. Создайте пользовательскую форму, позволяющую ввести с ее помощью все исходные данные таблицы и исключить непосредственное взаимодействие оператора с таблицей Excel.

Порядок выполнения работы

1. Создайте модуль формы. С помощью конструктора внедрите на заготовку формы необходимые элементы управления.

2. Перейдите в окно редактора кодов формы и создайте необходимые процедуры обработки событий внедренных в форму элементов управления.

3. Разработайте последовательность действий, которую надо выполнить при активизации формы и оформите ее в виде процедуры UserForm_Activate.

4. Внедрите на рабочий лист кнопку, позволяющую при нажатии запустить создаваемую форму (пример 31).

5. Разработайте последовательность действий, которую надо выполнить при удалении формы и оформите ее в виде процедуры UserForm_Terminate. Предусмотрите в ней действия по приведению в исходное положение кнопки на рабочем листе.

6. Убедитесь в работоспособности созданной программы.

7. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

Контрольные вопросы

1. Зачем нужны формы?
2. Что нужно сделать для того, чтобы создать форму?

3. Как можно задать действия, которые должны быть выполнены в форме при ее открытии?

4. Как задать свойства и методы формы?

5. Как завершить работу с формой?

6. Как обратиться к объекту, находящемуся в форме?

7. Какие события могут быть предусмотрены в форме? Как они программируются?

8. Как изменить значения свойств формы?

9. Как получить список методов формы?

10. Как перейти от режима программирования формы к ее работе и обратно?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, тексты написанных вами процедур обработки событий элементов управления формы, тексты процедур ее активации и удаления и текст программы обработки событий кнопки. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Лабораторная работа №11

Библиотечные классы VBA, связанные с Excel

Методические указания

Разработчики пакета Microsoft Office не могли обойтись без создания набора собственных классов, позволяющих реализовать те или иные возможности. Некоторые классы этого набора являются общими для всех составляющих пакета, а некоторые отражают специфику конкретной используемой программы. Мы сосредоточимся на библиотечных классах Excel, хотя изложенные далее методы работы с классами могут быть применимы и для других программ пакета, например, Word и Power Point.

В основе библиотеки классов лежит так называемая модель объектов. Она определяет структуру библиотечных классов, каждый из которых может быть непосредственно использован при программировании. Модель объектов Excel показана на рис. 17. Главным классом является сам Excel, рассматриваемый как приложение

(Application). В его составе присутствуют вложенные наборы классов и вложенные одиночные классы. Схожие классы могут объединяться в так называемые коллекции. Имена коллекций записываются на английском языке во множественном числе за счет добавления в конце буквы s. Один и тот же класс может относиться к нескольким коллекциям. Классы, отнесенные к одной коллекции, могут иметь общие свойства и методы, присущие именно этой коллекции. С другой стороны, каждый класс характеризуется собственным набором свойств и методов. Конкретные объекты могут создаваться как экземпляры класса и использоваться самостоятельно или в составе объекта боле высокого уровня иерархии.

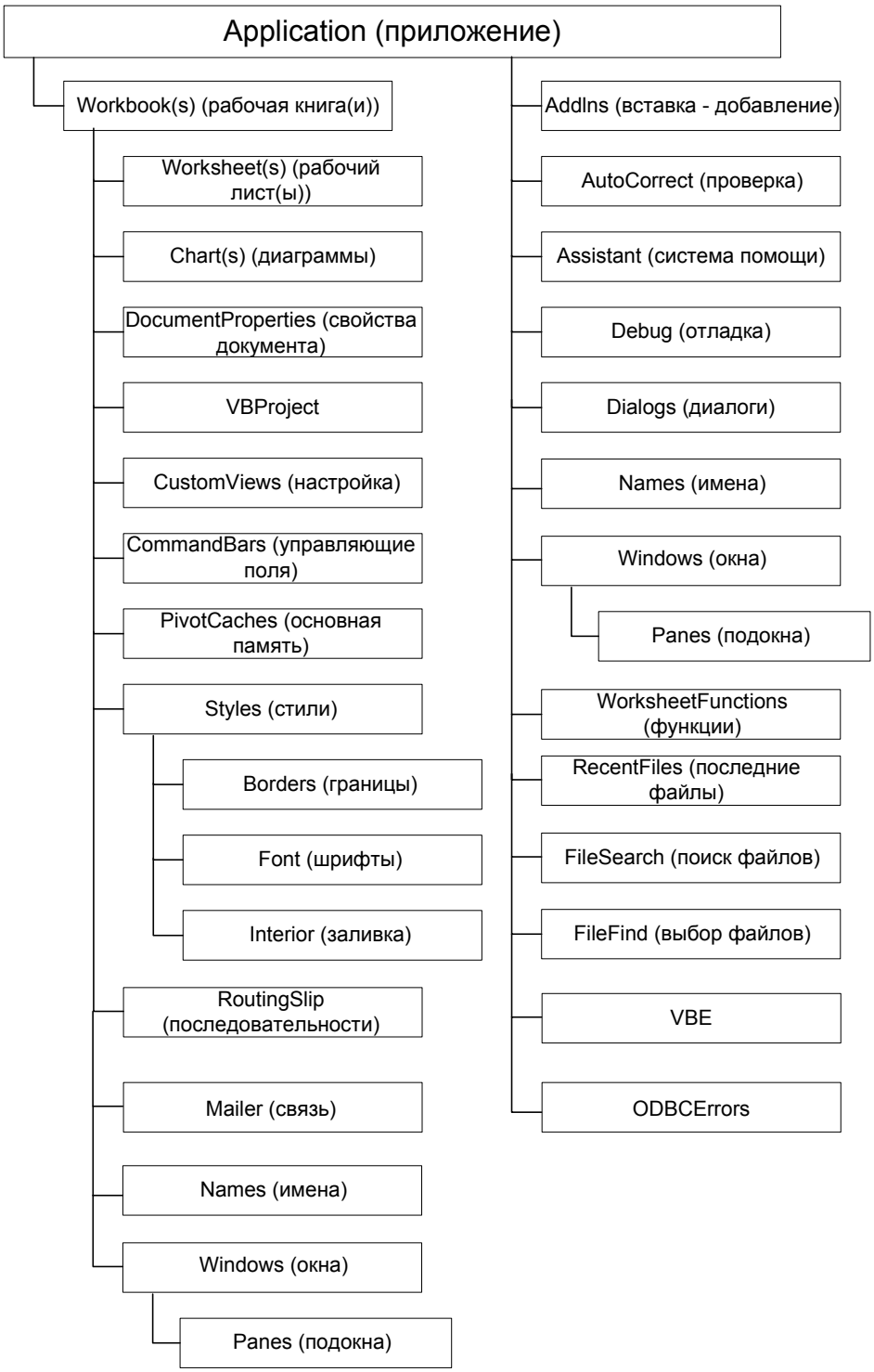


Рис. 17. Модель объектов Excel.

Воспользоваться моделью объектов для программирования действий со стандартными классами можно при наличии соответствующей программной документации. Если доступ к ней оказывается затруднительным, приходится получать необходимую информацию из системы помощи. В некоторых случаях при программировании действий с библиотечными классами Excel можно воспользоваться уже ранее

применявшимся приемом, связанным с анализом текста создаваемого системой макроса.

Рассмотрим задачу записи в файл информации из примера 23. Допустим, что мы хотим записать файл не с конкретным именем, явно указанным в программе, а предлагаем оператору выбрать новое имя файла на этапе выполнения программы. Конечно, мы можем воспользоваться уже ранее изученными приемами и запросить имя файла в диалоге (например, функцией InputBox). Тем не менее, работа программы выглядит гораздо аккуратнее, если в нужный момент мы воспользуемся библиотечным классом Excel, отвечающим за работу с файлами и генерирующим в нужный момент необходимый нам объект.

Наберем в окне редактора кодов в свободной строке текста любого модуля слово Application и введем после него точку. В ответ интегрированная среда разработки VBA откроет список имен свойств, методов и констант, доступных для этого объекта. Выбирая различные позиции списка, мы можем занести интересующее нас имя в строку и, при необходимости, прочитать дополнительную информацию после нажатия клавиши F1.

Пример 36. Модернизируем программу, рассмотренную в примере 23, так, чтобы оператор мог задавать имя файла принятым в рамках пакета Microsoft Office способом. Для этого воспользуемся свойством GetSaveAsFilename объекта Application.

```
fname = Application.GetSaveAsFilename _
("", "Файлы зарплаты (*.hhh), *.hhh, Все файлы (*.*), *.hhh", , "Зарплата")
If fname <> False Then
   Open fname For Output As #1
      Print #1, "Иванов В.Н."
      Print #1, 1234
      Print #1, "Трофимова Л.А."
      Print #1, 1234
      Print #1, "Семенова Е.Г."
      Print #1, 1000
      Print #1, "Степанов А.Г."
      Print #1, 900
   Close #1
Else
End If
```

Рассмотрим задачу построения диаграммы на основе таблицы рис. 1. Включим режим записи макроса и, отвечая на вопросы мастера, построим график. Изучая его текст и выполняя пошаговую отладку, отметим, что активизация диаграммы осуществляется в момент выполнения строки Charts.Add. Далее задается один из возможных типов графика, диапазон данных, место его размещения, а также значения

свойств. В последних строках программы задается его новое положение на листе, а сама диаграмма делается активной.

Пример 37. Текст макроса, записанного во время построения диаграммы.

```
Sub Построение_диаграммы()
' Построение_диаграммы Макрос
' Макрос записан 28.02.2006 (Администратор)
    Charts.Add
    ActiveChart.ChartType = xl3DColumnClustered
    ActiveChart.SetSourceData Source:=Sheets("Лист2").Range("C2:C5"), PlotBy:= _
      xlColumns
    ActiveChart.Location Where:=xlLocationAsObject, Name:="Лист2"
    With ActiveChart
      .HasTitle = True
      .ChartTitle.Characters.Text = "Зарплата"
      .Axes(xlCategory).HasTitle = True
      .Axes(xlCategory).AxisTitle.Characters.Text = "Сотрудник"
      .Axes(xlSeries).HasTitle = False
      .Axes(xlValue).HasTitle = False
    End With
    ActiveSheet.Shapes("Диагр. 25").IncrementLeft 231#
    ActiveSheet.Shapes("Диагр. 25").IncrementTop -82.5
    ActiveSheet.ChartObjects("Диагр. 25").Activate
End Sub
```

Часть кодов рассмотренного примера может быть непосредственно внедрена в программу VBA, причем изменение свойств графика в процессе ее выполнения непосредственно отражается на экране во время ее выполнения.

Задание

Используйте согласованный с преподавателем вариант задания (табл. 1), выполненную на его основе таблицу Excel, написанные программы вычислений в таблице, созданный вами класс и созданную пользовательскую форму. Напишите программу, позволяющую сохранять результаты вычислений в файл произвольного имени и считывать ранее созданные файлы. Постройте график или семейство графиков, связанных с данными вашей таблицы. Включите созданные программы в состав методов класса и напишите итоговую программу, демонстрирующую возможности работы с классом на основе всех созданных вами методов.

Порядок выполнения работы

1. Модифицируйте ранее написанную диалоговую программу, запрашивающую имя файла, так, чтобы использовалось свойство GetSaveAsFilename библиотечного объекта Application.

2. Модифицируйте ранее написанную программу так, чтобы процедура удаления объекта пользовательского класса использовала библиотечный объект.

3. Убедитесь в работоспособности созданной программы.

4. Внедрите программным путем на лист Excel библиотечный объект Charts (диаграмма), отражающий содержание данных исходной таблицы.

5. Продемонстрируйте возможности изменения внешнего вида диаграммы в процессе выполнения программы.

6. Проверьте правильность комментариев с учетом изменений в тексте программы, дополните и, при необходимости, скорректируйте их.

Контрольные вопросы

1. Зачем нужна модель объектов?
2. Как определить назначение классов объектов, входящих в модель?
3. В чем заключаются преимущества применения библиотечных классов?
4. Как использовать в программе VBA библиотечный класс?
5. Как управлять свойствами объектов библиотечного класса?
6. Как создать объект библиотечного класса?
7. Что надо сделать, чтобы изменить состояние объекта библиотечного класса?
8. Как проводить отладку программы с объектами библиотечного класса?
9. Как запустить на выполнение программу с объектами библиотечных классов?
10. Как снять с выполнения программу с объектами библиотечных классов?

Отчет о работе

Подготовьте отчет о выполненной лабораторной работе. Он должен содержать титульный лист, тексты написанных вами процедур работы с файлами и диаграммой. Сформулируйте выводы, которые можно сделать по результатам выполненной работы.

Предметный указатель

Литература

1. Гарнаев А. Ю. Самоучитель УВА. – СПб.: БХВ - Санкт-Петербург, 1999. – 512 с.

Малышев С.А. Самоучитель VBA. Как это делается в Word, Excel, Access. – СПб: Наука и техника, 2001. – 496 стр.

Visual Basic 6.0: Пер. с англ. – СПб.: БХВ-Петербург, 2002. – 992 стр.

Браун С. Visual Basic 5 с самого начала. – СПб: Питер, 1998. – 320 с.

Уокенбах Д. Подробное руководство по созданию формул в Excel 2002. : Пер. с англ. — М. : Издательский дом "Вильяме", 2002. – 624 с.

Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++, 2-е изд.: Пер. с англ. – М.: «Издательство Бином», СПб.: «Невский диалект», 2000. – 560 с.

Пол А. Объектно-ориентированное программирование на C++, 2-е изд.. : Пер. с англ. – СПб.; М.: «Невский Диалект» – «Издательство БИНОМ», 1999 г. - 462 с.

Приложение А. Пример титульного листа отчета о выполнении лабораторной работы[4].

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

Факультет 8 Специальность 351400 Кафедра 82

Отчет

по лабораторной работе на тему

РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОЙ ТАБЛИЦЫ СРЕДСТВАМИ ПРОЦЕССОРА EXCEL, СОЗДАНИЕ И ВЫПОЛНЕНИЕ МАКРОСОВ EXCEL

Дисциплина: Высокоуровневые методы информатики и программирования

Работу выполнил(а)
студент(ка) группы
№

	подпись, дата	инициалы, фамилия

Работу принял

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Санкт-Петербург
2005

[4] Отчет может выполняться как в рукописной, так и в печатной форме. Листы отчета должны иметь нумерацию (на первом титульном листе номер не ставится) и должны быть скреплены. С актуальным на текущий учебный год вариантом титульного листа можно ознакомиться на http://standarts.guap.ru/

Приложение Б. Пример содержания отчета о выполнении лабораторной работы.

Вариант задания № 31

Формулировка задания: расчет заработной платы.

Вид таблицы:

Фамилия, И.О.	*Начислено*	*Налог*	*К выдаче*
Иванов В.Н.	1 234,00р.	148,08р.	1 085,92р.
Трофимова Л.А.	1 234,00р.	148,08р.	1 085,92р.
Семенова Е.Г.	1 000,00р.	120,00р.	880,00р.
Степанов А.Г.	900,00р.	108,00р.	792,00р.
Итого	4 368,00р.	524,16р.	3 843,84р.
Ставка подоходного налога		12,00%	

Комментарий к содержанию таблицы:

Текст *Фамилия, И.О.* размещен в ячейке A1.

Исходные данные в виде констант размещены в полях *Фамилия, И.О., Начислено, Ставка подоходного налога*.

Средствами Excel вычислялись значения полей *Налог*, *К выдаче,* **Итого**

Формула для вычисления в поле *Налог* строки 2 имела вид =B2*C7

Формула для вычисления в поле *К выдаче* строки 2 имела вид =B2-C2

Таблица была заполнена за счет протаскивания ячеек вниз.

Значение **Итого** было рассчитано за счет выполнения функции группового суммирования =СУММ(C2:C5). Функция была скопирована в столбец D протаскиванием ячейки вправо.

Записанный макрос с внесенными комментариями имеет вид:

```
Sub Расчет_заработной_платы()
' Расчет _заработной_платы Макрос
' Макрос записан 01.12.2005 (Администратор)
    Range("C2").Select ' Активизируется ячейка C2
    ActiveCell.FormulaR1C1 = "=RC[-1]*R7C3" 'В C2 программируется формула =B2*$C$7
    Range("D2").Select ' Активизируется ячейка D2
    ActiveCell.FormulaR1C1 = "=RC[-2]-RC[-1]" 'В D2 программируется формула =B2-C2
    Range("C2:D2").Select 'Выделяются две ячейки (диапазон)
    Selection.AutoFill Destination:=Range("C2:D5"), Type:=xlFillDefault ' Запрограммированные
' формулы протаскиваются по столбцу
    Range("C6").Select ' Активизируется ячейка C6 для программирования строки Итого
    ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)" ' Вычисляется сумма по столбцу
    Range("D6").Select ' Активизируется ячейка D6 для программирования строки Итого
    ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)" ' Вычисляется сумма по столбцу
End Sub
```

Вывод. Установлено, что система Excel при включении режима записи макроса формирует текстовый файл на языке VBA. Содержимое этого файла может быть прочитано и исследовано.