

Ministerul Educației, Culturii și Cercetării al Republicii Moldova

Анатол ГРЕМАЛЬСКИ
Лудмила ГРЕМАЛЬСКИ

ИНФОРМАТИКА

Учебник для **8** класса

ȘTIINȚA, 2019

CZU 004 (075.8)

Г 803

Manualul este elaborat conform curriculumului disciplinar în vigoare și aprobat prin Ordinul ministrului educației (nr. 455 din 31 mai 2013). Finanțat din sursele financiare bugetare.

Comisia de evaluare: *Svetlana Golubev*, prof. șc., gr. did. superior, specialist principal la informatică, DGETS, mun. Chișinău; *Sergei Balic*, prof. șc., gr. did. al doilea, Liceul Teoretic „Vasile Alecsandri”, or. Călărași; *Valeriu Brodicico*, prof. șc., gr. did. superior, Liceul Teoretic „Boris Cazacu”, or. Nisporeni

Recenzenți: *Mircea Andronic*, șeful Catedrei de informatică economică; *Ruslan Bantea*, șeful Catedrei de informatică generală; *Vitalie Zavadschi*, director adjunct, Colegiul Republican de Informatică din Chișinău

Traducere din limba română: *Arcadie Malearovici*

Responsabil de ediție: *Valentina Ribalchina*

Redactori: *Natalia Cazacu, Tatiana Bolgar*

Lector: *Svetlana Bronschi*

Corectori: *Mariana Belenciuc, Maria Cornesco*

Redactor tehnic: *Nina Duduciuc*

Machetare computerizată: *Anatol Andrițchi*

Copertă: *Vitalie Pogolșa*

Întreprinderea Editorial-Poligrafică *Știința*,

str. Academiei, nr. 3; MD-2028, Chișinău, Republica Moldova;

tel.: (+373 22) 73-96-16; fax: (+373 22) 73-96-27;

e-mail: prini_stiinta@yahoo.com;

www.editurastiinta.md

Toate drepturile asupra acestei ediții aparțin Întreprinderii Editorial-Poligrafice *Știința*.

Descrierea CIP a Camerei Naționale a Cărții

Гремальски, Анато́л

Информатика: Учеб. для 8 кл. / Анато́л Гремальски, Людмила Гремальски; trad. din lb. rom.: Arcadie Malearovici; Comisia de evaluare: Svetlana Golubev [et al.]; Min. Educației, Culturii și Cercetării al Rep. Moldova. – Ch.: Î.E.P. *Știința*, 2019 (Tipografia „BALACRON” SRL). – 116 p.: fig., tab.

ISBN 978-9975-85-157-2

004 (075.8)

© *Anatol Gremalschi, Ludmila Gremalschi*. 2005, 2013, 2019

© Traducere: *Arcadie Malearovici*. 2005

© Î.E.P. *Știința*. 2005, 2013, 2019

ISBN 978-9975-85-157-2

СОДЕРЖАНИЕ

Глава I. Хранение информации на рабочих листах	5
1.1. Элементы рабочего листа	5
1.2. Типы данных	9
1.3. Ввод и редактирование данных	12
1.4. Форматирование данных	16
1.5. Числовые форматы	21
Глава II. Формулы и вычисления	26
2.1. Операторы и операнды	26
2.2. Адресация ячеек	30
2.3. Расчеты по формулам	35
2.4. Функции	39
Глава III. Диаграммы и графические объекты	46
3.1. Элементы диаграмм	46
3.2. Создание диаграмм	50
3.3. Редактирование диаграмм	55
3.4. Карты и графические объекты	60
3.5. Построение графиков	65
Глава IV. Базы данных	69
4.1. Элементы базы данных	69
4.2. Обработка списков	72
4.3. Сортировка записей	76
4.4. Отбор записей	80
Глава V. Алгоритмы	85
5.1. Алгоритмы и исполнители	85
5.2. Субалгоритмы	91
5.3. Циклические алгоритмы. Цикл со счетчиком	96
5.4. Циклические алгоритмы. Цикл с условием	101
5.5. Алгоритмы с разветвлениями	106
5.6. Алгоритм работы компьютера	109
5.7. Основные сведения об алгоритмах	111
Библиография	115

Дорогие друзья!

Информатика изменяет окружающую нас жизнь, причем иногда совсем неожиданным образом. Чтобы не отставать от этих изменений, чтобы извлекать из них пользу, чтобы быть конкурентоспособным, нужно изучать и уметь применять на практике весь объем методов и приемов информационных технологий.

Известно, что параметры современных компьютеров – быстрота действия, объем внутренней памяти, характеристики периферийных устройств – по существу удваиваются каждый год. Такой же впечатляющий прогресс, произошедший и в области коммуникаций, позволяет объединить в глобальную сеть практически все компьютеры, содержащие в себе безграничный объем информации. Эта информация создается людьми и для людей. Для того чтобы эффективно использовать эту информацию и участвовать в создании новой, каждый человек должен развивать свою информационную культуру и алгоритмическое мышление. Данная культура предполагает глубокие знания основных концепций информатики и умение разрабатывать алгоритмы решения задач, встречающихся в повседневной жизни.

Если в начале прошлого столетия для того, чтобы соответствовать требованиям времени, каждому члену общества достаточно было уметь читать, то в начале нашего тысячелетия в понятие грамотности входит и знание компьютера. Недаром во многих языках мира появился термин «право (лицензия) на владение компьютером». В последнее время без такого «права» доступ к большинству профессий, даже традиционных, становится невозможным.

Настоящий учебник предназначен для получения учащимися знаний, необходимых для автоматизированной обработки данных, при помощи табличных процессоров и формирования алгоритмического мышления.

Табличные процессоры, являясь наиболее распространенными компьютерными программами, используются для хранения данных, выполнения расчетов, представления информации в наглядной форме. Так же как текстовые и графические редакторы, табличные процессоры являются мощным инструментом для представления, хранения и обработки информации, ее организации в виде баз данных. Теоретические и практические знания в области табличных процессоров принесут Вам реальную пользу не только при изучении школьных предметов, таких как математика, физика, химия, география, история и др., но и после окончания гимназии.

Формирование алгоритмического мышления предполагает знание таких понятий как исполнитель, алгоритм, формы представления алгоритмов, свойства алгоритмов. Практический и теоретический материал, включенный в учебник, поможет Вам сделать первые шаги в мире алгоритмов и разработки собственных компьютерных программ.

Желаем успехов!

Авторы

ХРАНЕНИЕ ИНФОРМАЦИИ НА РАБОЧИХ ЛИСТАХ

1.1. ЭЛЕМЕНТЫ РАБОЧЕГО ЛИСТА

Ключевые термины:

- рабочий лист
- ячейка
- адрес/ссылка на ячейку
- значения и формулы
- табличные вычисления

Мы уже знаем, что компьютер обрабатывает самые различные типы информации: тексты, изображения, звук, видео и т.д. В компьютере обрабатываемая информация представляется в виде данных, которые, в свою очередь, хранятся в файлах. Файлы создаются и модифицируются при помощи приложений – программ, которые обеспечивают ввод, обработку и вывод соответствующих данных. Примерами приложений служат:

Notepad (Блокнот) – программа для ввода, исправления и печати небольших текстов.

Paint (Рисунок) – программа, которую начинающие художники могут использовать для создания и обработки простых изображений.

Media Player – программа для воспроизведения аудио- и видеофайлов.

Word – программа для работы со сложными документами, состоящими из текста, таблиц, изображений, аудио- и видеофрагментов.

В процессе развития информатики было установлено, что во многих областях современной деятельности нужны программные приложения, которые облегчили бы обработку данных, представленных в виде таблиц. В качестве примера можно привести учет книг в библиотеке или товаров в магазине, разработку семейного бюджета или бюджета малого предприятия, обработку данных определенного научного эксперимента и т.п. С целью упрощения ввода, обработки и представления часто встречающихся в таких задачах данных были разработаны специальные программы, называемые **табличными процессорами**. В таких приложениях обрабатываемые данные представлены в виде специальных таблиц, называемых **рабочими листами**. Структура рабочего листа представлена на *рисунке 1.1*.

Рабочий лист состоит из следующих элементов:

– **столбцов**, обозначаемых одной или двумя заглавными буквами латинского алфавита: А, В, С, ..., Z, АА, АВ, АС и т.д.;

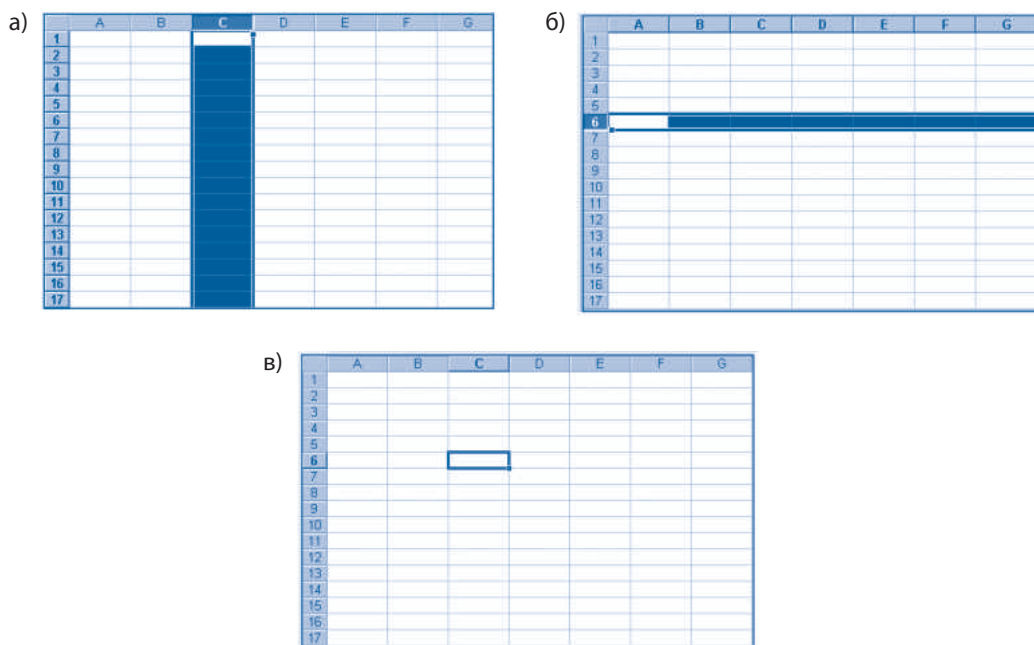


Рис. 1.1. Элементы рабочего листа:
а) столбец; б) строка; в) ячейка

– **строк**, обозначаемых числами 1, 2, 3, 4 и т.д.;

– **ячеек**, обозначаемых указанием столбца и строки, которым они принадлежат.

Например, ячейка, которая расположена на пересечении столбца С и строки 6 (рис. 1.1в) обозначается комбинацией С6. Такая комбинация называется **ссылкой** или **адресом ячейки**.

Будем называть ссылкой или адресом ячейки комбинацию, образованную путем соединения идентификаторов (имен) столбца и строки, на пересечении которых она находится.

Обрабатываемые данные и указания, определяющие способ их обработки, вводятся в ячейки рабочих листов. В общем случае ячейки рабочего листа могут содержать следующие данные:

1. Значения (текст, числа, календарные даты или время), которые представляют собой подлежащие обработке данные. Значения отображаются на экране, печатаются на принтере и могут быть использованы для вычислений.

2. Формулы, указывающие действия, которые должны быть выполнены над значениями других ячеек. Обычно формула состоит из адресов ячеек и математических знаков + (сложения), – (вычитания), * (умножения) и / (деления), которые показывают, какие именно операции должны быть выполнены в ходе обработки информации. Каждая формула вырабатывает значение, которое запоминается в соответствующей ячейке и может быть использовано при вычислениях в других формулах.

Очевидно, что в каждую отдельно взятую ячейку пользователь может вписать только данные одного вида – значение либо формулу. Непосредственно после изменения содержимого хотя бы одной ячейки программа обходит все ячейки рабочего листа и пере-

считывает их значения по записанным в них формулам, запоминая в соответствующих ячейках и отображая на экране результаты перерасчета. Поскольку производительность современных компьютеров очень велика, то перерасчет формул осуществляется быстро даже в тех случаях, когда рабочий лист содержит миллионы ячеек.

Будем называть табличными вычислениями автоматическую обработку данных в ячейках рабочего листа. Ячейки рабочего листа содержат значения, которые представляют собой обрабатываемые данные, и формулы, описывающие способ их обработки.

В операционной системе Windows обработка данных рабочих листов осуществляется при помощи приложения **Microsoft Excel**, окно которого представлено на *рисунке 1.2*.

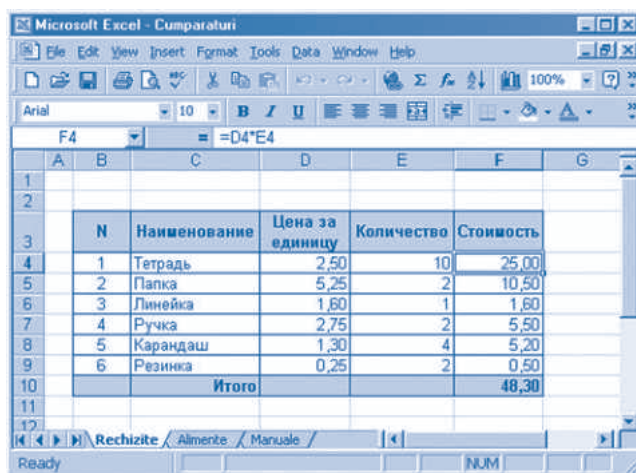


Рис. 1.2. Окно приложения **Microsoft Excel**

Рассматриваемое окно содержит следующие элементы, характерные для приложения **Microsoft Excel**:

1. Панель заголовка. Содержит наименование приложения и имя активного файла (документа), в нашем случае **Cumparaturi**. Отметим, что файлы, созданные приложением **Microsoft Excel**, называются **рабочими книгами**, причем в каждой книге может быть несколько рабочих листов.

2. Строка меню. Содержит меню **File** (Файл), **Edit** (Правка), **View** (Вид), **Insert** (Вставка), **Format** (Формат), **Tools** (Сервис), **Data** (Данные), **Window** (Окно) и **Help** (Справка). В свою очередь каждое из этих меню содержит команды, которые запускаются на выполнение стандартными методами операционной системы **Windows**. Например, команды меню **File** позволяют создать новую рабочую книгу, открыть существующую или сохранить текущую книгу, а команды меню **Edit** – удалить, копировать и вставить данные. Как и в случае других приложений, команды меню **Help** позволяют получать справку из системы помощи, которая содержит полную информацию о назначении и способе использования всех средств, предлагаемых табличным процессором.

3. Панель инструментов Стандартная. Эта панель содержит наиболее часто используемые инструменты, например, открытия и сохранения файлов (для **Microsoft Excel** – открытия и сохранения рабочих книг), печати рабочих листов, копирования и

вставки данных, обработки данных и другие. Вспомним, что стандартные инструменты упрощают доступ к часто используемым командам меню, а назначение соответствующих кнопок можно уточнить, задержав курсор над нужной пиктограммой.

4. Панель инструментов Форматирование. Эта панель похожа на соответствующую панель с таким же названием из приложения **Microsoft Word**. Она содержит набор инструментов, позволяющих устанавливать шрифты, размер символов, стиль отображения, выравнивание данных в ячейке и т.п.




5. Поле Имя. В этом поле отображается адрес (ссылка) выбранной ячейки. Например, для рабочего листа на *рисунке 1.2* таким адресом является F4. Отметим, что на рабочем листе выбранная ячейка автоматически выделяется утолщенной рамкой.

6. Строка формул. В этом поле отображается содержимое выбранной ячейки, причем пользователь имеет возможность модифицировать указанное содержимое. Например, в случае выбора ячейки F4 рабочего листа, изображенного на *рисунке 1.2*, в этом поле отображается формула $=D4*E4$, имеющая следующий смысл «умножьте содержимое ячейки D4 на содержимое ячейки E4 и отобразите полученное значение». Аналогичным образом ячейка F10 содержит формулу $=SUM(F4:F9)$, смысл которой «сложите значения, хранящиеся в ячейках F4, F5, F6, F7, F8 и F9, и отобразите полученное значение».

7. Окно документа, в котором отображается один из листов рабочей книги. Поскольку рабочие листы очень велики, окно документа содержит, в общем случае, линейки прокрутки как по горизонтали, так и по вертикали. Очевидно, что на экране помещается только часть рабочего листа, а выбор отображаемого участка остается за пользователем.

Обычно при выводе рабочего листа на экран или на принтер, отображаются не сами формулы, а значения, рассчитанные по ним. Таким образом, для ячейки F4 на *рисунке 1.2* на экране отображается значение 25,00 или, другими словами, результат вычислений по формуле $=D4*E4$, содержащейся в этой ячейке. Точно так же для ячейки F10 на экран выводится результат вычислений по формуле $=SUM(F4:F9)$, т.е. значение 48,30.

8. Закладки для выбора листов, которые расположены в нижней части экрана. Для выбора нужного листа используется соответствующая закладка. Например, для рабочей книги на *рисунке 1.2* выбрана закладка **Rechizite**, и соответствующий лист выведен на экран. Отметим, что рабочая книга **Cumparaturi** на *рисунке 1.2* содержит три рабочих листа: **Rechizite**, **Alimente** и **Manuale**, а переключение между ними осуществляется путем выбора нужной закладки.

В отличие от текстовых редакторов в табличных процессорах курсор (указатель) мыши имеет в большинстве случаев вид массивного знака , который всё же может меняться в зависимости от используемого инструмента, например на  или на .

Вопросы и упражнения

- 1 Назовите элементы рабочего листа.
- 2 Как обозначаются столбцы, строки и ячейки рабочего листа?
- 3 Объясните термин *адрес (ссылка) ячейки*.
- 4 Покажите на рабочем листе *рисунка 1.2* следующие ячейки: B3, C5, C8, D3, F4, F10, G11 и A12.
- 5 Какие виды данных может содержать ячейка? Какова разница между значениями и формулами?
- 6 Найдите на рабочем листе **Rechizite** (*рис.1.2*) ячейки, которые содержат значения и ячейки, содержащие формулы. Предполагается, что рабочая книга **Cumparaturi** имеется на компьютере, за которым Вы работаете.

- 7 Объясните, каким образом выполняется автоматическая обработка данных, содержащихся в ячейках рабочего листа.
- 8 Измените количество карандашей в ячейке E8 рабочего листа **Rechizite** на *рисунке 1.2*. Заметьте, насколько быстро изменятся и значения, отображаемые в ячейках F8 и F10. Как Вы считаете, чем это объясняется?
- 9 Какова разница между таблицей в текстовом документе и таблицей внутри какого-либо рабочего листа?
- 10 Объясните термин *табличные вычисления*.
- 11 Назовите все элементы окна приложения на *рисунке 1.2*. Как Вы думаете, для чего предназначены кнопки ◀ и ▶ в левом нижнем углу этого окна? В чем сходство и в чем различие окон приложений **Microsoft Word** и **Microsoft Excel**?
- 12 Назовите несколько задач, которые, по Вашему мнению, могут быть решены с помощью табличных процессоров.
- 13 Как Вы думаете, для чего предназначены команды меню **File: New, Open, Close, Save, Save As**?

1.2. ТИПЫ ДАННЫХ

Ключевые термины:

- тип данных
- тип данных *число*
- тип данных *дата*
- тип данных *время*
- тип данных *текстовой*
- тип данных *логический*
- тип данных *ошибка*

Для правильной обработки данных табличный процессор должен располагать информацией об их природе. Например, являются ли обрабатываемые данные текстом, числами, календарными датами, показаниями часов и т.п. Также табличный процессор должен знать, как эти данные будут представлены в компьютере, например, как цепочка байтов, как целые или действительные числа. Чтобы избавить пользователя от необходимости учета всех подробностей, связанных с внутренним представлением данных, а также для обеспечения правильности вычислений, в табличных процессорах используется понятие *тип данных*.

Будем называть типом данных множество определенных значений и множество операций, которые можно выполнять над указанными значениями.

Табличные процессоры используют различные типы данных. Перечислим главные из них:

1. Тип данных *число*. Множество значений типа данных *число* включает все действительные числа, которые могут быть представлены в памяти компьютера, причем множество операций, которые могут быть выполнены над ними, включает сложение

(обозначенное знаком +), вычитание (-), умножение (*) и деление (/). К значениям числового типа можно присоединять знак процента или денежных единиц: доллар, евро, лей, рубль и др.

Примеры:

7	215
-712	-3,14
234,3	428,56835
345,3289	89563,46935

2. Тип данных *дата*. В общем случае значения рассматриваемого типа являются числами, они обрабатываются табличным процессором как календарные даты. Табличные процессоры представляют дни, начиная с 1 января 1900 года и заканчивая 31 декабря 9999, числами 1, 2, 3, 4 и т.д. В момент отображения на экране программа, пользуясь электронным календарем, преобразует порядковый номер в соответствующую дату. Очевидно, что число 1 будет преобразовано в 1 января 1900 года, число 2 – во 2 января 1900 года, число 3 – в 3 января 1900 года и т.д. И наоборот, при вводе данных компьютер, используя тот же календарь, вычисляет порядковый номер дня относительно 1 января 1900 года. Для отображения календарных дат существует множество заранее заданных форматов. В нашей стране наиболее распространенными являются форматы день.месяц.год, день/месяц/год или день-месяц-год.

Примеры:

01.01.04	16.02.13
01/01/04	16/02/13
01-Янв-04	16-Дек-13

3. Тип данных *время*. Значения, принимаемые данными этого типа, представляют собой дробные числа от 0 до 1, которые обрабатываются табличным процессором как значения времени (показания часов). Например, числу 0 будет соответствовать 0.00, числу 0,25 – 6.00, числу 0,5 – 12.00, числу 0,75 – 18.00 и т.д. Как и в случае календарных дат, для отображения времени также существует множество заранее заданных форматов, самыми распространенными из которых являются часы:минуты или часы:минуты:секунды.

Примеры:

06:00	21:04:39
6:30	09:04 PM
18:05	9:04 PM

Табличные процессоры допускают одновременное использование календарных дат и времени, например, 01-Янв-13 18:00. Соответствующие значения представляются в компьютере рациональным числом, целая часть которого определяет дату, а дробная часть – время. Отметим, что в разных странах способы записи даты и времени отличаются. Следовательно, для каждого компьютера формат соответствующих данных должен определяться страной (региональными настройками), для которой было сконфигурировано приложение **Microsoft Excel**.

Поскольку типы данных *число*, *дата* и *время* представлены в памяти компьютера одинаково, то они имеют общее наименование – **числовые типы**.

4. Тип данных *текст*. Множество значений типа данных *текст* включает в себя строки, состоящие из отображаемых символов, причем одной из самых распространенных операций является их сцепление.

Примеры:

Тетради
Резинка
Административные затраты
Стоимость учебников

5. Тип данных логический. Этот тип данных включает значения TRUE (истина) и FALSE (ложь). Указанные значения могут быть получены в результате вычислений по определенным формулам из ячеек рабочего листа.

6. Тип данных ошибка. Значения рассматриваемого типа данных генерируются табличным процессором в тех случаях, когда появляются ошибки. Основными значениями типа данных *ошибка* являются:

- #DIV/0! – деление на ноль;
- #N/A! – значение отсутствует;
- #NAME! – неверное имя;
- #NULL! – пересечение областей, не имеющих общих ячеек;
- #NUM! – неверное число;
- #REF! – неверная ссылка;
- #VALUE! – неверное значение;
- ##### – невозможность отображения данных.

Для примера на *рисунке 1.3* представлен рабочий лист, значения из ячеек которого принадлежат следующим типам данных:

- а) тип данных *текст* – ячейки B2, B4, B5, B6, B8, B9, ..., C4, C5 и C6;
- б) тип данных *число* – ячейки C16, C17 и C18;
- в) тип данных *дата* – ячейки C8 и C11;
- г) тип данных *время* – ячейки C9 и C12.

	A	B	C	D
1				
2		Путевой лист		
3				
4		Тип транспортного средства	Daсia-2150	
5		Регистрационный номер	С ИК 096	
6		Водитель	Ротару Ион	
7				
8		Дата отбытия	12-май-03	
9		Время отбытия	7:00	
10				
11		Дата прибытия	14-май-03	
12		Время прибытия	19:00	
13				
14				
15		Показания счетчика		
16		При отбытии	345211	
17		При прибытии	346189	
18		Пройденное расстояние	978	
19				
20		Подпись		
21				

Рис. 1.3. Типы данных на рабочем листе

Тип данных каждой из ячеек рабочего листа устанавливается табличным процессором в момент ввода или редактирования соответствующих значений или в процессе расчета или пересчета формул. Выбор подходящих типов данных, которые отражали бы как можно точнее специфику решаемой на компьютере проблемы, является задачей пользователя.

Вопросы и упражнения

- ❶ Объясните термин *тип данных*.
- ❷ Назовите множество значений следующих типов данных:
 - а) текст;
 - б) число;
 - в) дата;
 - г) время;
 - д) логический;
 - е) ошибка.
- ❸ Определите тип данных для каждого из приведенных ниже значений:
 - а) Мунтяну Василе
 - б) 1998
 - в) Дата 15.06.03
 - г) Точное время 18:00
 - д) 18:00
 - е) 21.03.03 07:00
 - ж) 18:30:01
 - з) TRUE;
 - и) Всего
 - к) #####
 - л) 3,14
 - м) 314
- ❹ Назовите типы данных значений из ячеек рабочего листа **Rechizite** (рис. 1.2).
- ❺ Как Вы думаете, какие типы данных должны быть использованы для представления на рабочем листе следующей информации:
 - а) фамилия и имя ученика;
 - б) отметки, полученные каждым учеником на протяжении учебного года;
 - в) рост каждого ученика;
 - г) вес каждого ученика;
 - д) год рождения каждого ученика;
 - е) дата рождения каждого ученика;
 - ж) класс, в котором учится каждый ученик;
 - з) начало и окончание каждого урока в школьном расписании.

1.3. ВВОД И РЕДАКТИРОВАНИЕ ДАННЫХ

Ключевые термины:

- ввод значений
- ввод формул
- выбор объектов
- операции редактирования

Табличные процессоры предлагают несколько способов ввода и редактирования данных. Для того чтобы ввести данные в определенную ячейку, она предварительно должна быть выбрана. Указанная операция выполняется стандартными для всех графических интерфейсов приемами:

- щелчком левой кнопкой мыши по нужной ячейке;
- перемещением указателя выбранной ячейки (этот указатель представляет собой утолщенную рамку) при помощи клавиш ←, ↑, ↓, → или клавиши **Tab**;
- заданием адреса (ссылки) ячейки в диалоговом окне команд **Edit, Go To** (Правка, Перейти).

Значения можно вводить непосредственно в ячейку или при помощи строки формул. Для использования строки формул ее сначала необходимо активизировать, выполнив по ней щелчок левой. При вводе значений необходимо соблюдать следующие правила:

- знак плюс (+) перед числами опускается;
- отрицательным числам должен предшествовать знак минус (-);
- длина текста не должна превышать 255 символов;
- чтобы вводимое число считалось текстом, перед ним вводится апостроф (');
- в одну ячейку могут быть одновременно введены дата и время, отделенные друг от друга пробелом;
- ввод значения завершается выбором другой ячейки или нажатием клавиши **Enter**.

Формулы, как и значения, могут быть введены непосредственно в ячейки или при помощи строки формул. В обоих случаях перед формулой обязательно ставится знак равно (=).

Редактирование данных осуществляется при помощи специальных команд, большинство из которых находятся в меню **Edit** (Правка) и **Insert** (Вставка). Часто используемые команды можно также выполнить при помощи соответствующих кнопок панели инструментов **Стандартная**. Назначение команд редактирования представлено в *таблице 1.1*.

Таблица 1.1

Команды редактирования данных

Наименование опций	Назначение
Меню Edit (Правка)	
Undo (Отменить)	Отменяет последнюю команду редактирования, выполненную до активизации изучаемой опции.
Cut (Вырезать)	Вырезает выбранный объект. Объект или его содержимое помещается в буферную память.
Copy (Копировать)	Копирует выбранный объект или его содержимое в буферную память.
Paste (Вставить)	Вставляет объект или его содержимое из буферной памяти в то место, где расположена точка вставки.
Clear (Очистить)	Стирает содержимое или свойства выбранного объекта.
Delete (Удалить)	Удаляет с рабочего листа выбранный объект.
Delete Sheet (Удалить лист)	Удаляет рабочий лист из книги.
Move or Copy Sheet (Переместить или Копировать лист)	Перемещает или копирует рабочий лист в текущую либо в другую книгу, указанную пользователем.

Наименование опций	Назначение
Меню Insert (Вставка)	
Cells (Ячейки)	Вставляет ячейки по соседству с выбранным объектом.
Rows (Строки)	Вставляет строки по соседству с выбранным объектом.
Columns (Столбцы)	Вставляет столбцы по соседству с выбранным объектом.
Worksheet (Рабочие листы)	Вставляет в книгу рабочий лист.

Приемы редактирования данных представлены в *таблице 1.2*. Подчеркнем, что использование этих приемов предполагает выбор обрабатываемого объекта. Таким объектом может быть содержимое некоторой ячейки, сама ячейка, столбец или строка.

Таблица 1.2

Приемы редактирования данных

Операция редактирования	Последовательность действий
Изменение содержимого ячейки	<ul style="list-style-type: none"> – выполняем двойной щелчок на нужной ячейке; – курсор меняет вид на , указывая на переход при- ложения в режим редактирования содержимого ячейки; – изменяем содержимое ячейки, используя стан- дартные приемы редактирования текста; – после завершения редактирования нажимаем клавишу Enter.
Копирование ячейки	<ul style="list-style-type: none"> – выполняем щелчок (левой) по ячейке – источнику; – выполняем команду Copy; – выполняем щелчок по ячейке – приемнику; – выполняем команду Paste.
Перемещение содержимого ячейки	<ul style="list-style-type: none"> – выполняем щелчок (левой) по ячейке – источнику; – выполняем команду Cut; – выполняем щелчок по ячейке – приемнику; – выполняем команду Paste.
Вставка столбца (строки)	<ul style="list-style-type: none"> – выбираем столбец (строку), возле которого будет вставлен новый столбец (новая строка), выполнив для этого щелчок левой по соответствующему за- головку; – выполняем команду Insert, Columns (Insert, Rows).
Удаление ячейки	<ul style="list-style-type: none"> – выбираем нужную ячейку; – выполняем команду Edit, Delete; – указываем в диалоговом окне направление, в ко- тором будут смещаться оставшиеся ячейки.
Удаление столбца (строки)	<ul style="list-style-type: none"> – выбираем нужный столбец (строку); – выполняем команду Edit, Delete.
Очистка содержимого ячейки, столбца, строки	<ul style="list-style-type: none"> – выбираем нужную ячейку, столбец, строку; – выполняем команду Edit, Clear; – выбираем опцию Contents.

Отметим, что операции редактирования могут осуществляться не только над отдельными объектами (столбцом, строкой или ячейкой), но и над группами объектов, например, группами соседних столбцов или строк. Выбор соответствующих групп осуществляется с применением мыши и специальных клавиш клавиатуры, например, перемещением курсора по заголовкам нужных столбцов или строк при удерживаемой левой кнопке мыши.

Вопросы и упражнения

- ❶ Как можно выбрать ячейку на рабочем листе? Столбец? Строку?
- ❷ Объясните назначение команд меню **Edit: Undo, Cut, Copy, Paste, Clear, Delete**.
- ❸ Какая разница между командами **Cut** и **Copy**?
- ❹ Объясните назначение команд **Cells, Rows, Columns** меню **Insert**.
- ❺ Найдите при помощи системы помощи назначение и способ использования команды **Worksheet** из меню **Insert**.
- ❻ Как можно изменить содержание произвольной ячейки?
- ❼ Назовите тип данных каждого из значений ячеек рабочего листа **Salarii** на *рисунке 1.4*.

	A	B	C	D	E	F	G
1							
2							
3							
4			Фамилия, имя	Отработанных дней	Оплата за день	Зарботная плата	
5							
6							
7			Попеску Ион	20	225,8	4516	
8			Маржине Василе	15	132,15	1982,25	
9			Мунтяну Корнел	19	148,3	2817,7	
10			Ротару Ирина	21	229,5	4819,5	
11							
12							
13			Итого			14135,45	

Рис. 1.4. Рабочий лист **Salarii** (Зарплата)

- ❶ Введите в компьютер рабочий лист, представленный на *рисунке 1.4*. Отметьте, что в ячейки F7, F8, F9 и F10 вводятся не значения, показанные на рисунке, а, соответственно, формулы $=D7*E7$, $=D8*E8$, $=D9*E9$ и $=D10*E10$. В ячейку F13 вводится формула $=SUM(E7:E10)$, предназначенная для вычисления суммы значений из ячеек E7, E8, E9 и E10.
- ❷ Внесите в рабочий лист, изображенный на *рисунке 1.4*, следующие изменения:
 - а) замените значение 225,8 в ячейке E7 значением 330,1;
 - б) замените имя Ион в ячейке C7 на имя Сорин;
 - в) замените значение 19 в ячейке D9 значением 23;
 - г) удалите столбец A;
 - д) удалите строки 1 и 2.
 Объясните изменения, которые произошли на рабочем листе, в особенности в строке, содержащей текст Итого.
- ❸ Введите в компьютер рабочий лист, представленный на *рисунке 1.3*. Отметьте, что в ячейку C18 вводится не значение 978, а формула вычисления пройденного пути $=C17-C16$. Измените рабочий лист следующим образом:

- а) вставьте строку над названием Путевой лист;
 - б) вставьте строку между значениями Водитель и Дата отбытия;
 - в) измените дату отбытия на 13-Май-03;
 - г) измените дату прибытия на 15-Май-03;
 - д) замените по очереди значение 346189 в ячейке С17 на значения 346203, 346715 и 346000.
- Объясните изменения, которые происходят на рабочем листе.

1.4. ФОРМАТИРОВАНИЕ ДАННЫХ

Ключевые термины:

- объект
- свойства объекта
- формат
- форматирование
- формат столбца
- формат строки
- формат ячейки

Рабочие листы могут содержать различные **объекты**: значения, ячейки, столбцы, строки и т.п. Каждый из указанных объектов обладает определенными **свойствами**, например ячейка может характеризоваться размерами, формой рамки, цветом фона, на котором отображаются соответствующие значения. Аналогичным образом символы, используемые для представления содержимого ячейки, характеризуются шрифтом, например, *Times New Roman*, *Courier* или *Arial*, размерами и стилем отображения *Regular*, *Bold* или *Italic*. Как и в других приложениях, свойства объектов, используемых в табличных вычислениях, описываются при помощи форматов.

Формат представляет собой набор сведений об объекте, которые описывают его свойства. Процесс установки и изменения указанных свойств называется форматированием.

Формат столбца (строки) содержит следующую информацию:

- ширину столбца (высоту строки);
- способ отображения на экране.

Свойства столбцов могут быть установлены при помощи команды **Column** (Столбец) меню **Format**. При выполнении команды **Format, Column** табличный процессор выводит подменю, содержащее следующие опции:

Width (Ширина) – установка ширины столбцов. После выбора опции **Width** приложение выводит на экран диалоговое окно, в котором пользователь может указать нужную ширину. Эта команда воздействует только на столбцы, в которых находятся выбранные ячейки.



AutoFit Selection (Автоподбор ширины) – ширина столбца будет установлена табличным процессором автоматически таким образом, чтобы содержимое каждой ячейки было полностью отображено на экране.

Hide (Скрыть) – в дальнейшем столбцы, в которых расположены выбранные ячейки, не будут отображаться на экране.

Unhide (Отобразить) – ранее «скрытые» при помощи команды **Hide** столбцы будут вновь отображаться на экране. Для отображения нужных столбцов, до выполнения команды **Unhide**, пользователь должен выбрать хотя бы одну ячейку в столбце, прилегающем к скрытому.

Standard Width (Стандартная ширина) – задает стандартную ширину, которая по умолчанию будет установлена на всех столбцах рабочего листа. Подчеркнем, что ширина столбцов указывается не в миллиметрах или сантиметрах, а в максимальном количестве десятичных цифр, которые могут быть отображены в соответствующем столбце.

Аналогичным образом команда **Format, Row** (Формат, Строка) позволяет установить высоту и способ отображения строк рабочего листа. При выполнении указанной команды приложение выводит подменю, содержащее опции **Height** (Высота), **AutoFit** (Автоподбор высоты), **Hide** (Скрыть) и **Unhide** (Отобразить).

Отметим, что пользователь может изменять ширину столбцов и высоту строк при помощи мыши, поместив для этого курсор в область заголовков столбцов или строк. В тот момент, когда указатель попадает на линию, разделяющую два столбца (две строки), он меняет свою форму с  на , показывая этим, что ширина столбца (высота строки) может быть изменена перетаскиванием соответствующей линии.

Формат ячейки содержит следующую информацию:

- числовой формат, который определяет, каким образом будет обрабатываться содержимое ячейки: как текст, число, дата или время;
- способ выравнивания содержимого ячейки по вертикали и по горизонтали;
- формат символов: примененный шрифт, стиль отображения, размер символов, цвет отображения, специальные эффекты;
- вид рамки;
- описание фона, на котором будет выводиться содержимое ячейки;
- режим защиты каждой ячейки.

Форматирование ячеек осуществляется при помощи команды **Format, Cells** (Формат, Ячейка). При выполнении указанной команды табличный процессор выводит диалоговое окно, состоящее из нескольких страниц, каждая из которых содержит элементы управления процессом форматирования ячеек (*рис.1.5*).

Форматирование символов осуществляется при помощи элементов управления на странице **Font** (*рис.1.5*). Как и в случае текстового редактора, пользователь может выбирать требуемый шрифт (Arial, Courier, Times New Roman и др.), стиль отображения (Regular, **Italic**, **Bold**, **Bold Italic**), размеры символов, цвет символов и специальные эффекты. Быстрый доступ к некоторым элементам со страницы **Font** осуществляется при помощи панели инструментов **Форматирование**. Рассматриваемая панель содержит раскрывающиеся списки **Font**, **Font Size** (Размер шрифта), **Font Color** (Цвет шрифта) и кнопки **Bold**, **Italic**, **Underline** (Подчеркивание). Отметим, что элементы панели форматирования служат одновременно и индикаторами свойств. Они изменяют свое состояние в зависимости от свойств символов и ячеек, в которых находится точка вставки.

Выравнивание содержимого ячеек по вертикали и по горизонтали осуществляется при помощи элементов управления со страницы **Alignment** (Выравнивание) диалогового окна **Format Cells** (*рис. 1.6*).

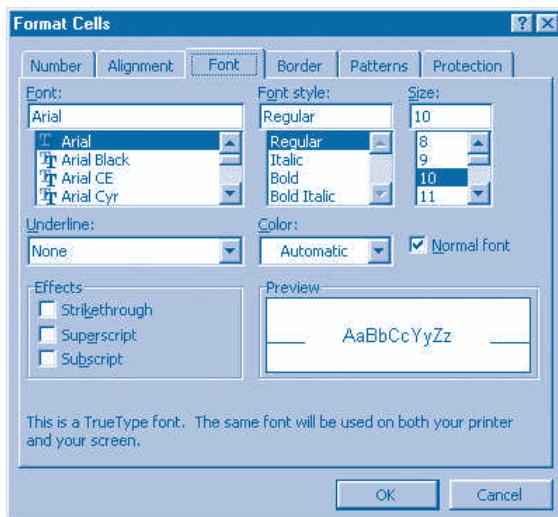


Рис.1.5. Диалоговое окно **Format Cells** (Формат ячеек), страница **Font** (Шрифт)

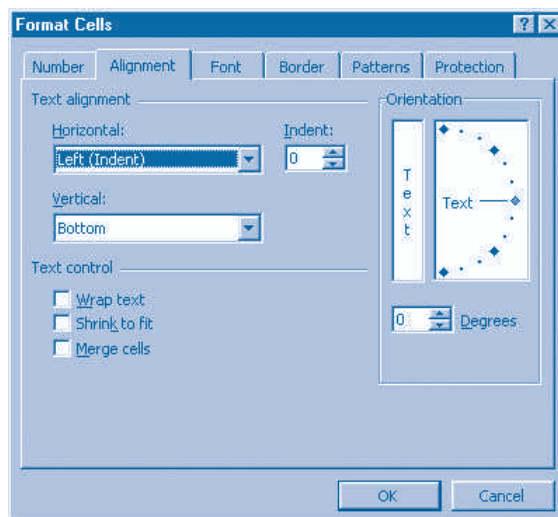


Рис.1.6. Страница **Alignment** (Выравнивание)

Раскрывающиеся списки на этой странице предоставляют пользователю различные способы выравнивания и ориентации содержимого каждой ячейки. Эти способы, в общем случае, аналогичны тем, что имеются в текстовых редакторах. Флажки на этой странице позволяют установить автоматический перенос текста (**Wrap text**), изменение размера символов в соответствии с размерами ячейки (**Shrink to fit**) и объединение соседних ячеек с целью полного отображения содержимого текущей ячейки (**Merge cells**).

Страница **Border** (Граница) диалогового окна **Format Cells** (рис.1.7) обеспечивает прорисовку рамки выбранных ячеек, предоставляя пользователю несколько опций: стиль линии, образующей рамку; цвет рамки; отсутствие или наличие каких-либо сторон рамки и др. Продуманное применение указанных опций позволяет зрительно выделить ячейки, содержащие важные для пользователя данные.

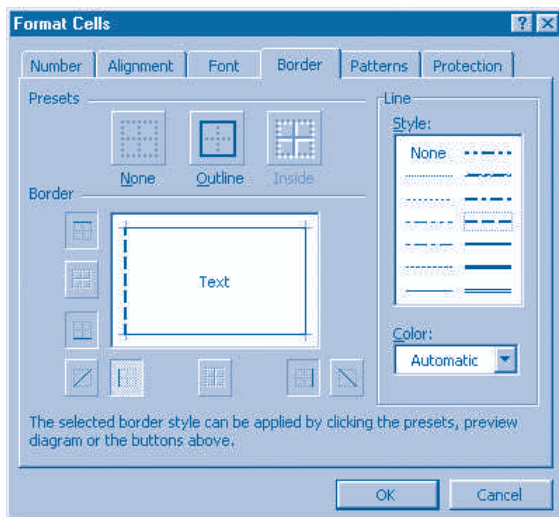


Рис.1.7. Страница **Border** (Граница)

Аналогичным образом страница **Patterns** (Образец) дает пользователю возможность выбора фона, на котором будет отображено содержание форматируемой ячейки (рис.1.8).

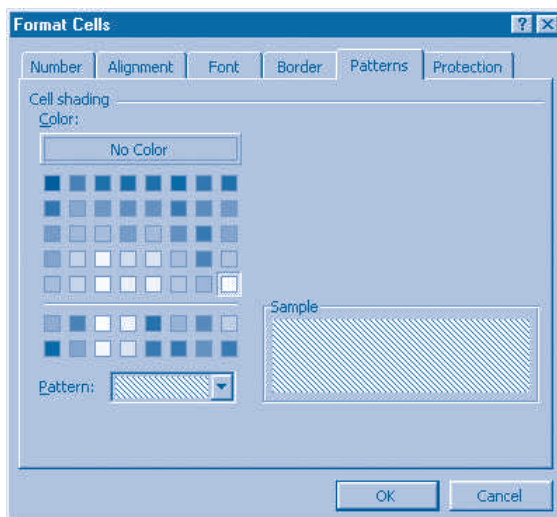


Рис.1.8. Страница **Patterns** (Образец)

Элементы управления страницы **Patterns** дают возможность выбора цвета фона (**Color**) и узора, изображенного на этом фоне (**Pattern**). Отметим, что неправильный выбор рассматриваемых параметров форматирования может затруднить чтение данных, выведенных на экран или на принтер.

Вопросы и упражнения

- 1 Объясните смысл следующих терминов: объект, свойства объекта, формат, форматирование.
- 2 Назовите свойства следующих объектов рабочего листа: строки, столбца, ячейки, содержимого ячейки.
- 3 Назовите свойства следующих объектов текстового документа: символа, абзаца, страницы. Какие из названных вами свойств являются общими и для объектов из состава рабочего листа?
- 4 Пользуясь системой помощи **Help**, определите назначение элементов управления страницы **Protection** (Защита) диалогового окна **Format Cells**.
- 5 Введите в компьютер рабочий лист, представленный на *рисунке 1.9*. Отформатируйте строки, столбцы и ячейки этого листа точно так же, как это показано на рисунке.

	A	B	C	D	E	F	G	H	I
1									
2		Шрифты и стили		Выравнивание текста		Границы		Узор	
3		Arial		По левому краю					
4		Courier		По центру					
5		Times New Roman		По правому краю					
6		Regular		По верхнему краю					
7		Bold		По центру					
8		<i>Italic</i>		По нижнему краю					
9		<i>Bold Italic</i>		По левому верхнему					
10		<u>Underline</u>		По правому нижнему					
11									

Рис.1.9. Форматирование ячеек

- 6 Используя в качестве образца *рисунок 1.10*, отформатируйте рабочий лист **Rechizite**.

	A	B	C	D	E	F	G
1							
2							
3		N	Наименование	Цена за единицу	Количество	Стоимость	
4		1	Тетрадь	2,50	10	25,00	
5		2	Папка	5,25	2	10,50	
6		3	Линейка	1,60	1	1,60	
7		4	Ручка	2,75	2	5,50	
8		5	Карандаш	1,30	4	5,20	
9		6	Резинка	0,25	2	0,50	
10			Итого			48,30 LEI	
11							

Рис.1.10. Рабочий лист **Rechizite** (Принадлежности)

1.5. ЧИСЛОВЫЕ ФОРМАТЫ

Ключевые термины:

- числовой формат
- категория
- хранимое значение
- отображаемое значение

Известно, что каждая ячейка рабочего листа может содержать значение либо формулу. При работе табличного процессора компьютер пересчитывает все формулы и отображает полученные результаты. Другими словами, каждая формула вырабатывает определенное значение, которое отображается в соответствующей ячейке и может быть использовано в качестве исходных данных в других формулах. Для правильной обработки значений и формул, табличный процессор предполагает, что каждое значение принадлежит определенному типу данных: *числовому*, *текстовому*, *дате*, *времени*, *логическому* или *ошибке*. Зная тип данных каждой ячейки, табличный процессор автоматически выбирает внутреннее представление данных, которое, будучи невидимым для пользователя, обеспечивает правильное выполнение соответствующих операций. Например, компьютер будет складывать числа, но не будет суммировать тексты, умножит, но не поделит число на ноль, не станет умножать и делить текстовые величины и т.п.

С развитием табличных процессоров было установлено, что знание только типов данных недостаточно для правильной обработки и отображения рабочего листа. Например, страницы школьного журнала пронумерованы натуральными числами 1, 2, 3 и т.д., порядковые номера учащихся, записанных в журнале, также задаются натуральными числами 1, 2, 3 и т.д., в то время как средние оценки учащихся выражаются действительными положительными числами из интервала 1–10, не более чем с двумя знаками после запятой. Безусловно, невозможно представить себе номер страницы или порядковый номер учащегося равным 8,14, в то время как для средней оценки такое значение является корректным. Во всех случаях рассматриваемого примера соответствующие значения принадлежат типу данных *число*, хотя операции, которые можно выполнять над ними и способ их отображения различаются.

Для того чтобы уточнить способ, которым будут обрабатываться и отображаться данные каждой ячейки, табличный процессор включает в их формат специальную информацию, называемую **числовым форматом**.

Числовой формат является составной частью формата ячейки и представляет собой набор сведений о том, как будут обработаны, выведены на экран и принтер соответствующие значения.

Подчеркнем, что остальные компоненты формата ячейки – выравнивание содержимого, формат символов, вид границы, описание фона и режим защиты – относятся только к виду данных на рабочем листе, в то время как числовой формат вместе с типом данных определяет то, как именно будет обрабатываться содержимое каждой ячейки.

Для того чтобы задать для выбранных ячеек требуемый формат, выполняется команда **Format, Cells** и выбирается страница **Number** (Число). После выполнения этих

действий на экране появится диалоговое окно, в котором пользователь выбирает один из предлагаемых форматов или определяет свой собственный числовой формат (рис.1.11).

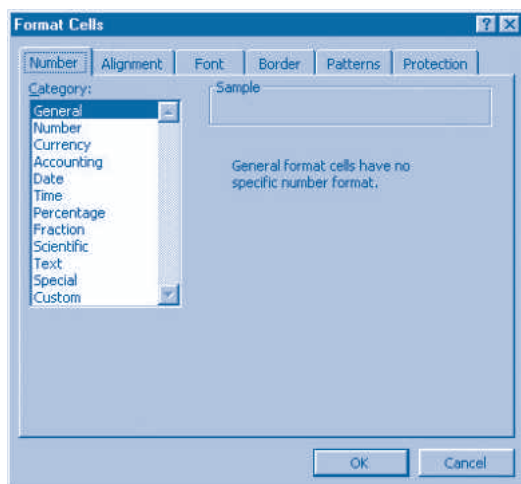


Рис.1.11. Страница **Number** окна **Format Cells**

Для упрощения процессов форматирования в табличных процессорах числовые форматы сгруппированы по категориям.

Категория представляет собой набор числовых форматов, характерных для определенных задач: целые числа, действительные числа, простые дроби, проценты, денежные суммы, календарные даты, показания времени, телефонные номера и т.п.

Табличный процессор **Microsoft Excel** содержит следующие категории числовых форматов:

General (Общий) – эта категория не устанавливает выбранным ячейкам какого-либо специального формата. Их содержимое отображается на экране точно так, как оно было введено пользователем с клавиатуры (рис.1.11). Обычно указанный формат устанавливается автоматически для всех ячеек рабочего листа в момент его создания.

Number (Числовой) – эта категория представляет возможность выбора определенного числового формата для действительных чисел как со знаком, так и без него. Элементы управления соответствующей страницы позволяют устанавливать количество десятичных знаков после запятой и способ отображения отрицательных чисел: со знаком минус или в красном цвете (рис.1.12). Отметим, что количество цифр после запятой относится только к тому, как число отображается на экране, а не к тому, как оно запоминается в самой ячейке. Табличные процессоры позволяют запоминать до 15 десятичных цифр. В качестве примера на рисунке 1.12 приведен рабочий лист, который содержит в ячейке B2 число $\pi = 3,14159265358979$ (см. поле ввода формул). Поскольку для указанной ячейки установлен числовой формат с двумя десятичными цифрами после запятой, отображаемое значение для ячейки B2 имеет вид 3,14.

Currency (Денежный) – для представления числовых величин, представляющих денежные суммы. Элементы управления рассматриваемой категории позволяют выбирать

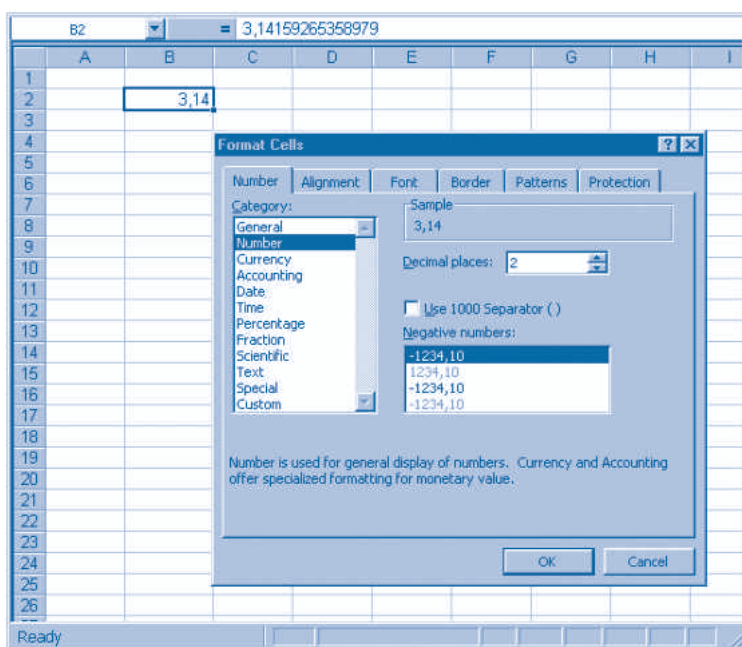


Рис.1.12. Категория **Number** (Числовой)

символы используемой валюты (леи, евро, доллары США, фунты стерлингов, японские йены и т.п.), количество десятичных знаков после запятой, а также способ отображения отрицательных чисел.

Accounting (Финансовый) – для представления числовых величин, часто используемых в бухгалтерском учете. Обычно такие числа имеют смысл денежных сумм, которые выравнены относительно запятой, отделяющей целую часть от дробной.

Date (Дата) – эта категория позволяет выбрать определенный числовой формат для представления календарных дат. Доступные форматы зависят от страны, на которую настроено приложение табличного процессора. В Республике Молдова используются форматы день.месяц.год, день/месяц/год или день-месяц-год.

Time (Время) – для представления числовых значений, являющихся по смыслу показаниями времени. Для отображения показаний времени существует несколько заранее определенных форматов, самыми распространенными из которых являются часы:минуты или часы:минуты:секунды.

Percentage (Процентный) – для представления чисел с процентным значением. Табличный процессор добавляет к соответствующим числам символ %.

Fraction (Дробный) – эта категория позволяет выбрать числовой формат для отображения простых дробей.

Scientific (Экспоненциальный или научный) – для представления числовых величин в форме, используемой в научных расчетах. Этот формат изучается в более углубленных курсах информатики.

Text (Текстовой) – для представления числовых величин в виде текста. Данная категория полезна в тех случаях, когда необходимо обработать по отдельности каждый из символов, рассматриваемого числа.

Special (Дополнительный) – эта категория позволяет выбрать числовой формат для отображения некоторых особых чисел, например почтовых кодов, номеров телефонов, кодов социального страхования и т.п. Отметим, что соответствующие форматы зависят от страны, на которую настроен табличный процессор.

Custom (Пользовательский) – для представления числовых значений в формате, созданном самим пользователем. Методы создания собственных числовых форматов изучаются в углубленных курсах информатики.

Как правило, непосредственно после создания нового рабочего листа всем его ячейкам автоматически назначается числовой формат **General**. В процессе ввода данных табличный процессор устанавливает их тип, пытаясь также автоматически подобрать им подходящий числовой формат. Естественно, иногда автоматически назначаемый компьютером формат не соответствует намерениям пользователя, и он должен явным образом указывать нужный ему числовой формат. Это действие можно выполнить как до, так и после ввода данных, используя команду **Format, Cells, Number**.

Отметим тот факт, что табличные процессоры запоминают и обрабатывают числовые значения с точностью до 15 значащих цифр, а числовые форматы действуют только при выводе соответствующих значений на экран или на принтер. Как следствие необходимо различать **хранимые значения**, т.е. значения, хранящиеся в ячейках рабочих листов, и **отображаемые значения**, т.е. значения, которые выводятся на экран или на принтер в соответствии с числовыми форматами, присвоенными соответствующим ячейкам.

В качестве примера на *рисунке 1.13* представлен рабочий лист, содержащий одно и то же значение $\pi = 3,14159265358979$, записанное в ячейки с различными числовыми форматами. На рисунке видно, что рассматриваемое значение отображается в соответствующих ячейках в зависимости от числового формата, присвоенного каждой из них: как целое число, как действительное число, как простая дробь, как значение, выраженное в процентах, как текст, как календарная дата или как показание времени. Вспомним, что календарные даты во внутренней форме представлены целой частью действительного числа, причем нумерация дней начинается с 1 января 1900 года, а показания времени (часы, минуты и секунды) представлены дробной частью соответствующего числа.

D11		= 3,14159265358979					
	A	B	C	D	E	F	G
		Неформатированное	Форматированное			Числовой формат	
1							
2							
3		3,14159265358979		3,14		Числовой, 2 цифры после запятой	
4		3,14159265358979		3,1416		Числовой, 4 цифры после запятой	
5		3,14159265358979		3		Числовой без цифр после запятой	
6		3,14159265358979	3,14159265358979			Текстовый	
7		3,14159265358979		3,14 LEI		Денежный, 2 цифры после запятой	
8		3,14159265358979		03-январь-00		Дата	
9		3,14159265358979		3:23:54		Время	
10		3,14159265358979		314,16%		Процентный, 2 цифры после запятой	
11		3,14159265358979		3 1/8		Дробный, с восьмью долями	
12							

Рис.1.13. Форматирование чисел

Вопросы и упражнения

- 1 Объясните значение следующих терминов: *числовой формат*, *категория*, *хранимое значение*, *отображаемое значение*.
- 2 Как Вы думаете, когда появляется необходимость назначения числовых форматов ячейкам рабочего листа?
- 3 Перечислите основные категории числовых форматов и объясните их назначение.
- 4 Какой формат автоматически присваивается всем ячейкам только что созданного рабочего листа?
- 5 Объясните, для чего предназначена каждая категория числовых форматов, представленных на *рисунке 1.11*.
- 6 Укажите на рабочем листе *рисунка 1.12* хранимое значение и отображаемое значение.
- 7 Введите в компьютер рабочий лист, изображенный на *рисунке 1.13*. Запишите в ячейки столбца D значение 3,14159265358979 и отформатируйте соответствующие ячейки так, как показано в столбце F. Объясните результаты, отображаемые в ячейках столбца D. Укажите хранимые значения и отображаемые значения.
- 8 Ячейки D3, D4, ..., D11 рабочего листа на *рисунке 1.14* содержат одну и ту же формулу =B3. Эта формула копирует числовое значение из ячейки B3, именуемой входной ячейкой, в ячейку, в которой эта формула находится. Для каждой из ячеек D3, D4, ..., D11 были установлены числовые форматы, показанные на *рисунке*.

	A	B	C	D	E	F	G
1		Входная ячейка		Отформатированная		Числовой формат	
2							
3		12345,11		12345,10		Числовой, 2 цифры после запятой	
4				12345,1000		Числовой, 4 цифры после запятой	
5				12345		Числовой без цифр после запятой	
6				12345,1		Текстовый	
7				12 345,10 LEI		Денежный, 2 цифры после запятой	
8				18-окт-33		Дата	
9				2:24:00		Время	
10				1234510,00%		Процентный, 2 цифры после запятой	
11				12345 1/8		Дробный, с восьмьюми долями	
12							

Рис.1.14. Отображение данных в соответствии с числовым форматом

Введите рабочий лист в компьютер и определите, как будут отображаться значения 1; 100; 1000; 1,879534; 32768,3146; 1504098987, последовательно записываемые в одну и ту же входную ячейку B3.

- 9 Внимательно просмотрите все числовые форматы на странице **Number** диалогового окна **Format Cells** (*рис.1.11*). Пользуясь системой подсказки, попытайтесь найти значение символов dd, mm, yy, # в числовых форматах, принадлежащих категории **Custom**.

2.1. ОПЕРАТОРЫ И ОПЕРАНДЫ

Ключевые термины:

- формула
- операнд
- оператор
- приоритет операций

Ячейки рабочего листа могут содержать как значения, так и формулы. Значения представляют собой подлежащие обработке данные, а формулы указывают операции, которые необходимо выполнить над содержимым ячеек. Сразу после изменения содержимого одной из ячеек табличный процессор обходит все ячейки рабочего листа и перерасчитывает встречающиеся в них формулы, запоминая в соответствующих ячейках полученные значения.

Формулы представляют собой записанные определенным образом математические выражения, показывающие, какие операции необходимо выполнить над значениями из ячеек.

Обычно табличные процессоры отображают в ячейках рабочего листа не сами формулы, а результаты выполненных по ним расчетов. Поскольку содержимое выбранной ячейки отображается в поле формул, пользователь может отобразить на экране соответствующие формулы, последовательно выбирая требуемые ячейки обрабатываемого рабочего листа. Если же требуется отобразить все формулы одновременно, то выполняется команда **Tools, Options, View** (Сервис, Параметры, Вид) и устанавливается флажок **Formulas** (Формулы). В качестве примера на *рисунке 2.1* представлен рабочий лист **Rechizite** до и после выполнения указанных выше действий.

В общем случае формула начинается со знака «=» и состоит из операндов, операторов и названий функций. Порядок выполнения операций может быть уточнен с помощью круглых скобок «(» и «)».

Операнды определяют значения, которые участвуют в расчетах. В качестве операндов в табличных процессорах используются константы, ссылки (адреса ячеек), имена ячеек и функций.

Например, в формуле

$$=A1+2$$

A1 является ссылкой, а 2 – это константа. Аналогичным образом в формуле

а)

	A	B	C	D	E	F	G
1							
2							
3		N	Наименование	Цена за единицу	Количество	Стоимость	
4		1	Тетрадь	2,50	10	25,00	
5		2	Палка	5,25	2	10,50	
6		3	Линейка	1,60	1	1,60	
7		4	Ручка	2,75	2	5,50	
8		5	Карандаш	1,30	4	5,20	
9		6	Резинка	0,25	2	0,50	
10			Итого			48,30	
11							

б)

	A	B	C	D	E	F	G
1							
2							
3		N	Наименование	Цена единицы	Количество	Стоимость	
4		1	Тетрадь	2,5	10	=D4*E4	
5		2	Палка	5,25	2	=D5*E5	
6		3	Линейка	1,6	1	=D6*E6	
7		4	Ручка	2,75	2	=D7*E7	
8		5	Карандаш	1,3	4	=D8*E8	
9		6	Резинка	0,25	2	=D9*E9	
10			Итого			=SUM(F4:F9)	
11							

Рис. 2.1. Формулы рабочего листа **Rechizite** (Принадлежности):
а) отображение вычисленных значений; б) отображение формул

=SUM(A1; A2; A3)*Cursul_valutar

SUM – это функция, а Cursul_valutar имя ячейки. Функции и имена ячеек будут изучаться в следующих параграфах.

Операторы описывают действия, которые необходимо выполнить над значениями из определенных ячеек рабочего листа. Операторы табличного процессора классифицируются следующим образом:

Арифметические операторы:

- + – сложение;
- – вычитание;
- * – умножение;
- / – деление;
- ^ – возведение в степень.

Арифметические операторы применимы только к числовым значениям. Их результатом являются также числовые значения.

Операторы отношения:

- = – равно;
- < – меньше;
- > – больше;
- <= – меньше или равно;
- >= – больше или равно;
- <> – не равно.

Операторы отношения сравнивают два значения одинакового типа, а результатом является логическое значение TRUE (истина) или FALSE (ложь).

Текстовый оператор & (амперсанд) сцепляет (объединяет) последовательно два и более текстовых значения в одно единственное текстовое значение.

В качестве примера на *рисунке 2.2* представлены формулы, содержащие арифметические операторы, операторы отношения и текстовый оператор &. Подчеркнем, что текстовые константы в формулах заключаются в кавычки “ и ”.

a)

	A	B	C	D	E
1					
2	Данные, подлежащие обработке				
3		1	2	Gimnaziu	
4		3	4	Liceu	
5					
6	Формулы, содержащие...				
7	арифметические операторы	операторы отношения	текстовый оператор		
8	=B3+C3	=B3<C3	=D3&D4		
9	=B3-C3	=B3>C3	=D4&D3		
10	=B4*C4	=B4>=C4	=D4&"! Teoretic Călărași"		
11	=B4/C4	=B4<>C4	=D3&"! nr. 2"		
12	=B4^C3	=C4>=B4	=D4&"! "&"nr.5"		
13	=(C4-B4)^2	=C4<=B4	=D4&D3&D4		
14					

б)

	A	B	C	D	E
1					
2	Данные, подлежащие обработке				
3		1	2	Gimnaziu	
4		3	4	Liceu	
5					
6	Формулы, содержащие...				
7	арифметические операторы	операторы отношения	текстовый оператор		
8	3	TRUE	GimnaziuLiceu		
9	-1	FALSE	LiceuGimnaziu		
10	12	FALSE	Liceul Teoretic Călărași		
11	0,75	TRUE	Gimnaziul nr. 2		
12	9	TRUE	Liceul nr. 5		
13	1	FALSE	LiceuGimnaziuLiceu		
14					

Рис. 2.2. Операторы и операнды на рабочем листе:
а) отображение формул; б) отображение вычисленных значений

В тех случаях, когда в формуле встречается несколько операторов, они выполняются в определенном порядке. Правила вычисления формул такие же, как и в математике:

- операции выполняются в соответствии с приоритетом операторов;
- в случае равных приоритетов операции выполняются слева – направо;
- в первую очередь вычисляются выражения, заключенные в скобки.

Приоритеты операций приведены в *таблице 2.1*.

Таблица 2.1




Приоритеты операторов

Оператор	Значение	Приоритет
^	Возведение в степень	1 (максимальный)
* и /	Умножение и деление	2
+ и -	Сложение и вычитание	3
&	Сцепление (конкатенация)	4
=, <, >, <>, <=, >=	Операторы отношения	5 (минимальный)

Например, для двух кажущихся идентичными формул $=2+3*4$ и $=(2+3)*4$ результаты вычислений будут различными. Формула $=2+3*4$ вычисляется в следующем порядке:

$3 \times 4 = 12$, затем $2 + 12 = 14$, а результат, отображаемый в соответствующей ячейке, будет равен 14. Формула $=(2+3)*4$ будет вычисляться в соответствии с порядком, задаваемым скобками: $2 + 3 = 5$, затем $5 \times 4 = 20$, а результат, отображаемый в соответствующей ячейке, будет равен 20.

Для того чтобы упростить ввод формул, табличные процессоры предлагают специальные приемы. Самый простой из них состоит в задании ссылок путем выбора (выделения) соответствующих ячеек. Для ввода формул путем выделения:

- активизируем поле для формул;
- нажимаем кнопку  (**Edit Formula**);
- если хотим ввести ссылку, выделяем мышкой нужную ячейку (выбранная ячейка будет окружена рамкой из пунктирных линий, а ее адрес появится в формуле);
- после ввода ссылки продолжаем формулу, вводя с клавиатуры оператор или скобку;
- для завершения ввода формулы нажимаем кнопку  или клавишу **Enter**;
- если отказываемся от ввода формулы, то нажимаем кнопку  (**Cancel Formula**).

Если в процессе ввода или вычисления формулы появляется ошибка, табличный процессор выводит сообщение, в котором указывается тип ошибки и предлагаются варианты для ее исправления. Очевидно, что пользователь не должен механически принимать предлагаемое компьютером решение, поскольку оно не всегда может соответствовать поставленной задаче.

Вопросы и упражнения

- 1 Для чего предназначены формулы рабочего листа?
- 2 Как можно отобразить на экране формулы рабочего листа?
- 3 Объясните метод ввода формул путем выделения.
- 4 Назовите операторы и операнды, входящие в состав следующих формул:

а) $=A1+2+B4$	е) $=B2$
б) $=(A3+B5)/(A1-B3)+2$	ж) $=B1<>B2$
в) $=B1\&"текущий"$	з) $="месяц"\&"май"$
з) $=A1*(B1+B2)/2$	и) $=A1\&"чисел"$
д) $=A1>B1$	к) $=A2=3$
- 5 Объясните значение следующих терминов: *формула, оператор, операнд, приоритеты операторов*.
- 6 Упорядочьте приведенные ниже операторы по порядку их приоритетов, начиная с максимального:

$<, +, *, \&, /, <>, \wedge, -, <=$.

- 7 Предположим, что ячейка A1 содержит значение 2, ячейка A2 – значение 3, ячейка B1 – значение Вниз, а ячейка B2 – значение Вверх. Вычислите значения приведенных ниже формул:

а) $=A1+A2$	е) $=A2$
б) $=A1+A2/A1-A2$	ж) $=A1<>A2$
в) $=(A1+A2)/(A1-A2)$	з) $=B1\&B2$
з) $=A1*A1+A2/2$	и) $=B2\&B1$
д) $=A1*(A1+A2)/2$	к) $=B1\&"$ или $"\&B2$

- 8 Каково назначение формул рабочего листа на *рисунке 2.1*? Назовите операторы и операнды, входящие в состав этих формул.
- 9 Введите в компьютер рабочий лист, представленный на *рисунке 2.2a*. Впишите в ячейки B3, C3 и D3 соответственно, значения 5, 6 и Школа. Заметьте, как изменятся отображаемые значения в ячейках, содержащих формулы. Объясните результаты, выводимые на экран.
- 10 Определите типы данных значений рабочего листа, изображенного на *рисунке 2.2b*. Как Вы считаете, влияет ли тип данных на результаты вычисления формул?
- 11 Впишите в ячейку B3 рабочего листа, изображенного на *рисунке 2.2*, значение Вверх. Табличный процессор отобразит в ячейках B8 и B9 значение #VALUE!. Как Вы считаете, чем это объясняется?
- 12 Введите в компьютер рабочий лист **Alimente** (*рис. 2.3*). Впишите в ячейки F4, F5, F6, F7, F8 и F9 формулы, предназначенные для вычисления стоимости каждого продукта, а в ячейку F10 – формулу, которая рассчитывает стоимость всех купленных продуктов.

	A	B	C	D	E	F	G
1							
2							
3		N	Наименование	Цена за единицу	Количество	Стоимость	
4		1	Молоко	3,20 LEI	3	9,60 LEI	
5		2	Хлеб	2,60 LEI	2	5,20 LEI	
6		3	Крупа	5,80 LEI	4,5	26,10 LEI	
7		4	Соль	4,70 LEI	2	9,40 LEI	
8		5	Сыр	18,90 LEI	0,5	9,45 LEI	
9		6	Колбаса	42,30 LEI	0,3	12,69 LEI	
10			Итого			72,44 LEI	
11							

Рис. 2.3. Рабочий лист **Alimente** (Продукты)

2.2. АДРЕСАЦИЯ ЯЧЕЕК

Ключевые термины:

- относительный адрес
- абсолютный адрес
- имя ячейки
- диапазон ячеек

Практика показывает, что большинство часто используемых рабочих листов содержит очень много формул, различающихся лишь некоторыми адресами строки или столбца. Например, на рабочем листе **Rechizite** (*рис. 2.1*) формулы =D4*E4, =D5*E5, =D6*E6, ..., =D9*E9 столбца F отличаются друг от друга лишь номером строки, который последовательно принимает значения 4, 5, 6, ..., 9. Аналогичным образом формулы столбца F рабочего листа **Alimente** (*рис. 2.3*), отличаются друг от друга только номерами строк, образуя такую же последовательность: =D4*E4, =D5*E6, =D6*E6, ..., =D9*E9.

Для упрощения ввода формул табличные процессоры используют следующие методы ссылки на ячейки (адресации ячеек):

- относительные адреса;
- абсолютные адреса;
- имена ячеек;
- диапазоны ячеек.

Адреса ячеек, образуемые путем склеивания идентификатора строки и идентификатора столбца, называются относительными адресами.

Когда формула копируется, табличный процессор изменяет относительные адреса, входящие в ее состав, таким образом, что положение операндов относительно ячейки, которая будет содержать копируемую формулу, останется прежним. Например, предположим, что ячейка B1 содержит формулу $=A1+2$ (рис. 2.4). Замечаем, что положение A1 относительно ячейки B1, содержащей саму формулу, может быть определено как «ячейка, находящаяся в той же строке, но в соседнем слева столбце». В ходе копирования формулы из ячейки B1 табличный процессор изменяет относительный адрес A1 таким образом, что положение операнда по отношению к ячейке, куда происходит копирование, останется неизменным. Вследствие этого в процессе копирования формулы $=A1+2$ из ячейки B1 в ячейки B2, B3, ..., B10, относительный адрес A1 будет автоматически изменен на A2, A3, ..., A10. Аналогичным образом в процессе копирования формулы $=A1+2$ из ячейки B1 в ячейки C1, D1, E1 относительный адрес A1 будет изменен на B1, C1, D1.

	A	B	C	D	E	F
1		=A1+2	=B1+2	=C1+2	=D1+2	
2		=A2+2				
3		=A3+2				
4		=A4+2				
5		=A5+2				
6		=A6+2				
7		=A7+2				
8		=A8+2				
9		=A9+2				
10		=A10+2				
11						

Рис. 2.4. Копирование ячеек, содержащих относительные адреса

Так как в некоторых случаях нет необходимости в автоматическом изменении адресов, то табличные процессоры предоставляют пользователю возможность использовать так называемые абсолютные адреса.

Адреса ячеек, в которых идентификаторам строки и столбца предшествует символ \$, называются абсолютными адресами.

Например, адреса A1, B1, A2 являются относительными, тогда как адреса \$A\$1, \$B\$1, \$A\$2 являются абсолютными адресами. Очевидно, что при копировании формул абсолютные адреса не будут изменяться (рис. 2.5).

В общем случае, копирование формул осуществляется путем использования команд **Edit, Copy** и **Edit, Paste** или соответствующих кнопок на стандартной панели инструментов. Если же требуется быстро скопировать определенную формулу сразу в несколько ячеек, то можно воспользоваться приемом «перетаски и отпусти», перетаскивая (при нажатой левой кнопке мыши) квадратик ■ из правого нижнего угла ячейки с формулой через требуемые соседние ячейки. Отметим, что при нажатии черного квадратика ■ из

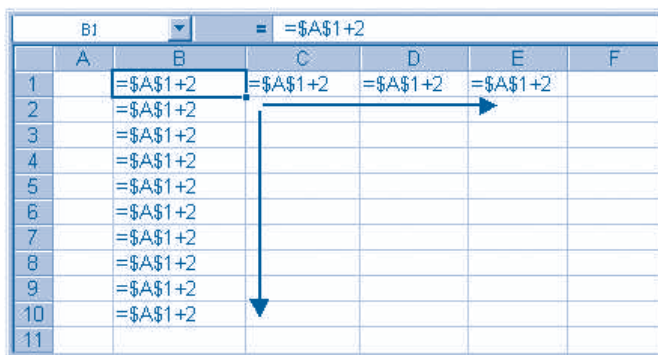




Рис. 2.5. Копирование формул, содержащих абсолютные адреса

правого нижнего угла ячейки, содержащей исходную формулу, курсор меняет свою форму из  в , показывая тем самым, что табличный процессор переходит в режим копирования формул.

Для того, чтобы сделать формулы легко читаемыми, табличный процессор предоставляет возможность обращения к ячейкам по именам. **Имя ячейки** должно:

- начинаться буквой или символом «_» (подчеркивание);
- продолжаться только буквами, цифрами, символом «.» (точка) или «_»;
- не содержать пробелов;
- не совпадать с другими ссылками или именами;
- содержать не более 255 символов.

Примеры имен:

<u>Правильных</u>	<u>Ошибочных</u>
Coeficient	lei/dolar
Cursul_valutar	tempe-ratura
temperatura	A2
Temperatura_Celsius	2S
Anul_2003	Anul 2003

Для того чтобы присвоить ячейке имя, выбираем требуемую ячейку, выполняем команду **Insert, Name, Define** (Вставка, Имя, Присвоить) и вводим в диалоговом окне соответствующее имя. Например, на рабочем листе **Alimente** (рис. 2.6) стоимость приобретаемых продуктов вычисляется в леях, евро и долларах США. Перерасчет стоимостей из леев в евро и доллары осуществляется с использованием коэффициентов из ячеек C13 и C14, названных для удобства Lei_Euro и Lei_Dolar.

Очевидно, что имена ячеек представляют собой абсолютные адреса, оставаясь неизменными при копировании.

Современные табличные процессоры допускают одновременное обращение сразу к нескольким ячейкам. Например, формула =SUM(F4:F9) рабочего листа **Alimente** (рис. 2.6) вычисляет сумму значений из ячеек F4, F5, F6, F7, F8 и F9, причем соответствующие ячейки указываются при помощи более компактной записи F4:F9.

Будем называть диапазоном подмножество ячеек рабочего листа. Диапазон может быть смежным (его ячейки являются соседними) или несмежным.

a)

	A	B	C	D	E	F	G	H	I
1									
2									
3		N	Наименование	Цена за ед., леи	Количество	Стоимость, леи	Стоимость, евро	Стоимость, доллары	
4		1	Молоко	3,20	3	9,60	0,61	0,66	
5		2	Хлеб	2,60	2	5,20	0,33	0,36	
6		3	Крупа	5,80	4,5	26,10	1,67	1,80	
7		4	Соль	4,70	2	9,40	0,60	0,65	
8		5	Сыр	18,90	0,5	9,45	0,60	0,65	
9		6	Колбаса	42,30	0,3	12,69	0,81	0,87	
10			Итого			72,44	4,63	4,99	
11									
12									
13		1 Евро =	15,65	леев					
14		1 Доллар =	14,52	леев					
15									

б)

	A	B	C	D	E	F	G	H	I
1									
2									
3		N	Наименование	Цена за ед., леи	Количество	Стоимость, леи	Стоимость, евро	Стоимость, доллары	
4		1	Молоко	3,2	3	=D4*E4	=F4/Lei Euro	=F4/Lei Dolar	
5		2	Хлеб	2,6	2	=D5*E5	=F5/Lei Euro	=F5/Lei Dolar	
6		3	Крупа	5,8	4,5	=D6*E6	=F6/Lei Euro	=F6/Lei Dolar	
7		4	Соль	4,7	2	=D7*E7	=F7/Lei Euro	=F7/Lei Dolar	
8		5	Сыр	18,9	0,5	=D8*E8	=F8/Lei Euro	=F8/Lei Dolar	
9		6	Колбаса	42,3	0,3	=D9*E9	=F9/Lei Euro	=F9/Lei Dolar	
10			Итого			=SUM(F4:F9)	=SUM(G4:G9)	=SUM(H4:H9)	
11									
12									
13		1 Евро =	15,65	леев					
14		1 Доллар =	14,52	леев					
15									

Рис. 2.6. Использование имен ячеек:
а) отображение значений; б) отображение формул

В случае смежных диапазонов соответствующие ячейки могут быть указаны адреса-ми любых двух ячеек, расположенных в противоположных углах диапазона. Для того чтобы разделить эти адреса используется символ «:» (двоеточие). Например, смежный диапазон на рисунке 2.7а может быть указан одной из следующих последовательностей: B2:E6, E6:B2, B6:E2 или E2:B6. Для несмежных диапазонов перечисляются все поддиапазоны, отделяя их друг от друга символом «;» (точкой с запятой). Например, ячейки несмежного диапазона на рисунке 2.7б можно указать следующим образом: B2:C7; E4:F5; E8:F8.

а)

	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

б)

	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

Рис. 2.7. Диапазоны ячеек:
а) смежные; б) несмежные

Вопросы и упражнения

- ❶ Какова разница между относительными и абсолютными адресами?
- ❷ Определите типы следующих адресов:
 - а) A1
 - б) B3
 - в) \$D\$8
 - г) C5
 - д) \$K\$12
 - е) K12
 - ж) \$B\$21
 - з) D8
- ❸ Как будет выглядеть формула =A1+2 ячейки B1 рабочего листа, изображенного на *рисунке 2.4*, если она будет скопирована в ячейки C2, C3 и C4?
- ❹ Как будет выглядеть формула =A1+2 ячейки B1 рабочего листа, изображенного на *рисунке 2.4*, если она будет скопирована в C2, D3 и E4?
- ❺ Введите в компьютер рабочий лист, изображенный на *рисунке 2.4*. Скопируйте формулу из ячейки B1 в столбцы C, D, E и строки 2, 3 и 4 этого рабочего листа. Заметьте, как будет меняться относительный адрес при копировании этой формулы. Объясните результаты, выводимые на экран.
- ❻ Объясните, в каких случаях используются относительные, а в каких – абсолютные адреса.
- ❼ Введите в компьютер рабочий лист, изображенный на *рисунке 2.5*. Скопируйте формулу из ячейки B1 в столбцы C, D, E и строки 2, 3 и 4 этого рабочего листа. Объясните результаты, выводимые на экран.
- ❽ Как Вы считаете, зачем некоторым ячейкам рабочего листа присваиваются имена?
- ❾ Какие из приведенных ниже имен ячеек являются правильными:
 - а) Greutatea
 - б) Suma_catetelor
 - в) 31A
 - г) C5
 - д) \$K\$12
 - е) b^3
 - ж) Perimetrul
 - з) AA3
- ❿ Введите в компьютер рабочий лист, изображенный на *рисунке 2.6*. Измените значения в ячейках Lei_Euro и Lei_Dolar на 15,95 и 14,78 соответственно. Объясните выводимые на экран результаты.
- ⓫ Замените имена ячеек Lei_Euro и Lei_Dolar в формулах рабочего листа, изображенного на *рисунке 2.6*, на адреса соответствующих ячеек. Сравните объем работы при использовании в формулах имен ячеек и адресов ячеек.
- ⓬ Покажите на *рисунке 2.7* следующие диапазоны:
 - а) A1:E8
 - б) A1; B2; C3
 - в) A1:B3; D5:F6
 - г) B6:G10
 - д) G10:B6
 - е) B2; C2; D2; E2
 - ж) B2; C3; D4; E5
 - з) F7:B2Какие из этих диапазонов являются смежными?
- ⓭ Покажите на рабочем листе, изображенном на *рисунке 2.6*, диапазоны, на которые ссылаются формулы в ячейках F10, G10 и H10.

2.3. РАСЧЕТЫ ПО ФОРМУЛАМ

Ключевые термины:

- преобразование типов данных
- влияющие ячейки
- зависимые ячейки

Под расчетом формулы понимается вычисление ее значения. Результат, который выдает подлежащая расчету формула, зависит от значений операндов и выполняемых над ними операций. Когда табличный процессор рассчитывает формулу, то ожидается, что каждому оператору будут соответствовать определенные типы операндов. Например, в случае арифметических операторов \wedge , $+$, $-$, $*$, $/$ соответствующие операнды должны иметь числовой тип, в случае оператора $\&$ оба операнда должны быть текстового типа, а в случае операторов отношения $<$, $>$, $<>$, $<=$, $>=$ оба операнда должны иметь одинаковый тип: числовой, логический либо текстовый. Типы результатов, которые выдают (возвращают) операторы, показаны в *таблице 2.2*.

Таблица 2.2

Типы результатов, возвращаемых операторами

Оператор	Тип операндов	Тип результата
\wedge , $+$, $-$, $*$, $/$	числовой	числовой
$\&$	текстовый	текстовый
$=$, $<$, $>$, $<>$, $<=$, $>=$	идентичные типы	логический

Вспомним, что во внутреннем представлении дата и время хранятся как числа, что позволяет обрабатывать их при помощи арифметических операторов или операторов отношения. Например, разность двух календарных дат равна числу отделяющих их друг от друга дней, а разность двух показаний времени – соответствующему интервалу времени.

В случае операторов отношения календарные даты и время сравниваются как числа, а текст – как строки символов, упорядоченных в лексикографическом порядке в соответствии с таблицей кодов ASCII.

В качестве примера на *рисунке 2.8* представлены результаты расчета некоторых формул, содержащих арифметические операторы, операторы отношения и текстовые операторы. Чтобы проиллюстрировать тот факт, что арифметические операторы будут работать с календарными датами, в формуле $=D4-C4$ вычитаются, а в формуле $=C4>D4$ сравниваются два значения рассматриваемого типа. Аналогичным образом в формуле $=D5-C5$ вычитаются, а в формуле $=C5+D5$ складываются два показания времени. В формуле $=C6\&D6$ сцепляются, а в формуле $=C6>D6$ сравниваются два значения текстового типа.

В случаях, когда типы данных, необходимые для выполнения какой-либо операции, не совпадают с ожидаемыми (см. *таблицу 2.2*), табличный процессор пытается преобразовать соответствующие значения к желаемому типу.

Под преобразованием типа данных произвольного значения будем понимать изменение соответствующего значения таким образом, что оно будет принадлежать другому типу данных.

а)

	A	B	C	D	E	F	G
1							
2		Тип операндов	Операнды		Формулы		
3		Числовой	1,00	2,00	3,00	-1,00	
4		Дата	01.01.2003	10.01.2003	9	FALSE	
5		Время	10:00	10:30	00:30	20:30	
6		Текстовый	один	два	одиндва	TRUE	
7							
8							

б)

	A	B	C	D	E	F	G
1							
2		Тип операндов	Операнды		Формулы		
3		Числовой	1	2	=C3+D3	=C3-D3	
4		Дата	37622	37631	=D4-C4	=C4>D4	
5		Время	0,416666666666667	0,4375	=D5-C5	=C5+D5	
6		Текстовый	один	два	=C6&D6	=C6>D6	
7							
8							

Рис. 2.8. Операции над числами, календарными датами, текстом и показаниями времени:
а) отображение значений; б) отображение формул

В процессе расчета формул табличные процессоры выполняют, если это необходимо, следующие преобразования:

1. Преобразование в числовые значения. Текст, если его можно интерпретировать (понять) как число, дату и время, преобразуется в числа. Логические значения FALSE и TRUE преобразуются соответственно в числа 0 и 1. Если преобразование невозможно, табличный процессор генерирует значение ошибки #VALUE!.

2. Преобразование в текстовые значения. В тех случаях, когда нужно текстовое значение, табличный процессор преобразует все числовые и логические значения в текстовые.

3. Преобразование в логические значения. Табличный процессор преобразует число 0 в значение FALSE, а остальные числа – в TRUE. Для текста в случаях, когда встречаются значения, отличные от „true” или „false”, будет сгенерировано значение ошибки #VALUE!.

В качестве примера на *рисунке 2.9* представлены результаты расчетов некоторых формул, операнды которых, когда это возможно, преобразованы в числовые значения.

Рабочий лист на *рисунке 2.9* содержит таблицу, в ячейках которой записаны формулы, которые складывают операнды из заголовков строк и столбцов. Заголовки содержат по одному числу, календарной дате, показанию времени, текстовому значению и логическому значению. Табличный процессор выводит в каждой ячейке таблицы результат расчета соответствующей формулы. Например, для формулы =B3+C2 в ячейке C3 выводится результат суммирования чисел 1 и 2, а для формулы =B3+D2 в ячейке D3 выводится результат суммирования числа 1 с календарной датой 10.01.2003. В процессе расчета эта дата преобразуется в число 37631, затем вычисляется сумма 1+37631=37632, которая далее отображается в ячейке D3 как календарная дата 11.01.2003. При расчете формулы =B3+F2 в ячейке F3 табличный процессор пытается преобразовать текст „doi” в число. Так как это невозможно, то в соответствующей ячейке выводится сообщение об ошибке #VALUE!.

Для того чтобы облегчить поиск и устранение ошибок, табличные процессоры определяют для каждой ячейки, содержащей формулу, множество влияющих и множество зависимых ячеек.

a)

	A	B	C	D	E	F	G	H
1								
2		Оператор "+"	2,00	10.01.2003	10:30	doi	TRUE	
3		1,00	3,00	11.01.2003	10:30	#VALUE!	2	
4		01.01.2003	03.01.2003	12.01.2106	01.01.2003 10:30	#VALUE!	02.01.2003	
5		10:00	10:00	10.01.2003	20:30	#VALUE!	01.01.1900 10:00	
6		unu	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	
7		FALSE	2	10.01.2003	10:30	#VALUE!	1	
8								

b)

	A	B	C	D	E	F	G	H
1								
2		Оператор "+"	2	37631	0,4375	doi	TRUE	
3		1	=B3+C2	=B3+D2	=B3+E2	=B3+F2	=B3+G2	
4		37622	=B4+C2	=B4+D2	=B4+E2	=B4+F2	=B4+G2	
5		0,416666666666667	=B5+C2	=B5+D2	=B5+E2	=B5+F2	=B5+G2	
6		unu	=B6+C2	=B6+D2	=B6+E2	=B6+F2	=B6+G2	
7		FALSE	=B7+C2	=B7+D2	=B7+E2	=B7+F2	=B7+G2	
8								

Рис. 2.9. Преобразование операндов в случае оператора "+":
а) отображение значений; б) отображение формул

Будем называть влияющими на заданную ячейку те ячейки, на которые ссылается формула из нее. Будем называть зависимыми от заданной ячейки те ячейки, которые на нее ссылаются.

Например, влияющими на ячейку B5 рабочего листа на рисунке 2.10, являются ячейки A2 и C2, а влияющей на ячейку D8 является ячейка C5. Зависимой от ячейки A2 является ячейка B5, а зависимыми от ячейки C5 являются ячейки B8 и D8.

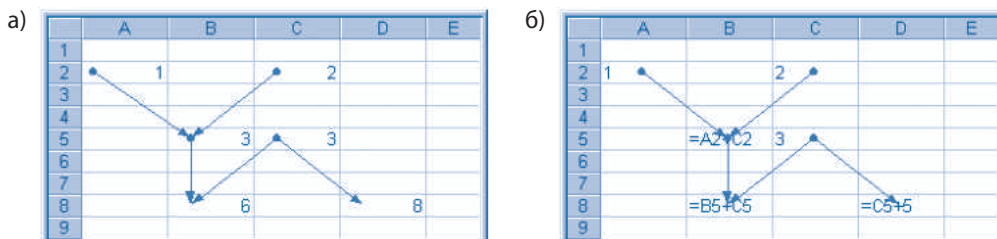


Рис. 2.10. Влияющие и зависимые ячейки:
а) отображение значений; б) отображение формул

Увидеть влияющие и зависимые ячейки для выбранной ячейки можно с помощью команд **Tools, Auditing** (Сервис, Зависимости). Эта команда содержит следующие опции:

Trace Precedents (Влияющие ячейки) – показывает на экране одну или несколько стрелок, которые обозначают перенос данных из ячеек-источников в выбранную ячейку.

Trace Dependents (Зависимые ячейки) – показывает на экране одну или несколько стрелок, которые обозначают перенос данных из выбранной ячейки в ячейки-приемники.

Trace Error (Источник ошибки) – выводит на экран одну или несколько стрелок, которые обозначают перенос данных из ячеек-источников в выбранную ячейку, где содержится значение ошибочного типа (#VALUE!, #DIV/0!, #NUM!, и др.).

Remove All Arrows (Убрать все стрелки) – убирает с экрана все стрелки, выведенные предыдущими командами.

В качестве примера на *рисунке 2.10* представлено изображение некоторого рабочего листа после запуска команд **Trace Precedents** и **Trace Dependents**.

Вопросы и упражнения

- ❶ Когда запускается процесс расчета формул?
- ❷ Назовите типы данных, допустимые для арифметических операторов.
- ❸ Какие типы данных допустимы для операторов отношения?
- ❹ Можно ли сравнивать два значения текстового типа?
- ❺ Введите в компьютер рабочий лист, изображенный на *рисунке 2.8*. Впишите в ячейку D4 календарную дату 12.09.51. Объясните выводимые на экран результаты.
- ❻ Объясните термин *преобразование типа данных*. Когда появляется необходимость в преобразовании типа данных?
- ❼ Как осуществляется преобразование календарных дат и времени в числа?
- ❽ Всегда ли текстовые значения могут быть преобразованы в числовые?
- ❾ Используя в качестве образца рабочий лист на *рисунке 2.9*, создайте аналогичные таблицы для арифметических операторов * и /, операторов отношения = и <>, текстового оператора &. Введите соответствующие таблицы в компьютер и объясните выводимые на экран результаты.
- ❿ В приведенной ниже таблице показаны типы данных операндов и выполняемых над ними операторов. Укажите типы данных, которые получаются в результате выполнения соответствующих операторов.

Тип операнда 1	Оператор	Тип операнда 2	Тип результата
число	+	текст	
календарная дата	+	число	
время	*	число	
текст	&	логический	
текст	>	число	
число	>=	текст	
календарная дата	&	текст	
календарная дата	&	время	

- ❶ Как для заданной ячейки определяются влияющие и зависимые ячейки?
- ❷ Укажите для каждой из ячеек рабочего листа, изображенного на *рисунке 2.9*, влияющие и зависимые ячейки.
- ❸ Введите в компьютер рабочий лист, изображенный на *рисунке 2.10*. Используя команду **Tools, Auditing** (Сервис, Зависимости), выведите на экран стрелки, показывающие перенос данных от влияющих ячеек к зависимым ячейкам.
- ❹ Покажите для каждой из ячеек рабочего листа **Alimente** (*рис. 2.6*) влияющие и зависимые ячейки. Выведите на экран стрелки, показывающие перенос данных от влияющих к зависимым ячейкам.
- ❺ Используя команду **Trace Error** (Источник ошибки), покажите на рабочем листе, изображенном на *рисунке 2.9*, стрелки, показывающие перенос данных от влияющих ячеек к одной из ячеек, содержащих ошибку.

2.4. ФУНКЦИИ

Ключевые термины:

- функция
- аргумент
- возвращаемое значение

Для упрощения процесса ввода данных и выполнения сложных вычислений табличный процессор предоставляет пользователю возможность включать в формулы predefined функции, назначение которых известно для всех рабочих листов, а также функции, определяемые самим пользователем. Вспомним, что в математике функция $f: A \rightarrow B$ описывается тремя элементами:

1) правилом, способом, законом f , посредством которого каждому элементу из A ставится в соответствие один элемент из B ;

2) областью определения или множеством возможных значений независимой переменной x ;

3) областью значений функции (множеством принимаемых функцией значений).

При помощи математической записи $y = f(x)$ подчеркивается тот факт, что x является независимой переменной или аргументом функции, $f(x)$ представляет значение функции в точке x , а y является зависимой переменной.

Функции представляют собой predefined формулы, которые получают на входе определенные значения, называемые аргументами, выполняют их обработку и возвращают вычисленные значения.

Например, на рабочем листе **Alimente** (рис. 2.6b) используется функция SUM, которая суммирует значения ячеек из диапазона, указанного в скобках. Таким образом, в формуле =SUM(F4:F9) складываются значения ячеек из диапазона F4:F9, в формуле =SUM(G4:G9) складываются значения ячеек из диапазона G4:G9, а в формуле =SUM(H4:H9) складываются значения ячеек из диапазона H4:H9. Очевидно, что формула =SUM(F4:F9) более компактна, чем формула =F4+F5+F6+F7+F8+F9, которая выполняет те же действия, но без использования функций.

В общем случае любая функция табличного процессора состоит из двух элементов (частей):

– имени функции, например, SUM, которое указывает, какие именно операции (действия) будут выполнены;

– одного или более аргументов, записанных через точку с запятой и заключенных в круглые скобки «(» и «)».

Например, в формуле =SUM(F4:F9) обозначение SUM – это имя функции, а запись F4:F9 представляет ее аргумент.

В качестве аргументов функций табличных процессоров могут выступать значения (числа, даты, время, текст, логические значения, значения ошибки), ссылки на ячейки (адреса, диапазоны, имена ячеек) или другие функции. Очевидно, что если в записи формулы функция стоит на первом месте, то ей должен предшествовать знак =.


Часто используемые функции табличных процессоров представлены в таблице 2.3.

Часто используемые функции табличных процессоров

Функция	Пример	Описание
AVERAGE (Среднее)	AVERAGE(A1:A10)	Вычисляет среднее арифметическое значение чисел из ячеек диапазона A1:A10.
	AVERAGE(1; 2; 3)	Вычисляет среднее чисел 1; 2; 3.
COUNT (Счет)	COUNT(A1:A5; C1:C5)	Подсчитывает количество всех ячеек, содержащих числовые значения, из несмежного диапазона A1:A5; C1:C5.
COUNTA (Счет A)	COUNTA(B3:D8)	Подсчитывает количество непустых ячеек из диапазона B3:D8.
MAX (Максимум)	MAX(A1:B5)	Возвращает максимальное значение из ячеек диапазона A1:B5.
	MAX(10; 20; A1:B5; D3:E8)	Возвращает максимальное значение из чисел 10; 20 и чисел из ячеек несмежного диапазона A1:B5; D3:E8.
MIN (Минимум)	MIN(C3:D8)	Возвращает минимальное значение из ячеек диапазона C3:D8.
SUM (Сумма)	SUM(A1:B10; C5:D10; F15)	Вычисляет сумму чисел из ячеек несмежного диапазона A1:B10; C5:D10; F15.
	SUM(10; 20,54; B5:D10)	Вычисляет сумму чисел 10; 20,54 и тех, что находятся в ячейках из диапазона B5:D10.
PRODUCT (Произведение)	PRODUCT(B1:B5)	Возвращает произведение чисел из ячеек диапазона B1:B5.
POWER (Степень)	POWER(A1; 2)	Вычисляет квадрат числа, содержащегося в ячейке A1.
	POWER(A1; 3)	Вычисляет куб числа, содержащегося в ячейке A1.
IF (Если)	IF(A1=0; «ноль»; «отлично от нуля»)	Если условие A1=0 является истинным (TRUE), возвращает текстовое значение «ноль». Если условие A1=0 является ложным (FALSE), возвращает текстовое значение «отлично от нуля».
	IF(A1=0; SUM(B1:B5); B1*2)	Если условие A1=0 истинно (TRUE), вычисляет SUM(B1:B5) и возвращает соответствующую сумму. Если условие A1=0 ложно (FALSE), вычисляет B1*2 и возвращает соответствующее произведение.
DATE (Дата)	DATE(2003, 9, 15)	Возвращает число, соответствующее дате 15.09.2003.

Функция	Пример	Описание
TIME (Время)	TIME(15; 20; 30)	Возвращает число, соответствующее времени 15:20:30.
CHAR (Символ)	CHAR(65)	Возвращает символ, код которого равен 65.
CODE (Код)	CODE("A")	Возвращает код символа "A".
LEN (Длина)	LEN("Text")	Возвращает количество символов в слове "Text".

Табличный процессор содержит сотни функций. Эти функции сгруппированы по категориям, например, математические функции, функции для обработки календарных дат и времени, функции для обработки текста, функции для финансовых расчетов и др. Полный список и подробное описание каждой функции можно вывести на экран или отпечатать на принтере при помощи справочной системы **Help**.

Для упрощения ввода функций табличный процессор предоставляет пользователю возможность воспользоваться **Function Wizard** (Мастер функций), который выводит подробную информацию о необходимых функциях и позволяет при помощи мыши выбирать как имена функций, так и соответствующие им аргументы. Запуск мастера осуществляется нажатием кнопки  (**Edit Formula**), расположенной слева от строки формул. Далее пользователь может выбрать имя функции из раскрывающегося списка, который отображается в строке для имен. После выбора нужной функции табличный процессор выводит диалоговое окно (панель формул), в котором пользователь может ввести аргументы функции (рис. 2.11).

Если требуемая функция отсутствует в раскрывающемся списке, необходимо выбрать опцию **More Functions** (Другие функции), которая выводит диалоговое окно **Paste Function** (Вставить функцию). Рассматриваемое окно содержит все функции табличного процессора, сгруппированные по категориям (рис. 2.12).

Такой же результат может быть достигнут запуском команды **Insert, Function** (Вставка, Функция) или нажатием кнопки  (**Paste Function**).

Отметим, что адреса ячеек, которые выступают в качестве аргументов функций, могут быть относительными или абсолютными. Очевидно, что при копировании формул, содержащих функции, относительные адреса будут изменены автоматически, что существенно уменьшает объем работы по их вводу и редактированию. В качестве примера на рисунке 2.13 представлен рабочий лист **Note**, на котором для каждого учащегося вычисляются минимальные, средние и максимальные оценки по каждому предмету. Хотя рассматриваемый рабочий лист содержит 60 формул, только 6 из них были введены непосредственно с клавиатуры, в то время как остальные просто были скопированы. Таким образом, формула =MIN(D3:J3) была введена в ячейку L3, а затем была скопирована во все ячейки диапазона L4:L12. Аналогичным образом формула =AVERAGE(D3:J3) была введена в ячейку M3, а затем скопирована во все ячейки диапазона M4:M12, формула =MAX(D3:J3) была введена в ячейку N3, а затем скопирована во все ячейки диапазона N4:N12.

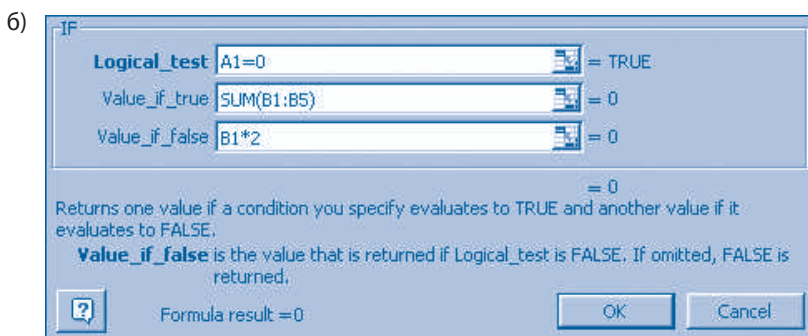
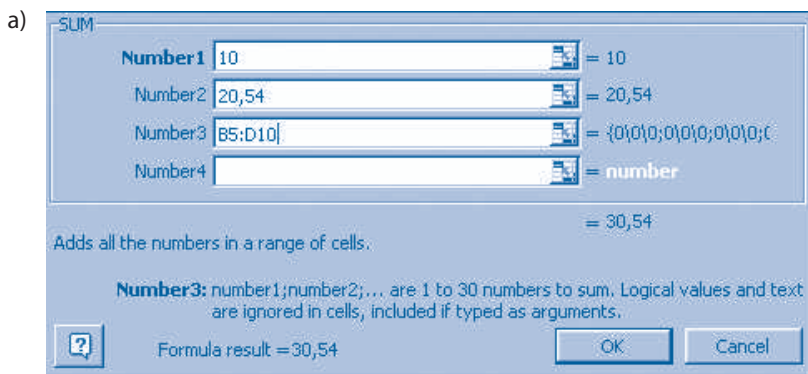


Рис. 2.11. Диалоговые окна для ввода функций (панель формул):
а) для функции SUM; б) для функции IF

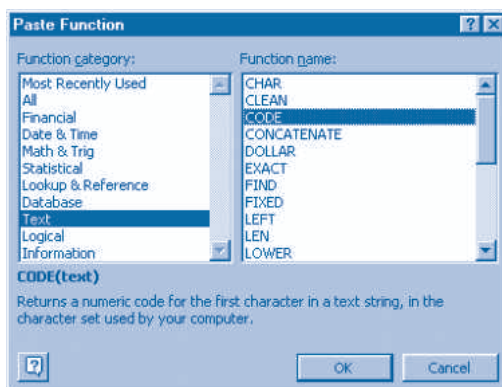


Рис. 2.12. Диалоговое окно **Paste Function**

Для того чтобы лучше понять, как используется функция IF, представим ее структуру в виде

IF(*аргумент 1*; *аргумент 2*; *аргумент 3*).

В процессе расчета программа вычисляет *аргумент 1*, называемый *условием*, и возвращает один из следующих двух аргументов. Если результат вычисления условия TRUE, то функция IF возвращает значение *аргумент 2*; а если результат вычисления условия FALSE, то функция IF вычисляет значение *аргумент 3*.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1															
2		N	Фамилия, имя	Родной язык	Английский язык	Математика	Информатика	Физика	Химия	Биология		Миним. оценка	Средн. оценка	Макс. оценка	
3		1	Букур Елена	5	6	6	7	5	5	8		5	6,00	8	
4		2	Буну Ион	7	7	5	5	5	6	9		5	6,29	9	
5		3	Чиботару Антон	8	8	9	10	10	6	6		6	8,14	10	
6		4	Чобану Кристина	9	9	6	7	5	5	7		5	6,86	9	
7		5	Догару Валентина	9	10	8	8	7	7	9		7	8,29	10	
8		6	Морару Ионица	6	6	6	7	5	5	7		5	6,00	7	
9		7	Мунтяну Александру	5	5	5	5	5	5	6		5	5,14	6	
10		8	Плугару Раиса	6	5	5	5	5	7	7		5	5,71	7	
11		9	Ротару Константин	4	5	5	5	4	4	6		4	4,71	6	
12		10	Вулпе Виктор	6	6	8	8	8	7	7		6	7,14	8	
13															
14			Миним. оценка	4	5	5	5	4	4	6					
15			Средн. оценка	6,50	6,70	6,30	6,70	5,90	5,70	7,20					
16			Макс. оценка	9	10	9	10	10	7	9					
17															

Рис. 2.13. Рабочий лист **Note** (Оценки)

Например, функция $IF(2=2; 3; 4)$ возвращает значение 3, а функция $IF(2<>2; 3; 4)$ возвращает значение 4.

В рабочих листах функция **IF** применяется для отбора данных, соответствующих определенным критериям. В качестве примера рассмотрим применение указанной функции для решения уравнений первого порядка с одним неизвестным. Вспомним, что такие уравнения имеют вид:

$$ax + b = 0.$$

Если a и b являются вещественными числами и $a \neq 0$, тогда уравнение имеет единственный корень $x = -\frac{b}{a}$; если же $a = 0$ и $b \neq 0$, то уравнение не имеет корней; если $a = 0$ и $b = 0$, то уравнение имеет бесконечное множество корней. Рабочий лист, предназначенный для решения уравнений первого порядка, представлен на *рисунке 2.14*.

Корни уравнения первого порядка вычисляются при помощи формулы:

$$=IF(A4=0; IF(B4=0; «Бесконечное множество корней»; «Нет корней»); -B4/A4),$$

записанной в ячейку F4. Первая функция **IF** анализирует содержимое ячейки A4, в ко-

	A	B	C	D	E	F	G
1							
2	a	b		Уравнение		Корень	
3							
4	1	1		$1x+1=0$		-1	
5	2	3		$2x+3=0$		-1,5	
6	2	4		$2x+4=0$		-2	
7	0	1		$0x+1=0$		Нет корней	
8	0	0		$0x+0=0$		Бесконечное множество корней	
9	3	-6		$3x-6=0$		2	
10	-3	6		$-3x+6=0$		2	
11							
12							

Рис. 2.14. Рабочий лист **Ecuatii** (Уравнения)

торой хранится коэффициент a . Если результатом вычисления условия $A4=0$ является логическое значение TRUE, то есть коэффициент $a = 0$, то происходит переход к вычислению внутренней функции IF; если же $a \neq 0$, то возвращается корень уравнения $-\frac{b}{a}$. Внутренняя функция IF анализирует содержимое ячейки B4, в которой хранится коэффициент b . Если в результате вычисления условия $B4=0$ получается логическое значение TRUE, т.е. коэффициент $b = 0$, то возвращается текст «Бесконечное множество корней», а в противном случае – текст «Нет корней».

Для того чтобы сделать рабочий лист более наглядным, в ячейке D4 отображается само уравнение, которое генерируется при помощи формулы

$$=A4\&"x"&IF(B4>=0; "+"; "")\&B4\&"=0"$$

хранящейся в ячейке D4. Эта формула сцепляет значение коэффициента a из ячейки A4, символ x , знак $+$ в случаях, когда коэффициент b положителен, значение коэффициента b из ячейки B4 и текстовой константы $=0$. Функция IF этой формулы возвращает знак $+$, если выполняется условие $B4=0$, и пустой текст – в противном случае.

Для одновременного решения нескольких уравнений формулы из ячеек D4, F4 копируются соответственно в ячейки D5:D10 и F5:F10.

Вопросы и упражнения

- ❶ Объясните, как определяются функции, которые изучались Вами на уроках математики?
- ❷ Какой смысл имеет термин *функция* в табличных процессорах?
- ❸ Из каких частей состоит функция табличного процессора?
- ❹ Определите имена и аргументы следующих функций:

а) MAX(1; 2; 3; A1)	д) AVERAGE(B5:B15; C5:D25)
б) SUM(6; B3; C8; D1:D10)	е) IF(A1=0; SUM(B1:G1); PRODUCT(B1:G1))
в) IF(B1=0; A1+5; A1-5)	ж) DATE(B5)
з) MIN(A1:A10; C1:C10)	з) IF(C1<>0; LEN(A1); LEN(B1))
- Используя *таблицу 2.3*, объясните назначение каждой из этих функций.
- ❺ При помощи системы подсказки **Help** определите, для чего предназначены следующие функции: CONCATENATE, LEFT, RIGHT, DAY, TODAY, YEAR, AND, OR, NOT, ABS, TYPE.
- ❻ Для чего предназначен мастер **Function Wizard**? Как вводятся формулы в рабочий лист с помощью этого мастера?
- ❼ Введите в компьютер рабочий лист **Note** (*рис. 2.13*). Измените этот лист в соответствии с данными журнала класса, в котором учитесь Вы. Объясните назначение каждой из формул рабочего листа.
- ❽ Найдите на экране влияющие и зависимые ячейки для ячеек D14 и L3 из рабочего листа **Note** (*рис. 2.13*). Объясните выводимые на экран результаты.
- ❾ Создайте рабочий лист, на котором будут отображаться коды заглавных букв А, В, С, ..., Z латинского алфавита.
- ❿ Создайте рабочий лист, на котором будут отображаться символы, соответствующие кодам 97, 98, 99, ..., 122.
- ⓫ В столбцах В и С рабочего листа **Piese**, изображенного на *рисунке 2.15*, записаны соответственно длина a и ширина b для 20 фигур прямоугольной формы. Введите в рабочий лист **Piese** формулы для вычисления периметра $2a + 2b$ и площади ab каждой фигуры.

	A	B	C	D	E	F
1						
2		Длина	Ширина	Периметр	Площадь	
3		a	b	$2a+2b$	ab	
4		2	3	10	6	
5		6	8	28	48	
6		
7						
8						
9						
10						
11						
12						
13						
14						

Рис. 2.15. Рабочий лист **Piese** (Фигуры)

- 12 Дополните рабочий лист **Piese** (рис. 2.15) формулами для нахождения фигуры максимальной площади.
- 13 Введите в компьютер рабочий лист **Ecuații** (рис. 2.14). Объясните назначение формул из ячеек рабочего листа. Выведите на экран влияющие и зависимые ячейки для формул из ячеек D4 и F4.
- 14 Для более точного определения длины некоторого стержня было выполнено n повторяющихся измерений, $n \leq 20$. Результаты измерений обозначаются $L_1, L_2, L_3, \dots, L_n$. Длина стержня L вычисляется как среднее арифметическое результатов измерений:

$$L = \frac{L_1 + L_2 + L_3 + \dots + L_n}{n}.$$

Также для каждого измерения определяется отклонение от среднего:

$$\Delta L_1 = L_1 - L; \Delta L_2 = L_2 - L; \Delta L_3 = L_3 - L; \dots; \Delta L_n = L_n - L.$$

Знание отклонений от среднего полезно в тех случаях, когда нужно исключить из расчетов те измерения, которые кажутся странными (подозрительными, ошибочными) по сравнению с другими.

Создайте рабочий лист для нахождения длины стержня L , максимального отклонения ΔL_{\max} :

$$\Delta L_{\max} = \max(|\Delta L_1|, |\Delta L_2|, |\Delta L_3|, \dots, |\Delta L_n|)$$

и минимального отклонения ΔL_{\min} от среднего:

$$\Delta L_{\min} = \min(|\Delta L_1|, |\Delta L_2|, |\Delta L_3|, \dots, |\Delta L_n|).$$

- 15 Создайте рабочий лист для решения уравнений второй степени:

$$ax^2 + bx + c = 0, \quad a \neq 0.$$

Вспомним, что для решения таких уравнений сначала вычисляется дискриминант:

$$D = b^2 - 4ac.$$

Далее анализируется знак дискриминанта D . Если $D > 0$, то уравнение имеет два различных корня:

$$x_1 = \frac{-b + \sqrt{D}}{2a}; \quad x_2 = \frac{-b - \sqrt{D}}{2a}.$$

Если $D = 0$, то уравнение имеет два равных корня:

$$x_1 = x_2 = \frac{-b}{2a}.$$

Если $D < 0$, то уравнение не имеет корней.

3.1. ЭЛЕМЕНТЫ ДИАГРАММ

Ключевые термины:

- диаграмма
- индикатор данных
- ряд (или серия) данных
- ось категорий
- категория данных
- ось значений

Числовые данные рабочего листа воспринимаются легче, если они представлены в графическом виде.

Диаграмма представляет собой изображение, на котором значения числовых данных передаются размерами определенных графических объектов.

Для примера на *рисунке 3.1* представлена диаграмма, построенная на основе данных рабочего листа **Note**.

Как и в случае текстовых редакторов, в табличных процессорах диаграммы рассматриваются как составные объекты, содержащие следующие элементы (*рис. 3.1*):

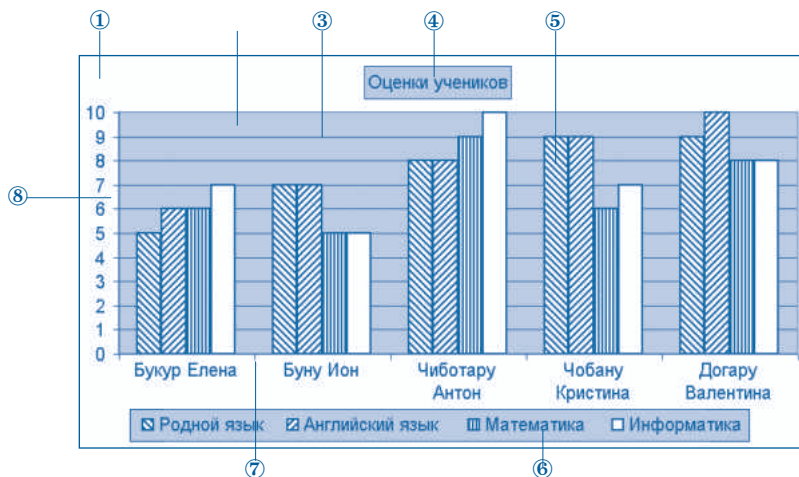


Рис. 3.1. Составные части диаграмм

1. Область диаграммы. Это область рисования самой диаграммы. В случае табличных процессоров диаграммы могут быть вставлены непосредственно в рабочий лист, содержащий отображаемые числовые данные, или в отдельные листы, называемые *листами диаграмм*.

2. Область построения. В этой области располагаются графические объекты, наглядно изображающие числовые данные из определенных ячеек рабочего листа.

3. Линии сетки. Эти линии облегчают чтение и сравнение числовых значений, изображенных в виде графических объектов в области построения.

4. Название диаграммы. Название является необязательным элементом, который в лаконичной форме отражает содержание диаграммы.

5. Индикаторы данных. Индикаторы данных передают при помощи своих геометрических размеров числовые значения из ячеек рабочего листа и, в зависимости от типа диаграммы, могут иметь различные формы: прямоугольники, линии, секторы круга и др. Каждый индикатор представляет значение из определенной ячейки рабочего листа.

В качестве примера на *рисунке 3.2* представлен рабочий лист **Note**, на котором при помощи утолщенной рамки показаны данные, используемые для создания диаграммы **Notele elevilor**.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1																
2		N	Фамилия, имя	Родной язык	Английский язык	Математика	Информатика	Физика	Химия	Биология		Миним. оценка	Средн. оценка	Макс. оценка		
3		1	Букур Елена	5	6	6	7	5	5	8		5	6,00	8		
4		2	Буну Ион	7	7	5	5	5	6	9		5	6,29	9		
5		3	Чиботару Антон	8	8	9	10	10	6	6		6	8,14	10		
6		4	Чобану Кристина	9	9	6	7	5	5	7		5	6,86	9		
7		5	Догару Валентина	9	10	8	8	7	7	9		7	8,29	10		
8		6	Морару Ионица	6	6	6	7	5	5	7		5	6,00	7		
9		7	Мунтяну Александру	5	5	5	5	5	5	6		5	5,14	6		
10		8	Плугару Раиса	6	5	5	5	5	7	7		5	5,71	7		
11		9	Ротару Константин	4	5	5	5	4	4	6		4	4,71	6		
12		10	Вулпе Виктор	6	6	8	8	8	7	7		6	7,14	8		
13																
14			Миним. оценка	4	5	5	5	4	4	6						
15			Средн. оценка	6,50	6,70	6,30	6,70	5,90	5,70	7,20						
16			Макс. оценка	9	10	9	10	10	7	9						
17																
18																

Рис. 3.2. Числовые значения, используемые при построении диаграммы **Notele elevilor** (Оценки учащихся)

Сопоставляя диаграмму на *рисунке 3.1* и рабочий лист на *рисунке 3.2*, замечаем, что индикаторы данных представляют оценки по четырем предметам, причем индикаторы для каждого из предметов нарисованы одинаковым образом.

Группа данных, расположенных в столбце или строке рабочего листа, называется рядом (или серией) данных.

На *рисунке 3.2* показаны четыре ряда данных:

- школьные оценки из диапазона D3:D7;
- школьные оценки из диапазона E3:E7;

- школьные оценки из диапазона F3:F7;
- школьные оценки из диапазона G3:G7.

Табличный процессор присваивает каждому ряду данных имя, которое он берет из ячейки, содержащей заголовки для соответствующих значений. Таким образом, ряду данных из столбца D присваивается наименование Родной язык, ряду данных из столбца E – наименование Английский язык, ряду данных из столбца F – наименование Математика, а ряду данных из столбца G – наименование Информатика.

Индикаторы, представляющие значения одного и того же ряда данных, имеют одинаковую цветовую окраску, одинаковую форму и образуют графический ряд данных.

Например, на диаграмме, представленной на *рисунке 3.1*, имеется четыре вида индикаторов данных, отличающихся раскраской: синий цвет – для индикаторов ряда данных Родной язык, красный – для индикаторов ряда данных Английский язык, желтый – для индикаторов ряда данных Математика и лазурный – для индикаторов ряда данных Информатика. Отметим, что при распечатке на черно-белом принтере соответствующие цвета заменяются на оттенки серого.

6. Легенда. Представляет собой панель, содержащую для каждого ряда данных образец цвета, формы или символа вместе с именем ряда. Использование легенды облегчает нахождение на диаграмме рядов данных.

7. Ось категорий. Для построения диаграммы табличный процессор группирует данные по категориям.

Категория представляет собой набор связанных между собой данных. Каждая категория имеет имя, которое берется из заголовка соответствующих значений.

Диаграмма на *рисунке 3.1* содержит пять **категорий** данных, названия которых взяты из рабочего листа **Note** (*рис. 3.2*):

- Букур Елена – диапазон D3:G3;
- Буну Ион – диапазон D4:G4;
- Чиботару Антон – диапазон D5:G5;
- Чобану Кристина – диапазон D6:G6;
- Догару Валенитина – диапазон D7:G7.

Обычно категории располагаются вдоль горизонтальной оси X, а ряды – вдоль вертикальной оси Y, хотя пользователь имеет возможность указать и другой порядок.

8. Ось значений. Вдоль этой оси расположены ряды данных.

На осях диаграмм могут содержаться **метки для значений**, которые представляют собой короткие отрезки прямых, пересекающих оси и помечающих категорию, масштаб или ряд данных на диаграмме. Эти метки могут быть обозначены определенным образом. Например, ось значений на *рисунке 3.1* содержит метки, обозначенные 0, 1, 2, ..., 10, а ось категорий – метки, обозначенные наименованиями категорий Букур Елена, Буну Ион, ..., Догару Валентина. Для большей ясности оси диаграммы могут также иметь и заголовки.

Чтобы легче находить нужные элементы диаграммы, при их выборе с помощью курсора, табличные процессоры отображают наименования соответствующих элементов. В случае индикатора данных отображается наименование ряда, наименование категории и соответствующее значение из рабочего листа (*рис. 3.3*).

В отличие от текстовых процессоров, табличные процессоры предоставляют пользователю гораздо более мощные инструменты создания и редактирования диаграмм. Напом-

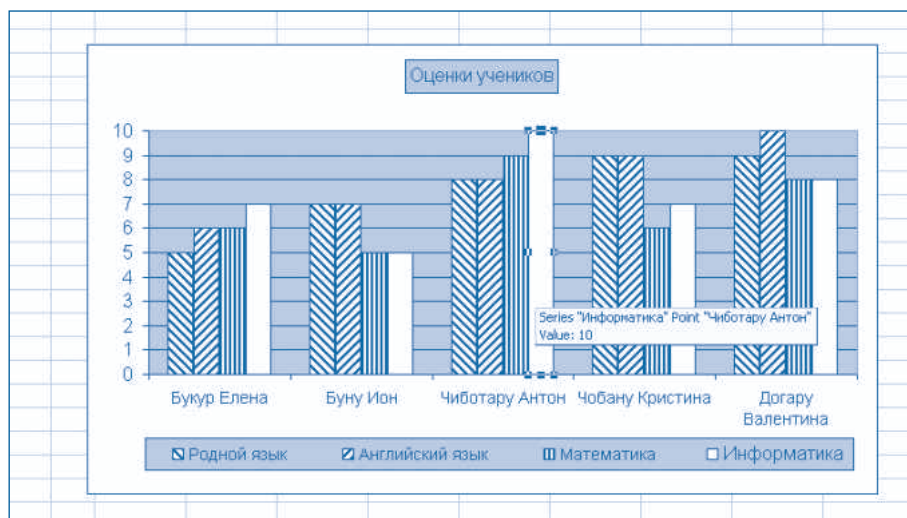


Рис. 3.3. Отображение информации об индикаторах данных

ним, что в случае текстовых редакторов диаграммы создаются при помощи специальных приложений, требующих ввода данных, необходимых для построения диаграмм, в особые таблицы, называемые **листами данных**. Обычно эти листы не считаются принадлежащими тексту и невидимы для читателя. Если всё же нужно создать диаграмму на основе таблицы, содержащейся в документе, то эту информацию необходимо скопировать из текста на лист данных. Вследствие этого, между таблицами в тексте и соответствующими диаграммами из редактируемого документа не существует непосредственной связи, и изменение таблиц из текста никак не отражается на соответствующих диаграммах.

В случае же табличных процессоров любое изменение данных рабочего листа вызывает автоматическое обновление диаграмм. Например, если пользователь изменит оценку ученика Буну Ион с 7 на 9, то соответствующий индикатор данных автоматически изменит свою длину, отражая текущее значение данных из рабочего листа.

Вопросы и упражнения

- ❶ Покажите на диаграмме (рис. 3.3) следующие элементы:
 - а) область диаграммы;
 - б) область построения;
 - в) заголовок диаграммы;
 - г) легенду;
 - д) ось категорий;
 - е) метки на оси категорий;
 - ж) обозначения меток на оси категорий;
 - з) ось значений;
 - и) метки на оси значений;
 - к) обозначения меток на оси значений;
 - л) линии сетки;
 - м) графический ряд данных Родной язык;
 - н) графический ряд данных Информатика;

- о) категорию Чиботару Антон;
 п) индикатор данных «ряд Английский язык, категория Буну Ион»;
 р) индикатор данных «ряд Информатика, категория Догару Валентина».
- 2 Сколько индикаторов данных содержит диаграмма на *рисунке 3.1*? Сколько категорий и сколько рядов данных изображено на этой диаграмме?
 - 3 Покажите на рабочем листе, представленном на *рисунке 3.2*, значения, соответствующие элементам диаграммы **Notele elevilor** на *рисунке 3.1*:
 - а) ряд Математика;
 - б) категория Чиботару Антон;
 - в) ряд Информатика;
 - г) категория Буну Ион;
 - д) индикатор данных «ряд Математика, категория Буну Ион»;
 - е) индикатор данных «ряд Информатика, категория Букур Елена».
 - 4 Объясните смысл терминов *ряд данных* и *категория данных*.
 - 5 Какова разница между диаграммами, созданными при помощи текстового редактора, и диаграммами, созданными при помощи табличного процессора?
 - 6 Вспомните, как создается диаграмма при помощи текстового редактора и вставьте в текстовый документ диаграмму, представленную на *рисунке 3.1*. Увеличьте на один балл все оценки в ряде данных Математика. Отразились ли каким-то образом на диаграмме, созданной в текстовом редакторе, произведенные Вами изменения?
 - 7 Увеличьте на один балл все оценки в ряде данных Математика (*рис. 3.2*). Заметьте, как изменяются индикаторы данных на диаграмме **Notele elevilor** на *рисунке 3.1*. Как Вы думаете, чем это объясняется?
 - 8 Измените на рабочем листе на *рисунке 3.2* фамилию Чиботару Антон на Маржине Василе. Как Вы думаете, изменится ли диаграмма **Notele elevilor** на *рисунке 3.1*? Проверьте Ваш ответ на компьютере.
 - 9 Последовательно выберите с помощью курсора все элементы диаграммы **Notele elevilor**, представленной на *рисунке 3.1*. Прокомментируйте появляющиеся на экране сообщения.
 - 10 Как и для текстовых редакторов, каждый элемент диаграммы, созданной в табличном процессоре, характеризуется определенными свойствами, например, размером, положением в пределах диаграммы, шрифтом, стилем отображения и др. Попробуйте перечислить эти свойства для каждого из элементов диаграммы, представленной на *рисунке 3.1*.

3.2. СОЗДАНИЕ ДИАГРАММ

Ключевые термины:

- тип диаграммы
- двумерная диаграмма
- трехмерная диаграмма

В табличном процессоре **Microsoft Excel** диаграммы создаются путем запуска команды **Insert, Chart** (Вставка, Диаграмма) или нажатием кнопки **Chart Wizard** (Мастер диаграмм) на стандартной панели инструментов.

Далее в учебных целях мы будем использовать рабочий лист **Note** на *рисунке 3.1* и создадим диаграмму **Notele elevilor**, конечный вид которой представлен на *рисунке 3.2*.

Сразу после запуска, мастер диаграмм **Chart Wizard** выводит на экран первую диалоговую страницу, с помощью которой можно выбрать тип создаваемой диаграммы (рис. 3.4).

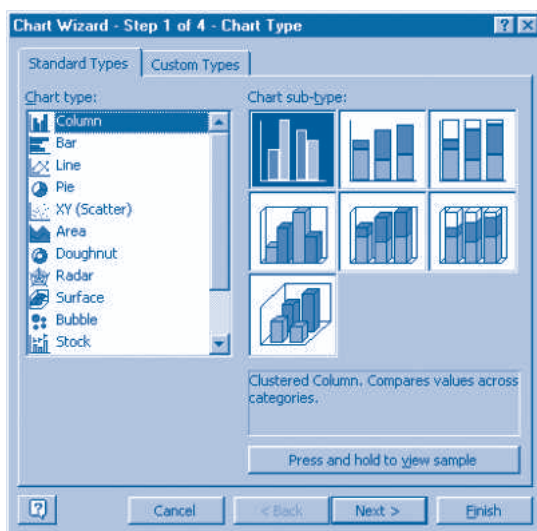


Рис. 3.4. Диалоговая страница **Chart Wizard – Step 1 of 4 – Chart Type**
(Шаг 1 из 4 – Тип диаграммы)

Список **Chart Type** этого окна предлагает пользователю несколько типов диаграмм. Напомним, что тип диаграммы определяется в соответствии с графическим объектом, использованным для представления числовых значений: столбики (вертикальные или горизонтальные), линии, сектора окружности и т.д. Наиболее часто используемыми типами диаграмм являются:

- Column** – гистограммы;
- Bar** – линейчатые диаграммы;
- Line** – графики;
- Pie** – круговые диаграммы;
- XY (Scatter)** – точечные диаграммы.

Очевидно, что каждый тип диаграммы отображает данные определенным образом. Внешний вид диаграммы каждого типа можно увидеть в правой части диалогового окна **Step 1 of 4**.

Выбор типа диаграммы осуществляется в соответствии с приводимыми ниже рекомендациями:

1. Гистограммы используются для представления различных рядов данных, меняющихся в пространстве или во времени. В качестве примера вспомним оценки учащихся по определенным предметам, число выпускников по годам окончания, ежемесячные доходы родителей, объемы коммерческих продаж и др.

2. Линейчатые диаграммы применяются для сравнения рядов неизменных во времени данных. Например, такая диаграмма может быть использована для представления производительности различных компьютеров.

3. Графики наиболее удобны для представления тенденций или зависимостей между определенными величинами в течение определенного периода времени. В качестве

примера можно привести температуру пациента в больнице, курс лея по отношению к доллару и др.

4. Круговые диаграммы используются для того, чтобы выделить соотношения между частями и целым. Например, в случае кулинарного рецепта каждый сектор окружности может представлять количество определенного продукта в блюде.

5. Точечные диаграммы применяются для представления зависимости между двумя рядами чисел. Этот тип диаграмм наиболее удобен для построения графиков функций, изучаемых на уроках математики, например, для линейной функции $y = ax + b$.

Панель **Chart sub-type** (Вид или подтип) позволяет выбрать для определенных типов диаграмм **двумерное** (плоское) либо **трехмерное** (объемное) представление. Очевидно, трехмерное представление дает более сильный визуальный эффект.

После выбора типа и подтипа нужной диаграммы выполняется переход к следующему шагу. После нажатия кнопки **Next** (Далее) на экран выводится диалоговое окно, изображенное на *рисунке 3.5*.

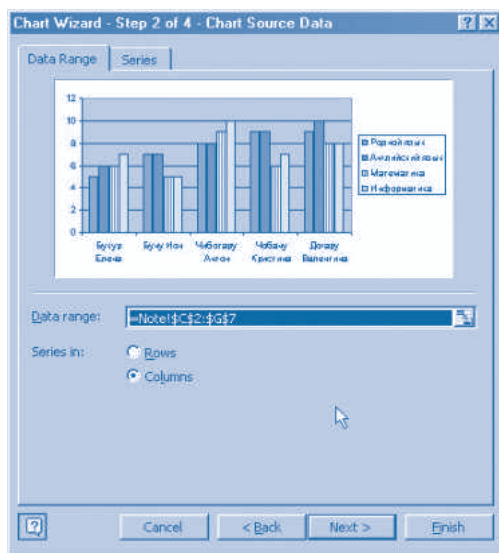


Рис. 3.5. Диалоговое окно **Chart Wizard – Step 2 of 4 – Chart Source Data** (Шаг 2 из 4 – Источник данных)

В текстовое поле **Data Range** (Диапазон) этого окна пользователь должен ввести диапазон ячеек рабочего листа, содержащий данные, которые будут представлены на диаграмме. Диапазон можно задать путем записи в соответствующее поле адреса ячеек или путем протягивания указателя мыши через нужные ячейки. Очевидно, что в случае диаграммы **Notele elevilor** на *рисунке 3.1* нужно протянуть указатель через ячейки C2:G7.

Отметим, что радиокнопки на странице **Data Range** позволяют выбирать ряды данных по строкам или по столбцам, а элементы управления на странице **Series** позволяют редактировать наименования рядов и изменять соответствующие диапазоны значений.

После указания необходимой информации нажимается кнопка **Next**, и на экране появляется диалоговое окно, представленное на *рисунке 3.6*.

Это окно состоит из нескольких страниц, которые позволяют редактировать элементы диаграммы непосредственно в процессе ее создания. Для начала укажем на странице

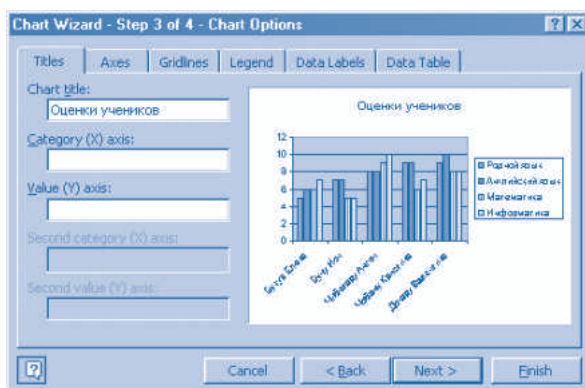


Рис. 3.6. Диалоговое окно **Chart Wizard – Step 3 of 4 – Chart Options**
(Шаг 3 из 4 – Опции диаграммы)

Titles (Заголовки) название диаграммы, а именно, **Notele elevilor (Оценки учащихся)**. Отметим, что все диалоговые окна мастера диаграмм содержат кнопку **Back** (Назад), при нажатии которой происходит возврат к предыдущему окну, что позволяет изменять ранее выбранные параметры.

Последнее диалоговое окно мастера диаграмм (рис. 3.7) позволяет указать место, куда будет вставлена создаваемая диаграмма – на лист диаграмм (**As new sheet**) или на рабочий лист, содержащий отображаемые данные (**As object in**). В случае рабочего листа **Notele elevilor** диаграмма вставляется на рабочий лист **Note**.

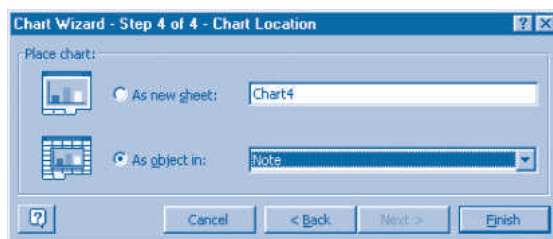


Рис. 3.7. Диалоговое окно **Chart Wizard – Step 4 of 4 – Chart Location**
(Шаг 4 из 4 – Размещение диаграммы)

Отметим, что диалоговые окна мастера диаграмм содержат несколько страниц, которые предлагают пользователю широкую гамму параметров, касающихся типа и подтипа диаграммы, выбора диапазона ячеек, содержащих значения, которые необходимо представить в графическом виде, размещения элементов диаграммы, способа отображения графических элементов и т.п. Очевидно, нет необходимости запоминать все указанные опции, поскольку информация о них содержится в системе помощи **Help** табличного процессора. От пользователя требуется лишь знание главных этапов построения диаграмм и основных возможностей, предоставляемых табличным процессором. Используя эти возможности, пользователь должен создавать простые, хорошо продуманные диаграммы, которые помогут читателю понять значение данных на рабочем листе. Считается, что диаграмма хорошо спроектирована в том случае, когда у читателя не возникает необходимости обращаться к числовым данным на рабочем листе для того, чтобы понять данные, представленные в графической форме.

Вопросы и упражнения

- 1 Назовите основные этапы создания диаграмм. Как Вы считаете, может ли быть изменен порядок выполнения этих этапов?
- 2 Пользуясь системой помощи **Help**, определите назначение каждого элемента управления страниц диалоговых окон мастера диаграмм.
- 3 Создайте при помощи мастера диаграмм диаграмму **Notele elevilor** (рис. 3.1). Объясните назначение каждого диалогового окна, выводимого на экран этим приложением.
- 4 На рисунке 3.8 представлена диаграмма **Число выпускников Гимназии «Константин Стере»**. Эта диаграмма была построена нами в процессе изучения текстовых редакторов. Создайте теперь данную диаграмму при помощи табличного процессора. Сравните возможности по созданию диаграмм, предоставляемые текстовым редактором и табличным процессором.



Рис. 3.8. Диаграмма **Число выпускников Гимназии «Константин Стере»**

- 5 Постройте диаграмму типа график **Курс американского доллара по отношению к молдавскому лею** (рис. 3.9).



Рис. 3.9. Диаграмма **Курс американского доллара по отношению к молдавскому лею**

- 6 В приведенной ниже таблице представлены ежемесячные доходы некоторой семьи. Постройте на основе этих данных гистограмму, линейчатую диаграмму и круговую диаграмму. Как Вы считаете, какой из типов диаграмм наиболее удобен для представления данных о доходах семьи?

	Январь	Февраль	Март	Апрель	Май	Июнь
Мама	2 700	500	900	200	1 000	1 400
Отец	1 000	1 500	1 100	1 800	300	1 200

- 7 Чем отличаются двумерные и трехмерные диаграммы? Как Вы считаете, когда предпочтительнее использовать трехмерные диаграммы?
- 8 Автомобильный парк некоторого предприятия состоит из 60 легковых автомобилей, 20 грузовиков и 15 автобусов. Создайте на основе этих данных две круговые диаграммы: одну – двумерную, а другую – трехмерную. Как Вы думаете, какой вид (подтип) диаграммы наиболее удобен для представления информации об автомобильном парке?
- 9 Создайте двумерную и трехмерную диаграммы для графического представления данных о ежемесячном доходе семьи (см. упражнение 6). Выберите диаграмму, которая, по Вашему мнению, наиболее наглядна для представления указанных данных. Обоснуйте Ваш выбор.
- 10 Дайте краткую характеристику каждого типа диаграмм. Каждая характеристика должна содержать:
 - описание графических элементов, используемых для представления числовых значений;
 - назначение и размещение (положение) оси значений и оси категорий;
 - примеры данных, для которых наиболее подходит применение соответствующего типа диаграммы.

3.3. РЕДАКТИРОВАНИЕ ДИАГРАММ

Ключевые термины:

- инкапсуляция объектов
- иерархия объектов
- объект диаграммы
- методы редактирования

Табличный процессор трактует каждую диаграмму как сложный объект, состоящий из других объектов, а именно: заголовка диаграммы, области построения, графических рядов данных, оси значений, оси категорий, легенды, линий сетки и др. Являясь составными частями диаграммы, некоторые из этих объектов, в свою очередь, содержат другие, относительно более простые объекты. Например, графический ряд данных состоит из индикаторов и меток данных, а ось значений, кроме самой оси, включает в себя заголовки, метки и обозначения. Другими словами, объекты, образующие диаграмму, могут входить один в состав другого.

Будем называть инкапсуляцией метод создания сложных объектов путем включения в их состав других относительно более простых объектов. Включаемые объекты могут содержать, в свою очередь, другие, еще более простые, объекты.

Подчеркнем, что **инкапсуляция объектов** является одним из основных методов, часто применяемых в информатике. Например, в табличном процессоре рабочая книга

представляет собой составной объект, который включает в себя более простые объекты, а именно рабочие листы. В свою очередь, любой рабочий лист представляет собой сложный объект, включающий в себя другие, относительно более простые объекты: строки, столбцы, ячейки, диаграммы и т.п. Аналогичным образом любой объект, созданный при помощи современного текстового редактора, также представляет собой составной объект, содержащий более простые объекты: тексты, таблицы, диаграммы, формулы, изображения, аудио- и видеофайлы.

Будем называть иерархией объектов порядок, в котором они содержатся (включаются) один в другом.

Иерархия объектов диаграмм, созданных при помощи табличных процессоров представлена на рисунке 3.10.

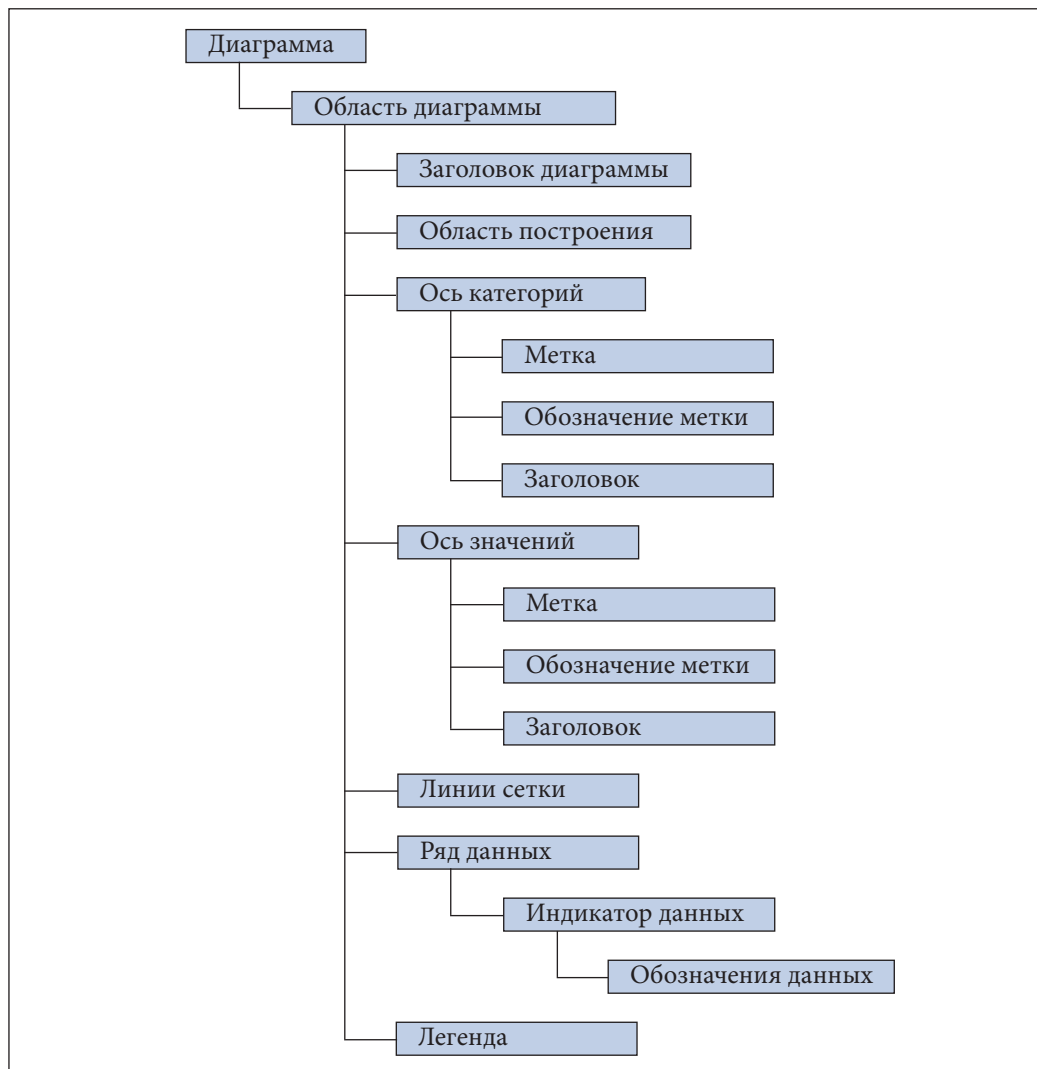


Рис. 3.10. Иерархия объектов диаграммы

На рисунке 3.10 видим, что на первом уровне иерархии находится сама диаграмма, а на втором уровне – область для диаграммы. Третий уровень содержит объекты, входящие в область для диаграммы, а именно: заголовок диаграммы, область построения, ось значений, ось категорий и т.п. Четвертый уровень образован объектами, включенными в те, что находятся на третьем уровне: метками, обозначениями меток, заголовками осей, индикаторами данных. Последний, пятый уровень, содержит обозначения данных, которые являются составными частями индикаторов данных.

Знание иерархии объектов упрощает процесс создания и изменения диаграмм, позволяя пользователю легко находить и выбирать обрабатываемые объекты.

Как и в других программах, любой объект, обрабатываемый при помощи табличного процессора, характеризуется определенными свойствами. Очевидно, что в этом смысле не являются исключением и объекты, входящие в диаграммы, которые характеризуются размерами, положением в рабочей книге или на рабочем листе, наличием или отсутствием рамки, способом изображения и т.д. Напомним, что набор свойств, которые описывают любой объект, называется **форматом**, а процесс установки и изменения этих свойств называется **форматированием**.

Обычно после создания диаграммы пользователь хочет добавить или удалить из нее некоторые элементы и/или изменить свойства определенных объектов. Например, заголовок диаграммы может располагаться над или под областью построения, а соответствующий текст может быть написан при помощи определенных шрифтов – **Arial**, **Times New Roman**, **Courier** и др. Аналогичным образом ось значений может иметь либо не иметь метки и обозначения.

Графическая подготовка диаграммы для печати или вставки в электронный документ называется редактированием диаграммы.

В процессе редактирования над объектами диаграммы могут выполняться следующие действия (операции):

- вставка;
- изменение размеров;
- перемещение;
- удаление;
- форматирование.

Команды, выполняющие эти операции, сгруппированы в меню **Edit** (Правка), **Format** (Формат), **Chart** (Диаграмма) и в контекстных меню. Например, меню **Edit** содержит команду **Clear** (Очистить), которая в зависимости от выбранного объекта предлагает пользователю опции **All** (Все), **Series** (Ряд) и **Formats** (Форматы). Аналогичным образом меню **Format** содержит команды **Selected Plot Area** (Выделенная область построения), **Selected Axis** (Выделенная ось), **Selected Axis Title** (Заголовок выделенной оси), **Selected Data Series** (Выделенный ряд) и др.

Современные табличные процессоры предоставляют пользователю очень подробные контекстные меню, набор опций которых изменяется в зависимости от объекта, выбранного с помощью курсора. Напомним, что выполнение команды такого меню осуществляется путем щелчка правой по редактируемому объекту и выбора нужной опции.

В общем случае **методы редактирования**, используемые при обработке диаграмм, предполагают выполнение следующих этапов:

- выбор подлежащего обработке объекта;
- выполнение нужной команды, используя для этого строки меню, контекстных меню или кнопок на стандартной панели инструментов;
- ввод, при необходимости, дополнительной информации, которая может потребоваться табличному процессору.

В качестве примера на *рисунке 3.11* изображен процесс форматирования области построения диаграммы **Notele elevilor**.

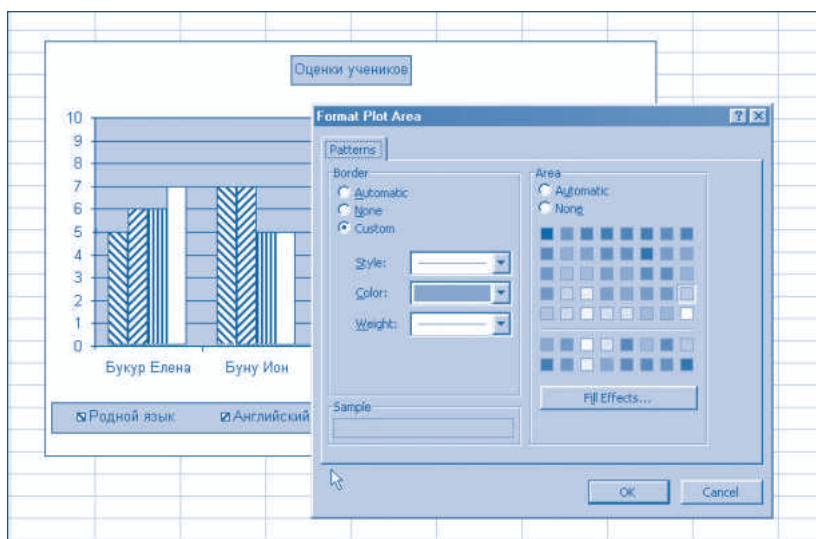


Рис. 3.11. Форматирование области построения

Чтобы придать области построения требуемый вид, пользователь вводит в диалоговом окне **Format Plot Area** (Формат области построения) информацию о наличии или отсутствии рамки (границ), стиле линий границ, наличии или отсутствии цвета фона, эффектах заливки и т.п.

Упрощение процесса создания и редактирования диаграмм обеспечивается при соблюдении следующих рекомендаций.

1. Устанавливайте диапазон ячеек рабочего листа, данные из которых необходимо отобразить в графическом виде, до начала создания диаграммы.

2. Выбирайте тип диаграммы в зависимости от характера информации, которую необходимо отобразить: гистограмму (изменение данных в пространстве или во времени), линейчатую (ряды данных, неменяющиеся во времени), круговую (отношения между частями и целым), точечную (взаимосвязь двух рядов чисел).

3. Выбирайте подтип (вид) диаграммы: двумерную (на плоскости) или трехмерную (в пространстве).

4. Включайте в диаграмму только те элементы, которые абсолютно необходимы для быстрого и правильного понимания данных.

5. Исключайте из диаграммы элементы, которые повторяют уже существующую информацию, или выполняют только декоративные функции.

6. Форматируйте каждый объект диаграммы, подбирая границы (рамки), цвет фона, варианты заливки, цвета рисования, размеры и используемые шрифты.

Отметим, что использование нескольких цветов оправдано лишь в тех случаях, когда диаграмма предназначена для распечатки на цветном принтере или вставки в электронный документ. Для черно-белых документов рекомендуется использование различных эффектов заливки области построения и индикаторов данных, что позволит отличать их друг от друга.

Вопросы и упражнения

- ❶ Объясните термины *инкапсуляция объектов* и *иерархия объектов*.
- ❷ Используя как образец *рисунок 3.10*, покажите на отдельных рисунках иерархию объектов диаграмм на *рисунках 3.8* и *3.9*.
- ❸ Покажите на рисунке иерархию объектов, входящих в состав документов, созданных с помощью современных редакторов текстов.
- ❹ Используя систему помощи и контекстные меню, определите свойства каждого объекта диаграмм **Число выпускников гимназии “Constantin Stere”** (*рис. 3.8*) и **Курс американского доллара по отношению к молдавскому лею** (*рис. 3.9*).
- ❺ Отредактируйте диаграмму **Notele elevilor** в соответствии с образцом, представленным на *рисунке 3.12*.

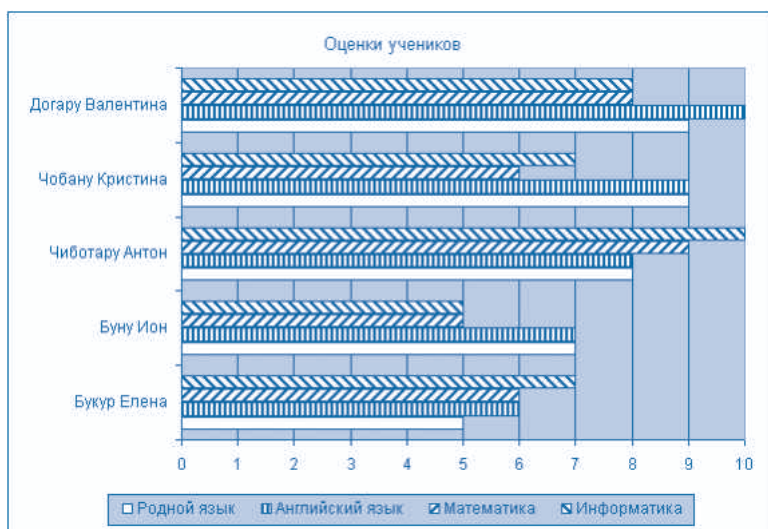


Рис. 3.12. Диаграмма **Notele elevilor**

- ❻ Выполнение команды **Chart Option** меню **Chart** выводит на экран диалоговое окно, которое содержит несколько страниц. Используя систему помощи, определите назначение элементов управления на каждой странице этого окна.
- ❼ Как Вы считаете, как можно подобрать цвета для элементов диаграммы. Объясните Ваш ответ.
- ❽ Как Вы считаете, в каких случаях нет необходимости включать в диаграмму следующие элементы:
 - заголовок диаграммы;
 - заголовки осей;
 - линии сетки;

- легенду;
 - метки;
 - обозначения меток.
- Обоснуйте Ваш ответ.

- 9 Преобразуйте диаграмму **Курс американского доллара по отношению к молдавскому лею** (рис. 3.9) в трехмерную. Как Вы считаете, какая из диаграмм более наглядна: двумерная или трехмерная.
- 10 Внимательно проанализируйте диаграммы из учебников географии и истории. Определите тип каждой диаграммы и назовите их элементы. Определите формат каждого объекта, входящего в их состав. Пользуясь табличным процессором, попробуйте воспроизвести одну из этих диаграмм.

3.4. КАРТЫ И ГРАФИЧЕСКИЕ ОБЪЕКТЫ

Ключевые термины:

- карта
- графические объекты
- инструменты для рисования

Для более наглядного представления числовых данных, характеризующих определенные географические регионы, табличные процессоры предлагают пользователю возможности вставлять в рабочие листы специальные диаграммы, называемые картами. Обычно в таких диаграммах числовые значения передаются путем раскраски каждой географической области в определенный цвет или вычерчиванием в соответствующих областях соответствующих индикаторов данных.

Карта представляет собой отдельный тип диаграммы, в которой индикаторы данных связаны с определенными географическими зонами: континентами, странами, регионами, уездами, районами и т.п.

Для примера на *рисунке 3.13* представлена карта, содержащая информацию о населении Республики Молдова. Цвет каждой географической области выбирается табличным процессором в зависимости от количества жителей, давая таким образом читателю возможность легче понять особенности каждой административно-территориальной единицы.

Визуальное воздействие карт более сильное, чем у обычных диаграмм, поскольку от читателя больше не требуется самостоятельного нахождения взаимосвязи между географическими зонами и числовыми данными, которые им соответствуют.

Как и в случае обычных диаграмм, табличный процессор вычерчивает карты на основе данных из ячеек рабочего листа. Соответствующая информация должна быть организована в виде таблицы, содержащей как минимум два столбца. Первый столбец в обязательном порядке должен содержать наименования географических зон, которые должны быть изображены на карте. В качестве примера на *рисунке 3.14* представлен рабочий лист, содержащий данные, на основе которых была создана карта **Население Республики Молдова**.

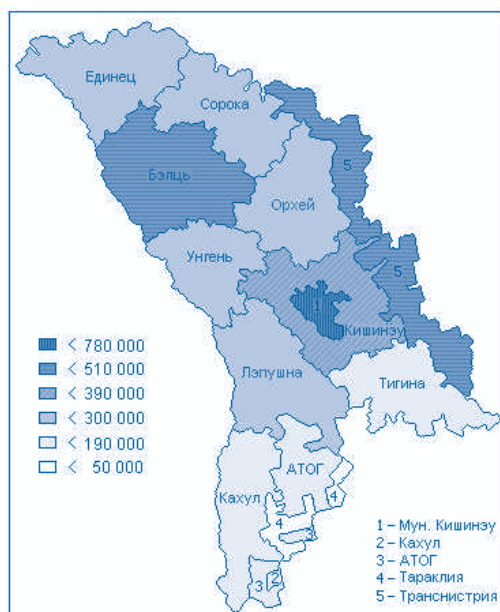


Рис. 3.13. Карта **Население Республики Молдова**

	A	B	C	D	E
1					
2		N	Административно-территориальные единицы	Население (тыс. человек)	
3		1	Мун. Кишинэу	779,0	
4		2	Бэлць	502,7	
5		3	Кахул	190,9	
6		4	Кишинэу	382,6	
7		5	Единец	281,7	
8		6	Лэпушна	277,1	
9		7	Орхей	299,5	
10		8	Сорока	276,2	
11		9	Тараклия	45,9	
12		10	Тигина	169,6	
13		11	Унгень	260,8	
14		12	АТО Гагаузия	159,2	
15					

Рис. 3.14. Рабочий лист **Populația Republicii Moldova** (Население Республики Молдова)

Очевидно, что любое изменение данных на рабочем листе ведет к автоматическому обновлению карт, созданных на их основе. Например, если пользователь изменит значение 299,5 в ячейке D9 на значение 301,5, цвет географической области Орхей (рис. 3.13) изменится автоматически.

В случае табличного процессора **Microsoft Excel**, создание карт выполняется при помощи специального приложения, именуемого **Microsoft Map** (Карта Microsoft). Различные меню этой программы содержат команды, предназначенные для выбора индикаторов данных, характера форматирования элементов карт, вставки и удаления обозначений и поясняющих текстов, выбора цвета и т.п. Подробно изучить каждую команду можно,

пользуясь системой помощи **Help**, которая содержит специальную главу, посвященную созданию и редактированию карт.

Отметим, что программы, предназначенные для создания и редактирования карт, требуют использования мощных компьютеров, которые являются быстродействующими и очень дорогостоящими. Вследствие этого, в школьных кабинетах информатики указанные программы устанавливаются только на таких компьютерах.

Как и в случае текстовых редакторов, в рабочие листы могут быть вставлены различные объекты: изображения, аудио- и видеофрагменты. В зависимости от способа создания объектов, которые могут быть вставлены в рабочий лист, будем различать:

- 1) объекты, созданные в приложении **Microsoft Excel**;
- 2) объекты, созданные при помощи других приложений.

Например, к первой группе принадлежат диаграммы и карты, а ко второй – рисунки, созданные в приложении **Paint**, звуковые фрагменты, записанные при помощи приложения **Sound Recorder** и др. Соответствующие объекты вставляются в рабочий лист с помощью одного из следующих методов:

- а) при помощи буферной памяти;
- б) путем обращения к приложению, в котором будет создан нужный объект;
- в) путем считывания нужного объекта из файла, указанного пользователем.

Вставка объектов производится аналогично тому, как это делается в текстовых редакторах: активизируется меню **Insert** (Вставка) и выбирается команда **Object** (Объект). Далее пользователь может указать файл, из которого будет взят нужный объект или запустить приложение, в котором он будет создан.

Практика показывает, что кроме диаграмм на рабочих листах используются и другие графические объекты, в частности, рисунки и изображения.

Будем называть графическими объектами рисунки и изображения, созданные в самом табличном процессоре или взятые из других приложений.
















Напомним, что графические объекты, создаваемые и обрабатываемые на компьютере, могут состоять из микрзон (точечно-ориентированная или растровая графика) или из более простых объектов (объектно-ориентированная или векторная графика). Обычно, растровые изображения берутся из других приложений, например, из программы **Paint**. В случае объектно-ориентированной (векторной) графики, табличные процессоры содержат в себе специальные программы, которые позволяют осуществлять вставку следующих объектов:




- линий;
- квадратов и прямоугольников;
- окружностей и эллипсов;
- дуг окружностей и эллипсов;
- неправильных фигур;
- поясняющих надписей;
- автофигур и т.д.

В табличном процессоре **Microsoft Excel** вставка указанных объектов осуществляется при помощи элементов управления панели инструментов рисования **Drawing** (Рисование), которая идентична имеющейся в текстовом редакторе **Microsoft Word**. Назначение кнопок панели инструментов рисования представлено в *таблице 3.1*.

После вставки выбранных графических объектов пользователь может изменять их свойства. Эти свойства индивидуальны для каждого графического объекта и выводятся

Инструменты рисования

Кнопка	Название	Назначение
	Draw (Действия)	Это меню содержит набор команд, обеспечивающих группировку объектов, изменение порядка наложения, выбор автофигур, выравнивание объектов и др.
	Select Objects (Выбор объектов)	Курсор для выбора графических объектов. При нажатии на эту кнопку курсор меняет свою форму на двоянную стрелку, приглашая этим пользователя выбрать один или несколько графических объектов.
	Free Rotate (Свободное вращение)	Вращение выбранного объекта. При нажатии на эту кнопку разметка выбранного объекта превращается в маленькие окружности, которые можно перетаскивать на требуемый угол.
	AutoShapes (Автофигуры)	Рисование автофигур: линий, стрелок, соединителей, геометрических фигур, логических схем и др.
	Line (Линия)	Рисование прямых линий.
	Arrow (Стрелка)	Рисование стрелок.
	Rectangle (Прямоугольник)	Рисование прямоугольников.
	Oval (Овал)	Рисование овалов.
	Text Box (Надпись)	Вставка надписей. Вставляемый текст может быть отформатирован независимо от других объектов рабочего листа.
	Insert WordArt (Вставка объекта WordArt)	Запускает приложение WordArt , которое позволяет вставлять художественно оформленные фрагменты текста.
	Fill Color (Цвет заливки)	Заливка замкнутой области выбранным цветом. Пользователь может также выбрать различные способы заливки (текстуры, узоры и др.).
	Line Color (Цвет линий)	Устанавливает цвет контура выбранного объекта.
	Font Color (Цвет шрифта)	Устанавливает цвет символов выбранного текста.
	Line Style (Тип линии)	Устанавливает толщину и форму контурных линий выбранного объекта.
	Dash Style (Тип штриха)	Устанавливает способ проведения контурных линий выбранных объектов: непрерывный, штриховой, точечный и т.п.

Кнопка	Название	Назначение
	Arrow Style (Вид стрелки)	Устанавливает форму краев отрезков выбранных прямых.
	Shadow (Тень)	Добавляет тень к выбранным объектам.
	3-D (Объем)	Превращает двумерные изображения объектов в трехмерные.

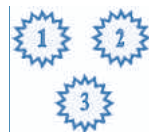
на экран при помощи контекстных меню. Для этого курсор устанавливается над формируемым объектом и выполняется щелчок левой. В зависимости от выбранного объекта табличный процессор выводит контекстное меню, содержащее опцию **Format xxxxxx** ..., где **xxxxxx** представляет наименование объекта – автоформа, **WordArt**, надпись и т.д. Очевидно, что это меню содержит и все команды, необходимые для копирования, вырезания и группирования объектов.

Вопросы и упражнения

- 1 Чем отличается карта, созданная при помощи табличного процессора, от обычной карты?
- 2 Как Вы считаете, какие индикаторы данных могут быть использованы в картах, создаваемых при помощи табличных процессоров?
- 3 Найдите в учебниках истории и географии карты, которые можно создавать при помощи табличных процессоров. Какой вид числовых данных представлен на этих картах? Назовите индикаторы данных, использованных на найденных Вами картах.
- 4 Как Вы считаете, в чем состоит отличие между диаграммой и картой, созданной при помощи табличного процессора?
- 5 Создайте на основе данных рабочего листа **Populația Republicii Moldova** (рис. 3.14) линейчатую диаграмму и гистограмму. Сопоставьте созданные диаграммы с картой на рисунке 3.13. Как Вы считаете, какое из этих представлений наиболее наглядно? Обоснуйте Ваш ответ.
- 6 Определите, установлена ли на Вашем компьютере программа для создания и редактирования карт.
- 7 Вспомните способы кодирования изображений в точно-ориентированной графике и объектно-ориентированной графике. Каковы преимущества и недостатки каждого из этих способов?
- 8 Назовите как минимум два источника, из которых можно брать растровые изображения.
- 9 Выведите на экран панель инструментов рисования и объясните назначение каждого элемента управления указанной панели.
- 10 Назовите операции, которые можно выполнять над графическими объектами.
- 11 Объясните, как могут быть отформатированы графические объекты, размещенные на рабочем листе.
- 12 Вставьте в рабочий лист **Notele elevilor** (рис. 3.2) следующие графические объекты таким образом, чтобы они показывали распределение учащихся в соответствии с их успеваемостью.



- 18 Вставьте в рабочий лист следующие графические объекты:



Отформатируйте указанные объекты в соответствии с приведенными выше образцами.

3.5. ПОСТРОЕНИЕ ГРАФИКОВ

Ключевые термины:

- график
- точечная диаграмма

Одной из часто встречающихся задач в математике, физике, химии, технике, медицине, экономике и т.п. является задача построения графиков.

График представляет собой рисунок, на котором показано изменение некоторой величины или связь между двумя переменными величинами.

В качестве примера вспомним графики математических функций $f: R \rightarrow R$, график механического движения $d = vt$, график нагревания воды, изменения температуры воздуха в течение дня или месяца и т.п.

В табличных процессорах графики можно построить на основе значений, содержащихся в ячейках рабочего листа, используя **точечные диаграммы – XY (Scatter)**. Соответствующие значения могут быть введены пользователем в рабочий лист, например, результаты измерений (температуры, скорости, расстояния, массы и т.д.) или вычислены автоматически при помощи самого табличного процессора. Для примера на *рисунке 3.15* представлен рабочий лист **Temperatura**, содержащий данные об изменении температуры в течение первых десяти часов одного из дней в марте.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2		t, час.	0	1	2	3	4	5	6	7	8	9	10	
3		T, град.	2,5	-0,3	-2,8	-3,4	-2,8	0,0	2,9	5,5	7,5	8,8	9,5	
4														
5														

Рис. 3.15. Рабочий лист **Temperatura**

График, построенный на основе данных рабочего листа **Temperatura**, представлен на *рисунке 3.16*. Отметим, что возможности мастера диаграмм не позволяют отобразить на графике стрелки, показывающие направление осей координат. Поэтому указанные элементы были вставлены как отдельные графические объекты.

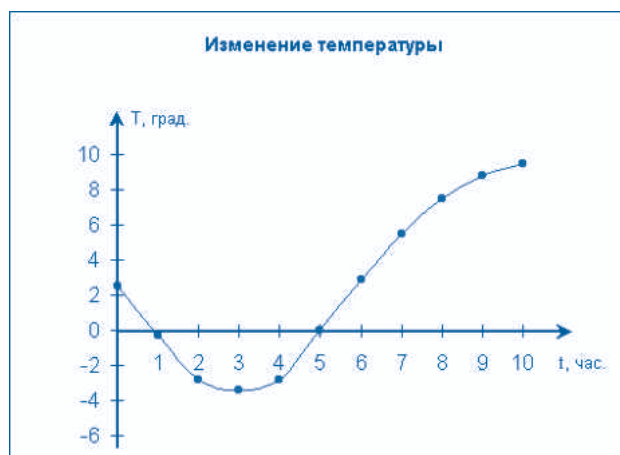


Рис. 3.16. График **Изменение температуры**

Графики математических функций могут быть построены аналогичным образом, а значения, необходимые для их построения, могут быть вычислены автоматически при помощи самого табличного процессора. Например, предположим, что нужно построить график функции

$$f(x) = x^2 - 5x + 3.$$

Очевидно, что для построения соответствующего графика, табличному процессору нужны значения аргумента x и значения функции $y = x^2 - 5x + 3$, представленные в виде таблицы. В принципе соответствующие значения могут быть вычислены пользователем «вручную» и введены в ячейки рабочего листа с клавиатуры, однако такой метод является неэффективным. Соответствующая таблица может быть заполнена автоматически самим табличным процессором путем записи в ячейки рабочего листа формул вида $=x*x-5*x+3$.

На *рисунке 3.17* представлен рабочий лист **Funcția de gradul doi**, который содержит таблицу, состоящую из двух строк. Первая строка этой таблицы содержит значения аргумента x , от 0 до 5 с шагом 0,5. Вторая строка таблицы содержит значения y , вычисленные по приведенной выше формуле.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2		x	0	0,5	1	1,5	2	2,5	3	3,5	4	4,5	5	
3		y	3	0,75	-1	-2,25	-3	-3,25	-3	-2,25	-1	0,75	3	
4														
5														

Рис. 3.17. Рабочий лист **Funcția de gradul doi** (Функция второй степени)

Чтобы записать в ячейки таблицы расчетные формулы, вначале вводим в ячейку C3 формулу $=C2*C2-5*C2+3$, которую затем копируем в ячейки D3:M3 рабочего листа.

График функции $y = x^2 - 5x + 3$, созданный на основе данных рабочего листа **Funcția de gradul doi**, представлен на *рисунке 3.18*.

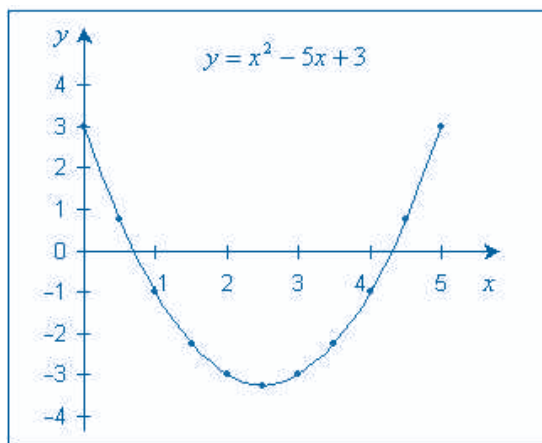


Рис. 3.18. График **Funcția de gradul doi**

Как и для предыдущего графика, стрелки, указывающие направление осей координат, вставлены в график функции второй степени как отдельные графические объекты.

В общем случае, для построения произвольного графика необходимо выполнить следующие этапы:

- 1) заполняем таблицу, содержащую значения функции;
- 2) строим на основе таблицы точечную диаграмму;
- 3) вставляем в диаграмму графические объекты, указывающие направления координатных осей;
- 4) вставляем, в случае необходимости, дополнительные элементы, облегчающие чтение графика (линии сетки, названия осей, название графика, единицы измерения и т.п.).

Создание таблицы, содержащей значения функции, требует от пользователя особого внимания, поскольку непродуманный выбор области значений аргумента может нарушить правильную работу мастера диаграмм.

Вопросы и упражнения

- ❶ Найдите в учебниках физики несколько графиков. Назовите элементы каждого графика.
- ❷ В каких случаях обосновано представление данных в виде графика?
- ❸ В следующей таблице представлено изменение температуры одного килограмма воды при ее нагревании. Постройте на основе этих данных график нагревания воды.

t, мин.	0	1	2	3	4	5	6	7	8
T, °C	60	65	72	80	88	95	100	100	100

- ❹ В следующей таблице представлен путь d , пройденный некоторым физическим телом за определенные интервалы времени t . Отобразите эти табличные данные в виде графика.

t, мин.	0	1	2	3	4	5
d, m	0	48	96	144	192	240

4.1. ЭЛЕМЕНТЫ БАЗЫ ДАННЫХ

Ключевые термины:

- простые данные
- составные данные
- база данных
- список данных
- запись
- поле

Подлежащая обработке информация представляется в компьютере в виде данных. Будем различать две категории данных: простые данные и составные данные.

Простые данные неделимы по отношению к операциям, которые над ними должны быть выполнены. Примеры таких данных: целые числа, действительные числа, слова текста, автофигуры и т.п.

Составные данные состоят из нескольких простых данных и могут быть разделены на составные части. Примеры: записи в классном журнале, которые содержат фамилию и имя ученика, год рождения, место жительства и т.п.; записи в платежном чеке, где указаны название купленного товара, количество, цена, налоги и т.п.

Очевидно, данные можно считать простыми, с одной точки зрения, и составными – с другой, в зависимости от вида предполагаемой обработки. Например, при арифметических операциях любое целое число можно рассматривать как простое данное, а в случае представления этого числа в компьютере оно может рассматриваться как составное данное, состоящее из более простых данных – двоичных или десятичных цифр.

В настоящее время объем обрабатываемых при помощи современных компьютеров данных очень велик, что усложняет хранение, извлечение, обработку и использование необходимой информации. Чтобы упростить эти процессы, были разработаны специальные средства, называемые базами данных.

База данных – это набор информации, организованной особым образом, что облегчает ее хранение и извлечение.

Например, если мы решили хранить в компьютере информацию об учениках класса в котором мы учимся, то можно создать базу данных, содержащую для каждого ученика составные данные, состоящие из совокупности простых данных – фамилии, имени, года рождения, места жительства и т.п. Аналогичным образом можно создать базу данных,

содержащую информацию обо всех учениках нашей школы или города, где мы живем. Впоследствии мы можем извлекать из созданной базы данных требуемую информацию, указывая для этого только фамилию или год рождения ученика, адрес проживания или класс, в котором он учится.

В общем случае, базы данных используются в самых различных областях: при управлении предприятиями, в учете населения, в таможенном контроле, в учете транспортных средств, в банковском учете и т.п. Обычно, создание и использование больших баз данных осуществляется при помощи специально для этого предназначенных программ, которые изучаются в углубленных курсах информатики.

Табличные процессоры содержат простые и эффективные средства для создания относительно малых баз данных, которые вполне достаточны для удовлетворения потребностей отдельных пользователей и небольших организаций, например, примэрий, школ, библиотек, магазинов и т.п. В табличных процессорах базы данных реализуются в виде списков.

Список данных представляет собой табличную структуру из строк и столбцов. Каждая строка списка называется записью, а каждый столбец – полем базы данных. Первая строка таблицы содержит имена полей.

В качестве примера на *рисунке 4.1* представлен рабочий лист, содержащий данные об участниках некоторого конкурса по информатике, организованные в виде списка данных.

	A	B	C	D	E	F	G	H	I
1									
2		N	Фамилия	Имя	Год рождения	Класс	Населенный пункт	Баллы	
3		1	Мунтяну	Ион	1995	12	Кишинэу	263	
4		2	Вулпе	Петру	1995	12	Орхей	290	
5		3	Плугару	Раиса	1996	11	Унгень	531	
6		4	Ротару	Константин	2000	7	Окница	208	
7		5	Морару	Ионица	1997	10	Кишинэу	412	
8		6	Чиботару	Михай	1997	10	Какул	585	
9		7	Мунтяну	Константин	2000	7	Калараш	301	
10		8	Маржине	Василе	1996	11	Орхей	531	
11		9	Бужур	Елена	1999	8	Тирасполь	115	
12		10	Олару	Анна	1995	12	Унгень	382	
13		11	Мунтяну	Анна	1999	8	Калараш	531	
14		12	Кобзару	Дорин	2000	7	Кишинэу	585	
15		13	Чиботару	Антон	1998	9	Орхей	314	
16		14	Мелинте	Виктор	1998	9	Окница	600	
17		15	Мунтяну	Александру	1996	11	Кишинэу	189	
18									

Рис. 4.1. Рабочий лист **Lista participanților** (Список участников)

Этот список содержит пятнадцать записей, по одной на каждого участника, и семь полей с именами: N, Фамилия, Имя, Год рождения, Класс, Населенный пункт, Баллы. Очевидно, что каждую запись можно рассматривать как составные данные, состоящие из простых данных, записанных в соответствующие поля. Например, запись в строке 5 состоит из следующих простых данных: 3, Плугару, Раиса, 1996, 11, Унгень, 531, записанных в ячейки B5:H5 рабочего листа.

Для того чтобы создать список, сначала вводим в одну строку наименования полей, а в следующие строки – сами записи. Каждая строка области, занимаемой списком, будет содержать одну запись, а каждый столбец – поле. В процессе создания списков нужно соблюдать следующие рекомендации:

- каждый список данных должен создаваться на отдельном рабочем листе;
- между именами полей и записями не должно быть пустых строк;
- ячейки из одного столбца должны содержать данные одного и того же типа;
- чтобы избежать путаницы, имена полей должны отличаться от содержащихся в них данных;
- данные в любой ячейке списка не должны начинаться с пробелов;
- ячейки, прилегающие к списку, должны оставаться пустыми;
- каждая запись должна быть пронумерована, для этого можно использовать первый или последний столбец списка.

Считается, что каждый список представляет собой отдельную базу данных, которая будет обрабатываться независимо от списков, находящихся на других рабочих листах одной и той же книги.

Вопросы и упражнения

- 1 Объясните термины *простые данные* и *составные данные*. Приведите примеры таких данных.
- 2 Что представляет собой база данных? Для чего предназначены базы данных?
- 3 Назовите несколько областей, в которых применяются базы данных.
- 4 Укажите на *рисунке 4.1* следующие элементы:
 - имена полей;
 - записи, содержащие фамилию Мунтяну;
 - записи, содержащие имя Анна;
 - записи, содержащие количество баллов, равное 531;
 - записи, содержащие населенный пункт Кишинэу.
- 5 Введите в компьютер рабочий лист **Lista participantilor** (*рис. 4.1*). Проверьте, соблюдены ли все рекомендации, сформулированные в данном параграфе.
- 6 Создайте по приведенному ниже образцу рабочий лист **Lista cărților** (Список книг).

№	Автор	Название	Издательство	Год	Страниц
1.	Гремальски Анатол	Информатика	Știința	2013	170
2.

Введите в созданный лист данные обо всех книгах, которыми Вы пользуетесь, по одной записи на каждую книгу.

- 7 Создайте по приведенному ниже образцу рабочий лист **Prietenii** (Друзья).

№	Фамилия	Имя	Дата рождения	Телефон	Адрес
1.	Андроник	Виталий	12-Сен-2000	324567	Дечебал, 13, кв. 4
2.

Введите в созданный лист данные о Ваших друзьях, по одной записи на человека.

- 8 Создайте рабочий лист **Lista telefoanelor**, который должен содержать данные из телефонной книжки, по одной записи на каждый часто используемый номер телефона.

4.2. ОБРАБОТКА СПИСКОВ

Ключевые термины:

- форма данных
- отображение записей
- поиск записей
- редактирование записей
- вставка записей
- удаление записей

Базы данных играют роль «хранилищ», откуда мы можем извлекать существующую или где мы можем хранить новую информацию. Ясно, что с течением времени базы данных необходимо обновлять, исключая из них ставшие ненужными записи, изменяя содержимое определенных ячеек указанных записей или добавляя новые записи. Следовательно, обработка списков данных предполагает выполнение следующих операций:

- поиск требуемых записей;
- вставка записей;
- удаление записей;
- изменение содержимого полей указанных записей.

Поскольку на рабочем листе списки данных не имеют особого статуса, то их обновление можно производить теми же стандартными способами, которые используются для обработки других видов информации:

- выбираем нужную ячейку;
- редактируем содержимое выбранной ячейки.

Очевидно, что для списков данных, состоящих из сотен и тысяч записей, поиск обрабатываемых ячеек вручную требует большого объема работы и становится утомительной операцией.

Для упрощения процессов управления списками данных табличные процессоры содержат специальные команды, предназначенные для обработки каждой записи как единого набора данных. Большинство из этих команд сгруппированы в диалоговом окне **Form** (Форма), которое можно вывести на экран при помощи одноименной команды из меню **Data** (Данные). После выбора произвольной ячейки из списка данных и запуска команды **Data, Form**, табличный процессор просматривает все заголовки столбцов и создает форму, содержащую текстовые поля, обозначенные с левой стороны именами соответствующих полей. Например, для рабочего листа **Lista participantilor** (рис. 4.1), форма имеет вид, представленный на рисунке 4.2.

Текстовые поля **формы данных** имеют двойное назначение. Во-первых, они служат для отображения информации соответствующих полей текущей записи. Во-вторых, они предназначены для редактирования отображенной информации, освобождая таким образом пользователя от необходимости утомительного просмотра всех ячеек списка. Отметим, что для быстрого перемещения по записям списка данных можно использовать линейку прокрутки, расположенную в центре формы. Номер текущей записи отображается в правом верхнем углу, благодаря чему упрощается выбор требуемых записей.





Назначение элементов управления форм данных приведено в *таблице 4.1*.

Рис. 4.2. Форма данных **Lista participantilor**

Таблица 4.1

Назначение элементов управления форм данных

Кнопка	Наименование	Назначение
	New (Добавить)	Очищает текстовые поля и готовит базу данных для ввода новой записи. В правом верхнем углу появляется сообщение New Record (Новая запись), которое является приглашением пользователю ввести в текстовые поля соответствующую информацию. Переход от одного текстового поля к другому осуществляется нажатием клавиши Tab или щелчком левой. Новая запись включается в базу данных только после нажатия клавиши Enter .
	Delete (Удалить)	Удаляет текущую запись из базы данных. Перед этим табличный процессор выводит сообщение, в котором предупреждает пользователя, что текущая запись будет безвозвратно утеряна.
	Restore (Вернуть)	Восстанавливает исходную информацию в текстовых полях формы данных. Восстановление информации может потребоваться в случае ввода нежелательных изменений в текущую запись. Восстановить информацию можно только до того, как нажата клавиша Enter .
	Find Previous (Назад)	Загружает в форму предыдущую запись, которая удовлетворяет критериям поиска, если они были указаны.
	Find Next (Далее)	Загружает в форму следующую запись, которая удовлетворяет критериям поиска, если они были указаны.

Кнопка	Наименование	Назначение
	Criteria (Критерии)	Очищает текстовые поля формы и готовит базу данных к поиску определенных записей. В правом верхнем углу формы данных появляется сообщение Criteria , которое является приглашением пользователю ввести в текстовые поля критерии поиска. Ими могут быть определенные значения, например, Мунтяну, или условия вида =2000. После нажатия одной из кнопок Find Prev или Find Next в форму данных будет загружена запись, удовлетворяющая указанным критериям поиска.
	Clear (Очистить)	Стирает критерии поиска в текстовых полях формы данных.
	Form (Правка)	Переключает форму данных из режима поиска в режим отображения записей.
	Close (Закреть)	Закрывает форму данных.

Чтобы облегчить процесс обновления баз данных, советуем использовать следующие типовые операции, предназначенные для обработки списков из рабочих листов.

Отображение записей:

- 1) выбираем произвольную ячейку списка данных;
- 2) запускаем команду **Data, Form**, которая выводит на экран форму данных, отображающую первую запись списка;
- 3) для того чтобы перейти к следующей записи, нажимаем кнопку **Find Next**;
- 4) для того чтобы перейти к предыдущей записи, нажимаем кнопку **Find Prev**.

Поиск записи:

- 1) выбираем произвольную ячейку списка данных;
- 2) запускаем команду **Data, Form**, которая выводит на экран форму данных, отображающую первую запись списка;
- 3) нажимаем кнопку **Criteria**;
- 4) вводим в одно или несколько текстовых полей критерии поиска;
- 5) для того чтобы перейти к следующей записи, удовлетворяющей критериям поиска, нажимаем кнопку **Find Next**;
- 6) для того чтобы перейти к предыдущей записи, удовлетворяющей критериям поиска, нажимаем кнопку **Find Prev**;
- 7) после того как запись найдена, нажимаем кнопку **Form**.

Редактирование записи:

- 1) загружаем в форму данных нужную запись;
- 2) вводим в текстовые поля необходимые изменения;
- 3) нажимаем клавишу **Enter**;
- 4) переходим к редактированию другой записи или нажимаем кнопку **Close**.

Вставка новой записи:

- 1) выбираем произвольную ячейку списка данных;

- 2) запускаем команду **Data, Form**, которая выводит на экран форму данных, отображающую первую запись списка;
- 3) нажимаем кнопку **New**;
- 4) вводим в текстовые поля данные для новой записи;
- 5) для вставки записи в список, нажимаем кнопку **Enter**;
- 6) вводим в текстовые поля данные для другой новой записи и т.д.;
- 7) для окончания процесса вставки нажимаем кнопку **Close**.

Удаление записи:

- 1) выбираем произвольную ячейку списка данных;
- 2) запускаем команду **Data, Form**, которая выводит на экран форму данных, отображающую первую запись списка;
- 3) загружаем в форму данных требуемую запись;
- 4) нажимаем кнопку **Delete**.

Эффективное использование команд, содержащихся в формах данных, подразумевает наличие определенных навыков, которые можно приобрести, решая как можно больше практических задач по обработке списков данных.

Вопросы и упражнения

- 1 Назовите часто используемые операции, предназначенные для обработки списков данных.
- 2 Для чего предназначена команда **Data, Form**? Откуда берутся обозначения текстовых полей из формы данных?
- 3 Объясните назначение элементов управления формы данных табличного процессора.
- 4 Откройте рабочий лист **Lista participanților** и выведите на экран форму данных, представленную на *рисунке 4.2*. Какую информацию содержат текстовые поля непосредственно после открытия формы?
- 5 Используя линейку прокрутки формы на *рисунке 4.2*, выведите на экран все записи рабочего листа **Lista participanților**. Отметьте, как изменяется информация, выводимая в текстовых полях формы данных. Укажите на рабочем листе ячейки, информация из которых отображается в соответствующих полях.
- 6 Выведите на экран форму, представленную на *рисунке 4.2*. Переведите форму в режим поиска (кнопка **Criteria**) и введите в текстовое поле **Фамилия** слово Мунтяну. Отметьте, как изменяется содержимое всех текстовых полей после нажатия кнопок **Find Prev** и **Find Next**. Укажите на рабочем листе записи, информация о которых выводится в текстовых полях формы.
- 7 Отобразите из списка данных, представленного на *рисунке 4.1*, только записи, относящиеся к:
 - а) участникам с фамилией Мунтяну;
 - б) участникам, которые учатся в 7 классе;
 - в) участникам, родившимся в 2000 году;
 - г) участникам с именем Анна;
 - д) участникам, которые учатся в 10 классе;
 - е) участникам, родившимся в 1996 году;
 - ж) участникам, набравшим более 200 баллов;
 - з) участникам, набравшим более 500 баллов.
- 8 Измените количество баллов, набранных Кобзару Дорином (*рис. 4.1*) с 585 на 588, а Мелинте Виктором – с 600 на 598.

- 9 Удалите с рабочего листа **Lista participanților** (рис. 4.1) записи с номерами 3, 7 и 10. Перенумеруйте записи, оставшиеся в списке.
- 10 Вставьте в рабочий лист **Lista participanților** (рис. 4.1) записи из следующей таблицы:

№	Фамилия	Имя	Год рождения	Класс	Населенный пункт	Баллы
16.	Петру	Ион	1995	12	Бэлць	149
17.	Мардаре	Лучия	1999	8	Сорока	523
18.	Ботнару	Елена	1997	10	Бэлць	487

- 11 Объясните, как можно выполнить следующие действия:
- отображение всех записей списка данных;
 - поиск определенных записей;
 - вставку новых записей;
 - удаление указанных записей;
 - изменение определенных данных в существующих записях.
- 12 Как Вы считаете, какой метод обработки списков данных проще: основанный на выборе каждой ячейки или основанный на использовании форм данных? Обоснуйте Ваш ответ.

4.3. СОРТИРОВКА ЗАПИСЕЙ

Ключевые термины:

- сортировка
- ключ сортировки
- порядок сортировки
- сортировка по возрастанию
- сортировка по убыванию

Информацию, которая содержится в базах данных, легче обрабатывать, если пользователь имеет возможность располагать соответствующие записи в определенном порядке. Например, записи из базы данных **Lista participanților** (рис. 4.1) можно упорядочить в алфавитном порядке по значениям полей Фамилия, Имя или в числовом порядке по значениям поля Баллы. Упорядочивание в алфавитном порядке полезно при регистрации участников, а упорядочивание в числовом порядке – при определении победителей конкурса.

Операция изменения порядка следования записей списка данных в зависимости от значений одного или нескольких полей называется сортировкой.

Табличные процессоры располагают специальными средствами сортировки записей. В случае приложения **Microsoft Excel** сортировка записей выполняется при помощи команды **Data, Sort** (Данные, Сортировка). После запуска указанной команды на экран выводится диалоговое окно **Sort** (рис. 4.3), в котором пользователь указывает параметры, необходимые для выполнения сортировки.

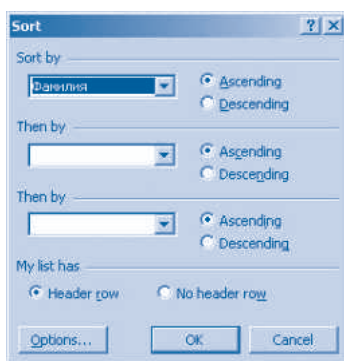


Рис. 4.3. Диалоговое окно **Sort** (Сортировка)

В диалоговом окне **Sort** можно указать до трех **ключей сортировки**, каждый из которых представляет имя некоторого поля из заголовка списка. **Порядок сортировки** – по возрастанию или по убыванию – устанавливается для каждого ключа при помощи радиокнопок **Ascending** (Сортировка по возрастанию) и **Descending** (Сортировка по убыванию).

Отметим, что в случае текстовых значений записи будут упорядочены в лексикографическом порядке в соответствии с таблицей кодировки, установленной в соответствующем компьютере. Естественно, в случае текстовых данных, не содержащих математических символов, соответствующие значения будут упорядочены в алфавитном порядке.

В качестве примера на *рисунке 4.4* представлен рабочий лист **Lista participantilor**, после сортировки базы данных по ключу **Фамилия** в возрастающем порядке.

Сопоставляя *рисунок 4.1*, на котором представлен список до сортировки, и *рисунок 4.4*, где представлен тот же список после сортировки, замечаем, что записи списка были упорядочены в алфавитном порядке только по текстовым значениям поля **Фамилия**. Так как поле **Имя** не было указано в качестве второго ключа сортировки (*рис. 4.3*), то соответствующие значения не были учтены. Например, в списке имеются две записи с фамилиями **Чиботару** и четыре записи с фамилиями **Мунтяну**, которые не упорядочены в алфавитном порядке по полю **Имя**. Для того чтобы отсортировать записи по обоим

	A	B	C	D	E	F	G	H	I
1									
2		N	Фамилия	Имя	Год рождения	Класс	Населенный пункт	Баллы	
3		9	Букур	Елена	1999	8	Тирасполь	115	
4		2	Вулпе	Петру	1995	12	Орхей	290	
5		12	Кобзару	Дорин	2000	7	Кишинэу	585	
6		8	Маржине	Василе	1996	11	Орхей	531	
7		14	Мелинте	Виктор	1998	9	Окница	600	
8		5	Морару	Ионица	1997	10	Кишинэу	412	
9		1	Мунтяну	Ион	1995	12	Кишинэу	263	
10		7	Мунтяну	Константин	2000	7	Калараш	301	
11		11	Мунтяну	Анна	1999	8	Калараш	531	
12		15	Мунтяну	Александру	1996	11	Кишинэу	189	
13		10	Олару	Анна	1995	12	Унгень	382	
14		3	Плугару	Раиса	1996	11	Унгень	531	
15		4	Ротару	Константин	2000	7	Окница	208	
16		6	Чиботару	Михай	1997	10	Кахул	585	
17		13	Чиботару	Антон	1998	9	Орхей	314	
18									

Рис. 4.4. Сортировка по ключу **Фамилия**

ключам – Фамилия и Имя, в диалоговом окне **Sort** указываются оба поля и нужный порядок сортировки (рис. 4.5).

База данных, отсортированная по ключам Фамилия и Имя, представлена на рисунке 4.6.

Сопоставляя список, отсортированный по ключу Фамилия (рис. 4.4), и список, отсортированный по ключам Фамилия и Имя (рис. 4.6), видим, что в случаях, когда фамилии совпадают, записи упорядочиваются в алфавитном порядке и с учетом имен участников конкурса. Таким образом, в случае фамилии Чиботару первым в списке появляется Чиботару Антон, а следом за ним – Чиботару Михай. Аналогичным образом участники с фамилиями Мунтяну, расположатся в следующем порядке: Александру, Анна, Ион и Константин.

Следовательно, упорядочивание записей осуществляется в том порядке, в котором были указаны ключи сортировки диалогового окна **Sort** (рис. 4.5):

- 1) в первую очередь записи сортируются по первому ключу;
- 2) далее, если он был указан, записи сортируются по второму ключу;
- 3) наконец, если он был указан, записи сортируются по третьему ключу.

При необходимости пользователь может указать дополнительные параметры сортировки. Так, после нажатия кнопки **Options** (Параметры), на экран будет выведено диа-

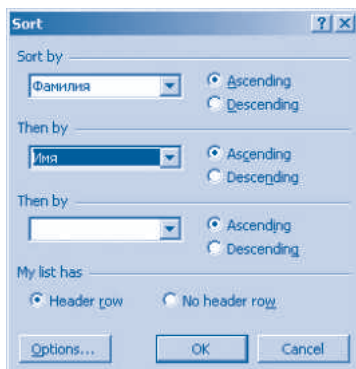


Рис. 4.5. Задание двух ключей сортировки

	A	B	C	D	E	F	G	H	I
1									
2		N	Фамилия	Имя	Год рождения	Класс	Населенный пункт	Баллы	
3		9	Букур	Елена	1999	8	Тирасполь	115	
4		2	Вулпе	Петру	1995	12	Орхей	290	
5		12	Кобзару	Дорин	2000	7	Кишинэу	585	
6		8	Маржине	Василе	1996	11	Орхей	531	
7		14	Мелинте	Виктор	1998	9	Окница	600	
8		5	Морару	Ионица	1997	10	Кишинэу	412	
9		15	Мунтяну	Александру	1996	11	Кишинэу	189	
10		11	Мунтяну	Анна	1999	8	Калараш	531	
11		1	Мунтяну	Ион	1995	12	Кишинэу	263	
12		7	Мунтяну	Константин	2000	7	Калараш	301	
13		10	Олару	Анна	1995	12	Унгень	382	
14		3	Плугару	Раиса	1996	11	Унгень	531	
15		4	Ротару	Константин	2000	7	Окница	208	
16		13	Чиботару	Антон	1998	9	Орхей	314	
17		6	Чиботару	Михай	1997	10	Кахул	585	
18									

Рис. 4.6. Сортировка списка по ключам **Фамилия** и **Имя**

логовое окно **Sort Options** (Параметры сортировки), в котором можно уточнить порядок сортировки: по месяцам года или по дням недели, по строкам или по столбцам и др.

В общем случае, **порядок сортировки по возрастанию** упорядочивает записи следующим образом:

- а) числа – от наименьшего отрицательного до наибольшего положительного;
- б) календарные даты и время – в хронологическом порядке;
- в) тексты – в алфавитном порядке;
- г) логические значения – сначала FALSE, а затем TRUE;
- д) значения ошибки – в том порядке, в котором они встречаются;
- е) пробелы.

Очевидно, что сортировка **по убыванию** упорядочивает записи строго в обратном порядке по отношению к сортировке по возрастанию, за исключением того факта, что пробелы также размещаются в последней позиции.

Отметим, что табличные процессоры предоставляют пользователю возможность сортировать список целиком или только его определенную часть. Для сортировки только части списка требуемые записи необходимо выбрать до запуска команды **Data, Sort**.

Вопросы и упражнения

- ❶ Объясните термин *сортировка записей*. Когда возникает необходимость сортировки записей базы данных?
- ❷ Для чего предназначены ключи сортировки? Каким образом указываются ключи, в соответствии с которыми будет выполняться сортировка?
- ❸ Как указывается порядок сортировки записей произвольной базы данных? Может ли быть изменен этот порядок?
- ❹ Объясните, как выполняется сортировка по возрастанию следующих данных:
 - числа;
 - календарные даты;
 - время;
 - тексты;
 - логические значения;
 - ошибки.Как выполняется сортировка указанных данных по убыванию?
- ❺ Имеет ли значение порядок, в котором перечислены ключи сортировки в диалоговом окне **Sort** (рис. 4.5)? Обоснуйте Ваш ответ.
- ❻ Откройте рабочий лист **Lista participanților** (рис. 4.1). Отсортируйте записи базы данных этого рабочего листа следующим образом:
 - а) по ключу *Фамилия*, по возрастанию;
 - б) по ключу *Фамилия*, по убыванию;
 - в) по ключам *Фамилия* и *Имя*, по возрастанию;
 - г) по ключам *Фамилия* и *Имя*, по убыванию;
 - д) по ключу *Баллы*, по возрастанию;
 - е) по ключу *Баллы*, по убыванию;
 - ж) по ключу *Год рождения*, по возрастанию;
 - з) по ключу *Год рождения*, по убыванию;
 - и) по ключу *Класс*, по возрастанию;
 - к) по ключу *Класс*, по убыванию;

- л) по ключам Год рождения и Класс, по возрастанию;
м) по ключам Год рождения и Класс, по убыванию;
н) по ключу Класс, по возрастанию, и по ключу Баллы, по убыванию.
Объясните результаты, выводимые на экран.

- 7 Упорядочьте записи рабочего листа **Lista cărților** (смотрите упражнение 6 параграфа 4.1) в следующем порядке:
- в алфавитном порядке авторов;
 - в алфавитном порядке названий;
 - в алфавитном порядке издательств;
 - в хронологическом порядке годов издания;
 - в порядке возрастания количества страниц.
- 8 Упорядочьте записи рабочего листа **Prietenii** (смотрите упражнение 7 параграфа 4.1) в следующем порядке:
- в алфавитном порядке фамилий и имен;
 - в хронологическом порядке дат рождения;
 - в алфавитном порядке названий улиц.
- 9 Откройте рабочий лист **Note** (рис. 2.13). Этот лист содержит три отдельные области ячеек:
B2:J12 – список учеников и оценок по определенным предметам;
L2:N12 – минимальная, средняя и максимальная оценки каждого ученика;
С14:J16 – минимальная, средняя и максимальная оценки по каждому предмету.
Поскольку каждая из этих областей окружена пустыми ячейками, приложение **Microsoft Excel** воспринимает их как отдельные списки данных. Отсортируйте список учеников следующим образом:
- в алфавитном порядке фамилий и имен;
 - по возрастанию оценок по одному из предметов, например, по информатике;
 - по возрастанию минимальных оценок в столбце L;
 - по убыванию средних оценок в столбце M;
 - по возрастанию максимальных оценок в столбце N.
- Попробуйте скопировать на другой рабочий лист данные о минимальных, средних и максимальных оценках по каждому предмету. Отсортируйте эти данные по возрастанию средних оценок.

4.4. ОТБОР ЗАПИСЕЙ

Ключевые термины:

- отбор записей
- фильтрация записей
- условия отбора
- копирование отобранных записей

Базы данных используются для хранения больших объемов информации. В процессе обработки этой информации появляется необходимость выбора записей, удовлетворяющих определенным условиям. Например, в случае рабочего листа **Lista participantilor** (рис. 4.1) представляет интерес извлечение записей об участниках, на-

бравших более 500 баллов, об участниках, которые учатся в 8 классе, об участниках из одного города и т.п.

Операция выбора из базы данных записей, удовлетворяющих определенным условиям, называется отбором или фильтрацией записей.

Для приложения **Microsoft Excel** отбор записей осуществляется при помощи команды **Filter** (Фильтр) из меню **Data**. Данная команда содержит три подкоманды: **AutoFilter** (Автофильтр), **Show All** (Отобразить всё) и **Advanced Filter** (Расширенный фильтр). После выбора произвольной ячейки из списка данных и выполнения команды **Data, Filter, AutoFilter** табличный процессор присоединяет к каждому имени поля раскрывающийся список, содержащий все значения, которые встречаются в соответствующем столбце (рис. 4.7).

	A	B	C	D	E	F	G	H
1								
2		N	Фамилия	Имя	Год рождения	Класс	Населенный пункт	Баллы
3		1	Мунтяну	Ион	(All)	12	Кишинэу	263
4		2	Вулле	Петру	(Top 10...)	12	Орхей	290
5		3	Плугару	Раиса	(Custom...)	11	Унгень	531
6		4	Ротару	Константин	1995	7	Окница	208
7		5	Морару	Июница	1996	10	Кишинэу	412
8		6	Чиботару	Михай	1997	10	Кахул	585
9		7	Мунтяну	Константин	1998	7	Калараш	301
10		8	Маржине	Василе	1999	11	Орхей	531
11		9	Букур	Елена	1995	8	Тирасполь	115
12		10	Олару	Анна	1999	12	Унгень	382
13		11	Мунтяну	Анна	1999	8	Калараш	531
14		12	Кобзару	Дорин	2000	7	Кишинэу	585
15		13	Чиботару	Антон	1998	9	Орхей	314
16		14	Мелинте	Виктор	1998	9	Окница	600
17		15	Мунтяну	Александр	1996	11	Кишинэу	189
18								

Рис. 4.7. Автоматическая фильтрация записей

Чтобы выбрать нужные записи, пользователь должен указать для одного или нескольких полей списка данных **условия отбора**. Ими могут быть:

- All** – все возможные значения ячеек данного поля;
- Top 10** – первые десять из всех значений, встречающихся в данном поле, упорядоченные по возрастанию;
- Custom** – условия отбора, задаваемые пользователем;
- одно из значений, встречающихся в данном поле.

Сразу после задания условий отбора табличный процессор просматривает список данных, проверяет для каждого поля соблюдение условия отбора и «прячет» записи, не удовлетворяющие им. Таким образом, после фильтрации в списке данных будут отображены только те записи, которые удовлетворяют условиям отбора, указанным для каждого поля. Чтобы удалить условия отбора и вернуться к режиму отображения всех записей, выполняется команда **Data, Filter, Show All** (Данные, Фильтр, Отобразить всё).

Например, предположим, что необходимо отобрать все записи об участниках конкурса по информатике, родившихся в 2000 году. Для отбора таких записей в раскрываемом списке поля *Год рождения* (рис. 4.7) указывается значение 2000. Непосредственно после выбора этой опции на экране отображается список, представленный на рисунке 4.8.

	A	B	C	D	E	F	G	H	I
1									
2		N	Фамилия	Имя	Год рождения	Класс	Населенный пункт	Баллы	
6		4	Ротару	Константин	2000	7	Окница	208	
9		7	Мунтяну	Константин	2000	7	Калараш	301	
14		12	Кобзару	Дорин	2000	7	Кишинэу	585	
18									
19									

Рис. 4.8. Отбор записей в соответствии с условием «Год рождения = 2000»

Аналогичным образом, если необходимо отобразить всех участников из Кишинэу, соответствующее значение указывается в раскрывающемся списке поля Населенный пункт. Записи, отобранные табличным процессором, представлены на рисунке 4.9.

	A	B	C	D	E	F	G	H	I
1									
2		N	Фамилия	Имя	Год рождения	Класс	Населенный пункт	Баллы	
3		1	Мунтяну	Ион	1995	12	Кишинэу	263	
7		5	Морару	Ионица	1997	10	Кишинэу	412	
14		12	Кобзару	Дорин	2000	7	Кишинэу	585	
17		15	Мунтяну	Александру	1996	11	Кишинэу	189	
18									
19									

Рис. 4.9. Отбор записей в соответствии с условием «Населенный пункт = Кишинэу»

Для того чтобы отобразить записи, относящиеся к участникам из Кишинэу, родившимся в 2000 году, соответствующие значения указываются в двух раскрывающихся списках: значение 2000 – в списке Год рождения и значение Кишинэу – в списке Населенный пункт. В этом случае табличный процессор выводит на экран единственную запись (рис. 4.10).

	A	B	C	D	E	F	G	H	I
1									
2		N	Фамилия	Имя	Год рождения	Класс	Населенный пункт	Баллы	
14		12	Кобзару	Дорин	2000	7	Кишинэу	585	
18									
19									

Рис. 4.10. Отбор записей в соответствии с условиями отбора «Год рождения = 2000» и «Населенный пункт = Кишинэу»

Если для каждого из полей пользователь выбирает в раскрывающихся списках опцию **Custom** (Условие), табличный процессор выводит на экран диалоговое окно **Custom AutoFilter** (Пользовательский автофильтр), в котором могут быть указаны условия отбора, определяемые пользователем. Например, если нужно отобразить записи об участниках, набравших 500 и более баллов, то в диалоговом окне вводится условие “Баллы is greater than or equal to 500” (рис. 4.11). Записи, отобранные табличным процессором, представлены на рисунке 4.12.

Критерии отбора, задаваемые пользователем, могут состоять из одного или двух условий, объединенных логической операцией И (**And**) / ИЛИ (**Or**). Как правило, такие

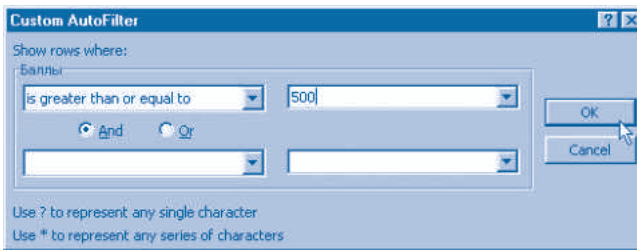


Рис. 4.11. Условия отбора, определяемые пользователем

	A	B	C	D	E	F	G	H	I
1									
2		N	Фамилия	Имя	Год рождения	Класс	Населенный пункт	Баллы	
5		3	Плугару	Раиса	1996	11	Унгень	531	
8		6	Чиботару	Михай	1997	10	Кахул	585	
10		8	Маржине	Василе	1996	11	Орхей	531	
13		11	Мунтяну	Анна	1999	8	Калараш	531	
14		12	Кобзару	Дорин	2000	7	Кишинэу	585	
16		14	Мелинте	Виктор	1998	9	Окница	600	
18									
19									

Рис. 4.12. Отбор записей в соответствии с условием «Баллы \geq 500»

условия используются в тех случаях, когда необходимо отобразить записи со значениями, принадлежащими определенной области.

В общем, отобранные записи могут быть подвергнуты дальнейшей обработке, например, отсортированы по определенным полям, вставлены в другой список, отпечатаны на принтере и т.п. В таких случаях рекомендуется скопировать отобранные записи в другую область рабочего листа или даже на другой лист.

Для **копирования отобранных записей** выполняются следующие операции:

- 1) отбираем нужные записи, используя команду **Data, AutoFilter**;
- 2) выбираем область ячеек, содержащих список отфильтрованных записей;
- 3) выполняем команду **Copy** из меню **Edit** (когда существует активный фильтр, данная команда копирует только выбранные записи);
- 4) выбираем левую верхнюю ячейку области, в которой требуется выполнить копирование;
- 5) выполняем команду **Paste** из меню **Edit**.

Табличный процессор копирует выбранные записи в новое место без внесения изменений в исходном списке. Напомним, что для того, чтобы вернуться к отображению всех записей исходного списка и отключить условия отбора, необходимо выполнить команду **Data, Filter, Show All**.

Вопросы и упражнения

- 1) Для чего предназначены раскрывающиеся списки, присоединяемые табличным процессором к каждому имени поля списка данных?
- 2) Объясните термин *отбор (фильтрация)* записей. В каком месте рабочего листа хранятся отобранные записи?

- ③ Для чего предназначены опции **All**, **Top 10** и **Custom** из раскрывающихся списков, которые присоединяются табличным процессором к именам полей списка данных?
- ④ Объясните назначение элементов управления диалогового окна **Custom AutoFilter** (рис. 4.11).
- ⑤ Как можно скопировать отобранные записи в другое место рабочего листа?
- ⑥ Объясните назначение опций **AutoFilter** и **Show All** команды **Filter** в меню **Data**.
- ⑦ Пользуясь системой помощи, попытайтесь найти назначение опции **Advanced Filter** команды **Data, Filter**.
- ⑧ Имеет ли значение порядок, в котором указаны условия отбора из раскрывающихся списков, присоединенных к именам полей? Обоснуйте Ваш ответ.
- ⑨ Откройте рабочий лист **Lista participanților** (рис. 4.1). Отберите из списка данных записи, удовлетворяющие следующим условиям:
- отобранные участники проживают в Орхее;
 - отобранные участники родились в 1984 году;
 - отобранные участники родились в 1984 году и живут в Орхее;
 - отобранные участники родились с 1984 по 1986 год;
 - отобранные участники проживают в Орхее или в Кишинэу;
 - отобранные участники родились с 1984 по 1986 год и живут в Орхее или в Кишинэу;
 - отобранные участники имеют фамилию Мунтяну или Чиботару;
 - отобранные участники учатся в 7 классе и живут в Окнице;
 - отобранные участники набрали 550 и более баллов;
 - отобранные участники набрали от 500 до 600 баллов;
 - фамилии отобранных участников начинаются с буквы М;
 - имена отобранных участников начинаются с буквы А;
 - отобранные участники учатся в классах с 7 по 9.
- Объясните результаты, выводимые на экран.
- ⑩ Откройте рабочий лист **Lista cărților** (смотрите упражнение 6 в параграфе 4.1). Отберите из списка данных записи, удовлетворяющие следующим условиям:
- книги, изданные в 1995–2003 годах;
 - книги, выпущенные Издательством *Știința*;
 - книги объемом до 200 страниц.
- ⑪ Отберите из списка данных на рабочем листе **Prietenii** (смотрите упражнение 7 в параграфе 4.1) записи о следующих людях:
- те, кто родились в год, указанный пользователем;
 - те, чье имя указано пользователем;
 - те, чей телефон указан пользователем.
- ⑫ Отберите на рабочем листе **Lista participanților** (рис. 4.1) записи, относящиеся к участникам, набравшим 500 и более баллов. Скопируйте отобранные записи на другой рабочий лист. Отсортируйте полученный список по ключу **Баллы** в порядке убывания.
- ⑬ Создайте из базы данных рабочего листа **Lista participanților** (рис. 4.1) отдельные базы данных, которые бы содержали:
- участников, обучающихся в 7 классе;
 - участников, обучающихся в 8 классе;
 - участников, обучающихся в 9 классе;
 - участников, которые проживают в Кишинэу;
 - участников, которые проживают в Орхее;
 - участников, родившихся в 1984 году;
 - участников, родившихся в 1989 году;
 - участников из 11 класса, набравших 500 баллов и более.
- Упорядочьте полученные списки данных по ключам **Фамилия** и **Имя**, по возрастанию.

5.1. АЛГОРИТМЫ И ИСПОЛНИТЕЛИ

Ключевые термины:

- алгоритм
- исполнитель
- ручное управление
- программное управление
- программа
- язык программирования

Обработка информации при помощи компьютера предполагает наличие определенной программы, которая загружается в его внутреннюю память. Мы уже знаем, что любая компьютерная программа представляет собой последовательность двоичных слов, которые указывают компьютеру порядок вычислений. Компьютерные программы разрабатываются на основе алгоритмов.

Интуитивно алгоритм можно представить как конечную совокупность правил и предписаний, выполнение которых обеспечивает решение поставленной задачи. Процесс разработки алгоритмов называется алгоритмизацией.

Напомним, что в повседневной жизни предписания представляют собой точные указания по выполнению определенных действий (определенной работы). Примеры таких указаний:

- 1) до запуска компьютера проверьте подключен ли сетевой кабель;
- 2) если $a \neq 0$, вычислите $x = b/a$;
- 3) чтобы найти корни уравнения второй степени, в первую очередь вычислите дискриминант $D = b^2 - 4ac$.

Слово **алгоритм** происходит от имени великого математика средневековья Аль-Хорезми (*Al-Khwarizmi Muhammed ibn Musa*, примерно 780–850 гг.), который впервые сформулировал точные и ясные правила для выполнения основных арифметических операций, которые сейчас изучаются в начальной школе: сложения, вычитания, умножения и деления. Позже эти правила появились под наименованием **алгоритм** в книге «Элементы Евклида». Считается, что одним из первых известных в математике алгоритмов является алгоритм Евклида для нахождения наибольшего общего делителя двух натуральных чисел.

Пример алгоритма:

Исходные данные: длины a , b катетов прямоугольного треугольника.

1. Вычислите $x=a^2$.

2. Вычислите $y=b^2$.

3. Вычислите $z=x + y$.

4. Вычислите $c = \sqrt{z}$.

Результат: длина гипотенузы c .

Очевидно, что приведенный выше алгоритм может быть выполнен любым человеком, знающим основные арифметические действия (операции), а также операции со степенями и радикалами.

В настоящее время смысл слова алгоритм стал значительно шире. Этот термин используется в различных областях науки и техники для точного и однозначного описания последовательности действий, направленных на (предназначенных для) решение определенной задачи. В информатике понятие алгоритма неразрывно связано с понятием исполнителя.

Исполнитель представляет собой объект, который может выполнять определенные команды. Множество таких команд образует систему команд исполнителя.

Например, телевизор может рассматриваться как объект, который выполняет следующие команды: включение, выключение, увеличение громкости, уменьшение громкости, прием канала №1, прием канала №2 и т.п. Соответствующие команды передаются телевизору с помощью кнопок пульта управления. Аналогичным образом компьютер представляет собой объект, который выполняет следующие команды: сложение, вычитание, умножение, деление и сравнение чисел, чтение данных с клавиатуры, вывод данных на экран, запись данных на магнитный диск, чтение данных с оптического диска и т.п.

Точное определение исполнителя включает:

- 1) описание системы (набора) команд, которые умеет выполнять («знает») исполнитель;
- 2) описание среды, в которой работает исполнитель.

В дальнейшем мы будем изучать исполнителей **Кенгуренок** и **Муравей**, разработанных в учебных целях для школ нашей страны.

Исполнитель Кенгуренок

Этот исполнитель представляет собой компьютерную программу, предназначенную для выполнения в операционной системе **Windows**. Окно приложения **Кенгуренок** представлено на *рисунке 5.1*.

Сам исполнитель представлен пиктограммой маленького кенгуру, который может выполнять следующие команды:

ШАГ – кенгуру перемещается на одну клетку, рисуя при этом соответствующий отрезок прямой;

ПРЫЖОК – кенгуру перемещается на одну клетку, но ничего не рисует;

ПОВОРОТ – кенгуру поворачивается на 90° по часовой стрелке.

Окно приложения **Кенгуренок** состоит из следующих элементов:

1) **Строка меню**, включающая в себя стандартные пункты меню Файл, Правка, Команды, Опции, Помощь.

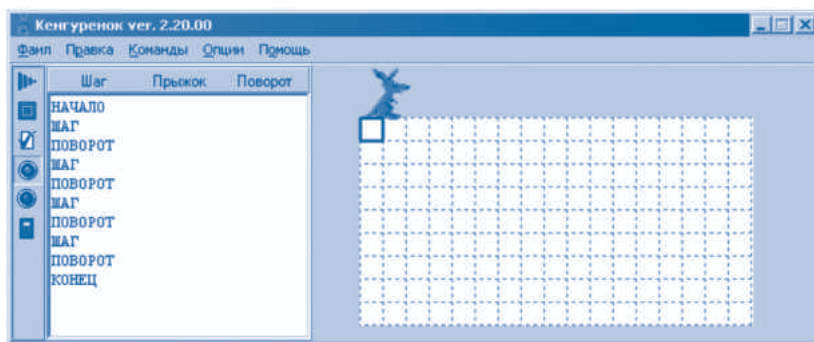


Рис. 5.1. Окно приложения **Кенгуренок**

2) **Центр управления**, содержащий кнопки Шаг, Прыжок, Поворот, Выполнить, Остановить, Проверить, Ручное выполнение, Автоматическое выполнение, Выход.

3) **Область редактирования** программ.

4) **Рабочая среда** Кенгуру, которая представляет собой прямоугольное поле, разбитое на клетки.

Будем различать два режима управления исполнителями: ручное управление и программное управление.

Режим ручного управления предполагает отдельный ввод каждой команды и немедленное ее выполнение исполнителем.

Например, предположим, что мы хотим нарисовать квадрат определенных размеров. В случае ручного управления пользователь должен нажимать кнопки ШАГ, ПОВОРОТ и ПРЫЖОК Панели управления таким образом, чтобы Кенгуренок рисовал соответствующий квадрат.

Режим программного управления предполагает предварительное сохранение определенной последовательности команд и их выполнение в автоматическом режиме, без вмешательства пользователя.

В случае программного управления пользователь должен ввести в область редактирования программ последовательность предписаний (команд), которые обеспечивают рисование квадрата (рис. 5.1). Отметим, что начало и конец каждой программы указываются при помощи вспомогательных слов НАЧАЛО и КОНЕЦ.

Программа представляет собой алгоритм, записанный на языке исполнителя. Процесс написания (разработки) программ называется программированием.

Очевидно, что программы можно сохранять на диске и использовать их многократно. В случае приложения **Кенгуренок** сохранение и открытие программ осуществляется при помощи команд, сгруппированных в меню Файл.

В общем случае, программы для исполнителей пишутся при помощи специальных языков, называемых **языками программирования**. Эти языки содержат предписания, называемые операторами, и вспомогательные слова. Обычно команды исполнителя являются одновременно и операторами языка программирования. Например, язык программирования исполнителя **Кенгуренок** содержит операторы ШАГ, ПРЫЖОК, ПОВОРОТ и вспомогательные слова НАЧАЛО и КОНЕЦ.

Исполнитель Муравей

В этом приложении исполнитель представлен изображением муравья, который может перемещаться по полю, разлинованному на клетки (рис. 5.2). В некоторые из клеток могут быть вписаны печатаемые символы. Обычно требуется разработать программу, которая располагает эти символы в определенном порядке.

Исполнитель может выполнять команды ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО, которые перемещают муравья из текущей в одну из соседних клеток. Если в соответствующей клетке находится какой-либо символ, он будет сдвинут, когда это возможно, по направлению движения. Начало и конец программы указываются при помощи вспомогательных слов НАЧАЛО и КОНЕЦ.

Окно приложения **Муравей** (рис. 5.2) содержит следующие элементы:

- 1) **Строка меню**, содержащая стандартные пункты меню Файл, Программа, Конфигурация, Помощь.
- 2) **Кнопки**: Новый, Открыть, Сохранить, Выход, Синтаксический анализ, Выполнить, Остановить.
- 3) **Область редактирования** программ.
- 4) **Рабочая среда** Муравья, которая представляет собой прямоугольное поле, разлинованное на клетки.
- 5) **Центр управления**, содержащий кнопки ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО, обозначенные стрелками.
- 6) **Поле вставки** печатаемых символов.
- 7) **Кнопки**: НАЧАЛО, КОНЕЦ, ВВЕРХ, ВНИЗ, ..., ПРОЦЕДУРА, предназначенные для упрощения процесса редактирования программ.

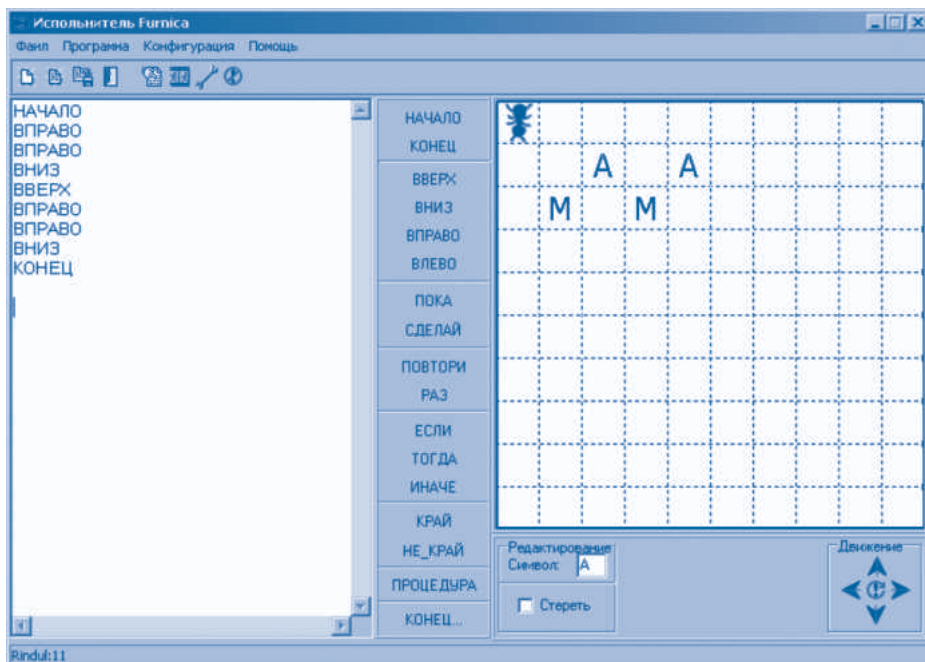
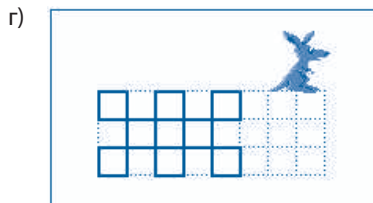
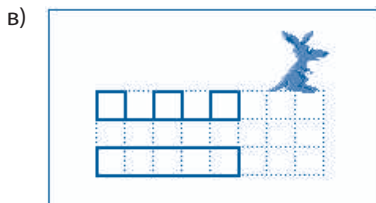
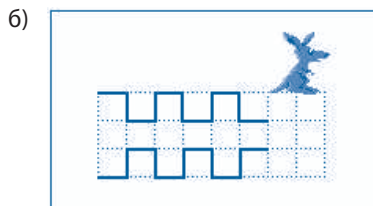
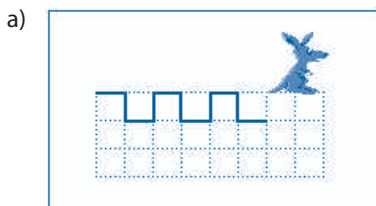


Рис. 5.2. Окно приложения **Муравей**

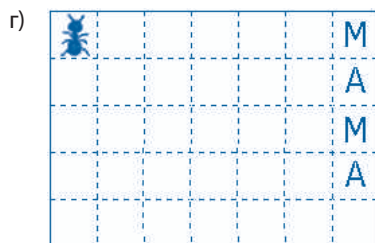
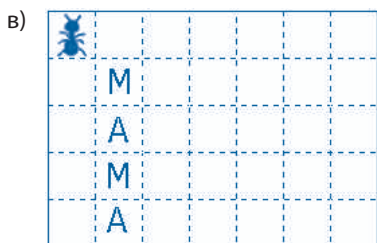
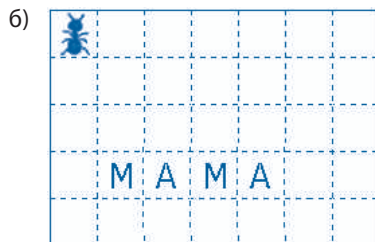
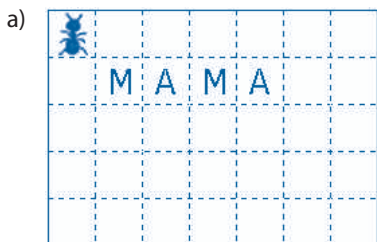
13 Напишите программы для рисования приведенных ниже фигур:



Сохраните эти программы на магнитном диске.

14 Запустите на выполнение приложение **Муравей**. Найдите при помощи системы подсказки назначение всех элементов управления окна этого приложения.

15 Выполняя команды ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО, упорядочьте символы *рисунка 5.2* по приведенным ниже образцам:



16 Напишите программы, предназначенные для упорядочения символов на *рисунке 5.2* в соответствии с образцами, представленными в упражнении 15. Сохраните эти программы на магнитном диске.

17 Изобразите на рисунке фигуру, которую нарисует исполнитель **Кенгуренок** в результате выполнения приведенной ниже программы:

1

```

НАЧАЛО
ШАГ
ШАГ
ПОВОРОТ
ШАГ
    
```

2

```

ШАГ
ПОВОРОТ
ШАГ
ШАГ
ПОВОРОТ
    
```

3

```

ШАГ
ШАГ
ПОВОРОТ
КОНЕЦ
    
```

- 18 Изобразите на рисунке размещение букв рисунка 5.2 после выполнения следующей программы:

1	2	3
НАЧАЛО	ВНИЗ	ВЛЕВО
ВНИЗ	ВПРАВО	ВЛЕВО
ВПРАВО	ВПРАВО	ВЛЕВО
ВПРАВО	ВВЕРХ	ВВЕРХ
ВЛЕВО	ВПРАВО	ВПРАВО
ВЛЕВО	ВВЕРХ	ВПРАВО
ВНИЗ	ВНИЗ	КОНЕЦ

- 19 Какую роль играет центр управления в ходе выполнения программ?
20 Кто дает команды исполнителю в режиме ручного управления? А в режиме программного управления?

5.2. СУБАЛГОРИТМЫ

Ключевые термины:

- подпрограмма
- главная программа
- процедура
- вызов процедуры (или обращение к процедуре)
- метод последовательных уточнений

Как и в случае языков, используемых людьми, язык программирования описывается его синтаксисом и семантикой. Известно, что синтаксис – это набор правил, которые описывают структуру предложений, а семантика – это набор правил, описывающих смысл соответствующих предложений. Отметим, что в случае языков программирования эквивалентом предложения является программа.

Синтаксис программ может быть описан при помощи **форматов**, которые представляют собой образцы (модели, шаблоны) для написания операторов, символов и вспомогательных слов. Например, для исполнителей **Кенгуренок** и **Муравей** программы, которые приводились выше, имели следующий формат:

```
НАЧАЛО
Оператор_1
Оператор_2
...
Оператор_k
КОНЕЦ
```

где *Оператор_k*, $k = 1, 2, 3$ и т.д. может быть любой из команд ШАГ, ПРЫЖОК, ПОВОРОТ для исполнителя **Кенгуренок** и ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО – для исполнителя **Муравей**.

Анализ многих практически важных задач позволяет выявить тот факт, что большинство из них содержат последовательности операторов, выполняющих одни и те

же операции (действия). Например, предположим, что необходимо нарисовать восемь квадратов, как это показано на *рисунке 5.3*.

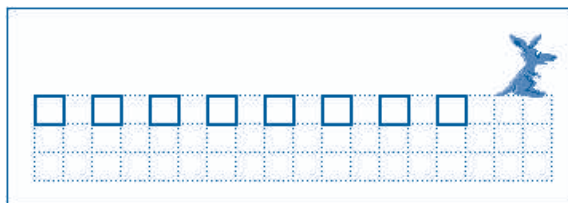


Рис. 5.3. Рисование квадратов

Решая эту задачу «в лоб», можно написать программу, в которой каждый квадрат будет получен путем рисования всех сторон при помощи операторов ШАГ и ПОВОРОТ. Очевидно, что такая программа получится очень длинной, а ее написание будет очень утомительным. Более элегантное решение состоит в написании вспомогательной программы для рисования квадрата требуемых размеров и последующем использовании этой программы для рисования произвольного числа квадратов. В таких случаях вспомогательная программа называется **подпрограммой**, а программа, которая к ней обращается (еще можно сказать – «ее вызывает») – **главной программой** или, короче, программой.

В случае исполнителей **Кенгуренок** и **Муравей**, подпрограммы называются **процедурами** и имеют следующий формат:

```
ПРОЦЕДУРА Имя
Оператор_1
Оператор_2
...
Оператор_k
КОНЕЦ ПРОЦЕДУРЫ
```

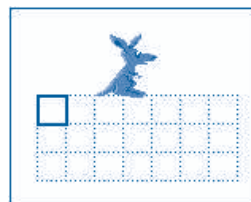
Имя процедуры представляет собой последовательность, состоящую из букв и цифр, начинающуюся с буквы. Слово ПРОЦЕДУРА – не оператор, а вспомогательное слово, которое сообщает центру управления имя процедуры и указывает место в тексте программы, где начинается ее описание. Окончание описания процедуры указано при помощи вспомогательных слов КОНЕЦ ПРОЦЕДУРЫ. Операторы, расположенные между вспомогательными словами ПРОЦЕДУРА и КОНЕЦ ПРОЦЕДУРЫ, образуют **тело процедуры**.

Примеры:

Процедура Квадрат

```
ПРОЦЕДУРА Квадрат
ШАГ
ПОВОРОТ
ШАГ
ПОВОРОТ
ШАГ
```

Получаемая фигура



```
ПОВОРОТ
ШАГ
ПОВОРОТ
ПРЫЖОК
ПРЫЖОК
КОНЕЦ ПРОЦЕДУРЫ
```

Процедура Квадрат рисует квадрат с начальным положением Кенгуренка в верхнем левом углу. Конечное положение Кенгуренка было выбрано таким образом, чтобы оно совпадало с начальным положением следующего квадрата (рис. 5.3).

Операторы, входящие в состав процедур, будут выполняться только тогда, когда в ходе выполнения главной программы встретится оператор **вызова процедуры**. Этот оператор имеет следующий формат:

```
ВЫПОЛНИТЬ Имя
```

где слово *Имя* представляет собой имя вызываемой процедуры.

Когда Центр управления встречает оператор вызова процедуры, выполнение главной программы приостанавливается и начинается выполнение операторов, входящих в состав вызываемой процедуры. После завершения выполнения операторов процедуры Центр управления возвращается к выполнению главной программы, а именно к оператору следующему за оператором вызова соответствующей процедуры. Следовательно, программа Восемь квадратов, предназначенная для рисования квадратов рисунка 5.3, имеет вид:

```
НАЧАЛО
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
КОНЕЦ
```

Очевидно, что главной программе должны быть известны описания всех процедур, которые будут в ней вызываться. В зависимости от типа исполнителя, соответствующие процедуры могут быть включены в основную программу или записаны в отдельных файлах на магнитных дисках.

Для исполнителей **Кенгуренок** и **Муравей**, процедуры включаются в начало основной программы. Для этих исполнителей общий формат программы имеет вид:

```
ПРОЦЕДУРА Имя_1
...
КОНЕЦ ПРОЦЕДУРЫ
ПРОЦЕДУРА Имя_2
...
КОНЕЦ ПРОЦЕДУРЫ
...
```

```

НАЧАЛО
Оператор_1
Оператор_2
...
Оператор_k
КОНЕЦ

```

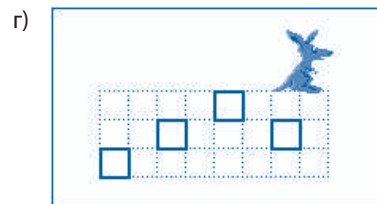
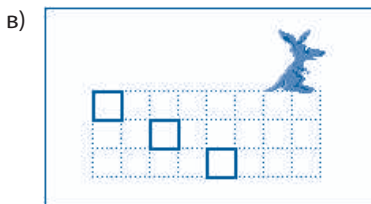
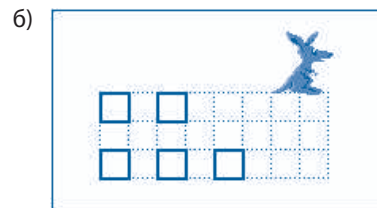
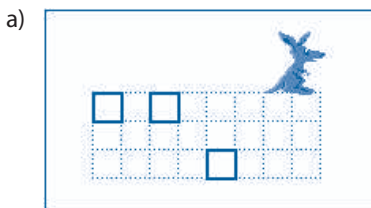
Из представленного формата видим, что программа состоит из описания подпрограмм *Имя_1*, *Имя_2* и т.д., за которыми следуют операторы главной программы, заключенные между вспомогательными словами НАЧАЛО и КОНЕЦ. Вспомогательные слова и операторы, заключенные между ними, образуют тело программы. Следовательно, программа состоит из **описаний подпрограмм** и собственно **тела программы**.

Вспомним, что программа представляет собой алгоритм, записанный на языке исполнителя. Ясно, что классификация программ на подпрограммы и главные программы применима и для алгоритмов. Как правило, для решения определенной задачи пользователь разрабатывает требуемые алгоритмы и субалгоритмы, создавая соответствующие программы и подпрограммы для каждого конкретного типа исполнителя.

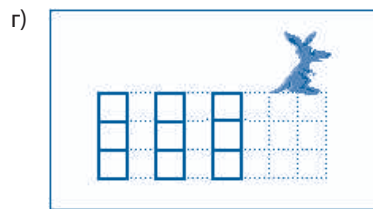
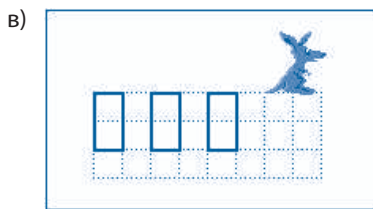
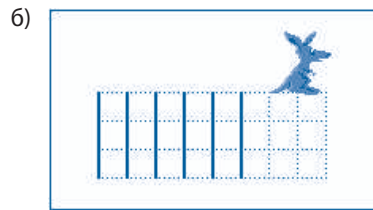
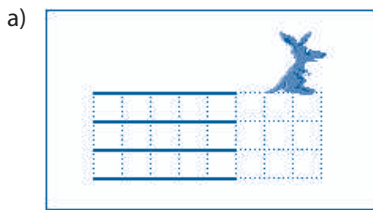
В общем случае, подпрограммы представляют собой субалгоритмы, предназначенные для решения отдельных задач, часто встречающихся при разработке главного алгоритма. Метод решения сложных задач путем разделения их на более простые задачи называется **методом последовательных уточнений**. Например, в случае *рисунка 5.3* начальная задача – рисование восьми квадратов – была разделена на восемь идентичных подзадач, а именно, – рисование одного квадрата.

Вопросы и упражнения

- ❶ Когда появляется необходимость использования субалгоритмов? Приведите примеры.
- ❷ В чем разница между алгоритмом и субалгоритмом?
- ❸ Каков формат процедур, записанных на языке исполнителей **Кенгуренок** и **Муравей**? Как вызываются эти процедуры?
- ❹ Используя процедуру **Квадрат**, разработайте программы для рисования следующих фигур:



- 5 Найдите на приведенных рисунках повторяющиеся фрагменты. Напишите процедуры для рисования этих фрагментов.



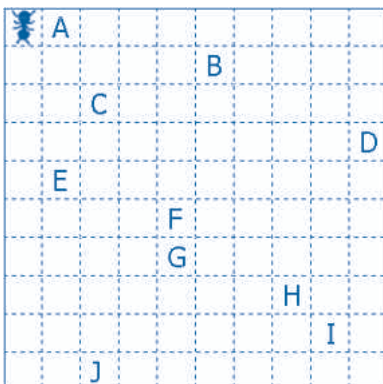
Используя разработанные процедуры, напишите программы для рисования представленных на рисунках фигур.

- 6 Используя метод последовательных уточнений, напишите программы рисования фигур *в* и *г* из упражнения 12 параграфа 5.1.
- 7 Напишите на языке исполнителя **Муравей** следующие процедуры:
 Влево_5 – перемещение Муравья влево на пять позиций;
 Вправо_5 – перемещение Муравья вправо на пять позиций;
 Вверх_5 – перемещение Муравья вверх на пять позиций;
 Вниз_5 – перемещение Муравья вниз на пять позиций.

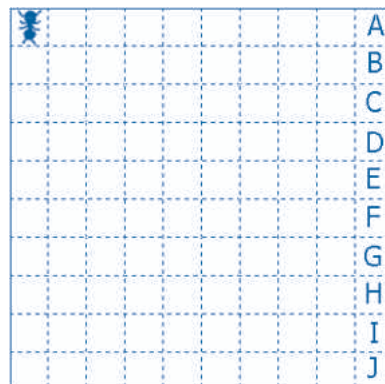
Используя разработанные процедуры, напишите программу, которая перемещает Муравья таким образом, чтобы траектория движения представляла квадрат со сторонами а) 5×5 ; б) 6×6 ; в) 8×8 .

- 8 На приведенных ниже фигурах представлен начальный и конечный вид рабочего поля исполнителя **Муравей**.

Начальный вид рабочего поля



Конечный вид рабочего поля







Напишите процедуру, которая перемещает печатаемый символ текущей строки в последнюю ячейку этой же строки. Напишите программу, которая перемещает все символы в последний столбец поля.

5.3. ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ. ЦИКЛ СО СЧЕТЧИКОМ

Ключевые термины:

- блок-схема
- линейный алгоритм
- составной оператор
- простой оператор
- цикл со счетчиком
- циклический алгоритм

С целью наглядного представления, алгоритмы могут быть описаны при помощи блок-схем. **Блок-схема** представляет собой рисунок, состоящий из следующих графических обозначений:

-  – точка запуска процесса выполнения алгоритма;
-  – точка остановки процесса выполнения алгоритма;
-  – выполнение указанного оператора;
-  – вызов указанного субалгоритма;

→ – стрелка, указывающая порядок, в котором должны выполняться операторы алгоритма.

В качестве примера на *рисунке 5.4* представлена блок-схема процедуры Квадрат и блок-схема программы Восемь_квадратов.

Из анализа *рисунка 5.4* видим, что процесс выполнения алгоритма можно представить в виде воображаемого перехода от одного графического символа блок-схемы к другому в направлении, указанном стрелками.

Алгоритмы, операторы которых выполняются в порядке их появления в тексте, называются линейными.

Очевидно, в случае линейных алгоритмов, воображаемый путь от графического символа СТАРТ до графического символа СТОП, представляет собой линию без самопересечений (*рис. 5.4*).

В процессе разработки алгоритмов было установлено, что очень часто определенные последовательности операторов должны выполняться многократно. Например, в случае процедуры Квадрат (*рис. 5.4a*) последовательность операторов ШАГ, ПОВОРОТ выполняется четыре раза, а оператор вызова процедуры в программе Восемь_квадратов – восемь раз. В таких случаях для упрощения процесса разработки алгоритмов можно воспользоваться оператором ПОВТОРИ. Формат и блок-схема названного оператора представлены на *рисунке 5.5*.

В описании оператора ПОВТОРИ *n* представляет число повторений, а ПОВТОРИ, РАЗ, КОНЕЦ ПОВТОРА являются вспомогательными словами.

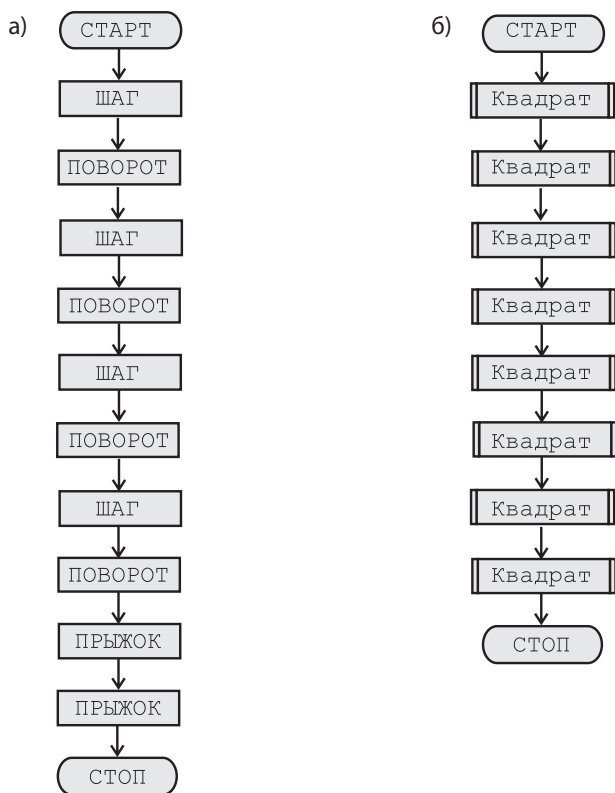


Рис. 5.4. Блок-схемы:

а – процедуры Квадрат; б – программы Восемь_квадратов

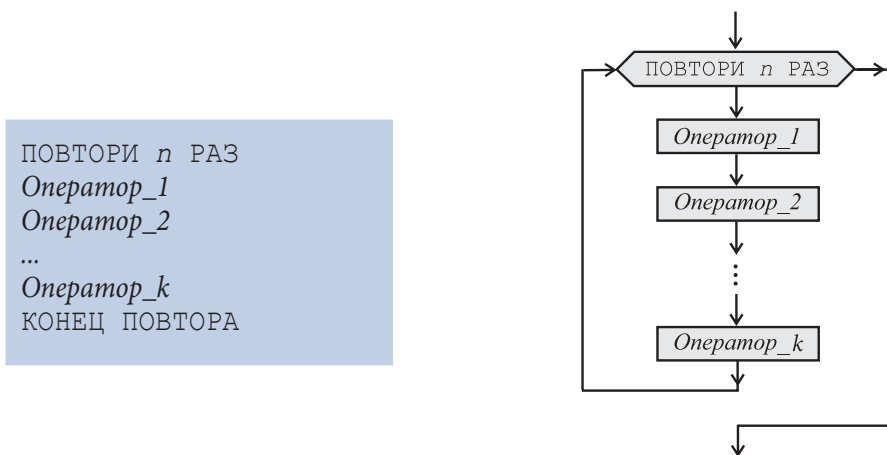


Рис. 5.5. Формат и блок-схема оператора ПОВТОРИ

Оператор ПОВТОРИ записывается в нескольких строках и включает в себя другие операторы. Операторы такого вида называются **составными операторами**, в отличие

от **простых операторов** ШАГ, ПРЫЖОК, ПОВОРОТ, ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО, вызов процедуры, рассмотренных в предыдущих параграфах.

В ходе выполнения оператора ПОВТОРИ Центр управления выполняет n раз группу операторов, расположенных между соответствующими вспомогательными словами. С помощью оператора ПОВТОРИ программы, содержащие последовательности многократно повторяемых операторов, можно переписать в более компактной форме. Например, программа Восемь_квадратов может быть переписана в следующем виде:

1

```
ПРОЦЕДУРА Квадрат
ПОВТОРИ 4 РАЗ
ШАГ
ПОВОРОТ
КОНЕЦ ПОВТОРА
ПРЫЖОК
ПРЫЖОК
КОНЕЦ ПРОЦЕДУРЫ
```

2

```
НАЧАЛО
ПОВТОРИ 8 РАЗ
ВЫПОЛНИТЬ Квадрат
КОНЕЦ ПОВТОРА
КОНЕЦ
```

Оператор ПОВТОРИ n РАЗ называется **цикл со счетчиком**, поскольку при его выполнении одна и та же последовательность операторов повторяется многократно, а само число повторений n известно в момент написания программы. Группа операторов, заключенная между вспомогательными словами ПОВТОРИ и КОНЕЦ ПОВТОРА, называется **телом цикла**.

Алгоритмы, которые содержат последовательности многократно выполняемых операторов, называются циклическими.

Блок-схемы, представляющие в графическом виде процессы выполнения процедуры Квадрат и программы Восемь_квадратов, представлены на *рисунке 5.6*. В этих блок-схемах используется графический символ ПОВТОРИ, из которого, в отличие от ранее изученных символов, выходят две стрелки: первая направлена к операторам тела цикла, а вторая – к оператору, который будет выполняться непосредственно после окончания цикла.

Из анализа блок-схем можно заметить, что в случае циклических алгоритмов воображаемый путь от символа СТАРТ до символа СТОП представляет собой линию, содержащую как минимум одну петлю. Эта петля включает в себя графический символ ПОВТОРИ и все символы, соответствующие операторам тела цикла (*рис. 5.6*).

В общем случае, оператор ПОВТОРИ может содержать в своем теле другие составные операторы, образуя, таким образом, вложенную структуру. Следовательно, относительно короткие программы могут описывать очень длинные последовательности требуемых операций. В качестве примера приведем программу, которая заставляет Кенгуренка переместиться 100 раз вдоль верхнего края рабочего поля:

```
НАЧАЛО
ПОВТОРИ 100 РАЗ
ПОВТОРИ 15 РАЗ
ПРЫЖОК
КОНЕЦ ПОВТОРА
```

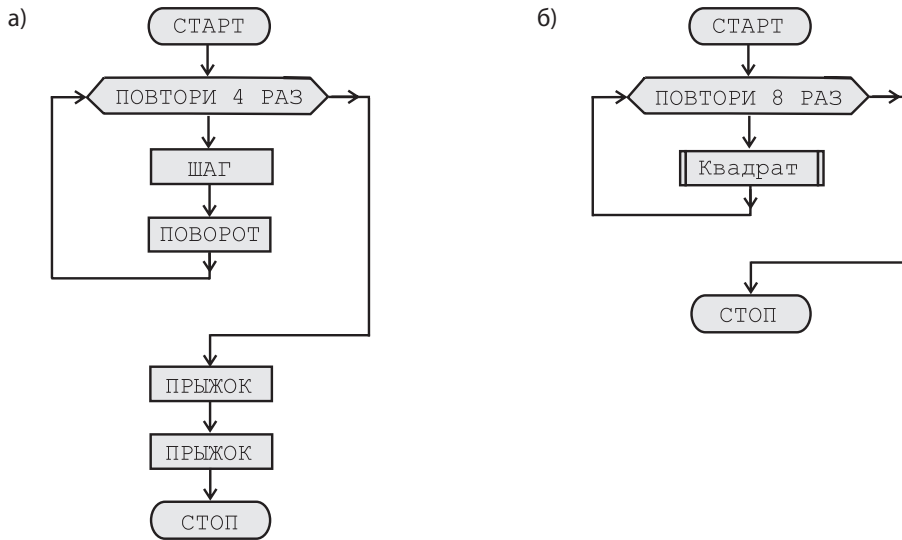


Рис. 5.6. Блок-схемы циклических алгоритмов:
 а – процедуры Квадрат; б – программы Восемь_квадратов

ПОВОРОТ
 ПОВОРОТ
 КОНЕЦ ПОВТОРА
 КОНЕЦ

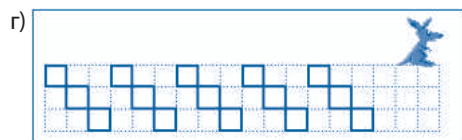
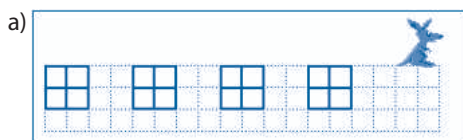
Вопросы и упражнения

1 Объясните смысл следующих графических элементов:

- а) б) в) г) д) е) →

Каков смысл стрелок в блок-схемах?

- 2 Нарисуйте блок-схемы алгоритмов рисования фигур из упражнений 4, 5 и 8 из параграфа 5.2.
- 3 Покажите на блок-схемах рисунка 5.4 воображаемый путь, представляющий процесс выполнения соответствующих алгоритмов.
- 4 Объясните термин *линейный алгоритм*. Приведите примеры.
- 5 Каков формат оператора ПОВТОРИ? Приведите несколько примеров использования этого оператора.
- 6 Как Вы думаете, когда применяется оператор ПОВТОРИ? Приведите примеры.
- 7 Объясните термин *циклический алгоритм*. Приведите примеры.
- 8 Покажите на блок-схемах рисунка 5.6 воображаемый путь, представляющий процесс выполнения соответствующих алгоритмов.
- 9 Покажите на блок-схемах рисунка 5.6 петли, представляющие циклы со счетчиком.
- 10 Перечислите известные Вам простые операторы. В чем отличие простого оператора от составного оператора?
- 11 Разработайте циклические алгоритмы для рисования следующих фигур:



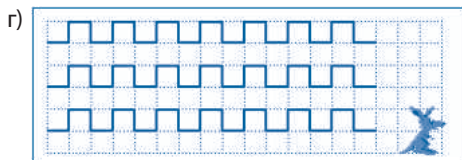
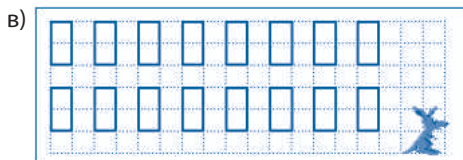
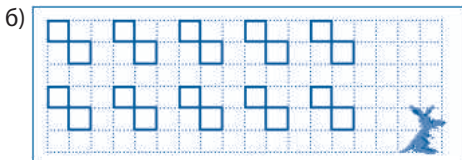
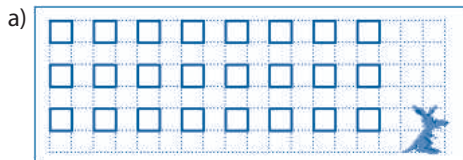
Подсчитайте, сколько команд выполнит Кенгуренок в процессе рисования этих фигур.

- 12 Сколько команд выполнит Кенгуренок в ходе выполнения следующих программ:

а) НАЧАЛО
ПОВТОРИ 10 РАЗ
ПРЫЖОК
ПОВОРОТ
ПОВОРОТ
ПРЫЖОК
КОНЕЦ ПОВТОРА
КОНЕЦ

б) НАЧАЛО
ПОВТОРИ 10 РАЗ
ПОВТОРИ 20 РАЗ
ПРЫЖОК
ПОВОРОТ
ПОВОРОТ
ПРЫЖОК
КОНЕЦ ПОВТОРА
КОНЕЦ ПОВТОРА
КОНЕЦ

- 13 Используя вложенные циклы, разработайте циклические алгоритмы для рисования следующих фигур:



Подсчитайте, сколько команд выполнит Кенгуренок при рисовании этих фигур.

- 14 Запишите на языке исполнителя **Муравей** следующие циклические субалгоритмы:

Влево_8 – перемещает Муравья влево на восемь позиций;

Вправо_8 – перемещает Муравья вправо на восемь позиций;

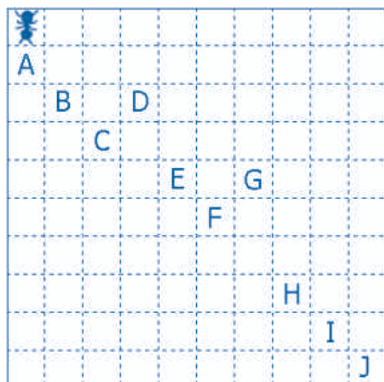
Вверх_8 – перемещает Муравья вверх на восемь позиций;

Вниз_8 – перемещает Муравья вниз на восемь позиций.

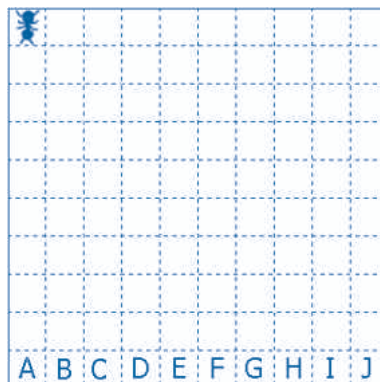
Используя эти субалгоритмы, напишите программу, которая перемещает Муравья таким образом, что траектория его движения будет представлять квадрат со сторонами: а) 8×8 ; б) 9×9 ; в) 10×10 .

- 15 На приведенных ниже фигурах показаны начальный и конечный вид рабочего поля исполнителя **Муравей**. Используя циклы со счетчиком, напишите процедуру, которая перемещает печатаемый символ из текущего столбца в последнюю ячейку столбца.

Начальный вид рабочего поля



Конечный вид рабочего поля



Используя циклы со счетчиком, напишите программу, которая сдвинет все печатаемые символы в последнюю строку поля.

5.4. ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ. ЦИКЛ С УСЛОВИЕМ

Ключевые термины:

- сенсор
- условие
- цикл с условием
- алгоритм с обратной связью
- ошибка выполнения (отказ)

До настоящего момента все алгоритмы, разработанные нами для исполнителей **Кенгуренок** и **Муравей**, не анализировали ситуацию рабочей среды, т.е. условия на рабочем поле. Другими словами, Центр управлял действиями исполнителей без проверки того факта, обеспечивают ли соответствующие команды достижение поставленной цели или, в состоянии ли исполнитель выполнить полученные от Центра команды. Такой способ разработки алгоритмов существенно усложняет решение многих задач, встречающихся в повседневной практике.

Например, предположим, что начальное положение Муравья нам неизвестно. Требуется разработать алгоритм, который перемещает Муравья в верхний левый угол рабочего поля. Очевидно, что изученные нами ранее операторы не позволяют решить рассматриваемую задачу, так как не ясно, сколько команд ВВЕРХ, ВЛЕВО должен получить исполнитель.

Для того чтобы избежать возникновения таких ситуаций, исполнители снабжены **сенсорами***, которые передают в Центр управления определенную информацию о рабочей

* Сенсор – устройство, обнаруживающее определенное явление.

среде. Например, исполнители **Кенгуренок** и **Муравей** снабжены сенсорами, которые обнаруживают наличие препятствий по ходу предполагаемого движения. Данные от сенсоров передаются Центру управления посредством логических переменных ЛИНИЯ и КРАЙ, которые могут принимать значения ЛОЖЬ или ИСТИНА.

В случае циклических алгоритмов, анализ ситуации в рабочей среде осуществляется при помощи составного оператора ПОКА. Этот оператор обеспечивает циклическое выполнение группы операторов до тех пор, пока ситуация в рабочей среде удовлетворяет определенным условиям. Формат и блок-схема оператора ПОКА представлены на рисунке 5.7.

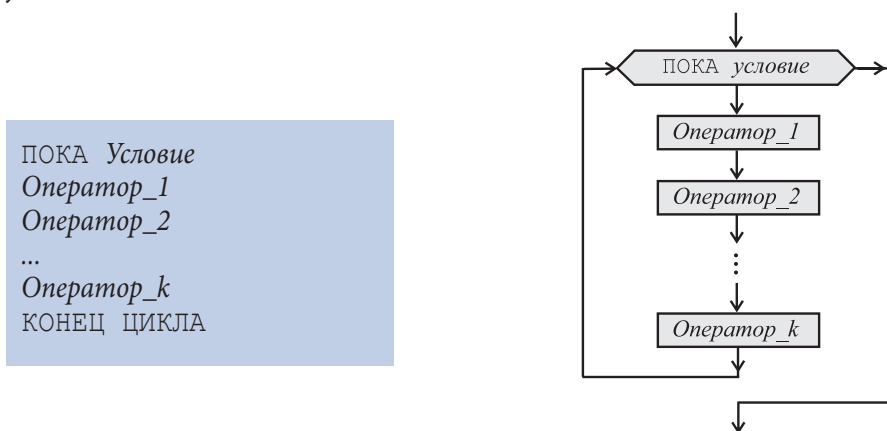


Рис. 5.7. Формат и блок-схема оператора ПОКА

В описании оператора ПОКА *Условие* является логическим выражением, а слова ПОКА и КОНЕЦ ЦИКЛА – вспомогательными словами.

Условия представляют собой логические выражения, принимающие значение Истина при наличии определенных ситуаций в рабочей среде исполнителя.

В языках программирования исполнителей **Кенгуренок** и **Муравей** логические выражения, используемые для выявления определенных ситуаций в рабочей среде, очень просты:

Кенгуренок

ЛИНИЯ
КРАЙ
НЕ ЛИНИЯ
НЕ КРАЙ

Муравей

КРАЙ
НЕ КРАЙ

Для более сложных исполнителей в состав условий могут входить переменные, которые сообщают Центру управления информацию о направлении и скорости движения других объектов, наличии препятствий, температуре, влажности, уровне радиации и т.п., а логические выражения могут быть построены при помощи операций отношения =, ≠, >, ≥, <, ≤ и логических операций НЕ, И, ИЛИ.

Для примера, рассмотрим следующую задачу. Предположим, что необходимо переместить Кенгуренка на один из краев рабочего поля, причем его начальное положение неизвестно. Очевидно, что в данном случае решить задачу с помощью оператора ПОВТО-

РИ невозможно, так как количество шагов, которые должен сделать Кенгуренок, заранее неизвестно. С помощью оператора ПОКА рассматриваемая задача решается очень легко:

```
ПОКА НЕ КРАЙ
ПРЫЖОК
КОНЕЦ ЦИКЛА
```

В процессе выполнения оператора ПОКА Центр управления, в первую очередь, вычисляет условие НЕ КРАЙ. Если указанное условие имеет значение ИСТИНА, выполняются операторы тела цикла, в случае рассматриваемого примера – оператор ПРЫЖОК. После выполнения оператора ПРЫЖОК сенсор Кенгуренка исследует соседнюю клетку и обновляет значение переменной КРАЙ. Если условие НЕ КРАЙ принимает значение ЛОЖЬ, выполнение цикла завершается и осуществляется переход к оператору, следующему непосредственно за вспомогательными словами КОНЕЦ ЦИКЛА.

Оператор ПОКА называется **циклом с условием**, поскольку он обеспечивает многократное выполнение содержащейся в нем группы операторов. Количество повторений устанавливается в ходе выполнения программы в зависимости от текущих значений указанного условия.

Реализация оператора ПОКА предполагает существование между исполнителем и Центром управления двух каналов передачи информации:

- 1) прямого канала, предназначенного для передачи команд от Центра управления к исполнителю;
- 2) обратного канала, предназначенного для передачи информации, собранной сенсорами исполнителя, в Центр управления.

Алгоритмы, содержащие группы операторов, выполнение которых зависит от информации в рабочей среде исполнителя, называются алгоритмами с обратной связью.

Отметим, что большинство алгоритмов, используемых в современной информатике, являются алгоритмами с обратной связью, что позволяет исполнителям адаптироваться (приспосабливаться) к конкретным ситуациям рабочей среды. В качестве примера рассмотрим очень маленькую программу, предназначенную для рисования спирали (рис. 5.8):

```
НАЧАЛО
ПОКА НЕ ЛИНИЯ
ПОКА НЕ ЛИНИЯ
ШАГ
КОНЕЦ ЦИКЛА
ПОВОРОТ
КОНЕЦ ЦИКЛА
КОНЕЦ
```

Внутренний цикл, телом которого является оператор ШАГ, рисует отрезок прямой, длина которого зависит от вида ранее нарисованных линий. Как только сенсор Кенгуренка обнаруживает в соседней клетке линию, условие НЕ ЛИНИЯ принимает значение ЛОЖЬ и выполнение этого цикла прекращается. Следующий оператор поворачивает Кенгуренка на 90°, готовя его к проведению новой линии. Внешний цикл обеспечивает проведение линий, образующих спираль до тех пор, пока все соседние клетки окажутся занятыми.

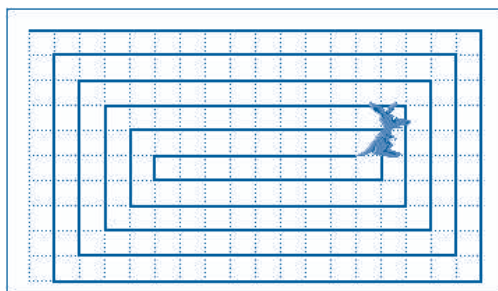


Рис. 5.8. Спираль, нарисованная Кенгуренком

Теоретически спираль можно нарисовать и без использования оператора цикла ПОКА, создавая соответствующие циклы с помощью оператора ПОВТОРИ. Однако в этом случае пользователь должен вычислить «вручную» длину каждой линии спирали и записать по четыре цикла со счетчиком для каждого витка. Очевидно, что при большом числе витков указанная работа становится очень утомительной или даже практически невозможной.

Из практики известно, что при разработке и отладке программ появляются ситуации, когда исполнителю посылаются такие команды, которые он не может выполнить. Например, программа

```
НАЧАЛО  
ПОВТОРИ 1000 РАЗ  
ШАГ  
КОНЕЦ ПОВТОРА  
КОНЕЦ
```

с синтаксической точки зрения является правильной, но ее полное выполнение невозможно, поскольку размеры рабочего поля значительно меньше чем 1000×1000 . В процессе выполнения рассматриваемой программы, независимо от исходного положения, возникает ситуация, при которой Кенгуренок не может в дальнейшем выполнять команду ШАГ, поскольку он уже находится на краю поля. Следовательно, в процессе выполнения приведенной выше программы появляется ошибка выполнения или, другими словами, отказ.

Ошибка выполнения (отказ) появляется в том случае, когда в процессе выполнения программы исполнитель не может выполнить полученную команду.

Реакция Центра управления на отказ зависит от типа исполнителя и возможностей соответствующей программы. Например, исполнители Кенгуренок и Муравей выводят сообщение об ошибке и останавливаются. В случае более сложных исполнителей, отказ понимается как особое условие, которое вызывает запуск специальных субалгоритмов для исправления возможных ошибок. В качестве примера вспомним системы подсказки текстовых редакторов или табличных процессоров, которые в случае возникновения ошибок выводят на экран подсказки, помогающие пользователю правильно вводить соответствующие команды.

5.5. АЛГОРИТМЫ С РАЗВЕТВЛЕНИЯМИ

Ключевые термины:

- разветвление
- алгоритм с разветвлениями

В процессе выполнения программ очень часто возникает необходимость задавать исполнителю команды в зависимости от ситуации в рабочей среде. Мы уже знаем, что информация о ситуации в рабочей среде собирается с помощью сенсоров, причем соответствующие данные представляются в программах в виде переменных **ЛИНИЯ**, **КРАЙ** и т.п. В предыдущем параграфе мы уже использовали эти переменные для записи условий в составных операторах **ПОКА**.

В общем случае, условия, которые описывают ситуацию в рабочей среде, могут быть проанализированы и с помощью другого составного оператора, а именно, – оператора **ЕСЛИ**. Формат и блок-схема оператора **ЕСЛИ** представлены на *рисунке 5.9*.

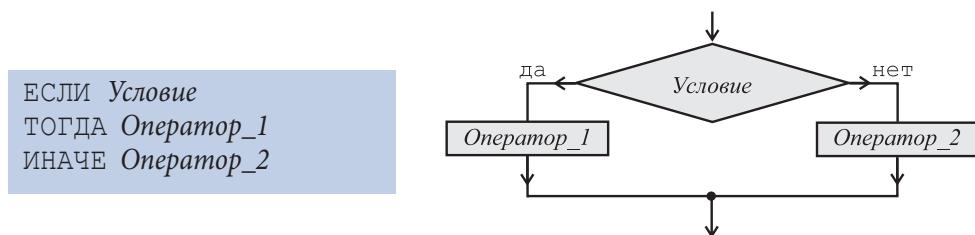


Рис. 5.9. Формат и блок-схема оператора **ЕСЛИ**

В описании оператора **ЕСЛИ** *Условие* – это логическое выражение, а **ЕСЛИ**, **ТОГДА** и **ИНАЧЕ** являются вспомогательными словами.

Как и в случае оператора **ПОКА**, для исполнителей **Кенгуренок** и **Муравей** можно использовать следующие логические выражения:

```
ЛИНИЯ
КРАЙ
НЕ ЛИНИЯ
НЕ КРАЙ
```

Очевидно, что в случае более сложных исполнителей в состав условий могут входить и другие переменные, а логические выражения могут записываться с использованием операторов отношения $=$, \neq , $>$, \geq , $<$, \leq и логических операторов **НЕ**, **И**, **ИЛИ**.

В процессе выполнения оператора **ЕСЛИ**, Центр управления анализирует в первую очередь соответствующее условие. Если это условие имеет значение **ИСТИНА**, то выполняется *Оператор_1*, а в противном случае (условие имеет значение **ЛОЖЬ**) – выполняется *Оператор_2*.

Составной оператор **ЕСЛИ** называется **условным оператором** или **оператором разветвления**, так как воображаемый путь, представляющий процесс выполнения

программы, проходит, в зависимости от текущего значения анализируемого условия, либо через символ *Оператор_1*, либо через символ *Оператор_2*, а ромб представляет место разветвления.

Алгоритмы, содержащие группы операторов, которые выполняются только при определенных значениях заданных условий, называются алгоритмами с разветвлениями.

В качестве примера рассмотрим программу, которая рисует квадрат, стороны которого совпадают с краями рабочего поля:

```
НАЧАЛО  
ПОВТОРИ 100 РАЗ  
ЕСЛИ КРАЙ  
ТОГДА ПОВОРОТ  
ИНАЧЕ ШАГ  
КОНЕЦ ПОВТОРА  
КОНЕЦ
```

Тело цикла ПОВТОРИ из приведенной программы содержит составной оператор ЕСЛИ, который будет выполнен 100 раз. При каждом выполнении оператора ЕСЛИ проверяется условие КРАЙ и, в зависимости от его текущего значения, выполняется либо оператор ПОВОРОТ, либо оператор ШАГ. Очевидно, оператор ПОВОРОТ выполняется только в тех случаях, когда условие принимает значение ИСТИНА, или, другими словами, когда Кенгуренок достигает края рабочего поля.

Блок-схема рассматриваемой программы представлена на *рисунке 5.10*. На блок-схеме видно, что при каждом повторном выполнении оператора ЕСЛИ, выбор одного из двух операторов ПОВОРОТ или ШАГ осуществляется в зависимости от текущего положения Кенгуренка относительно края рабочего поля.

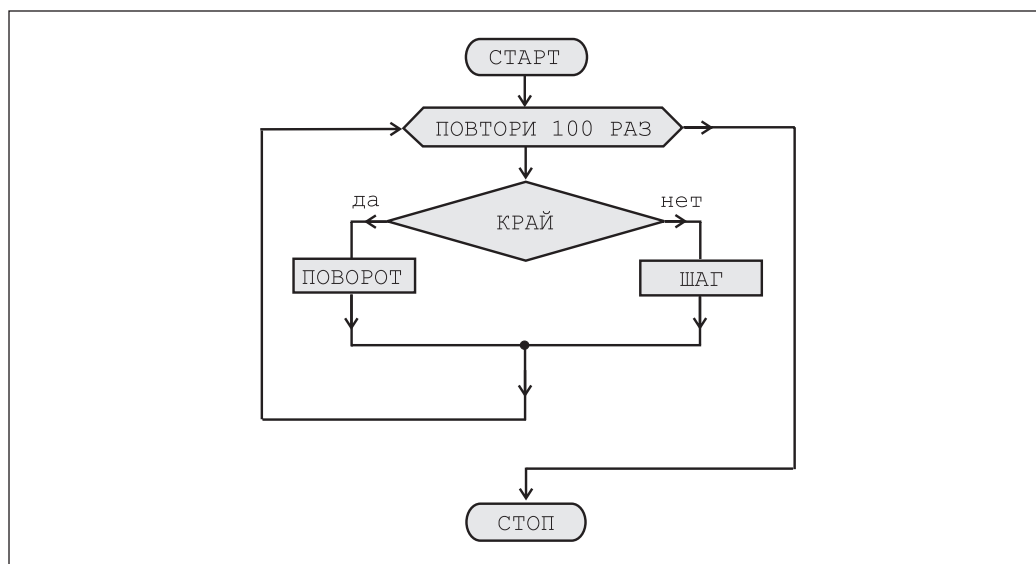


Рис. 5.10. Блок-схема алгоритма с разветвлением

Вопросы и упражнения

- 1 Для чего предназначен оператор ЕСЛИ? Напишите формат и нарисуйте блок-схему этого оператора.
- 2 Как выполняется оператор ЕСЛИ? Можно ли заменить оператор ЕСЛИ оператором ПОКА?
- 3 Как Вы считаете, является ли алгоритм с разветвлениями алгоритмом с обратной связью? Обоснуйте Ваш ответ.
- 4 Назовите как минимум три математические задачи, алгоритмы для решения которых содержат разветвления.
- 5 В чем разница между линейными алгоритмами и алгоритмами с разветвлениями?
- 6 В чем разница между циклическими алгоритмами и алгоритмами с разветвлениями?
- 7 Напишите две программы, которые рисуют спираль внутри прямоугольника, размеры которого заранее не известны (рис. 5.11). Для анализа ситуации на рабочем поле в первой программе должен использоваться оператор ЕСЛИ, а во второй – оператор ПОКА. Как Вы думаете, какая программа является более эффективной: та что использует оператор ЕСЛИ или та что использует оператор ПОКА? Обоснуйте Ваш ответ.

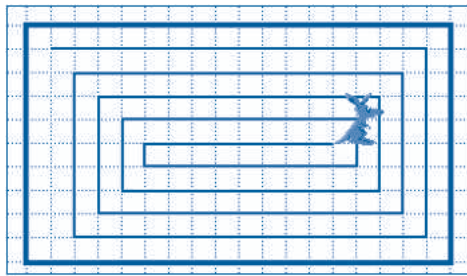


Рис. 5.11. Спираль внутри прямоугольника

- 8 Нарисуйте блок-схемы программ для рисования спирали внутри прямоугольника (рис. 5.11). Покажите на блок-схеме воображаемые пути, ход выполнения программы и точку разветвления.
- 9 Кенгуренок находится в начале коридора шириной в две клетки (рис. 5.12). Разработайте программу, которая рисует линию, проходящую точно посередине коридора. Предполагается, что остальные размеры коридора заранее неизвестны.

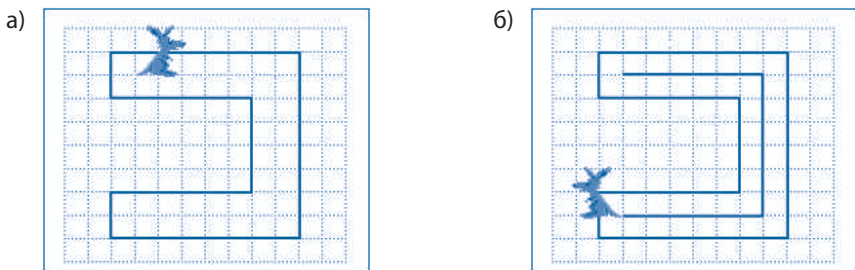


Рис. 5.12. Проведение линии внутри коридора
а – начальное положение; б – конечное положение

5.6. АЛГОРИТМ РАБОТЫ КОМПЬЮТЕРА

Ключевые термины:

- функциональные блоки
- центральное устройство управления
- арифметико-логическое устройство
- часто используемые команды
- алгоритм работы компьютера

Любой компьютер состоит из следующих **функциональных блоков** (рис. 5.13): процессора, внутренней памяти, внешней памяти, устройства ввода и устройства вывода. Процессор, в свою очередь, состоит из **центрального устройства управления и арифметико-логического устройства**.

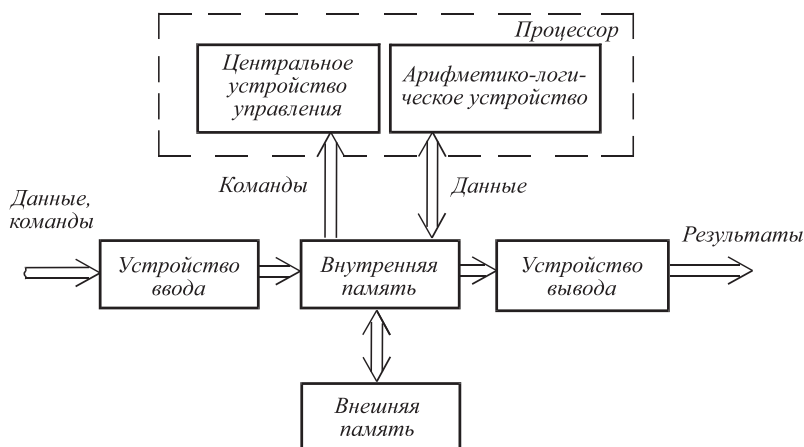


Рис. 5.13. Функциональная схема компьютера

За исключением центрального устройства управления, все функциональные блоки компьютера могут рассматриваться как исполнители, выполняющие определенные наборы команд. В качестве примера в *таблице 5.1* представлены **часто используемые команды** исполнителей, которые входят в состав компьютера.

Центральное устройство управления отдает команды исполнителям в соответствии с программой, записанной во внутренней памяти компьютера. Программа состоит из набора машинных команд, в которых указывается операция, которую необходимо выполнить и местоположение операндов. Команды закодированы с помощью двоичных чисел. Например, в любой арифметической команде указывается операция, которая должна быть выполнена (сложение, вычитание, умножение или деление) и местоположение операндов во внутренней памяти. В команде ввода информации указывается требуемое устройство ввода (клавиатура, сканнер) и место внутренней памяти, куда будет записана введенная информация. В команде вывода информации указывается место во внутренней памяти, содержащее извлекаемые данные, и требуемое устройство вывода (монитор или принтер).

**Часто используемые команды исполнителей
из состава персонального компьютера**

№	Исполнитель	Команды
1.	Клавиатура	Жди нажатия произвольной клавиши Прочти код нажатой клавиши Блокируй клавиатуру
2.	Монитор	Очисти экран Выведи указанный символ Выведи указанный графический элемент Установи цвет фона
3.	Устройство на магнитных дисках	Запиши данные на диск Прочитай данные с диска
4.	Принтер	Напечатай указанный символ Продвинь лист на строку Продвинь страницу
5.	Устройство на оптических дисках	Прочитай данные с диска Выдвинь (извлеки) диск
6.	Внутренняя память	Прочитай данные из указанной ячейки Запиши данные в указанную ячейку
7.	Арифметико-логическое устройство	Сложи Вычти Умножь Подели Сравни И ИЛИ НЕ

Аналогичным образом, в случае работы с магнитным диском (чтение или запись информации), в соответствующей машинной команде указывается место размещения данных во внутренней памяти и требуемое дисковое устройство.

Алгоритм работы центрального устройства управления, а значит, и **компьютера** в целом может быть описан следующим образом:

ПОКА ИЗВЛЕКАЕМАЯ ИЗ ВНУТРЕННЕЙ ПАМЯТИ МАШИННАЯ КОМАНДА \neq СТОП
ИЗВЛЕКИ МАШИННУЮ КОМАНДУ ИЗ ВНУТРЕННЕЙ ПАМЯТИ
ДЕКОДИРУЙ ИЗВЛЕЧЕННУЮ МАШИННУЮ КОМАНДУ
ПЕРЕДАЙ КОМАНДЫ ИСПОЛНИТЕЛЯМ
КОНЕЦ ЦИКЛА

Из приведенного алгоритма видно, что центральное устройство управления реализует принцип программного управления. Он состоит в том, что исполнителям передаются команды, получаемые из машинных команд, которые извлекаются из внутренней памяти компьютера. Подчеркнем, что система (набор) машинных команд современного компьютера включает в себя как команды обработки информации (сложение, вычитание,

деление, умножение и т.п.), так и команды вызова подпрограмм, циклов и разветвления. Такой набор дает возможность компактного написания очень сложных алгоритмов, которые при очень большом быстродействии (сотни миллионов операций в секунду) обеспечивают эффективное использование компьютеров во всех областях современной науки и техники.

Вопросы и упражнения

- 1 Определите набор команд каждого из периферийных устройств (клавиатуры, принтера, сканнера и т.п.), используемых в кабинете информатики.
- 2 Используя технические описания компьютеров из кабинета информатики, определите набор машинных команд компьютера, с которым Вы работаете. Укажите команды обработки, команды ввода/вывода и команды управления.
- 3 Объясните назначение каждого функционального блока, из которых состоит компьютер (рис. 5.13).
- 4 Какую роль играет центральное устройство управления, входящее в состав процессора?
- 5 Разработайте алгоритм функционирования цифрового компьютера. Как Вы думаете, команды какого вида посылает центральное устройство управления исполнителям компьютера?
- 6 От чего зависит производительность компьютера? Обоснуйте Ваш ответ.
- 7 Какие действия совершает центральное устройство управления в процессе выполнения программ? Как можно прервать выполнение программы?
- 8 Как реализуется принцип программного управления в случае цифровых компьютеров?
- 9 Как Вы думаете, для чего предназначены компьютеры, установленные в современных автомобилях? Какие виды периферийных устройств имеются у таких компьютеров? Какие программы выполняются на таких компьютерах?

5.7. ОСНОВНЫЕ СВЕДЕНИЯ ОБ АЛГОРИТМАХ

Ключевые термины:

- представление алгоритмов
- классификация алгоритмов
- свойства алгоритмов
- алгоритмическое мышление
- информационная культура

Для упрощения процесса разработки **алгоритмов** договорились, что они будут **представлены** (описаны, обозначены) при помощи специальных стандартных средств, самыми распространенными из которых являются:

1) обычные языки общения между людьми (например, русский) с использованием, при необходимости, математических формул;

- 2) блок-схемы;
- 3) алгоритмические языки (языки программирования).

Например, алгоритм решения уравнений первой степени $ax + b = 0$ можно описать на русском языке следующим образом:

1. Прочитай с клавиатуры коэффициенты a и b .
2. Если $a \neq 0$, то вычисли корень $x = -b/a$, выведи на экран сообщение «Уравнение имеет один корень» и значение x . СТОП.
3. Если $a = 0$ и $b = 0$, то выведи на экран сообщение «Уравнение имеет бесконечное множество корней». СТОП.
4. Если $a = 0$ и $b \neq 0$, то выведи на экран сообщение «Уравнение не имеет смысла». СТОП.

Основным преимуществом описания алгоритмов на обычном языке состоит в том, что такие описания являются понятными любому человеку, которому знакомы операторы и действия, встречающиеся в тексте алгоритма. К сожалению, современные компьютеры не понимают однозначным образом языки, на которых общаются люди. Следовательно, алгоритмы, написанные на таких языках, не могут использоваться в качестве компьютерных программ.

В случае блок-схем алгоритмы описываются при помощи графических символов *старт*, *стоп*, *оператор*, *вызов субалгоритма*, *повтори*, *пока*, *если* и др. В качестве примера на рисунке 5.14 представлена блок-схема алгоритма решения уравнений первой степени.

Анализ блок-схем позволяет легче **классифицировать алгоритмы** на линейные, циклические и с разветвлениями. Блок-схемы очень наглядны, но как и в случае языка общения между людьми, не могут однозначно восприниматься современными компьютерами. Как следствие, блок-схемы также не могут использоваться в качестве компьютерных программ.

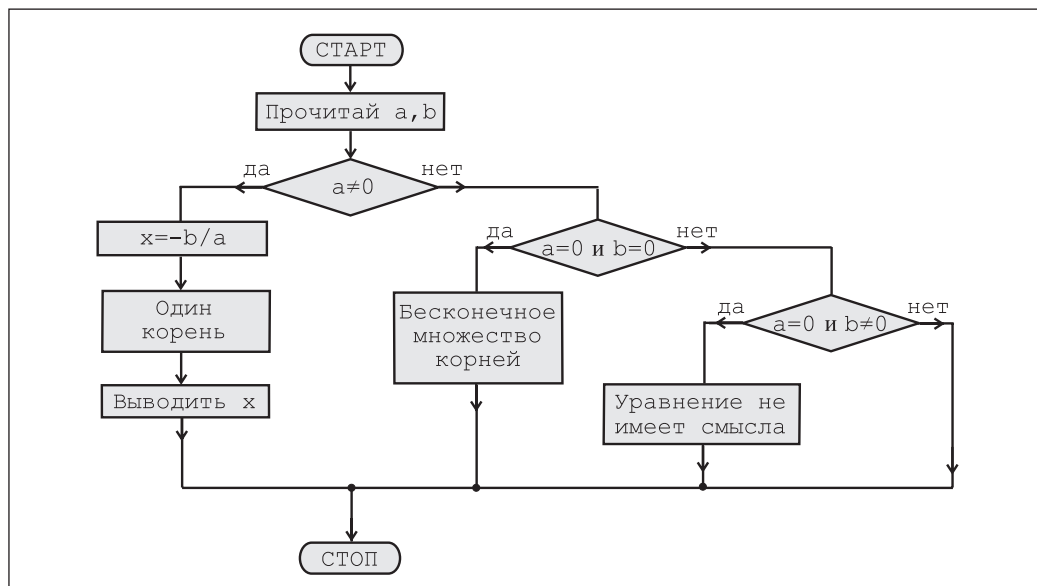


Рис. 5.14. Блок-схема алгоритма решения уравнений первой степени

Алгоритмические языки представляют собой искусственные языки, созданные с целью точного описания алгоритмов, и основаны на использовании строго определенных словаря, синтаксиса и семантики. В качестве примера рассмотрим алгоритм решения уравнений первой степени, записанный на алгоритмическом языке ПАСКАЛЬ:

```
Program Exemplan;
var a, b, x : real;
begin
  readln(a, b);
  if a<>0 then
    begin
      x:=-b/a;
      writeln('Уравнение имеет один корень');
      writeln(x);
    end;
  if (a=0) and (b=0) then
    writeln('Уравнение имеет бесконечное множество корней');
  if (a=0) and (b<>0) then
    writeln('Уравнение не имеет смысла');
end.
```

Английские слова в этом описании имеют следующий смысл:

program – программа;

var – переменная;

begin – начало;

end – конец;

if – если;

then – тогда;

readln – прочитай;

writeln – запиши;

and – и.

Из приведенного примера видно, что алгоритмический язык ПАСКАЛЬ содержит операторы и вспомогательные слова, изученные нами в предыдущих параграфах при разработке программ управления исполнителями **Кенгуренок** и **Муравей**.

Основным преимуществом алгоритмических языков является тот факт, что они обеспечивают однозначное описание алгоритмов, понятны людям и относительно легко переводятся на язык исполнителей. Очевидно изучение определенного алгоритмического языка, как и изучение любого иностранного языка, требует от человека определенных усилий. В школах нашей страны алгоритмический язык ПАСКАЛЬ изучается, начиная с 9 класса.

Языки программирования представляют собой алгоритмические языки, адаптированные (приспособленные) для описания алгоритмов в виде, понятном для исполнителей. Например, программы, разработанные в предыдущих параграфах, были написаны на языках исполнителей **Кенгуренок** и **Муравей**. Этот язык программирования содержит, кроме операторов ПОВТОРИ, ПОКА, ЕСЛИ, вызов процедуры и др., встречающихся во всех алгоритмических языках, и операторы, специфичные именно для данных исполнителей: ШАГ, ПРЫЖОК, ПОВОРОТ, ВВЕРХ, ВНИЗ и т.д.

Независимо от формы представления, **алгоритмы** имеют ряд общих **свойств**, которые отличают их от предписаний, рекомендаций или планов, предназначенных для решения отдельных задач. Установлено, что любой алгоритм должен иметь следующие три свойства:

- 1) **однозначность** – при выполнении алгоритма на каждом шаге должно быть точно известно как выполнить текущую операцию и какая именно операция будет следующей;
- 2) **универсальность** – алгоритм должен быть применим ко всем задачам, для которых он был разработан;
- 3) **конечность** – алгоритм конечен в пространстве (при описании) и во времени (при выполнении).

Обычно в ходе разработки пользователь делает набросок алгоритма, используя обычный язык, например, русский. При этом для большей наглядности могут использоваться и блок-схемы. Безусловно, что в конечном виде алгоритм будет записан на определенном алгоритмическом языке, или в случае конкретного исполнителя – непосредственно на соответствующем языке программирования.

В общем случае, разработка алгоритмов представляет собой творческий процесс, предполагающий, что пользователь обладает так называемым алгоритмическим мышлением.

Под алгоритмическим мышлением мы будем понимать способность человека разрабатывать алгоритмы для решения задач, с которыми он сталкивается в повседневной жизни.

Безусловно, алгоритмическое мышление формируется и развивается в процессе изучения всех школьных дисциплин, но центральная роль при этом отводится информатике. Отметим, что если в прошлом веке главная задача учебных заведений заключалась в формировании навыков и прививании потребности в работе с книгами, то сегодня указанная задача значительно расширилась и включает в себя формирование **информационной культуры** и развитие алгоритмического мышления.

Вопросы и упражнения

- ❶ Каковы основные средства представления алгоритмов? Какими преимуществами и недостатками обладает каждый из них?
- ❷ Попробуйте описать процесс разработки произвольного алгоритма. Какие средства используются при описании алгоритма в ходе его разработки?
- ❸ Существует ли разница между алгоритмическим языком и языком программирования? Аргументируйте Ваш ответ.
- ❹ Объясните термины *алгоритмическое мышление* и *информационная культура*. Пользуясь каким-нибудь поисковым сервером, найдите в Интернете информацию по этим терминам.
- ❺ Перечислите основные свойства алгоритмов. Проверьте, обладают ли программы, разработанные для управления исполнителями **Кенгуренок** и **Муравей**, этими свойствами.
- ❻ Разработайте алгоритм для решения уравнений второй степени. Представьте этот алгоритм в виде блок-схемы.

БИБЛИОГРАФИЯ

1. Cerchez, Emanuela, Șerban, Marinela, *Informatica pentru gimnaziu*, Iași, Editura POLIROM, 2005, 232 p.
2. Cerchez, Emanuela, Șerban, Marinela, *PC pas cu pas*, Iași, Editura POLIROM, 2008, 296 p.
3. Curteanu, Silvia, *EXCEL prin exemple*, Iași, Editura POLIROM, 2004, 352 p.
4. Eder, Bernhard, Kodym, Willibald, Lechner, Franz, *Excel: Calcul tabelar*, București, Editura ALL EDUCATIONAL, 2002, 112 p.
5. Garabet M.E., Voicu A.E., Huțanu E., *Fizică, biologie, chimie pentru gimnaziu utilizând Microsoft Office*, București, Editura ALL EDUCATIONAL, 2001, 224 p.
6. Harvey, Greg, *Excel pentru toți*, București, Editura Teora, 1996, 416 p.
7. Heathcote P.M., *Excel 2000... Pentru copii*, București, Editura ALL EDUCATIONAL, 2003, 64 p.
8. Ionescu, Bogdan, Ionescu, Iuliana, Oancea, Mirela, *Inițiere în Microsoft Office*, București, Editura InfoMega, 2006, 475 p.
9. Ionescu, Bogdan, Ionescu, Iuliana, *Tehnologia Aplicațiilor Office – Excel 2007*, București, Editura InfoMega, 2011, 502 p.
10. Ivanov, Lilia, Gremalschi, Anatol, Căpățână, Gheorghe ș.a., *Informatică. Curriculum pentru învățământul gimnazial (clasele VII–IX)*. Ministerul Educației al Republicii Moldova, 2010.
11. Mateiaș, Dorina, Matei, Rodica, *Tainele informaticii*. Manual de informatică pentru clasele V–VIII, Pitești, Editura Paralela 45, 2008, 280 p.
12. Mârșanu R., Voicu A.E., *Tehnologia informației. Informatică – Tehnologii asistate de calculator*, București, Editura ALL EDUCATIONAL, 2004, 112 p.
13. Reisner, Trudi, *Excel sub Windows 95 pentru începători*, București, Editura Teora, 1996, 112 p.
14. Босова Л.Л., *Информатика: Учебник для 7 класса*, М.: БИНОМ. Лаборатория знаний, 2012, 237 с.
15. Босова Л.Л., Босова А.Ю., *Информатика и ИКТ. Учебник для 9 класса*. В 2 ч., М.: БИНОМ, Лаборатория знаний, 2012, ч. 1, 244 с.; ч. 2, 79 с.
16. Горячев А.В., Макарина Л.А. и др., *Информатика*, 7 класс. В 2 кн., М.: 2012, кн. 1, 256 с.; кн. 2, 144 с.
17. Макарова Н.В., *Информатика и ИКТ. Практикум*, 8–9 класс, СПб.: Издательство «Питер», 2010, 384 с.
18. Первин Ю.А., *Информатика дома и в школе*. Книга для ученика, Серия «Основы информатики», СПб.: БХВ-Петербург, 2003, 352 с.
19. Первин Ю.А., *Информатика дома и в школе*. Книга для учителя, Серия «Основы информатики», СПб.: БХВ-Петербург, 2003, 144 с.
20. Семакин И.Г., Залогова Л.А., Русаков С.В., Шестакова Л.В., *Информатика. Базовый курс*. Учебник для 9 класса, М.: БИНОМ, Лаборатория знаний, 2012, 341 с.
21. Угринович Н.Д., *Информатика и ИКТ. Учебник для 9 класса*, М.: БИНОМ, Лаборатория знаний, 2009, 295 с.

Acest manual este proprietatea Ministerului Educației, Culturii și Cercetării
al Republicii Moldova

Gimnaziul/Liceul _____				
Manualul nr. _____				
Anul de folosire	Numele de familie și prenumele elevului	Anul școlar	Aspectul manualului	
			la primire	la restituire
1				
2				
3				
4				
5				

- Dirigințele verifică dacă numele elevului este scris corect.
- Elevul nu trebuie să facă niciun fel de însemnări în manual.
- Aspectul manualului (la primire și la restituire) se va aprecia folosind termenii: *nou, bun, satisfăcător, nesatisfăcător*.

Imprimare la Tipografia „BALACRON” SRL,
str. Calea Ieșilor, 10; MD-2069
Chișinău, Republica Moldova
Comanda nr.