БЕЛКООПСОЮЗ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БЕЛОРУССКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ
УНИВЕРСИТЕТ ПОТРЕБИТЕЛЬСКОЙ КООПЕРАЦИИ»

Кафедра информационно-вычислительных систем

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Пособие
для студентов специальности 1-25 01 07
«Экономика и управление на предприятии»
специализации 1-25 01 07 02
«Экономическая информатика»,
специальности 1-26 03 01 «Управление
информационными ресурсами»

Гомель 2008

УДК 681.31
ББК 32.973
А 45

Авторы-составители: С. М. Мовшович, канд. техн. наук, доцент;
О. А. Кравченко, канд. физ.-мат. наук, доцент

Рецензенты: В. П. Кудин, канд. техн. наук, доцент, зав. кафедрой математики и информационных технологий ГФ УО ФПБ «МИТСО»;
Т. В. Астапкина, ассистент кафедры информационно-вычислительных систем Белорусского торгово-экономического университета потребительской кооперации

Рекомендовано к изданию научно-методическим советом учреждения образования «Белорусский торгово-экономический университет потребительской кооперации». Протокол № 1 от 10 октября 2006 г.

А 45 **Алгоритмизация** и программирование : пособие для студентов специальности 1-25 01 07 «Экономика и управление на предприятии» специализации 1-25 01 07 02 «Экономическая информатика», специальности 1-26 03 01 «Управление информационными ресурсами» / авт.-сост. : С. М. Мовшович, О. А. Кравченко. – Гомель : учреждение образования «Белорусский торгово-экономический университет потребительской кооперации», 2008. – 104 с.
ISBN 978-985-461-548-6

УДК 681.31
ББК 32.973

ISBN 978-985-461-548-6

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Молодой экономист будет востребован и вовлечен в процессы развития информационных технологий и ресурсов организации, если он способен алгоритмизировать бизнес-процессы и владеет системным подходом как в своей предметной области, так и в области информационных технологий.

Развитие алгоритмического мышления у студента в рамках учебных планов традиционных экономических специальностей практически невозможно из-за отсутствия подходящей учебной дисциплины.

Следствием такой подготовки является тот факт, что молодые специалисты-экономисты редко привлекаются для проведения обследований с целью автоматизации бизнес-процессов предприятия и при разработке технических заданий на создание и внедрение автоматизированных информационных систем. Основной объем указанных работ выполняют программисты, которые по ходу разработки и внедрения информационных технологий в организации осваивают предметные области экономики. Таким образом, актуальной стала задача подготовки экономистов-менеджеров, являющихся квалифицированными пользователями информационных технологий.

Цель курса «Алгоритмизация и программирование» – формирование у студентов знаний о принципах алгоритмизации и технологии программирования в среде Delphi.

Задачи данного курса сводятся к получению студентами знаний об алгоритмах и способах их записи, представлений о понятиях модульного и объектно-ориентированного программирования, сведений о системе программирования СП Delphi и к выработке практических навыков составления алгоритмов расчета статистических характеристик и работы с инструментами среды СП Delphi.

Настоящее пособие содержит следующие разделы:

- Теоретические основы курса.
- Задания лабораторных работ по разработке приложений в среде системы программирования Delphi с примерами и необходимыми теоретическими сведениями.
- Вопросы для подготовки к тестированию и экзамену для студентов заочной формы обучения.
- Примерные тестовые задания.

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ КУРСА

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ АЛГОРИТМИЗАЦИИ И ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

1.1. Свойства модуля

Модульное программирование основано на понятии *модуля* – логически взаимосвязанной совокупности функциональных элементов, оформленных в виде отдельного программного элемента.

Модуль обладает следующими характеристиками:

- *Один вход и один выход*. На входе программный модуль получает определенный набор исходных данных, выполняет содержательную обработку и возвращает один набор результатов, т. е. реализуется стандартный принцип IPO (Input – Process – Output) – *вход – процесс – выход*.
- *Функциональная завершенность*. Модуль выполняет перечень регламентированных операций для реализации каждой отдельной функции в полном составе, достаточных для завершения начатой обработки.
- *Логическая независимость*. Результат работы программного модуля зависит только от исходных данных, но не зависит от работы других модулей.
- *Слабые информационные связи с другими программными модулями*. Обмен информацией между модулями должен быть по возможности минимизирован.
- *Обозримый по размеру и сложности программный элемент*.

Таким образом, модули содержат определение доступных для обработки данных, операции обработки данных, схемы взаимосвязи с другими модулями.

Каждый модуль состоит из спецификации и тела. Спецификации определяют правила использования модуля, а тело – способ реализации процесса обработки.

1.2. Модульная структура программных продуктов

Принципы модульного программирования во многом сходны с принципами нисходящего проектирования. Сначала определяются состав и подчиненность функций, а затем – набор программных модулей, реализующих эти функции.

Однотипные функции реализуются одними и теми же модулями. Функция верхнего уровня обеспечивается главным модулем. Он управляет выполнением нижестоящих функций, которым соответствуют подчиненные модули.

При определении набора модулей, реализующих функции конкретного алгоритма, необходимо учиты-

вать следующее:

• каждый модуль вызывается на выполнение вышестоящим модулем и, закончив работу, возвращает управление вызвавшему его модулю;

• принятие основных решений в алгоритме выносится на максимально «высокий» по иерархии уровень;

• для использования одной и той же функции в разных местах алгоритма создается один модуль, который вызывается на выполнение по мере необходимости.

В результате дальнейшей детализации алгоритма создается *функционально-модульная схема* (*ФМС*) алгоритма приложения, которая является основой для программирования (рис. 1).

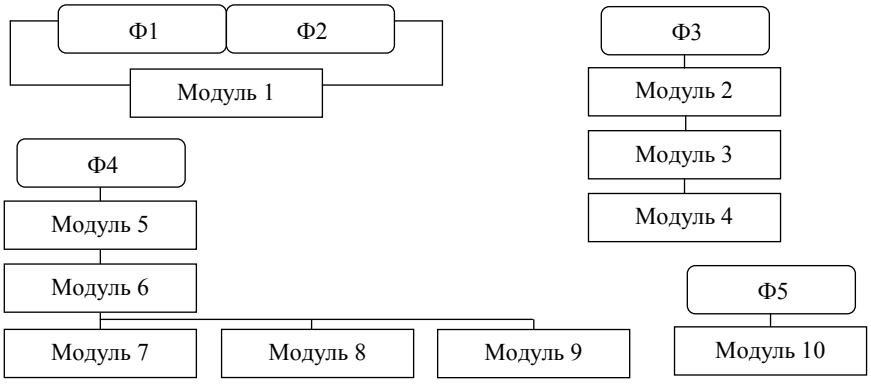


Рис. 1. **Функционально-модульная структура приложения**

Рис. 1 демонстрирует следующие особенности распределения функций по модулям:

• некоторые функции могут выполняться с помощью одного и того же программного модуля, например, функции Ф1 и Ф2;

• функция Ф3 реализуется в виде последовательности выполнения программных модулей;

• функция Ф4 осуществляется с помощью иерархии связанных модулей;

• функция Ф5 реализуется одним программным модулем;

• модуль 6 управляет выбором на выполнение подчиненных модулей.

Состав и вид программных модулей, их назначение и характер использования в программе в значительной степени определяются инструментальными средствами, т. е. системой программирования, в которой эти модули должны реализовываться.

1.3. Основные понятия объектно-ориентированного программирования

Метод объектно-ориентированного проектирования основывается на следующем:

• модели построения системы как совокупности объектов абстрактного типа данных;

• модульной структуре программ;

• нисходящем проектировании, используемом при выделении объектов.

Объектно-ориентированный подход использует следующие базовые понятия:

• объект;

• свойство объекта;

• метод обработки;

• событие;

• класс объектов.

Объект – совокупность свойств (параметров) определенных сущностей и методов их обработки (программных средств).

Объект содержит *инструкции* (программный код), определяющие действия, которые может выполнять объект, и обрабатываемые *данные*.

Свойство – характеристика объекта, его параметр. Все объекты наделены определенными свойствами, которые в совокупности выделяют объект из множества других объектов.

Объект обладает *качественной* определенностью, что позволяет выделить его из множества других объектов и обусловливает независимость процессов его создания и обработки от операций, выполняемых над другими объектами.

Например, объект можно представить перечислением присущих ему свойств: ОБЪЕКТ-А (свойство-1, свойство-2, ..., свойство-k).

Свойства объектов различных классов могут «пересекаться», т. е. возможны объекты, обладающие

одинаковыми свойствами:

- ОБЪЕКТ-B (...свойство-n, свойство-m, ..., свойство-r, ...);
- ОБЪЕКТ-C (...свойство-n, …, свойство-r,...).

Одним из свойств объекта является метод его обработки.

Метод – программа действий над объектом или его свойствами.

Метод рассматривается как программный код, связанный с определенным объектом, который осуществляет преобразование свойств и изменяет поведение объекта.

Объект может обладать набором заранее определенных встроенных методов обработки, созданных пользователем или заимствованных в стандартных библиотеках, которые выполняются при наступлении *заранее определенных событий* (однократное нажатие левой кнопки мыши, вход в поле ввода, выход из поля ввода, нажатие определенной клавиши и др.).

По мере развития систем обработки данных создаются *стандартные* библиотеки методов, в состав которых включаются типизированные методы обработки объектов определенного класса. Эти методы можно заимствовать для различных объектов.

Событие – изменение состояния объекта.

События бывают внешними и внутренними.

Внешние события генерируются пользователем, например, клавиатурный ввод или нажатие кнопки мыши, выбор пункта меню, запуск макроса, внутренние события – системой.

Класс – совокупность объектов, характеризующихся общностью применяемых методов обработки или свойств.

Объекты могут объединяться в классы, группы или наборы. В различных программных системах возможна различная терминология.

Один объект может выступать объединением вложенных в него по иерархии других объектов.

Схематично связь основных понятий объектно-ориентированного программирования представлена на рис. 2.

В объектно-ориентированном программировании используется следующий формат записи работы с объектами:

- ОБЪЕКТ. МЕТОД;
- ОБЪЕКТ.СВОЙСТВО.МЕТОД.

Программный продукт, созданный с помощью инструментальных средств объектно-ориентированного программирования, содержит объекты с их характерными свойствами, для которых разработан графический интерфейс пользователя. Как правило, работа с программным продуктом осуществляется с помощью экранной формы с объектами управления, которые содержат методы обработки, вызываемые при наступлении определенных событий. Экранные формы также используются для выполнения заданий и перехода от одного компонента программного продукта к другому. Каждый объект управления обладает определенными свойствами, значения которых могут изменяться. Для них уточняется перечень событий и создаются пользовательские методы обработки – программный код на языке программирования в виде событийных процедур.

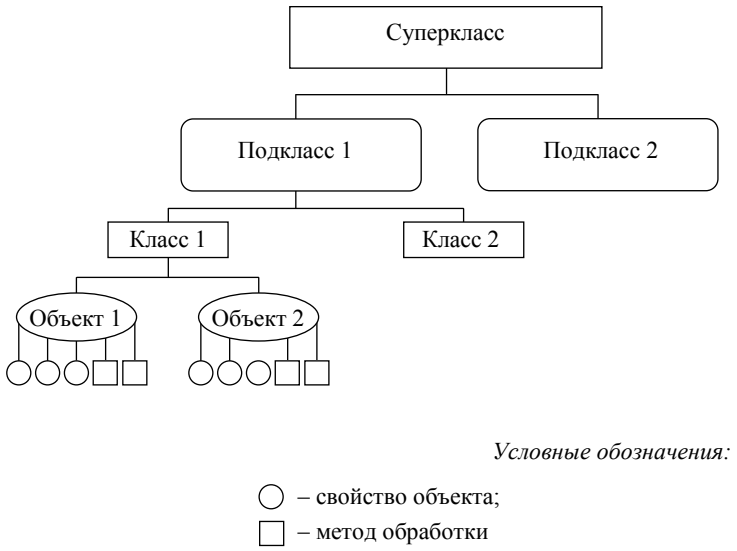


Рис. 2. **Соотношение основных понятий объектно-ориентированного подхода**

1.4. Понятие алгоритма

Алгоритм – понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение указанной цели или на решение поставленной задачи.

Для раскрытия приведенного определения составим простую классификацию всех окружающих нас правил. В своей жизни мы постоянно имеем дело с различными правилами – разрешающими, запрещающими и предписывающими.

Примерами разрешающих правил могут быть следующие:

- Каждый студент может получать повышенную стипендию.
- Стоянка разрешена.
- На дискотеку вход свободный.

Никакое разрешающее правило не является алгоритмом. Это же утверждение справедливо и для запрещающих правил, примерами которых являются следующие:

- При плохой успеваемости студент не может получать повышенную стипендию.
- Во всех корпусах университета курение запрещено.
- Стоянка запрещена.

К алгоритмам относятся только предписывающие правила. Например, кулинарный рецепт приготовления какого-либо блюда является алгоритмом. Алгоритмом является также инструкция по включению какого-либо прибора, например, компьютера, составленная из предписывающих правил.

Но не всякое предписывающее правило можно считать алгоритмом. Так, например, предписывающее правило *«Взвейтесь кострами, синие ночи!»* не является алгоритмом, так как предполагаемый исполнитель (синие ночи) желаемого действия (взвейтесь кострами) не в состоянии выполнить.

Таким образом, алгоритм – последовательность предписывающих правил, понятно и точно указывающих исполнителю, какую последовательность действий он должен исполнить по переработке исходных данных в искомые результаты.

1.5. Свойства алгоритма

К свойствам алгоритма относятся точность, понятность, дискретность, массовость, результативность и другие свойства.

Рассмотрим несколько свойств алгоритма.

Точность определяется как свойство, согласно которому исполнителю точно известно, какая команда должна выполняться следующей.

Для иллюстрации важности этого свойства рассмотрим предписывающее правило «Уходя, гасите свет». Является ли это правило алгоритмом? С одной стороны, рассматриваемое правило предписывает вполне определенному исполнителю (человеку, уходящему из данного помещения) выполнить в определенный момент (в момент ухода) вполне определенное действие (выключить свет). Исходя из этого, можно считать рассматриваемое правило алгоритмом. Но, с другой стороны, для человека, «страдающего» излишним формализмом, указанное правило не вполне точно определяет последовательность необходимых действий, так как для этого человека остается неясным, должен ли он, покидая помещение, выключить свет, если в помещении остаются люди. Анализируемый пример показывает на существование серьезной проблемы: то, что однозначно (точно) понимается одним исполнителем, может совсем не так восприниматься другим.

Понятность – это свойство, заключающееся в том, что каждая команда алгоритма должна входить в систему команд исполнителя. *Система команд исполнителя* – это совокупность команд, которые могут быть выполнены исполнителем. Например, система команд микрокалькулятора описывается в его паспорте. В нее входят такие команды, как четыре арифметические операции, запись числа в ячейку памяти, вычисление элементарных функции и т. д.

Дискретность алгоритма заключается в том, что предписание представляет собой последовательность четко выраженных отдельных команд. Выполнив одну команду, исполнитель выполняет следующую команду и т. д. Бессмысленно ставить вопрос о том, что делает исполнитель между выполнением двух последовательных команд, так как таких моментов просто нет.

Массовость алгоритма состоит в том, что для каждого алгоритма существует класс объектов, допустимых в качестве исходных данных. Так, например, алгоритм нахождения наибольшего общего делителя двух целых чисел применим для любой пары целых чисел. Из двух алгоритмов, разработанных для решения одной задачи, более ценным является тот, у которого шире класс допустимых исходных данных. Можно сказать, что у такого алгоритма массовость больше.

Результативность заключается в том, что результат выполнения алгоритма исполнитель должен получить, произведя конечное число действий. Если для каких-то допустимых исходных данных исполнитель не может получить результат выполнения алгоритма за конечное число шагов, то говорят, что алгоритм не применим для этих исходных данных. Так, например, алгоритм получения точного значения частного при делении уголком не применим для чисел 20 и 3. При этом класс допустимых исходных данных для указанного алгоритма – пара целых чисел, второе из которых не равно нулю.

1.6. Средства записи алгоритма

Для записи алгоритмов используются различные способы – словесная и графическая схемы алгоритма (блок-схема), алгоритмический язык, языки программирования. Рассмотрим причины такого разнообразия и области употребления каждого способа записи.

Алгоритм разрабатывается и записывается для каждого конкретного исполнителя. Можно выделить два класса исполнителей – люди и вычислительные устройства. К последним относятся роботы, микрокалькуляторы, компьютеры и т. д. Так как разные исполнители имеют различные системы команд, то в записях алгоритмов должны использоваться команды, зависящие от конкретного исполнителя (свойство понятности). Кроме наличия различных систем команд, каждый исполнитель отличается своим способом восприятия команд, алгоритма, т. е. человек может прочитать или выслушать данные ему для исполнения команды; микрокалькулятор воспринимает команды, вводимые в него нажатием клавиши; в компьютер команды можно вводить нажатием клавиш на клавиатуре или с помощью указателя мыши.

Итак, причина разнообразия средства записи алгоритмов – различия в системах команд исполнителей и в способах восприятия исполнителем команд алгоритма.

Язык программирования – это способ записи алгоритма, ориентированный на исполнение его системой программирования компьютера. Для записи алгоритмов, предназначенных для безмашинного исполнения, можно использовать естественный язык, например, русский, дополненный математическими символами. Это так называемая словесная запись алгоритма.

Наиболее часто при разработке и документировании программы лежащие в ее основе алгоритмы изображаются в виде графических схем алгоритма или блок-схем.

Алгоритм большой сложности обычно представляется с помощью схем двух видов:

- *обобщенной схемы алгоритма*, которая раскрывает общий принцип функционирования алгоритма и основные логические связи между отдельными модулями на уровне обработки информации: ввод и редактирование данных, вычисления, печать результатов и т. д.;
- *детальной схемы алгоритма*, представляющей содержание каждого элемента обобщенной схемы с использованием управляющих структур в блок-схемах алгоритма, псевдокода либо алгоритмических языков высокого уровня.

1.7. Графические схемы алгоритмов

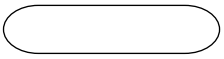
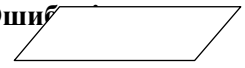
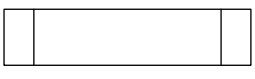
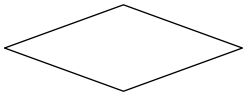
Перед тем, как записать алгоритм в виде программы, его, как правило, представляют в виде схемы алгоритма, а не наоборот, как пытаются сделать многие начинающие. Схема алгоритма, если она правильно составлена, способствует правильному и более быстрому написанию программы.

Правила выполнения схем алгоритмов регламентированы ГОСТ 19.701-90 (ИСО 5807-85), входящим в единую систему программной документации (ЕСПД) под названием «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». Согласно этому стандарту схема алгоритма – это графическое представление метода решения задачи, в котором используются символы для отображения операций, данных, потока, оборудования и т. д. Схема алгоритма состоит из следующих элементов:

- символов процесса, указывающих фактические операции обработки данных и определяющих путь, которого следует придерживаться с учетом логических условий;
- линейных символов, указывающих поток управления;
- специальных символов, используемых для облегчения написания и чтения схемы.

Обозначение наиболее часто употребляемых символов и описание отображаемых ими действий приведено в табл. 1.

Таблица 1. **Символы схемы алгоритма**

Название символа	Обозначение	Значение
Терминатор		Начало или конец схемы алгоритма
Данные		Ввод или вывод данных. Носитель данных не определен
Процесс		Обработка данных любого вида, приводящая к изменению значения, формы или размещения информации
Предопределенный процесс		Использование подпрограммы или модуля
Решение		Проверка условия и выбор одного из нескольких альтернативных выходов

Окончание табл. 1

Название символа	Обозначение	Значение
Подготовка		Модификация команды, группы команд, индексного регистра (создание цикла)
Линия		Отображает поток данных и управления. При необходимости могут быть добавлены стрелки-указатели
Соединитель		Обрыв линии и продолжение ее в другом месте. Соответствующие символы должны иметь одно и то же уникальное обозначение
Комментарий		Пояснения к выполняемым действиям. Располагается около ограничивающей фигуры (символа или блока символов, обведенных пунктирной линией)

Символы могут быть вычерчены в любой ориентации, но, по возможности, предпочтительной является горизонтальная ориентация.

Текст, необходимый для понимания функции данного символа, следует помещать внутри данного символа и записывать слева направо и сверху вниз независимо от направления потока. Если объем текста превышает размеры символа, нужно использовать символ комментария. В схемах может применяться идентификатор символа в виде номера, который является ссылкой на графический символ, расположенный на других страницах документации. Идентификатор символа должен располагаться слева над символом.

Правила выполнения соединений. Потоки данных или потоки управления в схемах показываются линиями. Направление потока слева направо и сверху вниз считается стандартным. Если необходимо внести большую ясность в схему (например, при соединениях), на линиях используются стрелки. Если направление потока отличается от стандартного, то стрелки должны указывать это направление.

В схемах следует избегать пересечения линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменения направления потока в точках пересечения не допускаются. Две или более входящие линии могут объединяться в одну исходящую.

Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линии должны быть направлены к центру символа. При необходимости линии в схемах нужно разрывать во избежание излишних пересечений или слишком длинных линий, а также если схема состоит из нескольких страниц. Соединитель в начале разрыва называется внешним соединителем, а соединитель в конце разрыва – внутренним соединителем. Совместно с символом комментария можно указать, с какой страницы или на какую страницу схемы совершается переход.

Примеры соединителей приведены на рис. 3.

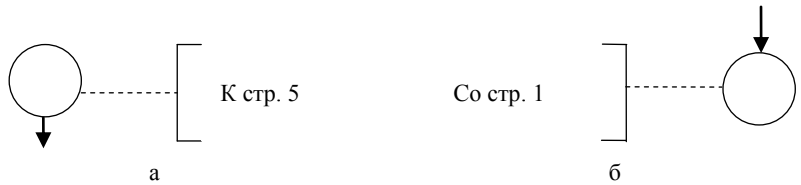


Рис. 3. **Виды соединителей:** а – внешний соединитель;
б – внутренний соединитель

Несколько выходов из символа можно показать несколькими линиями от данного символа к другим символам; одной линией от данного символа, которая затем разветвляется на соответствующее число линий.

Примеры изображения выходов из символа приведены на рис. 4.

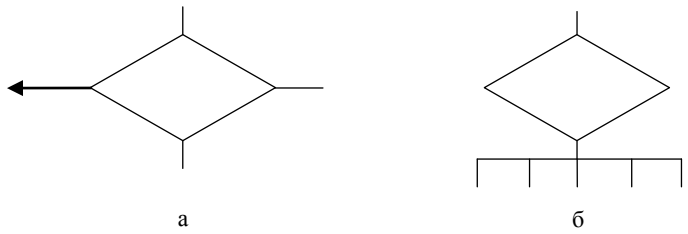


Рис. 4. **Изображение выходов из символа:**
а – одинарные выходы; б – мультивыход

Каждый выход из символа нужно пометить значениями соответствующих условий, чтобы показать ло-

гический путь, который он представляет, с тем, чтобы эти условия и соответствующие ссылки были идентифицированы.

Примеры идентификации ссылок приведены на рис. 5.

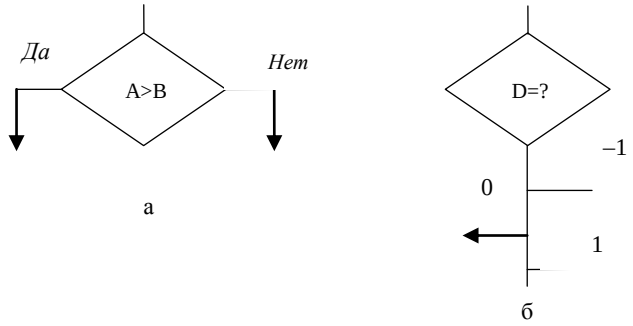


Рис. 5. **Примеры идентификации ссылок:** а – блок ветвления; б – блок выбора

При всем разнообразии структур алгоритмов можно выделить четыре типовых структуры (рис. 6), из которых, как из кирпичиков, можно построить здание алгоритма любой сложности: *следование, ветвление, цикл* и *выбор*.

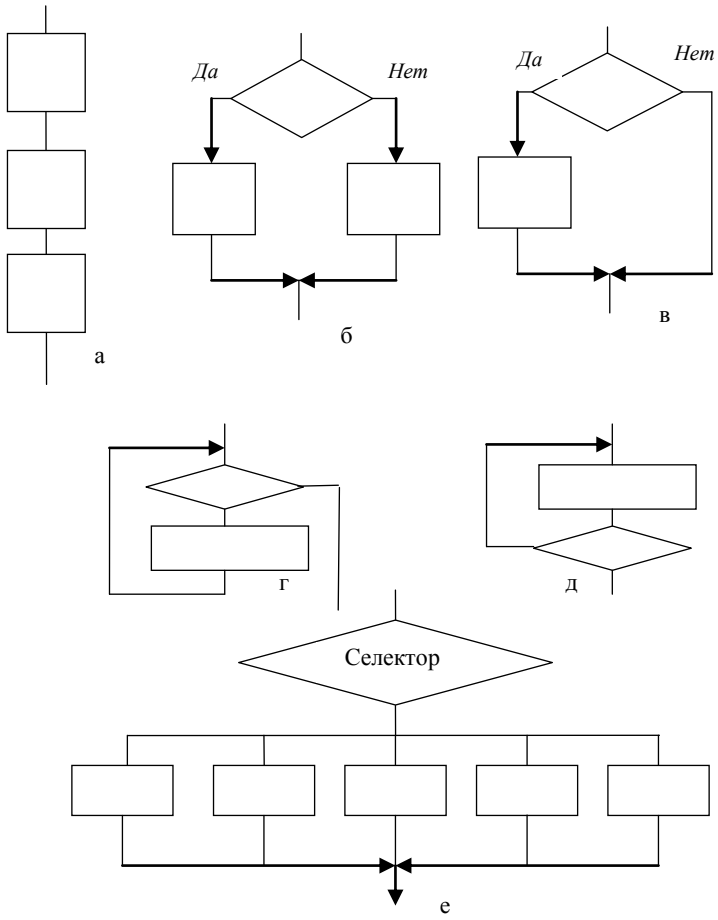


Рис. 6. **Типовые структуры алгоритмов:** а – следование;
б, в – ветвление (полное и неполное); г – цикл с предусловием;
д – цикл с постусловием; е – выбор

Типовые структуры хороши тем, что все они имеют одну точку входа и одну точку выхода. Это упрощает просмотр и понимание алгоритма; сокращает количество возможных ошибок, сделанных при составлении схемы, и время на их поиск. Типовые структуры могут быть вложенными друг в друга. Например, символ, входящий в структуру *следование*, может включать в себя структуру *ветвление* или *цикл*, а каждая из ветвей структуры *ветвление* или *выбор* может содержать структуры *следование*, *цикл* или еще одно *ветвление* и *выбор*.

1.8. Типы алгоритмов

На основе перечисленных в пункте 1.7 структур строятся следующие типы алгоритмов:

• *линейный* (на основе структуры *следование*), характеризующийся тем, что все действия, определяемые символами, входящими в схему, выполняются последовательно, в порядке их написания;

• *разветвляющийся* (на основе структур *ветвление и выбор*), характеризующийся тем, что в ходе выполнения решение задачи идет только по одному из имеющихся направлений, выбор которого зависит от выполнения заданного условия;

• *циклический* (на основе структуры *цикл*), характеризующийся многократным повторением определенной группы действий.

В последующих разделах на конкретных примерах рассматриваются проекты в системе программирования Delphi, основанные на всех вышеперечисленных типах алгоритмов.

2. ЭТАПЫ СОЗДАНИЯ ПРОСТОГО ПРИЛОЖЕНИЯ В СИСТЕМЕ ПРОГРАММИРОВАНИЯ DELPHI

2.1. Запуск и завершение работы системы программирования Delphi

Запуск Delphi выполняется с помощью ярлыка на *Рабочем столе* или через *Главное меню*. После загрузки на экране обычно находятся шесть окон (рис. 7): главное окно (содержит заголовок окна, строку меню, панель инструментов и палитры компонентов), окно инспектора объектов, окно формы и окно редактора кода, включающее окно браузера кода, окно дерева объектов.

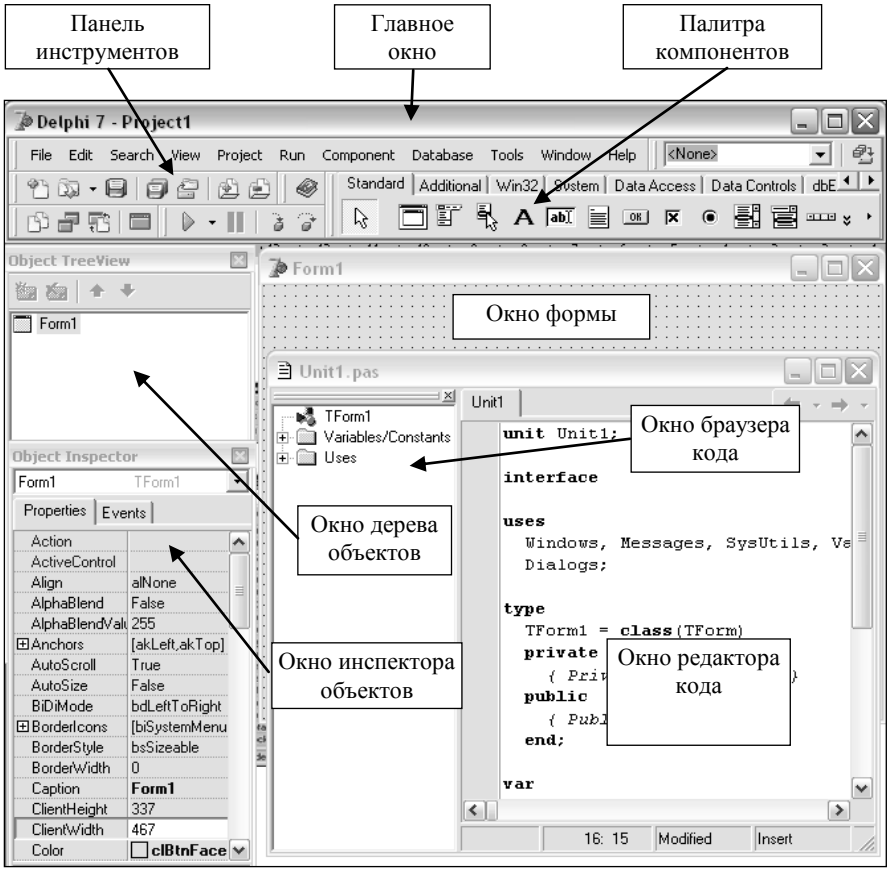


Рис. 7. **Окна СП Delphi**

Если какого-то окна нет на экране, то его можно вывести на экран с помощью следующих команд меню *View*:

• окно инспектора объектов → *View* / *Object Inspector* (*F11*);

• окно браузера кода → *View* / *Code Explorer*;

• окно формы → *View* / *Forms*…(*Shift+F12*);

• окно кода другого модуля → *View* / *Units*…(*Ctrl+F12*);

• окно дерева объектов → *View* / *ObjectTreeView*.

Переключение между формой и соответствующим ей модулем выполняется командой *View* / *Toggle Form* / *Unit* (*F12*).

Файл формы можно просмотреть с помощью команды *View as Text* контекстного меню формы в режиме проектирования. Возврат к форме выполняется по команде *View as Form* контекстного меню текстового файла формы.

Окно проекта при необходимости можно вывести на экран с помощью команды *Project* / *View Source*.

Закрытие окна редактора кода вызывает закрытие файла проекта, приводит к окончанию работы с ним. Для работы с данным проектом его необходимо снова открыть (пункт 2.2).

Закрытие главного окна приводит к окончанию работы в СП Delphi.

2.2. Открытие существующего проекта или создание нового

При запуске Delphi обычно уже создан проект *Project1*, содержащий одну главную форму *Form1* и соответствующий ей модуль *Unit1*. Для создания нового проекта необходимо выполнить команду *File* / *New Application*, а для открытия существующего – использовать команду *File* / *Open Project*… или кнопку на панели инструментов.

2.3. Создание интерфейса приложения

Создание интерфейса заключается в размещении на каждой форме визуальных компонентов в соответствии с разработанным интерфейсом проекта. Если проект должен содержать несколько форм, то для создания новой формы (дополнительно к главной форме, которая уже есть) используется команда *File* / *New Form* или кнопка *New Form* на панели инструментов. Для смены активной формы используется команда *View* / *Forms*….

Для размещения каждого визуального компонента необходимо выполнить следующее:

- сделать активной форму, где должен располагаться данный компонент;
- щелкнуть кнопкой мыши по визуальному компоненту на палитре компонентов, а затем щелкнуть кнопкой мыши в месте его расположения в окне формы;
- на вкладке *Properties* инспектора объектов найти свойства, которые нужно изменить, и установить для них необходимые значения.

Выделенный щелчком кнопки мыши визуальный компонент можно удалить клавишей *Delete*, с помощью мыши его можно переместить в другое место окна формы, изменить размеры компонента или скопировать его в буфер обмена. При копировании компонентов их копии получают новые имена, а свойства и имена процедур обработки событий копируются.

2.4. Первое сохранение проекта

Рекомендуется сохранять файлы в личных папках на диске, отличном от *C:*. Для сохранения каждого проекта целесообразно создавать отдельную папку и сохранять проект каждые 5–10 мин во время разработки интерфейса и записи алгоритмов (пункты 2.3 и 2.5).

Если проект ни разу не сохранялся и имеет имя *Project1*, то для его сохранения используется команда *Save Project As*…. В появившемся диалоговом окне необходимо указать папку для его сохранения, а имена проекта и модуля следует оставить системные – *Project1* и *Unit1*.

2.5. Создание процедур-обработчиков событий

Для создания и изменения процедур-обработчиков каждого события необходимо выполнить следующее:

- выделить компонент, для которого создается процедура;
- найти на вкладке *Events* инспектора объектов нужное событие;
- выполнить двойной щелчок по полю, находящемуся справа от названия события (если процедура создается, то поле пустое, а если процедура изменяется, то поле содержит имя процедуры);
- ввести текст процедуры или изменить его в окне редактора кода.

Для создания процедуры обработки щелчка левой кнопкой мыши по выбранному компоненту (событие *OnClick*) можно выполнить двойной щелчок кнопкой мыши по этому компоненту.

При вводе кода процедуры после набора на клавиатуре имени компонента редактор автоматически выводит список свойств и методов для этого компонента, а для формы – и список всех расположенных на ней компонентов. Таким образом, можно не набирать свойство или метод на клавиатуре, а выделить его в списке и нажать клавишу *Enter*.

2.6. Сохранение проекта в процессе работы

При повторном сохранении проекта с модулями без изменения имен и папки, где находятся файлы, используется команда *File* / *Save All* или кнопка *Save All* на панели инструментов, а для сохранения активного модуля и формы применяется команда *File* / *Save* или кнопка *Save* на панели инструментов.

2.7. Выполнение (запуск) проекта

Выполнение (запуск проекта) может быть осуществлено с помощью команды *Run* / *Run* (*F9*) или кнопки *Run* на панели инструментов.

Нельзя запустить вторую копию приложения во время работы первой.

При выполнении приложения Delphi сначала выполняет компиляцию программы, выявляет ошибки и, если они есть, выводит их в специальном окне, появляющемся обычно под окном модуля. При наличии ошибок выполнение программы прекращается. Строка в модуле, при компиляции которой найдена ошибка, выделяется красным цветом.

Если ошибок компиляции не найдено, программа из режима проектирования переходит в режим выполнения. При этом выводится окно приложения, соответствующее окну главной формы. Теперь пользователь может вводить исходные данные в поля окна приложения, запускать события, реакция на которые запрограммирована в модуле формы, и получать результаты. На этом этапе тоже могут быть найдены ошибки, например, данные не введены или тип введенных данных не соответствует объявленному в программе типу. В этом случае выводится сообщение об ошибке в окне сообщений, а затем (после нажатия кнопки *OK* в окне сообщения) выводится окно проекта. В этом случае для завершения этапа выполнения программы и перехода в режим проектирования (как и для выхода из «зацикливающейся» программы) нужно использовать команду *Run* / *Program Reset* или комбинацию клавиш *Ctrl+F2*.

Если никаких ошибок обнаружено не было, введены исходные данные и получены ожидаемые результаты, то для завершения выполнения программы и перехода в режим проектирования необходимо закрыть окно главной формы.

2.8. Внесение изменений в проект

При обнаружении в проекте ошибок или при несовпадении результатов работы с подготовленными тестами необходимо внести изменения в интерфейс приложения или устранить ошибки в процедурах-обработчиках событий.

Технология внесения изменений в интерфейс приложения описана в пункте 2.3. Редактирование программного кода модулей описано в пункте 2.5. При этом используются методы работы с текстом, принятые в любых текстовых редакторах ОС Windows – выделение, копирование, перемещение, вставка и т. д.

Удаление модуля из проекта выполняется с помощью команды *Project* / *Remove from Project*, а добавление модуля к проекту – с помощью команды *Project* / *Add to Project*.

Действия, выполняемые в пунктах 2.6–2.8, повторяются до тех пор, пока на экране не будут получены результаты, совпадающие с подготовленными тестами, а интерфейс и функционирование проекта не будут соответствовать поставленной задаче.

3. СОЗДАНИЕ ПРОЕКТА НА ОСНОВЕ ЛИНЕЙНОГО АЛГОРИТМА

Реализацию в проекте линейного алгоритма рассмотрим на примере вычисления значения функции

$$b = \lg\sqrt{e^{x-y} + x^{|y|} + z}$$

для заданных значений *x*, *y* и *z*.

Действия выполните в следующей последовательности:

1. Для рассматриваемого проекта создайте папку *Функция*.

2. Первый шаг создания проекта состоит в разработке его *интерфейса*, т. е. проектировании окна приложения. Для создания окна приложения поместите в окно формы с вкладки *Standard* палитры компонентов пять объектов *Label*, четыре объекта *Edit* и объект *Button*. Кроме того, поместите с вкладки *Additional* объект *Image*. Окно формы данного проекта приведено на рис. 8.

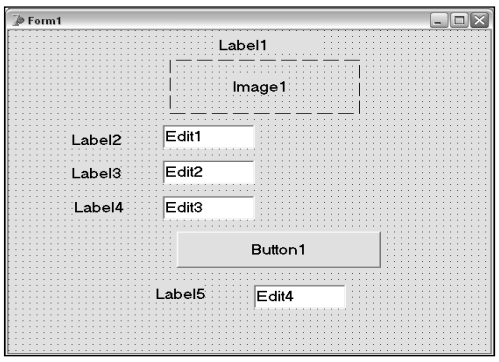


Рис. 8. **Объекты окна формы для проекта вычисления функции**

3. Наберите в текстовом процессоре Word с помощью редактора формул Microsoft Equation заданную формулу. После этого скопируйте формулу в графический редактор Paint. Сохраните функцию в файле *Функция.jpg* в папке *Функция*.

4. Наберите необходимые надписи (*Вычисление значения функции*, *x=*, *y=*, *z=*, *b=*, *Вычислить*) для значений свойства *Caption* объектов *Label* и *Button*. В качестве значений свойства *Text* объектов *Edit* укажите пустые строки. Свойству *Enabled* объектов *Edit1*, *Edit2* и *Edit3* присвойте значение *true*, а значение этого свойства для объекта *Edit4* установите равным *false*. В инспекторе объектов щелкните кнопкой мыши справа от свойства *Picture* и в открывшемся диалоговом окне по команде *Load* укажите файл *Функция.jpg*. В результате окно формы примет вид, представленный на рис. 9.

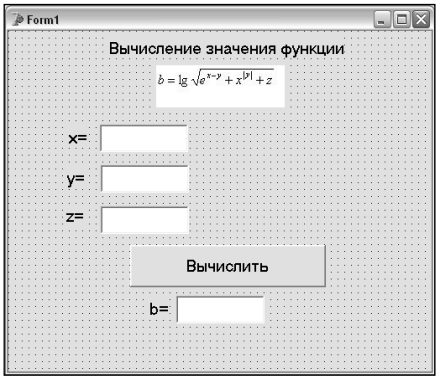


Рис. 9. **Окно формы для проекта вычисления функции**

5. Второй шаг создания проекта – разработка *алгоритма*. Схема алгоритма приведена на рис. 10.

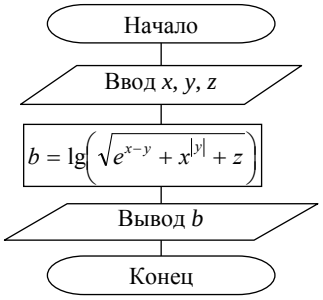


Рис. 10. **Схема алгоритма вычисления функции**

6. Третий шаг создания проекта заключается в написании процедуры обработки щелчка по кнопке *Вычислить*. При написании процедуры необходимо знать, какие стандартные математические функции имеются в СП Delphi. Перечень этих функций приведен в табл. 2.

Таблица 2. **Стандартные математические функции Delphi**

Математическое обозначение функции	Запись функции в СП Delphi	Описание
$\lvert x\rvert$	abs(*x*)	Абсолютное значение аргумента целого или вещественного типа
arctg *x*	arctan(*x*)	Арктангенс от вещественного аргумента
cos *x*	cos(*x*)	Косинус от вещественного аргумента
e^x	exp(*x*)	Экспонента от вещественного аргумента
ln *x*	ln(*x*)	Натуральный логарифм от вещественного аргумента
sin *x*	sin(*x*)	Синус от вещественного аргумента
x^2	sqr(*x*)	Квадрат аргумента (целого или вещественного типа)
$\sqrt{x}$	sqrt(*x*)	Квадратный корень от вещественного аргумента
–	round(*x*)	Округление вещественного аргумента до ближайшего целого
[*x*]	trunc(*x*)	Выделение целой части вещественного аргумента

Текст искомой процедуры приведен на рис. 11.

Unit1

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    x,y,z,b:real;
begin
  x:=StrToFloat(Edit1.Text);// ввод значения x
  y:=StrToFloat(Edit2.Text);// ввод значения y
  z:=StrToFloat(Edit3.Text);// ввод значения z
  b:=ln(sqrt(exp(x-y)+ exp(abs(y)*ln(x))+z))/ln(10,0);
  Edit4.Text:=FloatToStr(b)// вывод результата
end;
```

Рис. 11. **Текст процедуры обработки щелчка по кнопке *Вычислить***

При написании арифметического выражения для вычисления значения *b* использована следующая формула перехода от десятичного логарифма к натуральному:

$$\lg a = \frac{\ln a}{\ln 10}.$$

Кроме того, в этом выражении использована следующая формула для записи степени:

$$x^{|y|} = e^{|y|\cdot \ln x}.$$

7. Заключительным этапом работы над проектом является его *отладка*. Процесс отладки заключается в проверке правильности проекта, поиске и устранении ошибок. Отладка базируется на *тестировании* – сравнении результата выполнения проекта с заранее вычисленным значением. Для проведения отладки проекта надо подготовить следующий тест: вычислите в Excel значение искомой функции при *x*=2, *y*=3 и *z*=4. Затем выполните проект при тех же значениях аргументов. На рис. 12 приведены окно Excel и окно приложения. Совпадение полученных результатов показывает, что созданный проект функционирует правильно.

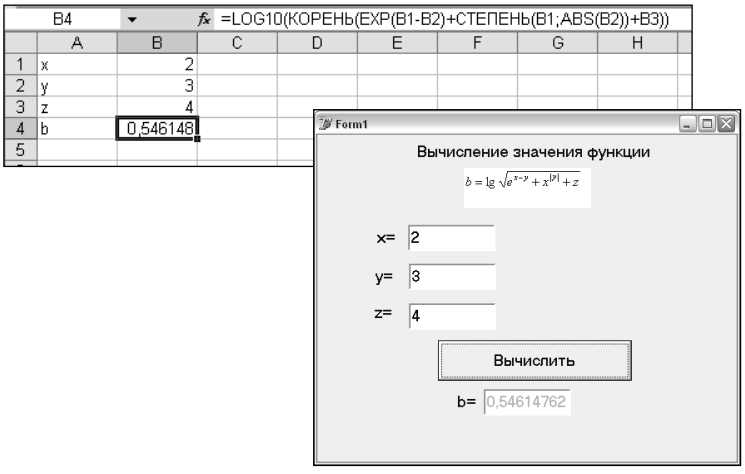


Рис. 12. **Тестирование проекта вычисления функции**

4. СОЗДАНИЕ ПРОЕКТА НА ОСНОВЕ РАЗВЕТВЛЯЮЩЕГОСЯ АЛГОРИТМА

Реализацию в проекте разветвляющегося алгоритма рассмотрим на примере вычисления для заданного *x* значения следующей кусочно-непрерывной функции:

$$y = \begin{cases} x^2, \text{ если } x \geq 7 \text{ или } x = 2; & (1) \\ x + 5, \text{ если } x \leq 0; & (2) \\ 2 \cdot x \text{ в остальных случаях.} & (3) \end{cases}$$

Кроме значения функции, необходимо вывести номер формулы, по которой производился расчет.

Порядок выполнения работы следующий:

1. *Проверка корректности условия задачи*. При вычислении функции, вид которой меняется в зависимости от диапазона значений аргумента, необходимо вначале проверить *корректность условия* задачи. Задача является корректной, если каждому значению аргумента соответствует единственное значение функции. Для проверки корректности заданной функции нарисуем числовую ось, отметим на ней точки, фигурирующие в условиях, и напишем над каждым диапазоном номер одной из трех формул, составляющих заданную функцию. Указанный анализ интервалов значений *x* представлен на рис. 13.

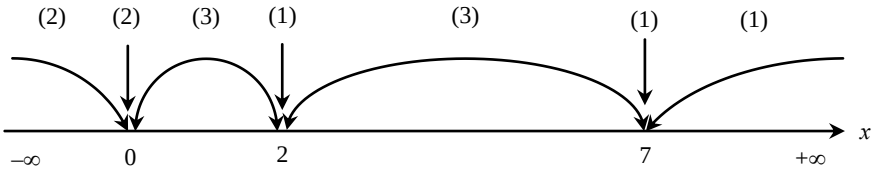


Рис. 13. **Виды функции на различных интервалах значения аргумента**

Как видно из рисунка, каждому значению аргумента соответствует единственная формула, по которой рассчитывается значение функции. Таким образом, можно считать, что условие задачи корректно.

2. *Задание места сохранения проекта*. Для рассматриваемого проекта создайте папку *Разветвляющаяся функция*.

3. *Вызов СП Delphi*. Вызовите СП Delphi и сохраните новый (пока еще пустой) проект в созданной папке.

4. *Разработка интерфейса*. Создайте интерфейс, окно формы которого приведено на рис. 14. Сохраните проект.

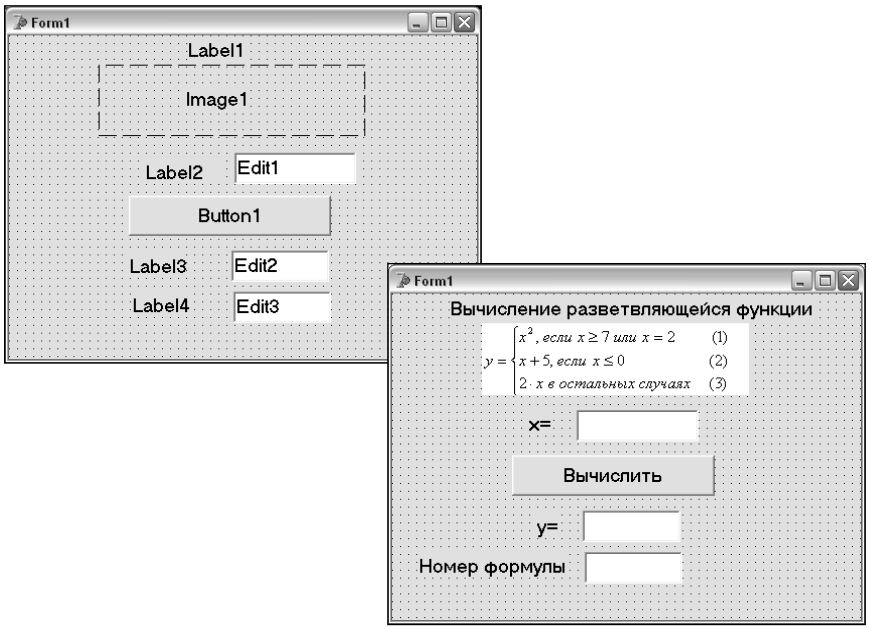


Рис. 14. **Окно формы для проекта вычисления разветвляющейся функции**

5. *Разработка алгоритма*. Схема алгоритма рассматриваемой задачи представлена на рис. 15. В схеме алгоритма имеются два элемента ветвления, которые обеспечивают три альтернативные ветви выполнения алгоритма, соответствующие трем видам рассматриваемой функции.

6. *Программирование*. Для составления процедуры обработки щелчка по кнопке *Вычислить* необходим оператор, реализующий ветвление. Таким оператором является *условный оператор*, который в общем виде записывается следующим образом:

```
if логическое выражение
  then   оператор1
  else   оператор2;
```

Условный оператор выполняется по следующей схеме:

- вычисляется логическое выражение;
- если получилось значение *true* (*истина*), то выполняется *оператор1*;
- если получилось значение *false* (*ложь*), то выполняется *оператор2*;
- вслед за тем, как выполнится *оператор1* или *оператор2*, выполняется оператор, стоящий после условного оператора.

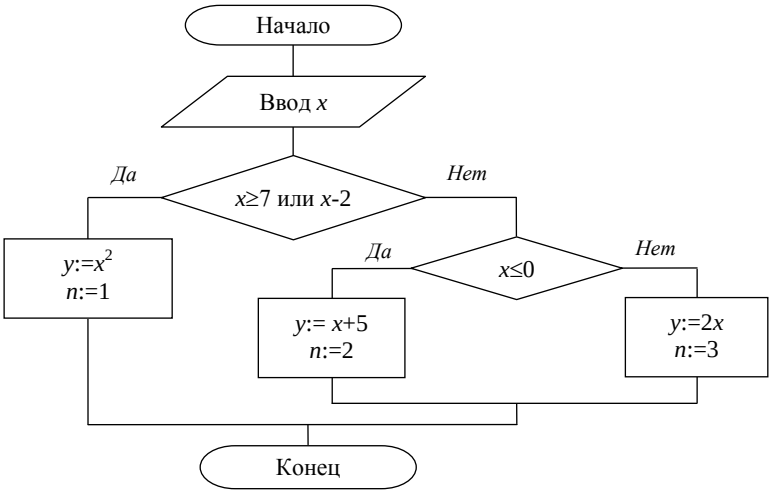


Рис. 15. **Схема алгоритма вычисления разветвляющейся функции**

Если за *then* или *else* надо указать несколько операторов, то эту последовательность операторов заключают в операторные скобки *begin – end*. Такая конструкция называется *составным оператором*.

Существует также неполная форма условного оператора, которая выглядит следующим образом:

if логическое выражение
  then оператор1;

Текст процедуры обработки щелчка по кнопке *Вычислить* приведен на рис. 16.

7. *Отладка проекта*. Отладку проекта надо провести на семи тестах, каждый из которых соответствует значению аргумента, принадлежащему одному из семи интервалов, изображенных на рис. 13. Набор тестов может быть следующим:

1) $x=-1$, $y=4$, $n=2$;
2) $x=0$, $y=5$, $n=2$;
3) $x=1$, $y=2$, $n=3$;
4) $x=2$, $y=4$, $n=1$;
5) $x=3$, $y=6$, $n=3$;
6) $x=7$, $y=49$, $n=1$;
7) $x=8$, $y=64$, $n=1$.

Unit1

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    x,y:real;
    n:integer;
begin
  x:=StrToFloat(Edit1.Text);
  if (x>=7) or (x=2)
    then
      begin
        y:=sqr(x);
        n:=1
      end
    else
      if x<=0
        then
          begin
            y:=x+5;
            n:=2
          end
        else
          begin
            y:=2*x;
            n:=3
          end;
  Edit2.Text:=FloatToStr(y);
  Edit3.Text:=IntToStr(n)
end;
```

Рис. 16. **Текст процедуры вычисления значения разветвляющейся функции**

Необходимо семь раз выполнить проект при указанных значениях *x*. Вид окна приложения для теста 7 приведен на рис. 17.

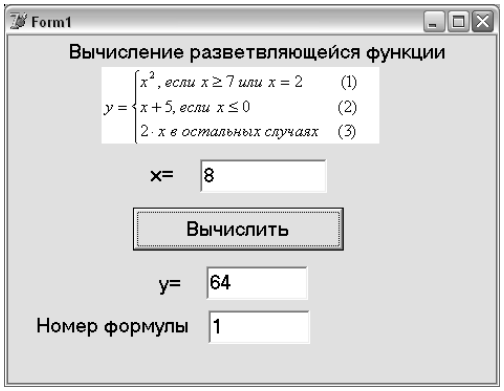


Рис. 17. **Окно приложения вычисления значения разветвляющейся функции**

5. ТАБУЛИРОВАНИЕ ФУНКЦИИ

5.1. Пример разработки проекта табуляции функции в область *Memo*

Дана функция $y = \dfrac{arctg(x^2 - 5) + 3}{\sin^2 x + 2}$.

Требуется рассчитать значения этой функции при значениях аргумента *x*, изменяющегося от начального значения $x_н$ до конечного значения $x_к$ с шагом изменения Δx. Результаты расчета следует представить в визуальном компоненте *Memo* в виде трех столбцов, озаглавленных *n*, *x* и *y*.

Порядок выполнения работы следующий:

1. Запустите СП Delphi.

2. Расположите в окне формы визуальные компоненты и установите значения их свойств так, как это показано на рис. 18.

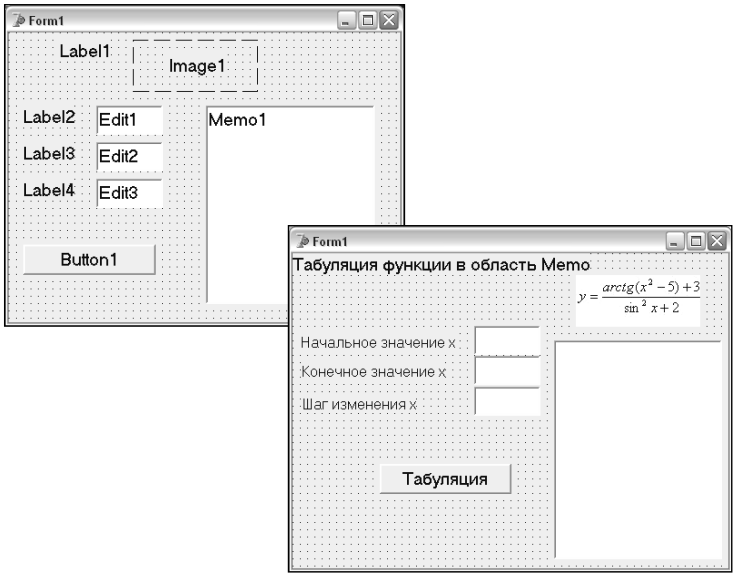


Рис. 18. **Окно формы проекта табулирования функции в область *Memo***

3. Сохраните проект в папке *Табуляция в Memo*.

4. Алгоритм табулирования функции является циклическим, т. е. действия по вычислению очередного значения *y*, вывода *x* и *y* в область *Memo* и увеличения текущего *x* на значение шага повторяются до тех пор, пока *x* не превысит конечное значение. При этом перед циклом необходимо присвоить *x* начальное значение и вывести в область *Memo* заголовки столбцов. Схема алгоритма приведена на рис. 19.

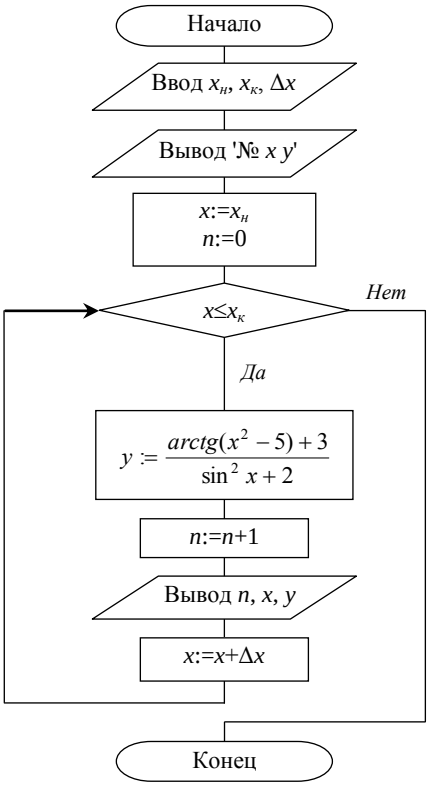


Рис. 19. **Схема алгоритма табулирования функции**

5. После двойного щелчка по кнопке *Табуляция* заполните заготовку процедуры обработки щелчка по данной кнопке. Для составления этой процедуры необходимо использовать *оператор цикла*. В языке программирования *Паскаль* существуют несколько операторов цикла. Наиболее употребительным из них является *оператор цикла с предусловием*, который имеет следующий вид:

while логическое выражение do
  оператор;

Оператор, стоящий после служебного слова *do*, составляет *тело цикла*. Логическое выражение, стоящее после *while*, является *условием повторения цикла*.

Тело цикла повторяется до тех пор, пока условие повторения цикла имеет значение *true* (истина). Как только условие повторения цикла примет значение *false* (ложь), тело цикла пропускается и выполняется оператор, стоящий после цикла.

Текст процедуры данного проекта приведен на рис. 20.

Unit1

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    x,xn,xk,dx,y:real;
    n:integer;
    s1,s2,s3:string;
begin
  Memo1.Clear;
  xn:=StrToFloat(Edit1.Text); // ввод начального значения x
  xk:=StrToFloat(Edit2.Text); // ввод конечного значения x
  dx:=StrToFloat(Edit3.Text); // ввод шага изменения x
  Memo1.Lines.Add(' №  x   y'); // вывод заголовков столбцов
  x:=xn;  // в качестве текущего x берется начальное значение
  n:=0;   //номер текущей строки в области Memo
  while x<=xk do  // проверка условия повторения цикла
    begin
      y:=(arctan(x*x-5)+3)/(sqr(sin(x))+2);
      n:=n+1; // увеличение номера текущей строки на 1
      {в следующих трех операторах значения n, x и y
        преобразуются из числового формата в текстовый}
      s1:=IntToStr(n);
      s2:=FloatToStr(x);
      s3:=FloatToStr(y);
      Memo1.Lines.Add(s1+' '+s2+' '+s3);  // вывод полученных значений
      x:=x+dx  // увеличение  текущего x на значение шага
    end;
end;
```

Рис. 20. **Текст процедуры обработки щелчка по кнопке *Табуляция* с выводом результатов в область *Memo***

6. Сохраните проект и запустите его на выполнение. Введите исходные данные и щелкните по кнопке *Табуляция*. Вариант результата выполнения проекта приведен на рис. 21. Обратите внимание на формат представления чисел в окне *Memo*. В приведенной программе не были заданы форматы для выводимых числовых значений, поэтому СП Delphi выводит результаты в формате, установленном по умолчанию.

7. Для управления форматом представления результатов табуляции можно использовать стандартную процедуру *Str*. Поясним работу этой процедуры на следующих примерах:

- пусть *n* – целая переменная, а *s1* – строковая, тогда в результате выполнения оператора *Str(n:2,s1);* в переменной *s1* будет записано значение *n* двумя символами;
- пусть *x* – вещественная переменная, а *s2* – строковая, тогда в результате выполнения оператора *Str(x:5:1,s2);* в переменной *s2* будет записано значение *x* пятью символами, причем на дробную часть отводится одна позиция, а на знак и целую часть – три позиции (еще одна позиция отводится на десятичную запятую).

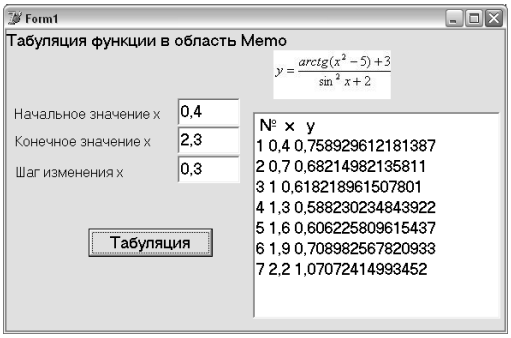


Рис. 21. **Пример результата табулирования**

8. В процедуре, приведенной на рис. 20, замените операторы присваивания, определяющие значения переменных *s1*, *s2* и *s3*, на операторы обращения к процедуре *Str*: *Str(n:2,s1)*; *Str(x:5:1,s2)*; *Str(y:6:2,s3)*.

9. Сохраните проект и снова запустите его на выполнение. Подберите число пробелов в строке *'№ x y'* так, чтобы заголовки располагались над числами. Пример выполнения новой версии проекта представлен на рис. 22.

Рис. 22. **Пример результата табулирования с заданием формата представления результатов**

5.2. Пример разработки проекта табуляции функции в текстовую таблицу

Реализуем задачу табулирования функции, сформулированную в пункте 5.1, с выводом результатов в текстовую таблицу (таблицу строк).

Порядок выполнения работы следующий:

1. Запустите СП Delphi.

2. Расположите в окне формы визуальные компоненты аналогично тому, как это показано на рис. 18, но вместо компонента *Memo* поместите компонент *StringGrid*. Данный компонент находится на вкладке *Additional* палитры компонентов.

3. Установите следующие свойства данного компонента:

- число фиксированных строк (*FixedRows*) – 1;
- число фиксированных столбцов (*FixedCols*) – 0;
- число столбцов в таблице (*ColCount*) – 3;
- число строк в таблице (*RowCount*) – 20;
- запрет ввода информации в ячейки с клавиатуры (*Options – goEditing – False*).

Искомое окно формы приведено на рис. 23.

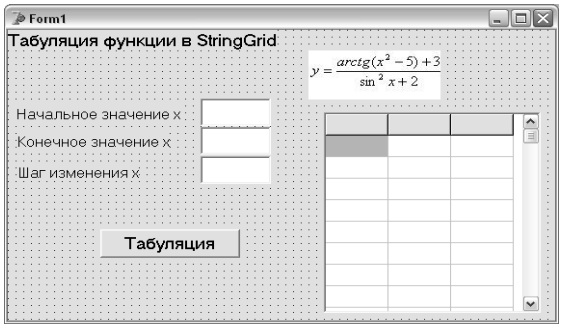


Рис. 23. **Окно формы с компонентом *StringGrid***

4. Сохраните проект в папке *Табуляция* в *StringGrid*.

5. Для создаваемого проекта справедлив алгоритм, приведенный на рис. 19, однако для реализации в процедуре блоков вывода здесь потребуются другие операторы. Для занесения информации в ячейки строковой таблицы надо иметь в виду, что нумерация столбцов и строк начинается с нуля. Ячейка таблицы, находящаяся на пересечении столбца с номером *j* и строки с номером *i*, идентифицируется как *StringGrid1.Cells[j,i]*.

6. В результате замены в процедуре (см. рис. 20) каждого оператора вывода в область *Memo* тремя операторами присваивания ячейкам таблицы получим процедуру, приведенную на рис. 24.

Unit1

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    x,xn,xk,dx,y:real;
    n:integer;
    s1,s2,s3:string;
begin
  xn:=StrToFloat(Edit1.Text); // ввод начального значения x
  xk:=StrToFloat(Edit2.Text); // ввод конечного значения x
  dx:=StrToFloat(Edit3.Text); // ввод шага изменения x
  // вывод заголовков столбцов
  StringGrid1.Cells[0,0]:=' №';
  StringGrid1.Cells[1,0]:='  x';
  StringGrid1.Cells[2,0]:='     y';
  x:=xn;  // в качестве текущего x берется начальное значение
  n:=0;   //номер текущей строки в области Memo
  while x<=xk do  // проверка условия повторения цикла
    begin
      y:=(arctan(x*x-5)+3)/(sqr(sin(x))+2);
      n:=n+1; // увеличение номера текущей строки на 1
      {в следующих трех операторах значения n, x и y
        преобразуются из числового формата в текстовый}
      Str(n:2,s1);
      Str(x:5:1,s2);
      Str(y:6:2,s3);
      // вывод полученных значений
      StringGrid1.Cells[0,n]:=s1;
      StringGrid1.Cells[1,n]:=s2;
      StringGrid1.Cells[2,n]:=s3;
      x:=x+dx  // увеличение  текущего x на значение шага
    end;
end;
```

Рис. 24. **Текст процедуры обработки щелчка по кнопке *Табуляция* с выводом результатов в текстовую таблицу**

7. Сохраните проект и запустите его на выполнение. Введите исходные данные и щелкните по кнопке Табуляция. Вариант результата выполнения проекта приведен на рис. 25.

Обратите внимание, что в результирующей таблице на рис. 25 отсутствует строка с *x=1,3*. Это обстоятельство связано с некорректностью записи условия повторения цикла в виде *x<=xk* (см. рис. 24). Дело в том, что вещественные числа представляются в памяти компьютера приближенно, и результат выполнения оператора *x:=x+dx* при *x=1,2* и *dx=0,1* может получиться 1,30000000000000 или 1,29999999999999, или 1,30000000000001 (точность округления чисел типа *real*). Если *x* получится равным 1,30000000000001, то значение выражения *x<=xk* будет *false* (ложь), и следовательно, цикл расчета значений функции завершится.

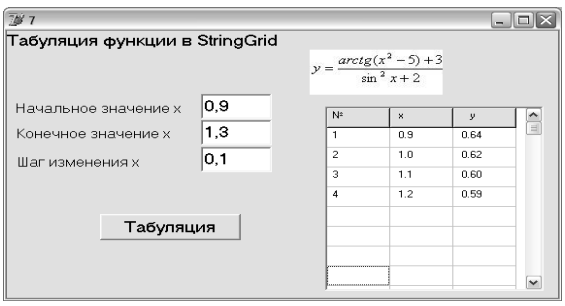


Рис. 25. **Пример результата выполнения проекта *Табуляция в StringGrid***

Для нейтрализации указанного эффекта округления вместо выражения *x<=xk* можно в качестве условия повторения цикла использовать *x<xk+0,0001*. В этом случае тело цикла будет выполняться при любом из вышеуказанных приближенных значений числа 1,3.

Таким образом, в тексте процедуры обработки щелчка по кнопке *Табуляция* оператор цикла *while x<=xk do* (см. рис. 24) надо заменить оператором *while x<xk+0,0001 do*.

5.3. Пример использования арифметического оператора цикла для табулирования функции в текстовую таблицу

Рассмотрим второй способ организации цикла табуляции, который гарантирует вычисление функции при всех значениях аргумента из заданного диапазона. Этот способ основывается на следующей формуле, определяющей количество значений аргумента в заданном диапазоне:

$$n = \left[\frac{xk - xn}{dx} \right] + 1.$$

Здесь квадратные скобки означают, что из выражения выделяется целая часть. Зная, для какого числа аргументов надо повторять вычисления функции, мы можем организовать цикл не по выполнению условия *x<xk+0,0001*, а с помощью арифметического оператора цикла.

Общий вид арифметического оператора цикла следующий:

```
for i:=k to m do
  оператор;
```

В данном фрагменте программы *i* – некоторая целая переменная, а *k* и *m* – целые выражения.

Оператор, стоящий после служебного слова *do*, составляет *тело цикла*.

Тело цикла выполняется сначала при *i=k*, затем при *i=k+1*, потом при *i=k+2* и т. д. Последний раз тело цикла выполняется при *i=n*.

Реализуем задачу табулирования функции, сформулированную в пункте 5.1, с выводом результатов в текстовую таблицу и с использованием арифметического оператора цикла.

Поскольку интерфейс нового проекта тот же, что и в предыдущем примере (см. рис. 23), то необходимо создать копию папки *Табуляция в StringGrid* и переименовать ее в папку *Табуляция2 в StringGrid*.

Для создаваемого проекта надо составить новую схему алгоритма с учетом организации цикла по номеру очередного значения аргумента. Схема алгоритма приведена на рис. 26.

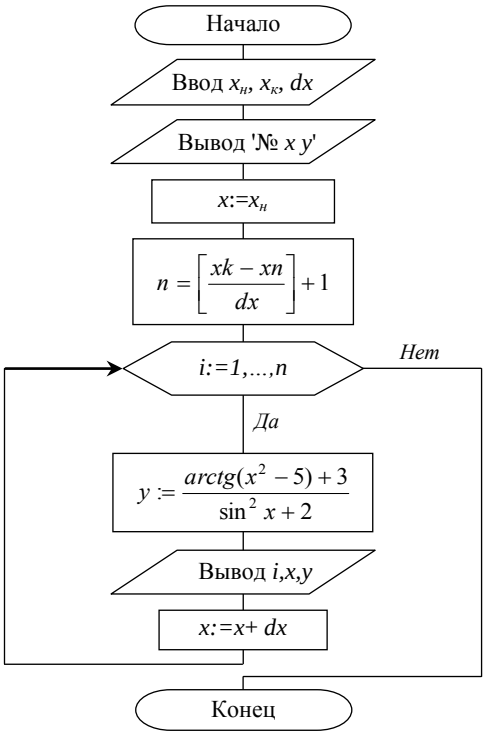


Рис. 26. **Схема алгоритма табулирования функции с использованием арифметического оператора цикла**

Порядок выполнения работы следующий:

1. Запустите СП Delphi.

2. Откройте проект *Табуляция2 в StringGrid*.

3. Внесите изменения в процедуру обработки щелчка по кнопке *Табуляция* в соответствии со схемой алгоритма (рис. 26). В итоге получится процедура, результат которой приведен на рис. 27.

4. Сохраните проект и запустите его на выполнение.

5. Введите исходные данные и щелкните по кнопке *Табуляция*.

Вариант результата выполнения проекта приведен на рис. 25.

Unit1

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    x,xn,xk,dx,y:real;
    i,n:integer;
    s1,s2,s3:string;
begin
  xn:=StrToFloat(Edit1.Text); // ввод начального значения x
  xk:=StrToFloat(Edit2.Text); // ввод конечного значения x
  dx:=StrToFloat(Edit3.Text); // ввод шага изменения x
  // вывод заголовков столбцов
  StringGrid1.Cells[0,0]:=' №';
  StringGrid1.Cells[1,0]:='  x';
  StringGrid1.Cells[2,0]:='     y';
  x:=xn;  // в качестве текущего x берется начальное значение
  n:=trunc((xk-xn)/dx)+1;   //количество значений аргумента
  for i:=1 to n do  // проверка условия повторения цикла
    begin
      y:=(arctan(x*x-5)+3)/(sqr(sin(x))+2);
      {в следующих трех операторах значения n, x и y
        преобразуются из числового формата в текстовый}
      Str(i:2,s1);
      Str(x:5:1,s2);
      Str(y:6:2,s3);
      // вывод полученных значений
      StringGrid1.Cells[0,i]:=s1;
      StringGrid1.Cells[1,i]:=s2;
      StringGrid1.Cells[2,i]:=s3;
      x:=x+dx  // увеличение  текущего x на значение шага
    end;
end;
```

Рис. 27. **Использование арифметического оператора цикла для табуляции функции**

6. ОБРАБОТКА ОДНОМЕРНЫХ ЧИСЛОВЫХ МАССИВОВ

6.1. Краткие сведения о работе с массивом чисел

Числовым массивом (массивом чисел) называется совокупность числовых переменных одного типа (элементов массива), каждой из которых поставлено в соответствие целое число, определяющее порядковый номер этой переменной в массиве. Указанный порядковый номер называется *индексом элемента массива*.

СП Delphi в соответствии с описанием массива отводит для хранения значений его элементов последовательность ячеек памяти. Размер ячейки в байтах зависит от объявленного типа массива.

Ниже приведен пример описания числовых массивов:

```
var
  a: array[1..20]of real; {массив из 20 вещественных чисел с нумерацией от 1 до 20}
  b: array[0..5]of integer; {массив из 6 целых чисел, пронумерованных от 0 до 5}
  x,y: array[1..3]of real; {два массива, каждый из которых содержит по три вещественных числа с индексами от 1 до 3}
```

Операции с числовыми массивами производятся поэлементно. Например, оператор присваивания *x[1]:=-4,2;* изменяет значение первого элемента массива. Для обнуления всех элементов массива можно написать следующий цикл:

```
for i:=1 to 20 do
  a[i]:=0;
```

Организацию ввода и вывода элементов массива рассмотрим на примере следующей задачи, приведенной ниже.

Задача. Дан массив целых чисел. Предполагается, что в массиве не более 20 чисел. Требуется увеличить каждый его элемент в два раза.

Интерфейс задачи представлен на рис. 28. Для свойств объекта *StringGrid1* и *StringGrid2* надо установить следующие значения:

- число фиксированных строк (*FixedRows*) – 0;
- число фиксированных столбцов (*FixedCols*) – 0;
- число столбцов в таблице (*ColCount*) – 1;
- число строк в таблице (*RowCount*) – 20.

Для объекта *StringGrid1* свойства *goEditing* и *goTabs* надо установить значение *true*. Для объекта *StringGrid2* эти свойства должны иметь значение *false*.

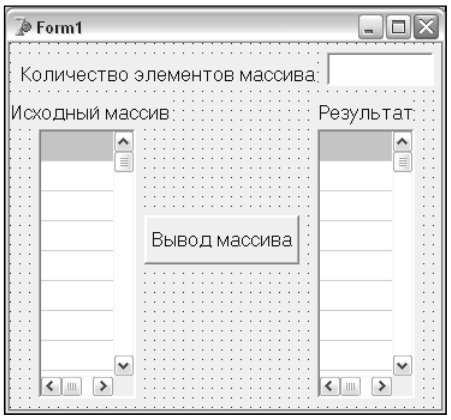


Рис. 28. **Интерфейс проекта ввода-вывода массива**

На рис. 29 представлены эквивалентные изображения цикла ввода элементов массива.

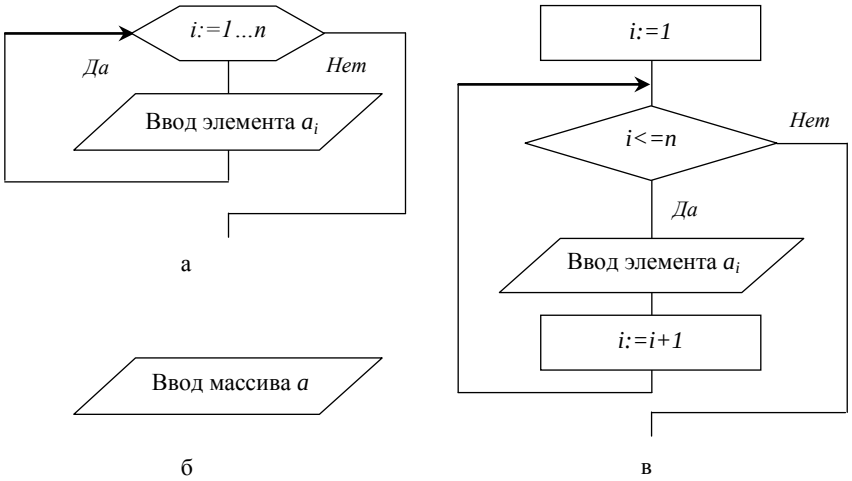


Рис. 29. **Варианты изображения алгоритма ввода элементов массива:**
а – на основе блока *Подготовка*; б – краткая форма;
в – на основе блока *Решение*

Схема алгоритма решения задачи приведена на рис. 30.

Для решения искомой задачи достаточно составить процедуру, обрабатывающую щелчок по кнопке *Button1*. Текст этой процедуры приведен на рис. 31.

Для отладки достаточно одного теста при условии, что в массиве больше одного элемента. Пример выполнения проекта приведен на рис. 32.

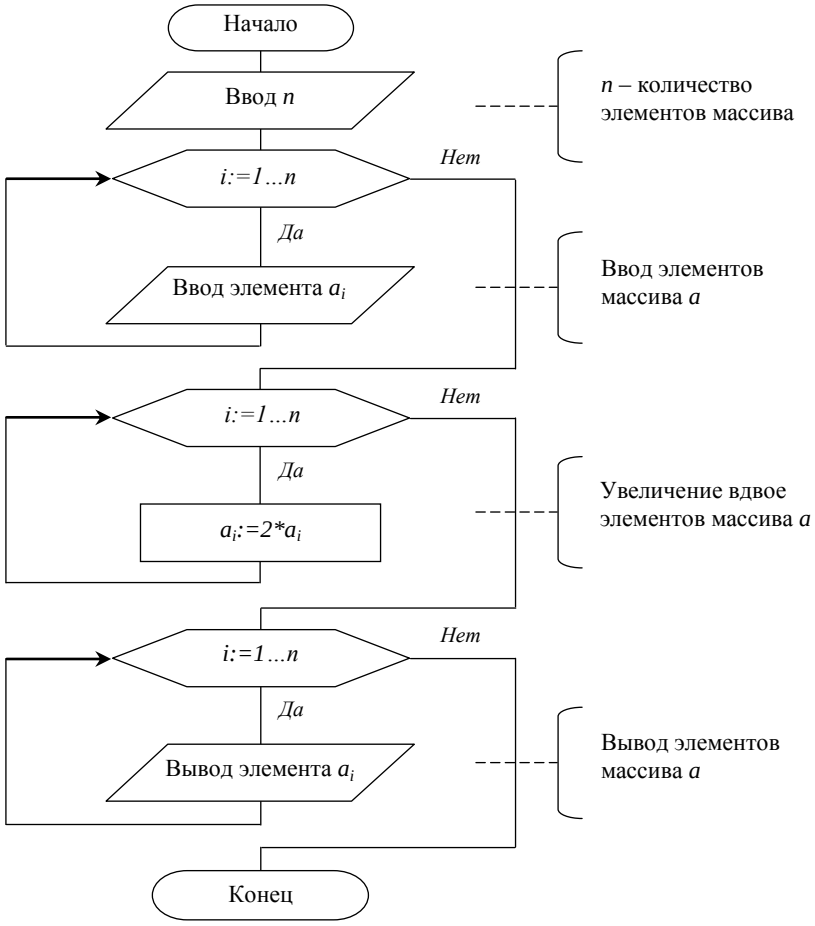


Рис. 30. **Схема алгоритма удвоения элементов массива**

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    n,i:integer;
    a:array[1..20] of integer;
begin
  n:=StrToInt(Edit1.Text); //число элементов массива
  for i:=1 to n do
  {в этом цикле происходит заполнение массива а
  числами, введенными в ячейки текстовой таблицы StringGrid1}
    a[i]:=StrToInt(StringGrid1.Cells[0,i-1]);
  for i:=1 to n do //цикл изменения значений элементов массива
    a[i]:=2*a[i];
  for i:=1 to n do //цикл вывода массива в таблицу StringGrid2
    StringGrid2.Cells[0,i-1]:=IntToStr(a[i]);
end;
```

Рис. 31. **Текст процедуры удвоения значений элементов массива**

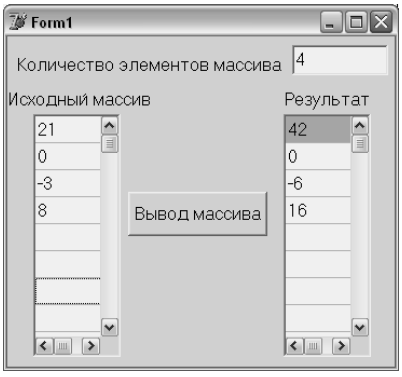


Рис. 32. **Пример выполнения проекта удвоения значений элементов массива**

6.2. Пример создания проекта по обработке числового массива

Технологию обработки числового массива рассмотрим на примере задачи, приведенной ниже.

Задача. В заданном целочисленном массиве вычислить сумму элементов, не меньших заданного числа и стоящих на четных местах. Кроме того, надо определить значение и местоположение последнего максимального элемента массива.

Интерфейс задачи определяется окном формы, вид которого представлен на рис. 33.

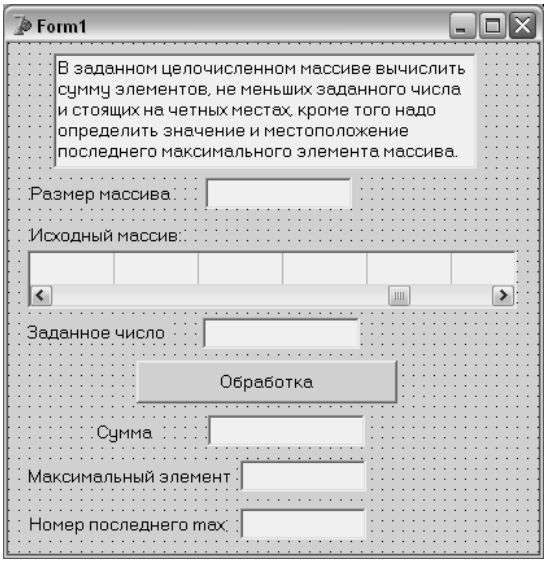


Рис. 33. **Окно формы задачи**

Исходный массив задается с помощью объекта *StringGrid1*, в свойствах которого указана одна строка, 10 столбцов и отсутствие фиксированных строк и столбцов.

Схема алгоритма решения задачи приведена на рис. 34.

Текст процедуры обработки щелчка по кнопке *Обработка* приведен на рис. 35.

Для отладки проекта можно сконструировать следующий тест: на четных местах в массиве расположить элементы большие, меньшие и равные заданному числу, при этом в массив поместить несколько максимальных чисел.

Пример результата выполнения проекта для указанного теста приведен на рис. 36.

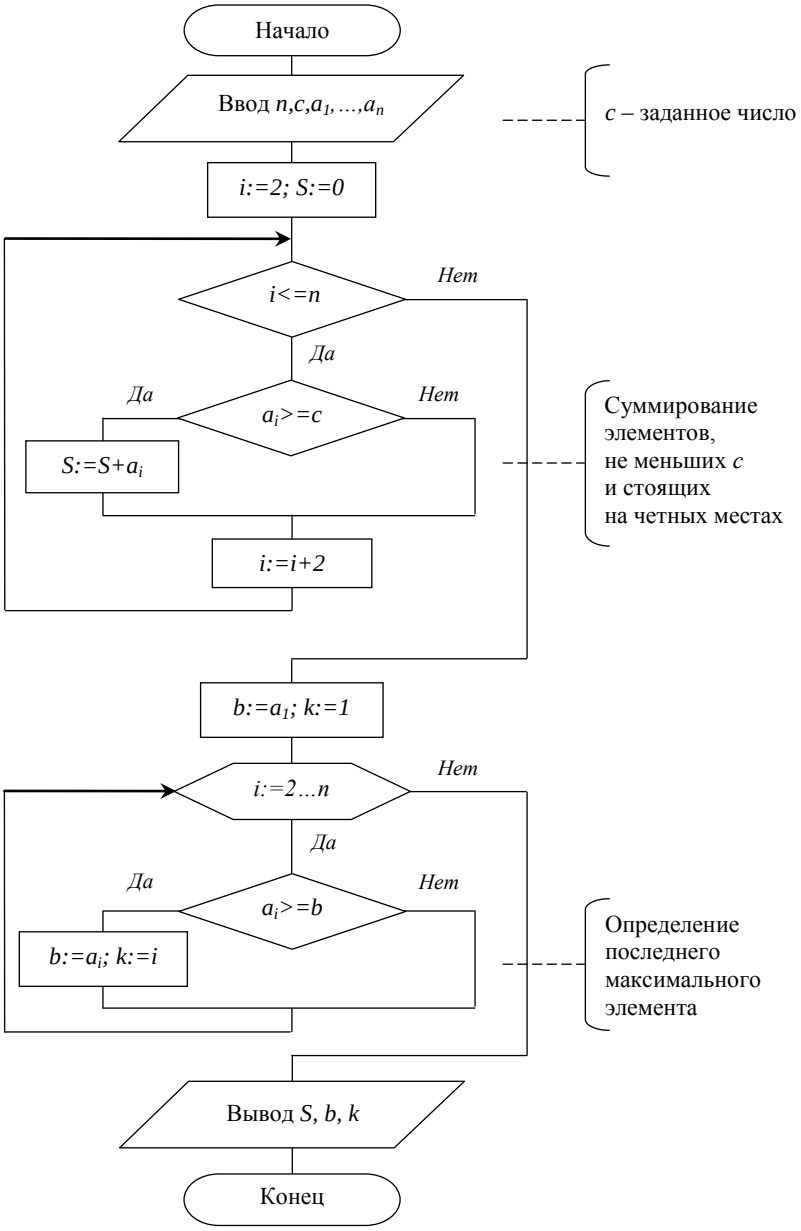


Рис. 34. **Схема алгоритма решения задачи**

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    n,c,i,S,b,k:integer;
    a:array [1..20] of integer;
begin
   n:=StrToInt(Edit1.Text);
   c:=StrToInt(Edit2.Text);
   for i:=1 to n do
     a[i]:=StrToInt(StringGrid1.Cells[i-1,0]);
   S:=0;
   i:=2;
   while i<=n do
   { цикл накопления суммы элементов,
     не меньших b и стоящих на четных местах}
     begin
       if a[i]>=c
         then S:=S+a[i];
       i:=i+2
     end;
   b:=a[1]; k:=1;
   for i:=2 to n do
   { цикл определения последнего максимального}
     if a[i]>=b
       then
         begin
           b:=a[i]; k:=i
         end;
   Edit3.Text:=IntToStr(S);
   Edit4.Text:=IntToStr(b);
   Edit5.Text:=IntToStr(k);
end;
```

Рис. 35. **Текст процедуры обработки щелчка по кнопке *Обработка***

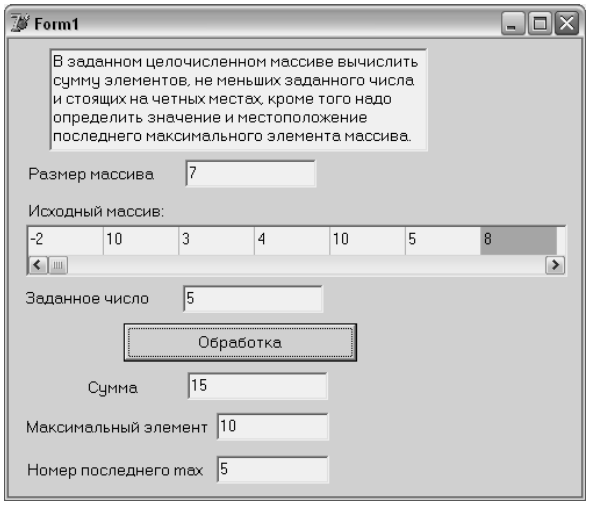


Рис. 36. **Пример выполнения заданного проекта**

7. ФЛАЖКИ И ПЕРЕКЛЮЧАТЕЛИ

7.1. Основные свойства объектов *Флажок, Переключатель* и *Группа переключателей*

Все три рассматриваемых здесь объекта находятся на странице *Standart* палитры компонентов.

Компонент *Флажок* (*CheckBox*) используется для указания, установлен или не установлен данный параметр. После размещения визуального компонента *Флажок* в окне формы возникнет контейнер (в виде квадрата) для отметок с заголовком, как показано на рис. 37.

Рис. 37. **Вид компонента *Флажок* после размещения в окне формы**

В окне формы может быть установлено несколько компонентов *CheckBox*, причем состояние любого из них никак не зависит от состояния остальных. Установка параметра отмечается флажком, который имеет вид птички, в контейнере для отметок (квадрат).

Основные свойства объекта *CheckBox* (флажок) следующие:

- *Caption* – заголовок.
- *Alignment* – возможные значения *taLeftJustify* или *taRightJustify* – расположение заголовка соответственно слева или справа относительно контейнера для отметок.
- *Checked* – возможные значения *true* или *false*, т. е. флажок соответственно включен или выключен.

Компонент *Перелючатель* (*RadioButton)* предназначен для выбора одного из нескольких взаимоисключающих параметров. После размещения визуального компонента *Перелючатель* в окне формы возникнет объект (радиокнопка с заголовком), вид которого представлен на рис. 38.

Рис. 38. **Вид компонента *Переключатель* после размещения в окне формы**

В окне формы должно быть размещено не менее двух переключателей. Выбранный параметр отмечается точкой в кружке.

Основные свойства объекта *RadioButton* (переключатель) такие же, как и у флажка:

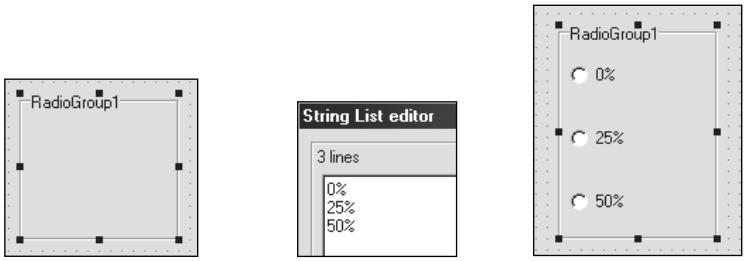
- *Caption* – заголовок.
- *Alignment* – возможные значения *taLeftJustify* или *taRightJustify* – расположение заголовка соответственно слева или справа относительно радиокнопки.
- *Checked* – возможные значения *true* или *false*, т. е. кнопка включена или выключена соответственно.

При установке значения *true* для свойства *Checked* одного из переключателей у всех остальных переключателей этой группы свойство *Checked* автоматически принимает значение *false*.

Назначение объекта *Группа переключателей* (*RadioGroup*) заключается в следующем:

- разделении всех переключателей, находящихся в окне форме, на независимо работающие группы;
- упрощении обработки переключательного списка за счет обеспечения доступа по индексу переключателя.

После размещения компонента в окне формы возникнет объект, приведенный на рис. 39, на котором размещаются переключатели. Размещение переключателей в области группы выполняется с помощью свойства *Items*.



а б в

Рис. 39. **Создание компонента *Группа переключателей*:** а – первоначальный вид компонента; б – окно задания имен переключателей; в – окончательный вид компонента

Items содержит список строк с заголовками переключателей, включенных в группу. Для включения переключателя в группу надо щелкнуть кнопкой мыши справа от названия свойства *Items* и в открывшемся окне *String List editor* (см. рис. 39) набрать текст заголовка включаемого переключателя. После окончания набора в области *RadioGroup* появятся переключатели с соответствующими заголовками (рис. 39).

Примечания:

1. Для исключения переключателя из группы надо удалить заголовок переключателя из списка *Items*.

2. Компоненты *RadioButton*, которые помещены в область созданной *RadioGroup* не через список *Items*, а с помощью обычной технологии (щелчок мыши), не будут относиться к данной группе переключателей, а войдут в независимую группу переключателей наряду с переключателями, располагающимися в окне формы вне области *RadioGroup*.

Кроме свойства *Items*, к основным свойствам объекта *RadioGroup* относятся следующие свойства:

- *Caption* – заголовок.
- *ItemIndex* – индекс установленного переключателя, нумерация которого начинается с 0. После образования группы данный индекс принимает значение –1 (минус единица). При замене этого значения на 0 первый переключатель перейдет в состояние *установлен*. Если оставить свойству *ItemIndex* исходное значение (–1), то после запуска проекта все переключатели группы будут в состоянии *не установлен*. Именно этот случай представлен на рис. 39.
- *Columns* – число столбцов, в которых представлены переключатели в области *RadioGroup*. Если значение этого свойства равно единице, то объект *RadioGroup* примет вид, изображенный на рис. 39.

7.2. Пример использования объектов *Флажок* и *Группа переключателей*

Задача. Определить размер платы за гостиницу при условии, что за одноместный номер взимается 100% от базовой величины, двухместный – 75, трехместный – 50%. При этом для участников войны делается скидка в размере 15%, а в случае предоплаты – дополнительная скидка 10%.

Для создания интерфейса проекта поместим в окно формы следующие визуальные компоненты для ввода исходных данных:

- компоненты *Label1* и *Edit1* для ввода базовой величины;
- компонент *RadioGroup1* для задания типа номера;
- объекты *CheckBox1* и *CheckBox2* для отметки полагающихся клиенту льгот.

В окно формы поместим также командную кнопку *Button1* и поле *Edit2* с надписью *Label2* для вывода результата.

Установим следующие значения свойства *Caption*:

- для объекта *Label1* и *Label2* – значения *Базовая величина* и *Плата за проживание* соответственно;
- для объектов *CheckBox1* и *CheckBox2* – *Ветеран* и *Предоплата* соответственно;
- для объекта *RadioGroup1* – *Тип номера*;
- для объекта *Button1* – *Расчет*.

Интерфейс проекта приведен на рис. 40.

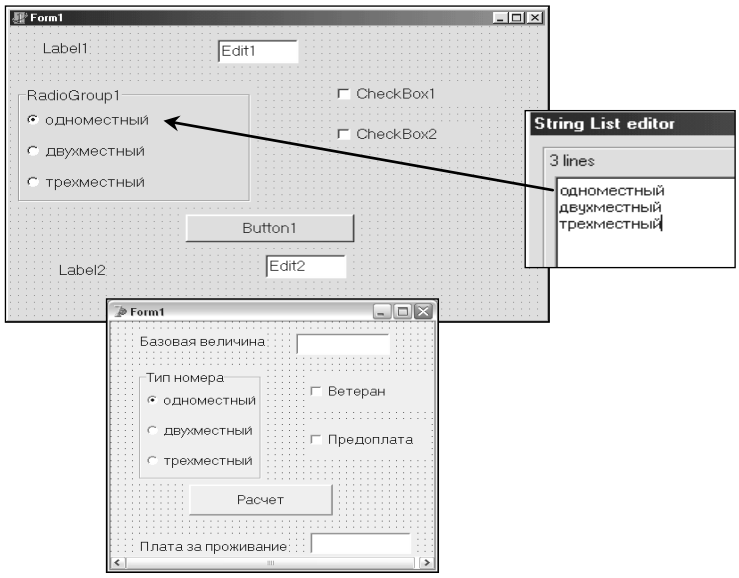


Рис. 40. **Интерфейс проекта, использующего флажки и группу переключателей**

В данном случае переключатели появились в области *RadioGroup1* после набора их заголовков в списке свойства *Items*. В результате стало возможным использовать значение свойства *ItemIndex*. Установим значение этого свойства, равное нулю. В результате будет включен первый переключатель, а остальные два будут выключены. Свойству *Columns* присвоим значение 1.

Свойству *Checked* объектов *CheckBox1* и *CheckBox2* надо присвоить значение *false*, в результате оба флажка будут *не установлены*.

Свойству *ReadOnly* объекта *Edit2* надо присвоить значение *true*, чтобы нельзя было изменить поле результата с клавиатуры.

Схема алгоритма решения задачи приведена на рис. 41.

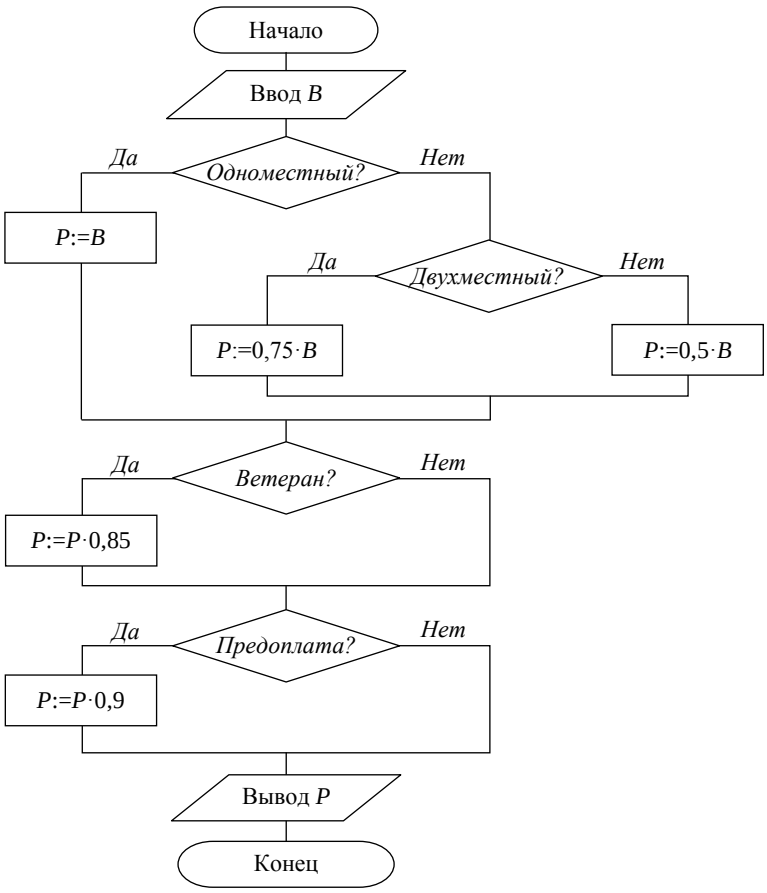


Рис. 41. **Схема алгоритма расчета платы за проживание**

Текст процедуры для обработки щелчка по кнопке *Расчет* приведен на рис. 42.

Отладка проекта должна быть проведена для следующих тестов:

- количество комнат – одна, скидки для ветеранов нет, предоплаты не было;
- количество комнат – две, действует скидка для ветеранов, предоплаты не было;
- количество комнат – три, скидки для ветеранов нет, была сделана предоплата;
- количество комнат любое, действует скидка для ветеранов, была сделана предоплата.

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    B,P:real;
begin
  B:=StrToFloat(Edit1.Text);
  if RadioGroup1.ItemIndex=0
    then P:=B
    else
      if RadioGroup1.ItemIndex=1
        then P:=0.75*B
        else P:=0.5*B;
  if CheckBox1.Checked
    then P:=P*0.85;
  if CheckBox2.Checked
    then P:=P*0.9;
  Edit2.Text:=FloatToStr(P)
end;
```

Рис. 42. **Текст процедуры обработки щелчка по кнопке *Расчет* проекта по расчету платы за проживание**

7.3. Использование оператора выбора

В СП Delphi существует специальный оператор выбора, который заменяет несколько условных операторов в тех случаях, когда надо выбрать один из нескольких вариантов значений целочисленной переменной.

Общий вид оператора следующий:

case <целая переменная> of
<значение 1>: <оператор 1>;
<значение 2>: <оператор 2>;
…
<значение n>: <оператор n>
else <оператор n+1>
end;

Если значение целой переменной равно одному из указанных в операторе выбора значений, то выполняется соответствующий оператор, а затем оператор, стоящий после оператора выбора. Если значение целой переменной не совпадает ни с одним из перечисленных значений, то выполняется оператор, стоящий после *else*, а затем оператор, стоящий после оператора выбора.

Существует и вариант оператора выбора без *else*, который имеет следующий вид:

case <целая переменная> of
<значение 1>: <оператор 1>;
<значение 2>: <оператор 2>;
…
<значение n>: <оператор n>
end;

Проиллюстрируем использование оператора выбора на примере задачи расчета платы за проживание, рассмотренной в пункте 7.2.

Текст процедуры для обработки щелчка по кнопке *Расчет* с использованием оператора выбора представлен на рис. 43.

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    B,P:real;
begin
  B:=StrToFloat(Edit1.Text);
  case RadioGroup1.ItemIndex of
    0: P:=B;
    1: P:=0.75*B;
    2: P:=0.5*B
  end;
  if CheckBox1.Checked
    then P:=P*0.85;
  if CheckBox2.Checked
    then P:=P*0.9;
  Edit2.Text:=FloatToStr(P)
end;
```

Рис. 43. **Использование оператора выбора в процедуре обработки щелчка по кнопке *Расчет* проекта по расчету платы за проживание**

Схема алгоритма задачи с использованием конструкции *Выбор* представлена на рис. 44.

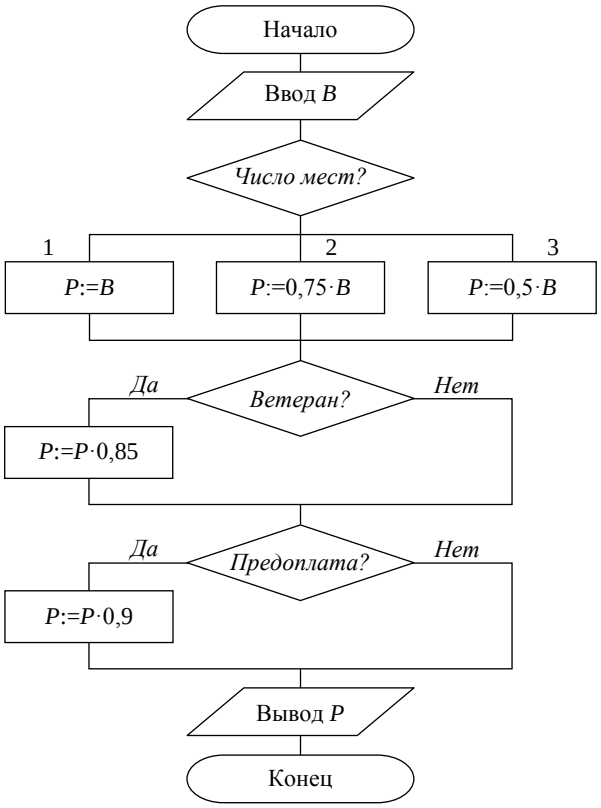


Рис. 44. **Схема алгоритма расчета платы за обучения с элементом *Выбор***

ЗАДАНИЯ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа 1.
Основные команды, используемые при создании проекта

Цели работы:

1. Получить представление о визуальном программировании и о технологии работы в системе программирования Delphi.

2. Ознакомиться с объектами *форма* (*Form*), *текстовое поле* или *надпись* (*Label*), *поле редактирования* (*Edit*), *кнопка* (*Button*) и их основными свойствами.

Порядок выполнения работы

1. На диске *D* в папке *STUD* создайте папку, в качестве ее имени укажите шифр Вашей группы. Вложи-

те в созданную папку еще одну папку и в качестве ее имени укажите Вашу фамилию. В последней папке будут сохраняться все проекты, разрабатываемые на лабораторных занятиях. Файлы каждого проекта целесообразно сохранять в отдельной папке. Для сохранения первого проекта создайте в Вашей папке папку *Пустой проект*.

2. Запустите в работу СП Delphi. Убедитесь в том, что на экране появились четыре окна: окно дерева объектов, окно инспектора объектов, окно браузера (редактора) кода и окно формы.

Если какого-то окна нет на экране, то его можно вывести на экран с помощью соответствующих команд меню *View*:

- окно инспектора объектов → *View* / *Object Inspector (F11)*;
- окно браузера кода → *View* / *Code Explorer*;
- окно формы → *View* / *Forms…(Shift+F12)*;
- окно дерева объектов → *View* / *ObjectTreeView*.

Обратите внимание на то, что окно формы называется *Form1*, окно браузера кода – *Unit1*, а в названии главного окна СП Delphi содержится имя проекта *Project1*.

3. Сохраните проект в Вашей папке. Для этого выполните команду *File* / *Save All*. В окне *Save Unit1 As* сохранения модуля не изменяйте установленное по умолчанию имя *Unit1.pas*.

В следующем окне *Save Project As* не изменяйте установленное по умолчанию имя *Project1.dpr*. Убедитесь в том, что после сохранения проекта автоматически создались следующие файлы: *Project1.dpr, Project1.cfg, Project1.res, Project1.dof, Unit1.pas, Unit1.dfm*.

4. Выполните переключение между окном формы и окном соответствующего ей модуля с помощью команды *View* / *Toggle Form* / *Unit* или нажатием клавиши *F12*.

5. Просмотрите текстовый файл формы *Unit1.dfm*. Этот файл можно просмотреть с помощью команды *View as Text* контекстного меню формы. В тексте файла указаны установленные по умолчанию значения параметров окна формы (ширина, высота, цвет и т. д.).

Для возвращения к графическому изображению формы выполните команду *View as Form* контекстного меню текстового файла формы.

6. Выведите окно проекта и просмотрите его содержимое. Окно проекта можно вывести на экран с помощью команды *Project* / *View Source*. В окне проекта находится заготовка текста головной программы проекта. Окно проекта содержит две вкладки: *Unit1* и *Project1*.

7. Откройте окно редактора кода. Это можно сделать с помощью вкладки *Unit1* окна проекта. Просмотрите текст кода модуля, содержащего заготовку модуля *Unit1*. Закрытие окна редактора кода вызывает закрытие файла проекта и приводит к окончанию работы с ним. Для продолжения работы с закрытым проектом его необходимо будет открыть с помощью команды *Open Project*.

8. Завершите работу с Delphi путем закрытия главного окна Delphi.

9. Продемонстрируйте и прокомментируйте преподавателю содержимое созданной папки *Пустой проект*.

Лабораторная работа 2. Создание проекта *Приветствие*

Цель работы: разработка проекта, использующего операторы ввода, вывода и присваивания.

Порядок выполнения работы

1. Создайте в Вашей папке новую папку *Привет* для нового проекта. Запустите СП Delphi и сохраните файлы проекта со стандартными именами.

2. Запустите программу на выполнение с помощью команды *Run* / *Run* главного меню или щелкните по кнопке *Run* панели инструментов (ПИ), или нажмите функциональную клавишу *F9*. На экране появится *окно приложения*. Вид этого окна в данном случае практически совпадает с окном формы, так как оно пока пустое. Единственное отличие – вид ярлыка в левом верхнем углу окна. В окне приложения ярлык имеет вид горящего факела, а в ярлыке окна формы факел погашен.

3. Поупражняйтесь с окном приложения и убедитесь в том, что оно обладает всеми свойствами стандартного окна операционной системы Windows. Закройте окно приложения.

4. Ознакомьтесь со свойствами форм *Left, Top, Width, Height* (координаты левого верхнего угла, его ширина и высота). Для этого перемещайте окно формы с помощью мыши по экрану и следите за изменением значений свойств *Left* и *Top* в окне *Object Inspector*, изменяйте размеры формы и следите за значениями свойств *Width* и *Height* в окне *Object Inspector*. Затем изменяйте значения свойств *Left, Top, Width, Height* в окне *Object Inspector* и следите за положением формы на экране и ее размерами.

5. Измените цвет фона формы, для чего в окне свойств формы *Object Inspector* в строке *Color* выберите значение цвета фона.

6. Выполните программу еще раз, при этом появится пустая форма другого цвета, закройте ее.

7. Откройте окно модуля *Unit1.pas* и просмотрите его текст. Текст модуля имеет вид шаблона.

8. Вставьте в окно формы текстовое поле (объект типа *Label*) с текстом *Приветствие студента*. Для этого активизируйте окно формы и выполните следующие действия:

• перейдите на вкладку *Standard* палитры компонентов и дважды щелкните кнопкой мыши по кнопке *Label*;

• появившийся объект с текстом *Label1* перетяните мышью к верхней границы окна формы;

• в окне *Object Inspector* измените значение свойства *Caption* с *Label1* на текст *Приветствие студента*;

• установите следующие значения свойства *Font* (шрифт) текстового поля *Label1*: *Times New Roman*, полужирный, размер 16, цвет пурпурный.

9. Аналогично вставьте в форму еще три текстовых поля *Label2*, *Label3* и *Label4*. Замените значения, установленные по умолчанию, свойства *Caption* первых двух полей следующими: *Введите имя и фамилию, Введите группу*, соответственно. Расположите их на форме по своему усмотрению. Значения свойства *Font* этих полей установите также по своему усмотрению. В качестве значения свойства *Caption* объекта *Label4* установите пустую строку, т. е. сотрите установленное по умолчанию значение *Label4.*

10. Запустите программу на выполнение. Обратите внимание на тот факт, что в процессе выполнения никакие свойства формы и визуальных компонентов изменить нельзя. Изменить можно только положение формы на экране и ее размеры. Закройте форму.

11. Сделайте активным окно модуля *Unit1.pas*. Обратите внимание на строки, автоматически добавленные в текст модуля.

12. Вставьте в форму два поля редактирования – объекты *Edit1* и *Edit2*. Для этого перейдите на вкладку *Standard* палитры компонентов, активируйте окно формы и щелкните кнопкой мыши по кнопке *Edit*, а затем – в нужном месте формы. Вставьте второй объект *Edit2* в форму. Для каждого из этих объектов в качестве значения свойства *Text* задайте пустую строку, т. е. сотрите значения по умолчанию. Запустите программу и поупражняйтесь со вставленными объектами, для чего щелкните кнопкой мыши в поле редактирования, введите текст, удалите его. Закройте окно программы.

13. Сделайте активным окно модуля *Unit1.pas*. Обратите внимание на строки, которые автоматически добавились в текст модуля.

14. Создайте в окне формы кнопку для вывода приветствия – объект *Button1*. Для этого перейдите на вкладку *Standard* палитры компонентов и щелкните кнопкой мыши по кнопке *Button*, а затем – в нужном месте формы. Изменением значения свойства *Caption* поменяйте надпись на кнопке *Button* на текст *Нажмите здесь*. Запустите программу, щелкните по кнопке и убедитесь в отсутствии реакции на это событие. Чтобы выполнились какие-либо действия, надо написать процедуру обработки события *OnClick* (щелчок по кнопке). Закройте окно программы.

15. Для создания процедуры обработки события *OnClick* надо выполнить двойной щелчок мышью по кнопке *Нажмите здесь* или справа от события *OnClick* на вкладке *Events* инспектора объектов. Выполните двойной щелчок одним из указанных способов. На экране появится окно редактора модуля с видимым шаблоном процедуры:

```
procedure TForm1.Button1Click(Sender: TObject):
begin
end;
```

После первой строки этого шаблона вставьте три следующие строки:

```
var
  im:string;
  gr:string;
```

Этими строками в процедуре создается раздел (начинается словом *var*) описания переменных. В данном случае описываются две переменные *im* и *gr* строкового типа (*string*) для сохранения в памяти компьютера вводимых пользователем с клавиатуры во время выполнения программы имени и группы.

После слова *begin* вставьте следующие три строки:

```
im:=edit1.Text;
gr:=edit2.Text;
Label4.Caption:= 'Привет '+im+' из группы '+gr+' !';
```

В этих строках записаны операторы присваивания. После выполнения первого оператора *im:=edit1.Text;* переменная *im* примет значение текста, введенного в поле объекта *Edit1*. После выполнения второго оператора *gr:=edit2.Text;* переменная *gr* примет значение текста, введенного в поле объекта *Edit2*. После выполнения третьего оператора *Label4.Caption:= 'Привет '+im+' из группы '+gr+' !';* изменится значение свойства *Caption* объекта *Label4*. При выполнении программы в окно приложения будет выведен текст, составленный из слов *Привет,* введенных ранее имени и фамилии (значения переменной *im*), слов *из группы* и введенного ранее названия группы (*gr*). Обратите внимание на то, что после каждого оператора записывается символ «;».

Получится текст процедуры в следующем виде:

```
procedure TForm1.Button1Click(Sender: TObject):
  var
    im:string;
    gr:string;
begin
    im:=edit1.Text;
    gr:=edit2.Text;
    Label4.Caption:= 'Привет '+im+' из группы '+gr+' !'
end;
```

16. Сохраните созданную на данный момент форму и программу в папке *Привет* с помощью команды *File* / *Save All*. В дальнейшем периодически, в частности, перед очередными запусками проекта на выполнение, сохраняйте файлы проекта по этой команде.

17. Запустите проект на выполнение. Введите имя и фамилию в первое поле, название группы во второе поле и щелкните по кнопке *Нажмите здесь*. В окне приложения будет выведен результат выполнения проекта.

18. Закройте СП Delphi.

19. Запустите на выполнение созданный проект не из среды СП Delphi. Для этого найдите в папке *Привет* файл *Project1.exe* и запустите его на выполнение. Введите новые данные, получите новое приветствие. Закройте форму.

20. Загрузите СП Delphi и откройте созданный проект.

21. Для открытия существующего проекта (файл *Project1.dpr*) используется команда *File* / *Open Project*… или кнопка *Open Project* панели инструментов, или комбинация клавиш *Ctri+F11*.

22. Внесите в проект следующие изменения:

- измените расположение надписей, полей редактирования и кнопок в окне формы;
- измените цвет формы и объектов по своему усмотрению;
- измените содержание приветствия.

23. Сохраните изменения в проекте и продемонстрируйте преподавателю выполнение проекта.

24. Закройте СП Delphi.

Лабораторная работа 3. Создание проекта *Калькулятор*

Цель работы: использование в проекте базовых арифметических операций.

Порядок выполнения работы

1. В папке *Калькулятор* создайте приложение вычисления суммы, разности и произведения двух вещественных чисел *x* и *y*. Интерфейс приложения представлен на рис. 45.

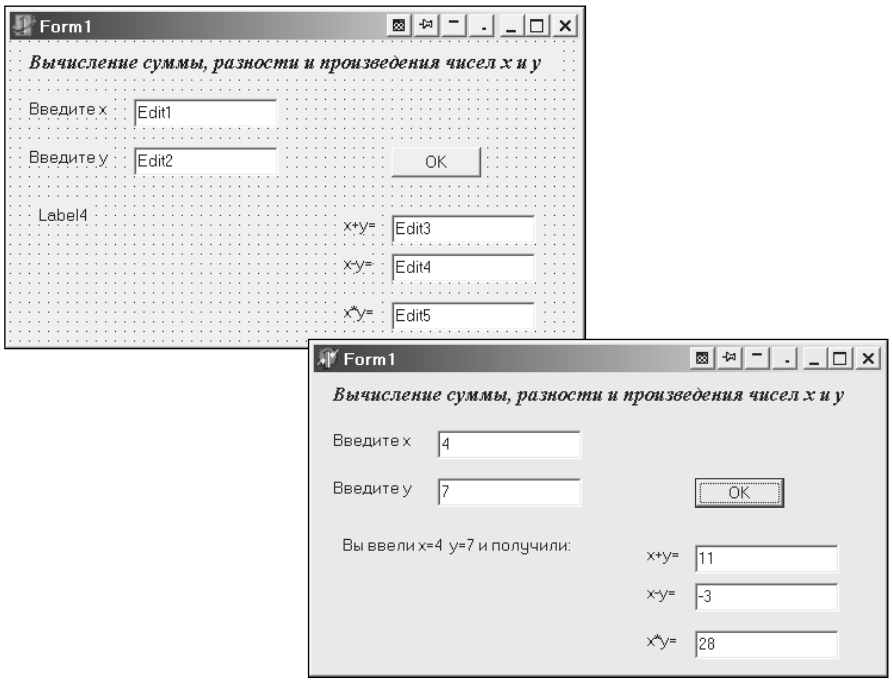


Рис. 45. **Интерфейс проекта *Калькулятор***

2. Необходимо заблокировать возможность ввода данных с клавиатуры в поля *Edit3, Edit4* и *Edit5*, которые предназначены для вывода результатов. С этой целью установите значение *false* для свойства *Enabled* каждого из указанных объектов.

3. Процедура обработки события *OnClick* для кнопки *OK* записывается в модуле после двойного щелчка кнопкой мыши по кнопке *OK* или справа от события *OnClick* на вкладке *Events*. Выполните это действие и в появившийся шаблон процедуры вставьте следующие описания и операторы:

```
procedure TForm1.Button1Click(Sender: TObject):
  var
    x,y,z1, z2, z3:real;
begin
    x:=StrToFloat(Edit1.Text);
    y:=StrToFloat(Edit2.Text);
    Label4.Caption:='Вы ввели x='+FloatToStr(x)+' y='
                    +FloatToStr(y)+' и получили:';
    z1:=x+y;
    z2:=x-y;
    z3:=x*y;
    Edit3.Text:=FloatToStr(z1);
    Edit4.Text:=FloatToStr(z2);
    Edit5.Text:=FloatToStr(z3);
end;
```

4. После создания искомого проекта произведите его отладку.

5. Добавьте в окно формы надпись и текстовое поле для вычисления значения x^y. Добавьте в текст процедуры описание новой переменной и оператор присваивания для вычисления степени. Учтите, что в Delphi нет операции возведения в степень, поэтому искомые вычисления следует выполнить в соответствии с формулой

$$x^y = e^{y \cdot \ln x}.$$

Лабораторная работа 4.
Создание проекта вычисления значений функции

Цель работы: запись арифметических выражений в СП Delphi.

Для варианта, указанного преподавателем из табл. 3, разработайте интерфейс проекта, напишите текст процедуры, вызываемой при щелчке по кнопе *Вычислить*, произведите отладку с помощью табличного процессора MS Excel.

Таблица 3. **Виды функций**

Вариант	Вид функции	Вариант	Вид функции
1	$y = \frac{arctg\ bx}{1+\sin^2 x}$	16	$y = \frac{arctg\ (a+x)}{\sqrt{a^3+b^3}}$
2	$y = \frac{\sin^2 x + a}{\sqrt{x}+bx}$	17	$y = \frac{1+\sqrt{bx}}{0{,}5+\sin^2 ax}$
3	$y = \sqrt{\frac{a+bx}{\ln^2 x}}$	18	$y = \frac{a-e^{bx}}{\ln\lvert 2x\rvert}$
4	$y = \frac{\ln^2(x-b)}{a\sqrt{x}}$	19	$y = \frac{(a+bx)^2}{1+\cos^3 ax}$
5	$y = \frac{a\ln^2 x}{b+\sqrt{x}}$	20	$y = \frac{b+\sin^2 ax}{e^{-x/2}}$
6	$y = \frac{e^{ax}+b}{1+\cos^2 x}$	21	$y = \frac{\sin^2 x - a}{bx}$
7	$y = \frac{a+\sqrt[3]{x}}{\sin^2 bx}$	22	$y = \frac{arctg^2 ax}{b+0{,}5x}$
8	$y = \frac{a\sqrt{\lvert x\rvert}-bx}{\ln^3 x}$	23	$y = \frac{\ln(a^2-x)}{b\sin^2 x}$

Окончание табл. 3

Вариант	Вид функции	Вариант	Вид функции
9	$y = \frac{\sqrt{ax} - b}{tg^2 x}$	24	$y = \frac{a - \sqrt{bx}}{1 + \cos\lvert 2x\rvert}$
10	$y = e^{-x}\frac{a + bx}{\ln^2\lvert x + 1\rvert}$	25	$y = \frac{\ln^2(a + x)}{(b + x)^2}$
11	$y = \frac{tg^2 x - b}{e^{ax}}$	26	$y = \frac{\sqrt{\lvert a\ \ln\ x\rvert}}{1 + tg^2 bx}$
12	$y = \frac{arctg\ bx}{1 + \sqrt[3]{ax}}$	27	$y = \frac{1 + tg^2 x}{b + e^{x/a}}$
13	$y = \frac{\sin^3 ax}{ax + b}$	28	$y = \frac{\cos^2 2x + b}{\sqrt{1 + e^{ax}}}$
14	$y = \frac{e^{-ab}}{b + \cos^3 ax}$	29	$y = \frac{\sqrt{ax + b}}{\ln^2\lvert x\rvert}$
15	$y = \frac{\ln^2\lvert x\rvert + b}{a\sqrt{x}}$	30	$y = \frac{1 + \sin^2 ax}{b^2 + x^2}$

Лабораторная работа 5.
Создание проекта вычисления экономических показателей (линейный алгоритм)

Цель работы: реализация простых экономико-математических моделей.

Для указанного преподавателем варианта разработайте интерфейс проекта, напишите текст процедуры, вызываемой при щелчке по кнопке *Вычислить*, произведите отладку. Данные для отладки подберите самостоятельно. Следует учесть, что интерфейс должен отражать экономическую сущность задачи.

Вариант 1

Рассчитайте плату ($П_x$, р.) за хранение транспортного средства на платной стоянке для заданных значений $Т_м$, $К_{мх}$, N и $К_{сх}$ по формуле

$$П_x = Т_м \cdot \left(К_{мх} + \frac{К_{сх}}{N} \right),$$

где $Т_м$ – тариф на услуги автостоянки за одно машино-место в месяц, р.;
$К_{мх}$ – количество полных месяцев хранения транспортного средства;
N – количество суток в текущем месяце;
$К_{сх}$ – количество суток хранения транспортного средства в текущем месяце.

Вариант 2

Для заданных значений *РКР* и *ГВ* рассчитайте резерв увеличения выпуска продукции предприятия (*РВП*, т) по формуле

$$РВП = РКР \cdot ГВ,$$

где *РКР* – резерв увеличения количества рабочих мест;
ГВ – фактическая среднегодовая выработка производственного персонала, т.

Вариант 3

Определите уровень задолженности предприятия по заработной плате ($К_з$, %) по формуле

$$К_з = \frac{ЗП_{нач} - ЗП_{выпл}}{ЗП_{нач}} \cdot 100,$$

где $ЗП_{нач}$ – сумма начислений заработной платы за определенный период, р.;
$ЗП_{выпл}$ – сумма выплаченной заработной платы за определенный период, р.

Вариант 4

Для заданных значений $СЗ_1$, $СЗ_0$ и $I_ц$ вычислите индекс роста средней заработной платы ($I_{зп}$) в условиях инфляции:

$$I_{зп} = \frac{СЗ_1}{СЗ_0 \cdot I_ц},$$

где $СЗ_1$ – среднегодовая зарплата работников в отчетном году, р.;
$СЗ_0$ – среднегодовая зарплата работников в прошлом году, р.;
$I_ц$ – индекс роста цен на потребительские товары и услуги.

Вариант 5

Для заданных значений *ЗАЦБ* и *СЗБ* вычислите относительное изменение объема в целом по затратам (*ОИО*) по формуле

$$ОИО = \frac{ЗАЦБ}{СЗБ},$$

где *ЗАЦБ* – стоимость затрат анализируемого периода в ценах базового периода, р.;
СЗБ – стоимость затрат в базовом периоде, р.

Вариант 6

Для заданных значений *ЗАЦБ* и *СЗА* вычислите относительное изменение средневзвешенной цены в целом по затратам (*ОИЦ*) по формуле

$$ОИЦ = \frac{СЗА}{ЗАЦБ},$$

где *ЗАЦБ* – стоимость затрат анализируемого периода в ценах базового периода, р.;
СЗА – стоимость затрат в анализируемом периоде, р.

Вариант 7

Для заданных значений *ЗАЦБ, ОА* и *ЦБ* вычислите производительность затрат в целом в анализируемом периоде (*ПЗ*) по формуле

$$ПЗ = \frac{ОА \cdot ЦБ}{ЗАЦБ},$$

где *ЗАЦБ* – стоимость затрат анализируемого периода в ценах базового периода, р.;
ОА – объем товаров в анализируемом периоде, т;
ЦБ – цена одной тонны товара в базисном периоде, р.

Вариант 8

Для заданных значений $СТ^{баз}$, $СТ^{ан}$, $СЭ^{баз}$ и $СЭ^{ан}$ вычислите влияние на прибыль использования электроэнергии (*ВПЭ*, р.) по формуле

$$ВПЭ = СЭ^{баз} \cdot \left(\frac{СТ^{ан}}{СТ^{баз}} - \frac{СЭ^{ан}}{СЭ^{баз}} \right),$$

где $СТ^{баз}$ – стоимость товаров в базисном периоде, р.;
$СТ^{ан}$ – стоимость товаров в анализируемом периоде, р.;
$СЭ^{баз}$ – стоимость электроэнергии в базисном периоде, р.;
$СЭ^{ан}$ – стоимость электроэнергии в анализируемом периоде, р.

Вариант 9

Для заданных значений *a, b* и *x* найдите сумму затрат (*Y*, р.) на эксплуатационное содержание грузового автотранспорта по формуле

$$Y = a + b \cdot x,$$

где *a* – сумма постоянных затрат, р.;
b – сумма переменных затрат на 1 т·км, р.;
x – объем грузооборота, т·км.

Вычислите также себестоимость 1 т·км (C, р.) по формуле

$$C = \frac{a}{x} + b.$$

Вариант 10

Для заданных значений $К_{лик0}$, $К_{лик1}$, $К_{норм}$, $Т_в$ и $Т_о$ рассчитайте коэффициент восстановления платежеспособности предприятия ($К_в$) за заданный период по формуле

$$К_в = \frac{1}{К_{норм}} \cdot \left(К_{лик1} + \frac{Т_в}{Т_о} \cdot \left(К_{лик1} - К_{лик0} \right) \right),$$

где $К_{лик1}$ – фактические значения коэффициента ликвидности в конце отчетного периода;
$К_{лик0}$ – фактические значения коэффициента ликвидности в начале отчетного периода;
$К_{норм}$ – нормативное значение общего коэффициента ликвидности;
$Т_в$ – период восстановления платежеспособности, мес.;
$Т_о$ – отчетный период, мес.

Вариант 11

Рассчитайте средневзвешенные цены затрат в базовом и анализируемом периодах ($Ц^{баз}$ и $Ц^{ан}$, р.) по следующим формулам:

$$Ц^{баз} = \frac{С^{баз}}{О^{баз}};$$

$$Ц^{ан} = \frac{С^{ан}}{О^{ан}},$$

где $С^{баз}$ – стоимость затрат в базовом периоде, р.;
$С^{ан}$ – стоимость затрат в анализируемом периоде, р.;
$О^{баз}$ – общий (условный) объем затрат в базовом периоде, т;
$О^{ан}$ – общий (условный) объем затрат в анализируемом периоде, т.

Вариант 12

Рассчитайте показатели скорости оборачиваемости капитала предприятия:
а) коэффициент оборачиваемости ($К_{об}$) по формуле

$$К_{об} = \frac{ВР}{СК},$$

где $ВР$ – выручка от реализации продукции, работ, услуг, р.;
$СК$ – среднегодовая стоимость капитала или его части, р.;
б) продолжительность одного оборота ($П_{об}$, дней) по формуле

$$П_{об} = \frac{Д}{К_{об}},$$

где $Д$ – количество календарных дней в анализируемом периоде (год, квартал, месяц).

Вариант 13

Рассчитайте показатели движения рабочей силы предприятия:
а) коэффициент оборота по приему персонала ($К_{пр}$) по формуле

$$К_{пр} = \frac{К_р}{S},$$

где $К_р$ – количество принятого на работу персонала;
S – среднесписочная численность персонала;

б) коэффициент оборота по выбытию ($К_в$) по следующей формуле:

$$К_в = \frac{К_{ув}}{S},$$

где $К_{ув}$ – количество всех уволившихся работников;

в) коэффициент текучести кадров ($К_т$) следующим образом:

$$К_т = \frac{К_{ут}}{S},$$

где $К_{ут}$ – количество уволившихся работников по собственному желанию и за нарушение трудовой дисциплины;

г) коэффициент постоянного состава персонала предприятия ($К_{пс}$) по формуле

$$К_{пс} = \frac{К_г}{S},$$

где $К_г$ – количество работников, проработавших весь год.

Вариант 14

Рассчитайте показатели непроизводительных затрат рабочего времени:

а) удельный вес заработной платы производственных рабочих в производственной себестоимости товарной продукции ($У_в$) по формуле

$$У_в = \frac{ЗП}{ПС} \cdot 100\%,$$

где *ЗП* – заработная плата производственных рабочих, р.;

ПС – производственная себестоимость товарной продукции, р.;

б) сумму заработной платы в себестоимости окончательного брака (*ОБ*) по формуле

$$ОБ = \frac{СБ \cdot У_в}{100\%},$$

где *СБ* – себестоимость забракованной продукции, р.;

в) удельный вес заработной платы производственных рабочих в производственной себестоимости товарной продукции за вычетом машиностроительных затрат ($У_{вв}$) по следующей формуле:

$$У_{вв} = \frac{ЗП}{ПС - М} \cdot 100\%,$$

где *М* – машиностроительные затраты, р.;

г) заработную плату рабочих по исправлению брака (*ЗИБ*) по формуле

$$ЗИБ = \frac{ИБ \cdot У_{вв}}{100\%},$$

где *ИБ* – затраты по исправлению брака, р.

Вариант 15

Рассчитайте показатели движения и технического состояния основных производственных фондов (*ОПФ*):

а) коэффициент обновления ($К_{обн}$) по следующей формуле:

$$К_{обн} = \frac{СПФ}{СКП},$$

где *СПФ* – стоимость поступивших ОПФ, р.;

СКП – стоимость ОПФ на конец периода, р.;

б) коэффициент выбытия ($К_{выб}$) по формуле

$$К_{выб} = \frac{СВФ}{СКП},$$

где *СВФ* – стоимость выбывших ОПФ, р.;

в) коэффициент прироста ($К_{пр}$) следующим образом:

$$К_{пр} = \frac{СПР}{СНП},$$

где *СНП* – стоимость ОПФ на начало периода, р.;
СПР – сумма прироста ОПФ, р.

Вариант 16

Рассчитайте показатели работы грузового автотранспорта:
а) коэффициент использования машин в работе ($К_{м}$) по формуле

$$К_{м} = \frac{КОД}{КНХ},$$

где *КОД* – количество дней, отработанных автомобилями;
КНХ – количество машино-дней, в течение которых автомобили находились в хозяйстве;

б) коэффициент использования рабочего времени машин ($К_{р}$) по следующей формуле:

$$К_{р} = \frac{ВП}{ВН},$$

где *ВП* – время нахождения машин в пробеге, ч;
ВН – время нахождения машин в наряде, ч;
в) коэффициент использования пробега ($К_{п}$) по формуле

$$К_{п} = \frac{ПГ}{ОП},$$

где *ПГ* – пробег с грузом, км;
ОП – общий пробег, км;

г) коэффициент использования грузоподъемности машин ($К_{гр}$) по формуле

$$К_{гр} = \frac{СЗ}{СГ},$$

где *СЗ* – средняя загруженность одной машины, т;
СГ – средняя грузоподъемность одной машины, т.

Вариант 17

Рассчитайте следующие показатели доходности инвестиционного, собственного и заемного капиталов:
а) общую рентабельность инвестированного капитала до уплаты налогов (*РБН*) по формуле

$$РБН = \frac{ЭП}{СК + ЗК},$$

где *ЭП* – валовая (эксплутационная) прибыль до выплаты налогов и процентов по займам;
СК – собственный капитал;
ЗК – заемный капитал;

б) общую рентабельность инвестированного капитала после уплаты налогов (*РСН*) по формуле

$$РСН = \frac{ЧП + ИФ}{СК + ЗК},$$

где *ЧП* – чистая прибыль после выплаты налогов и процентов;

ИФ – финансовые издержки по обслуживанию кредиторской задолженности;

в) ставку рентабельности заемного капитала (*РЗК*) по формуле

$$РЗК = \frac{ПКИ}{СДО},$$

где *ПКИ* – сумма выплаченных процентов за кредиты инвесторам;

СДО – среднегодовая сумма долговых обязательств предприятия;

г) рентабельность собственного капитала (*РСК*) по следующей формуле:

$$РСК = \frac{ЧП}{СК}.$$

Лабораторная работа 6.
Вычисление значения разветвляющейся функции

Цель работы: получение навыков по использованию условного оператора.

Для варианта, указанного преподавателем, разработайте интерфейс проекта; напишите текст процедуры, вызываемой при щелчке по кнопке *Вычислить*; произведите отладку с помощью табличного процессора MS Excel. Проект должен предусматривать вывод значения функции и номера формулы, по которой выполнялся расчет. Отладка должна быть проведена для полного набора тестов, отражающего все возможные значения аргумента, при которых могла изменяться формула расчета.

Вариант 1

$$y = \begin{cases} 1/x, \text{ если } x \ge -5,\ x \ne 0; & (1) \\ x^2, \text{ если } x \le -10; & (2) \\ \sqrt{|x+1|} \text{ в остальных случаях.} & (3) \end{cases}$$

Вариант 2

$$y = \begin{cases} x^2, \text{ если } x \le 0,\ x \ne -10; & (1) \\ \sqrt{x+1}, \text{ если } x > 1; & (2) \\ 1/x \text{ в остальных случаях.} & (3) \end{cases}$$

Вариант 3

$$y = \begin{cases} x + e^{2x}, \text{ если } x \le 0, \quad x \ne -1; & (1) \\ \cos^2 x, \text{ если } 0 < x \le 3{,}14; & (2) \\ x \text{ в остальных случаях.} & (3) \end{cases}$$

Вариант 4

$$y = \begin{cases} x^3, \text{ если } x > 10,\ x \ne 20; & (1) \\ x^2, \text{ если } -5 \le x \le 5; & (2) \\ \lg|x| \text{ в остальных случаях.} & (3) \end{cases}$$

Вариант 5

$$y = \begin{cases} \sqrt{x}, \text{ если } x \ge 100,\ x \ne 105; & (1) \\ \sqrt[3]{x}, \text{ если } x = 20 \text{ или } x = 40; & (2) \\ x^2 + 1 \text{ в остальных случаях.} & (3) \end{cases}$$

Вариант 6

$$y=\begin{cases}\sqrt[3]{x}, \text{ если } x>20; & (1)\\ 1/x, \text{ если } x\le 2 \text{ и } x\ne 0; & (2)\\ x^2-1 \text{ в остальных случаях.} & (3)\end{cases}$$

Вариант 7

$$y=\begin{cases}8x+1, \text{ если } x\ge 5,\ x\ne 9; & (1)\\ x^2+|x|, \text{ если } x\le 1; & (2)\\ x^3+\sqrt{x} \text{ в остальных случаях.} & (3)\end{cases}$$

Вариант 8

$$y=\begin{cases}1-3x, \text{ если } x>0,\ x\ne 8; & (1)\\ x^2-\sin x, \text{ если } x\le -1; & (2)\\ \cos x \text{ в остальных случаях.} & (3)\end{cases}$$

Вариант 9

$$y=\begin{cases}x^3+1, \text{ если } x\ge 8,\ x\ne 10; & (1)\\ 2x^2+\sqrt[3]{|x|}, \text{ если } x\le 1; & (2)\\ \sqrt{x} \text{ в остальных случаях.} & (3)\end{cases}$$

Вариант 10

$$y=\begin{cases}\sqrt{x}, \text{ если } x\ge 4; & (1)\\ 2x+3, \text{ если } x\le 1; & (2)\\ |x^3-4| \text{ в остальных случаях.} & (3)\end{cases}$$

Вариант 11

$$y=\begin{cases}\lg^2 2x, \text{ если } x\ge 5; & (1)\\ 2x^2, \text{ если } x<-2; & (2)\\ \sin x \text{ в остальных случаях.} & (3)\end{cases}$$

Вариант 12

$$y=\begin{cases}\sqrt{x}, \text{ если } x\ge 4 \text{ или } x=1; & (1)\\ \ln|x+1|, \text{ если } x\le -2; & (2)\\ e^x \text{ в остальных случаях.} & (3)\end{cases}$$

Вариант 13

$$y=\begin{cases}x/3, \text{ если } -3\le x\le 3; & (1)\\ \lg(x^2+1), \text{ если } x<-3; & (2)\\ \sqrt{x^3-2} \text{ в остальных случаях.} & (3)\end{cases}$$

Вариант 14

$$y=\begin{cases}|x^3+4|, \text{ если } x\le -1 \text{ или } x=0; & (1)\\ \sqrt{x/2}, \text{ если } x\ge 8; & (2)\\ x^3 \text{ в остальных случаях.} & (3)\end{cases}$$

Вариант 15

$$y = \begin{cases} \sqrt{3x^2 + 4}, \text{ если } x \geq 2; & (1) \\ \ln|x - 2|, \text{ если } x < 0; & (2) \\ \cos x \text{ в остальных случаях.} & (3) \end{cases}$$

Вариант 16

$$y = \begin{cases} tg\ x/2, \text{ если } 0 < x \leq 2; & (1) \\ x^2 + 1, \text{ если } x \leq 0; & (2) \\ \cos^2 x \text{ в остальных случаях.} & (3) \end{cases}$$

Лабораторная работа 7.
Вычисление экономических показателей
(разветвляющийся алгоритм)

Цель работы: получение навыков реализации экономико-математических моделей, содержащих альтернативные расчетные формулы.

Для указанного преподавателем варианта разработайте интерфейс проекта; напишите текст процедуры, вызываемой при щелчке по кнопке *Оценить*; произведите отладку с помощью табличного процессора MS Excel. Интерфейс проекта должен отражать экономическую сущность задачи. Отладка должна быть проведена для полного набора тестов, отражающего все возможные значения аргумента, при которых могла изменяться формула расчета.

Вариант 1

Для заданного расстояния до магазина рассчитайте стоимость транспортных расходов на обслуживание магазина при условии, что тариф зависит от расстояния следующим образом:

- если расстояние до магазина меньше 3 км, то тариф составляет 1 500 р. за 1 км;
- если расстояние от 3 км (включительно) до 7 км – 1 200 р. за 1 км;
- если расстояние не менее 7 км – 1 000 р. за 1 км.

Вариант 2

Для заданного времени разгрузки одного транспортного средства, приведенного в минутах, рассчитайте стоимость разгрузочных расходов базы при условии, что тариф зависит от времени разгрузки следующим образом:

- если время разгрузки не превышает 20 мин, то тариф равен 7 000 р. за 1 мин;
- если время разгрузки больше 20 мин, но не превышает 50 мин, то тариф – 12 000 р. за 1 мин;
- если время разгрузки больше 50 мин, то тариф – 15 000 р. за 1 мин.

Вариант 3

Для заданного объема продажи, приведенного в штуках, и цены товара в рублях рассчитайте стоимость продажи товаров при условии, что величина скидки зависит от объема продажи следующим образом:

- при объеме продажи не более 10 шт. товара скидка не предусматривается;
- при объеме продажи свыше 10 шт., но не более 50 шт., скидка составляет 5%;
- при объеме продажи свыше 50 шт. скидка – 10%.

Вариант 4

Для заданной продолжительности проживания в гостинице, приведенной в днях, рассчитайте стоимость затрат на проживание при условии, что тариф проживания определяется следующим образом:

- если период проживания в гостинице менее 7 дней, то тариф равен 25 000 р. за 1 день;
- если период находится в пределах от 7 дней (включительно) до 14 дней, то тариф – 20 000 р. за день;
- если период не менее 14 дней, то тариф – 15 000 р. за день.

Вариант 5

Для заданного объема закупок товара в тоннах и величины торговой надбавки в процентах рассчитайте объем реализации товара за месяц в тоннах как произведение объема закупок на коэффициент реализации при условии, что коэффициент реализации определяется следующим образом:

- если торговая надбавка не превышает 10%, то значение коэффициента реализации равно 0,9;
- если торговая надбавка находится в пределах от 10 до 20% (включительно), то коэффициент – 0,75;
- если торговая надбавка больше 20%, то коэффициент – 0,5.

Вариант 6

Для заданного объема произведенной партии продукции в тоннах и цены 1 т сырья в рублях рассчитайте объем и стоимость используемого сырья. Объем требуемого сырья вычисляется как произведение объема произведенной партии продукции на норматив расхода сырья. Норма расхода сырья определяется следующим образом:

- при производстве продукции объемом менее 140 т расходуется 1,4 т сырья на 1 т продукции;
- при производстве продукции объемом от 140 т (включительно) до 200 т расходуется 1,3 т сырья на 1 т продукции;
- при производстве продукции объемом не менее 200 т расходуется 1,2 т сырья на 1 т продукции.

Вариант 7

Заданы величины объема продажи в штуках и цены товара в рублях. Вычислите стоимость продажи как произведение объема продажи на отпускную цену. Отпускная цена рассчитывается как сумма цены товара и величины торговой надбавки, приведенная в рублях. Величина торговой надбавки зависит от объема продажи следующим образом:

- при объеме продажи не более 40 т надбавка составляет 18%;
- при объеме продажи от 40 до 60 т включительно – 10%;
- при объеме продажи свыше 60 т – 5%.

Вариант 8

Определите стоимость билета для пассажиров, едущих на заданное расстояние. Стоимость проезда в транспорте зависит от расстояния, которое проезжает пассажир, и тарифа за 1 км пути, который устанавливается в зависимости от дальности следующим образом:

- если расстояние не превышает 10 км, то тариф проезда равен 500 р. за 1 км;
- если расстояние находится в пределах от 10 до 100 км, то тариф проезда – 400 р. за 1 км;
- если расстояние не меньше 100 км, то тариф проезда – 300 р. за 1 км.

Вариант 9

Заданы следующие значения: оклад в рублях, стаж, возраст, пол («м» или «ж»), количество детей. Вычислите размер «надбавки за качество» в соответствии со следующими правилами:

- надбавка начисляется в размере 50% от оклада всем сотрудникам, имеющим стаж работы более 20 лет, и тем сотрудникам с меньшим стажем работы, у которых на иждивении имеется не менее трех детей;
- надбавка начисляется в размере 30% от оклада сотрудникам со стажем работы более 15 лет (но не более 20) и сотрудникам с меньшим стажем, имеющим на иждивении двух детей;
- надбавка начисляется в размере 20% от оклада сотрудникам со стажем работы более 10 лет (но не более 15) при условии, что они достигли пенсионного возраста (55 лет для женщин и 60 лет для мужчин);
- остальным сотрудникам надбавка не начисляется.

Вариант 10

Оцените скорость оборота денежных средств (S), рассчитанную в днях по формуле

$$S = \frac{E \cdot T}{N},$$

где E – средний остаток оборотных средств, р.;

T – длительность периода в днях (год – 365 дней, квартал – 90, месяц – 30 дней);

N – выручка от реализации продукции в оптовых ценах предприятия, р.

Результат оценки определяется следующими условиями:

$S \geq 20$ – низкий уровень оборачиваемости;

$S \in [10;20)$ – средний уровень оборачиваемости;

$S < 10$ – высокий уровень оборачиваемости.

Вариант 11

Оцените уровень (d) заработной платы в полной себестоимости продукции (U) в рублях, учитывая следующее:

$$U = Z + R^{об} + P + R; \quad d = \frac{Z}{U},$$

где Z – прямая заработная плата, р.;

$R^{об}$ – расходы на содержание и эксплуатацию оборудования, р.;

P – прямые материальные затраты, р.;
R – расходы, р.

Результат оценки определяется следующими условиями:
$d \geq 0{,}8$ – высокий;
$d \in [0{,}6;0{,}8)$ – средний;
$d < 0{,}6$ – низкий.

Вариант 12

Оцените уровень потерь от брака (d) в общей сумме расходов (R), учитывая следующее:

$$R = R^{нак} + R^{непр} + R^{бр}; \quad d = \frac{R^{бр}}{R},$$

где $R^{нак}$ – накладные внепроизводственные расходы, р.;
$R^{непр}$ – непроизводственные расходы, р.;
$R^{бр}$ – потери от брака, р.

Результат оценки определяется следующими условиями:
$d > 0{,}5$ – недопустимо высокий;
$d \in (0{,}3;0{,}5]$ – высокий;
$d \leq 0{,}3$ – допустимый.

Вариант 13

Оцените уровень расходов по исправлению брака (d) в сумме потерь от брака (P), учитывая следующее:

$$P = S + R - W; \quad d = \frac{R}{P},$$

где S – себестоимость забракованных изделий, р.;
R – расходы по исправлению брака, р.;
W – сумма возвращенной стоимости брака, р.

Результат оценки определяется следующими условиями:
$d > 0{,}8$ – высокий;
$d \in [0{,}6;0{,}8]$ – допустимый;
$d < 0{,}6$ – средний.

Вариант 14

Оцените уровень товарной продукции за отчетный период (N^p) в полном объеме реализации продукции с использованием следующих формул:

$$N^p = \frac{N^T}{N}; \quad N = N^T + N^O + N^S,$$

где N – полный объем реализации продукции, р.;
N^T – реализация товарной продукции за отчетный период, р.;
N^O – остатки отгруженной продукции, р.;
N^S – остатки готовой продукции на складе, р.

Результат оценки определяется следующими условиями:
$N^p > 0{,}85$ – высокий;
$N^p \in (0{,}7;0{,}85]$ – средний;
$N^p \leq 0{,}75$ – низкий.

Вариант 15

Оцените уровень расходов по сбыту (U) в общей сумме расходов с использованием следующих формул:

$$U = \frac{R^s}{R}; \quad R = R^{себ} + R^s + R^a + R^{пр},$$

где R – общая сумма расходов, р.;

$R^{себ}$ – себестоимость реализованной продукции, р.;
R^{s} – расходы по сбыту, р.;
R^{a} – административные расходы, р.;
$R^{пр}$ – прочие расходы, р.

Результат оценки определяется следующими условиями:
$U > 0{,}5$ – недопустимо высокие расходы;
$U \in (0{,}2;0{,}5]$ – высокие расходы;
$U \le 0{,}2$ – допустимый уровень расходов.

Вариант 16

Оцените долю расходов в семейном бюджете на питание (P) в общем объеме расходов с использованием следующих формул:

$$P = \frac{R^{пит}}{R}; \quad R = R^{пит} + R^{ком} + R^{од} + R^{пр},$$

где R – расходы семьи, р.;
$R^{пит}$ – расходы на питание, р.;
$R^{ком}$ – оплата жилья и коммунальных услуг, р.;
$R^{од}$ – расходы на одежду и обувь, р.;
$R^{пр}$– прочие расходы, р.

Результат оценки определяется следующими условиями:
$P > 0{,}7$ – высокая;
$P \in [0{,}3;0{,}7]$ – средняя;
$P < 0{,}3$ – низкая.

Вариант 17

Оцените уровень транспортных расходов (P) в общей сумме расходов предприятия с использованием следующих формул:

$$P = \frac{R^{тр}}{R}; \quad R = R^{себ} + R^{сб} + R^{тр} + R^{накл},$$

где R – сумма расходов предприятия, р.;
$R^{себ}$ – себестоимость реализованной продукции, р.;
$R^{сб}$ – расходы по сбыту, р.;
$R^{тр}$ – транспортные расходы, р.;
$R^{накл}$ – накладные расходы, р.

Результат оценки определяется следующими условиями:
$P > 0{,}3$ – высокий;
$P \in [0{,}1;0{,}3]$ – средний;
$P < 0{,}1$ – низкий.

Лабораторная работа 8.
Табуляция функции в область *Memo*

Цель работы: получение навыков использования оператора цикла с предусловием и вывод результатов в область *Memo*.

Для варианта, указанного преподавателем в табл. 3 лабораторной работы 4, разработайте интерфейс проекта; напишите текст процедуры, вызываемой при щелчке по кнопке *Табуляция*; произведите отладку с помощью табличного процессора MS Excel. Проект должен предусматривать вывод в область *Memo* порядкового номера строки результата, значений аргумента и функции. В проекте должно быть предусмотрено управление форматом представления результатов.

Лабораторная работа 9.
Табуляция функции в текстовую таблицу на основе оператора цикла с предусловием

Цель работы: получение навыков использования текстовой таблицы.

Для варианта, указанного преподавателем в табл. 3 лабораторной работы 4, разработайте интерфейс проекта; напишите текст процедуры, вызываемой при щелчке по кнопке *Табуляция*; произведите отладку с помощью табличного процессора MS Excel. Проект должен предусматривать вывод в область *StringGrid* порядкового номера строки результата, значений аргумента и функции. В проекте должно быть предусмотрено управление форматом представления результатов. При тестировании убедитесь, что вычисление функции происходит при всех значениях аргумента из заданного диапазона.

Лабораторная работа 10.
Табуляция функции в текстовую таблицу на основе арифметического оператора цикла

Цель работы: получение навыков использования арифметического оператора цикла.

Для варианта, указанного преподавателем в табл. 3 лабораторной работы 4, разработайте интерфейс проекта; напишите текст процедуры, вызываемой при щелчке по кнопке *Табуляция*; произведите отладку с помощью табличного процессора MS Excel. Проект должен предусматривать вывод в область *String Grid* порядкового номера строки результата, значений аргумента и функции. В проекте должно быть предусмотрено управление форматом представления результатов. При написании процедуры используйте арифметический оператор цикла.

Лабораторная работа 11.
Табуляция разветвляющейся функции в текстовую таблицу

Цель работы: получение навыков управления форматом вывода результатов.

Для варианта, приведенного в лабораторной работе 6, разработайте интерфейс проекта; напишите текст процедуры, вызываемой при щелчке по кнопке *Табуляция*; произведите отладку с помощью табличного процессора MS Excel. Проект должен предусматривать вывод в область *String Grid* порядкового номера строки результата, значений аргумента, функции и номера функции. В проекте должно быть предусмотрено управление форматом представления результатов. При составлении процедуры используйте арифметический оператор цикла. Для отладки подберите начальное, конечное значение и шаг изменения значения аргумента так, чтобы за минимальное число запусков проекта можно было бы проверить все варианты расчета функции.

Лабораторная работа 12.
Создание проектов по простой обработке числовых массивов

Цель работы: получение навыков реализации типовых алгоритмов обработки числовых массивов.

Этап 1

Дан массив целых чисел. Предполагается, что в массиве не более 20 чисел. Требуется найти сумму всех его элементов.

Порядок выполнения работы

1. Запустите СП Delphi.
2. Расположите в окне формы визуальные компоненты так, как это показано на рис. 46.
3. Сохраните проект в папке *Сумма элементов массива*.
4. Установите значения свойств указанных компонентов так, чтобы окно формы приняло вид, приведенный на рис. 47.

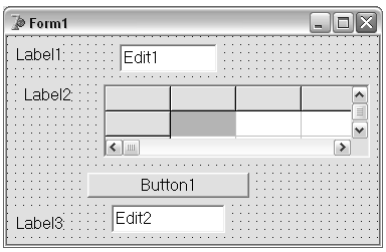


Рис. 46. **Визуальные компоненты для нахождения суммы элементов массива**

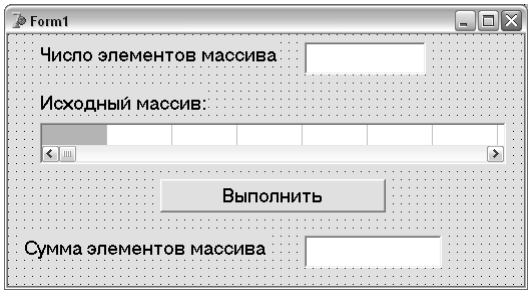


Рис. 47. **Результат изменения значений свойств визуальных компонентов**

Для того, чтобы можно было заполнять ячейки компонента *StringGrid* набором чисел на клавиатуре и пользоваться при этом клавишей *Tab*, в составном свойстве *Options* данного компонента задайте значение *true* для свойства *goEditing* и значение *true* для свойства *goTabs*.

Чтобы исключить возможность изменения содержимого компонента *Edit2*, задайте значение *false* для свойства *Enabled*.

5. После двойного щелчка по кнопке *Выполнить* заполните заготовку процедуры обработки щелчка по данной кнопке так, как это показано на рис. 48.

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    n,i,S:integer;
    a:array[1..20] of integer;
begin
  n:=StrToInt(Edit1.Text); //число элементов массива
  for i:=1 to n do
    a[i]:=StrToInt(StringGrid1.Cells[i-1,0]);
  {в этом цикле происходит заполнение массива а
  числами, введенными в ячейки текстовой таблицы}
  S:=0; // задание начального значения для накопления суммы
  for i:=1 to n do
    S:=S+a[i];  //цикл накопления суммы
  Edit2.Text:=IntToStr(S); //вывод результата в поле Edit2
end;
```

Рис. 48. **Текст процедуры вычисления суммы элементов массива**

6. Сохраните проект и запустите его на выполнение. Введите исходные данные и щелкните по кнопке *Выполнить*. Пример результата выполнения проекта приведен на рис. 49.

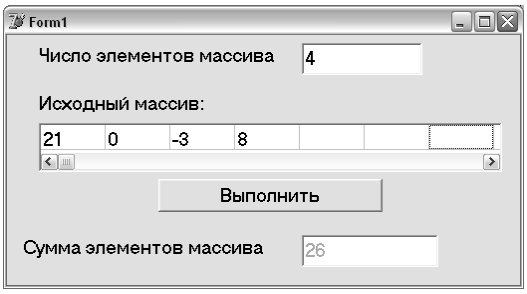


Рис. 49. **Пример вычисления суммы элементов целочисленного массива**

7. После проверки преподавателем правильности программы прекратите процесс выполнения проекта. Закройте окно Delphi.

Этап 2

Дан массив вещественных чисел. Предполагается, что в массиве не более 20 чисел. Требуется найти сумму всех его элементов.

Порядок выполнения работы

1. С помощью программы *Проводник* в Вашей папке создайте копию папки *Сумма элементов массива*. Полученной копии присвойте имя *Сумма элементов массива 2.*

2. Запустите СП Delphi и откройте проект из папки *Сумма элементов массива 2*.

3. Двойным щелчком по кнопке *Выполнить* войдите в окно редактора кода. В результате появится текст процедуры обработки щелчка по этой кнопке.

4. Внесите исправления в текст процедуры с учетом того, что массив *a* должен быть вещественным и результат *S* – тоже вещественным.

Сохраните созданный проект и проверьте его работу. Продемонстрируйте работу проекта преподавателю.

Этап 3

Дан массив вещественных чисел. Предполагается, что в массиве не более 20 чисел. Требуется найти сумму его положительных элементов.

Порядок выполнения работы

1. С помощью программы *Проводник* в Вашей папке создайте копию папки *Сумма элементов массива 2*. Полученной копии присвойте имя *Сумма положительных элементов.*

2. Запустите СП Delphi и откройте проект из папки *Сумма положительных элементов*.

3. Двойным щелчком по кнопке *Выполнить* войдите в окно редактора кода. В результате появится текст процедуры обработки щелчка по этой кнопке.

4. Вставьте в тело цикла необходимый условный оператор (*if – then – else*).

5. Сохраните созданный проект и проверьте его работу. Продемонстрируйте работу проекта преподавателю.

Этап 4

Дан массив вещественных чисел. Предполагается, что в массиве не более 20 чисел. Требуется найти произведение всех его элементов.

Порядок выполнения работы

1. С помощью программы *Проводник* в Вашей папке создайте копию папки *Сумма элементов массива 2*. Полученной копии присвойте имя *Произведение элементов массива.*

2. Запустите СП Delphi и откройте проект из папки *Произведение элементов массива*.

3. Двойным щелчком по кнопке *Выполнить* войдите в окно редактора кода. В результате появится текст процедуры обработки щелчка по этой кнопке.

4. Внесите необходимые исправления в текст процедуры: замените вычисление суммы на произведение и назовите переменную, которой присваивается результат, *P*.

5. Сохраните созданный проект и проверьте его работу. Продемонстрируйте работу проекта преподавателю.

Этап 5

Дан массив вещественных чисел. Предполагается, что в массиве не более 20 чисел. Требуется найти произведение всех его элементов, не равных нулю.

Порядок выполнения работы

1. С помощью программы *Проводник* в Вашей папке создайте копию папки *Произведение элементов массива*. Полученной копии присвойте имя *Произведение ненулевых элементов.*

2. Запустите СП Delphi и откройте проект из папки *Произведение ненулевых элементов*.

3. Двойным щелчком по кнопке *Выполнить* войдите в окно редактора кода. В результате появится текст процедуры обработки щелчка по этой кнопке.

4. Вставьте в тело цикла необходимый условный оператор (*if – then – else*).

5. Сохраните созданный проект и проверьте его работу. Продемонстрируйте работу проекта преподавателю.

Этап 6

Дан массив вещественных чисел. Предполагается, что в массиве не более 20 чисел. Требуется найти произведение его элементов, не равных нулю и стоящих на четных местах.

Порядок выполнения работы

1. С помощью программы *Проводник* в Вашей папке создайте копию папки *Произведение ненулевых элементов*. Полученной копии присвойте имя *Произведение ненулевых на четных местах.*

2. Запустите СП Delphi и откройте проект из папки *Произведение ненулевых на четных местах*.

3. Двойным щелчком по кнопке *Выполнить* войдите в окно редактора кода. В результате появится текст процедуры обработки щелчка по этой кнопке.

4. Замените оператор цикла *for-do* на оператор *while-do* и добавьте необходимые операторы присваивания.

6. Сохраните созданный проект и проверьте его работу. Продемонстрируйте работу проекта преподавателю.

Этап 7

Дан массив вещественных чисел. Предполагается, что в массиве не более 20 чисел. Требуется найти произведение отрицательных его элементов, стоящих на нечетных местах.

Порядок выполнения работы

1. С помощью программы *Проводник* в Вашей папке создайте копию папки *Произведение ненулевых на четных местах*. Полученной копии присвойте имя *Произведение отрицательных на нечетных местах.*
2. Запустите СП Delphi и откройте проект из папки *Произведение отрицательных на нечетных местах*.
3. Двойным щелчком по кнопке *Выполнить* войдите в окно редактора кода. В результате появится текст процедуры обработки щелчка по этой кнопке.
4. Внесите необходимые исправления в текст процедуры.
5. Сохраните созданный проект и проверьте его работу. Продемонстрируйте работу проекта преподавателю.

Этап 8

Дан массив вещественных чисел. Предполагается, что в массиве не более 20 чисел. Требуется найти минимальный элемент этого массива.

Порядок выполнения работы

1. С помощью программы *Проводник* в Вашей папке создайте копию папки *Сумма элементов массива 2*. Полученной копии присвойте имя *Минимальный элемент.*
2. Запустите СП Delphi и откройте проект из папки *Минимальный элемент*.
3. Двойным щелчком по кнопке *Выполнить* войдите в окно редактора кода. В результате появится текст процедуры обработки щелчка по этой кнопке.
4. Внесите необходимые исправления в текст процедуры.
5. Сохраните созданный проект и проверьте его работу. Продемонстрируйте работу проекта преподавателю.

Этап 9

Дан массив вещественных чисел. Предполагается, что в массиве не более 20 чисел. Требуется найти минимальный элемент этого массива и определить его порядковый номер.

Порядок выполнения работы

1. С помощью программы *Проводник* в Вашей папке создайте копию папки *Минимальный элемент*. Полученной копии присвойте имя *Положение минимального.*
2. Запустите СП Delphi и откройте проект из папки *Положение минимального*.
3. Двойным щелчком по кнопке *Выполнить* войдите в окно редактора кода. В результате появится текст процедуры обработки щелчка по этой кнопке.
4. Дополните текст процедуры необходимыми операторами.
5. Сохраните созданный проект и проверьте его работу. Продемонстрируйте работу проекта преподавателю.

Лабораторная работа 13.
Вычисления в одномерных числовых массивах

Цель работы: получение навыков реализации проекта, использующего вычисления в одномерных массивах.

Для указанного преподавателем варианта разработайте интерфейс проекта; напишите текст процедуры, вызываемой при щелчке по кнопке *Определить*; произведите отладку для самостоятельно подготовленных тестов. Проект должен предусматривать ввод размерности и элементов массива с использованием компонента *StringGrid,* ввод некоторых дополнительных чисел, выполнение действий в соответствии с условием задачи, вывод результатов вычислений. Отладка должна быть проведена для полного набора тестов, отражающего всевозможные варианты выполнения вычислений.

Вариант 1

Найдите количество чисел, принадлежащих промежутку [*A, B*], и сумму чисел, стоящих на местах, кратных 3.

Вариант 2

Найдите сумму чисел, меньших заданного *D*, и количество чисел, стоящих на четных местах и больших заданного *C*.

Вариант 3

Найдите произведение всех чисел, стоящих на местах кратных 4, и количество чисел, не больших заданного *A*.

Вариант 4

Найдите количество чисел, меньших заданного *X*, и произведение всех отрицательных чисел, стоящих на нечетных местах.

Вариант 5

Найдите количество чисел, не принадлежащих промежутку (*X, Y*], и сумму отрицательных чисел, стоящих на четных местах.

Вариант 6

Найдите количество неотрицательных чисел и определите сумму чисел, стоящих на местах, кратных 3 и неравных заданному *F*.

Вариант 7

Вычислите произведение чисел, принадлежащих промежутку (*A, B*], и количество отрицательных чисел, стоящих на местах, кратных 3.

Вариант 8

Найдите количество нулей во всем массиве и определите сумму квадратов чисел, принадлежащих промежутку (*A, B*) и стоящих на местах, кратных 4.

Вариант 9

Найдите среднее арифметическое отрицательных чисел и определите количество чисел, по величине больших *A* и стоящих на четных местах.

Вариант 10

Найдите среднее арифметическое положительных чисел, стоящих на нечетных местах, и количество чисел, меньших заданного *B*.

Вариант 11

Найдите среднее арифметическое чисел, принадлежащих промежутку [*A, B*), и количество положительных чисел, стоящих на местах, кратных 4.

Вариант 12

Найдите среднее арифметическое чисел, неравных заданному *C*, и произведение неположительных чисел, стоящих на четных местах.

Вариант 13

Найдите среднее арифметическое чисел, больших заданного *D* и стоящих на нечетных местах, и определите количество чисел, не больших заданного *F*.

Вариант 14

Найдите среднее арифметическое чисел, не попадающих в промежуток [*A, B*], и количество положительных чисел, стоящих на местах, кратных 3.

Вариант 15

Найдите среднее арифметическое ненулевых чисел и количество чисел, по величине не больших *A* и стоящих на четных местах.

Вариант 16

Найдите среднее арифметическое положительных чисел, стоящих на нечетных местах, и произведение чисел, меньших заданного *C*.

Лабораторная работа 14.
Перестановки элементов в одномерных числовых массивах

Цель работы: получение навыков реализации проекта по перестановке элементов в одномерных массивах.

Для указанного преподавателем варианта разработайте интерфейс проекта; напишите текст процедуры, вызываемой при щелчке по кнопке *Формировать*; произведите отладку для самостоятельно подготовлен-

ных тестов. Проект должен предусматривать ввод размерности (*N*) и элементов массива, определение значения и местоположения минимального или максимального элемента, вывод полученного массива. Отладка должна быть проведена для полного набора тестов, отражающего различные варианты расположения искомых элементов в массиве. Для ввода и вывода элементов массива используйте компонент *String Grid*.

Вариант 1
Найдите первый максимальный элемент и поменяйте его местами с пятым элементом массива.

Вариант 2
Найдите последний минимальный элемент и поменяйте его местами с предыдущим элементом массива.

Вариант 3
Найдите первый минимальный элемент и поменяйте его местами с последующим элементом массива.

Вариант 4
Найдите первый максимальный элемент и поменяйте его местами с шестым элементом массива.

Вариант 5
Найдите последний минимальный элемент и поменяйте его местами с третьим элементом массива.

Вариант 6
Найдите первый минимальный элемент и поменяйте его местами с третьим элементом массива.

Вариант 7
Найдите последний максимальный элемент и поменяйте его местами с четвертым элементом массива.

Вариант 8
Найдите последний максимальный элемент и поменяйте его местами с четвертым элементом массива.

Вариант 9
Найдите первый максимальный элемент и поменяйте его местами с последним минимальным элементом массива.

Вариант 10
Найдите последний максимальный элемент и поменяйте его местами с первым минимальным элементом массива.

Лабораторная работа 15. Формирование новых массивов

Цель работы: получение навыков по реализации проектов формирования новых массивов.

Для указанного преподавателем варианта разработайте интерфейс проекта; напишите текст процедуры, вызываемой при щелчке по кнопке *Формировать* и предусматривающей ввод двух целочисленных массивов и формирование нового массива. Учтите, что исходные массивы могут быть разной длины. Для ввода и вывода элементов массива используйте компонент *String Grid*.

Вариант 1
Сформируйте массив из элементов исходных массивов, больших второго элемента первого массива и положительных элементов второго массива.

Вариант 2
Сформируйте массив из отрицательных элементов первого массива и элементов обоих массивов, больших первого элемента второго массива.

Вариант 3
Сформируйте массив из положительных элементов первого массива и элементов обоих массивов, больших последнего элемента второго массива.

Вариант 4
Сформируйте массив из элементов исходных массивов, меньших произведения последних элементов заданных массивов и отрицательных элементов второго массива.

Вариант 5

Сформируйте массив из элементов исходных массивов, не превышающих первого элемента первого массива и элементов второго массива, больших заданного числа.

Вариант 6

Сформируйте массив из элементов исходных массивов, не превышающих третий элемент каждого из них соответственно и положительных элементов первого массива.

Вариант 7

Сформируйте массив из элементов исходных массивов, не превышающих сумму первых элементов исходных массивов и элементов первого массива, больших заданного числа.

Вариант 8

Сформируйте массив из положительных элементов исходных массивов, меньших 10, и отрицательных элементов второго массива.

Вариант 9

Сформируйте массив из нечетных элементов первого массива, больших 5, и элементов обоих массивов, не меньших заданного числа.

Вариант 10

Сформируйте массив из отрицательных элементов исходных массивов и четных элементов второго массива, больших заданного числа.

Вариант 11

Сформируйте массив из элементов исходных массивов, больших первого элемента второго массива, и нечетных элементов второго массива.

Вариант 12

Сформируйте массив из отрицательных четных элементов первого массива и положительных элементов исходных массивов, не меньших заданного числа.

Вариант 13

Сформируйте массив из нечетных элементов первого массива и элементов исходных массивов, не больших 3.

Вариант 14

Сформируйте массив из нечетных элементов второго массива, больших последнего элемента второго массива, и элементов исходных массивов, меньших последнего элемента первого массива.

Вариант 15

Сформируйте массив из тех элементов исходных массивов, которые меньше заданного числа, и нечетных элементов второго массива, больших последнего элемента первого массива.

Вариант 16

Сформируйте массив из элементов исходных массивов, принадлежащих промежутку [–4; 6], и четных элементов первого массива, больших 12.

Лабораторная работа 16. Обработка экономической информации на базе числовых массивов

Цель работы: получение навыков обработки экономической информации на базе числовых массивов.

Для указанного преподавателем варианта разработайте интерфейс проекта; напишите текст процедуры, вызываемой при щелчке по кнопке *Вычислить* и предусматривающей ввод числового массива и вычисление требуемых показателей. Для ввода элементов массива используйте компонент *String Grid.*

Вариант 1

По данным о ежемесячных материальных затратах организации за отчетный период определите следующее:

- сколько месяцев организация отработала неэффективно, т. е. превысила отраслевой норматив;
- какова доля затрат этих месяцев в общей сумме материальных затрат организации за отчетный период.

Вариант 2

По данным о ежемесячном объеме выпуска товарной продукции предприятием за отчетный период определите следующее:

- сколько месяцев предприятие отработало «хуже» и «лучше» среднего уровня;
- какова доля «лучших» месяцев в общем объеме выпущенной продукции организацией за отчетный период.

Вариант 3

По данным о ежемесячном объеме выплаченной заработной платы работникам организации за отчетный период определите следующее:

- среднемесячный размер фонда заработной платы;
- долю суммы заработной платы, выплаченной в первые два месяца анализируемого периода в общей сумме выплат за весь период.

Вариант 4

По данным о ежемесячно получаемой прибыли организацией в течение отчетного периода определите следующее:

- сколько месяцев организация отработала «лучше» среднего уровня;
- какова доля прибыли этих месяцев в общей сумме прибыли, полученной организацией за анализируемый период.

Вариант 5

По данным о ежемесячных материальных затратах организации за отчетный период определите следующее:

- сколько месяцев организация отработала лучше, чем заложено по нормативу;
- какова доля этих месяцев в общей сумме материальных затрат организации за отчетный период.

Вариант 6

По данным о среднемесячной численности персонала фирмы за отчетный период определите следующее:

- количество месяцев, когда фирма работала в составе, меньшем заданного значения;
- удельный вес этих месяцев в продолжительности отчетного периода.

Вариант 7

По данным о месячных объемах фактически привлеченных средств во вклады филиалами банка за отчетный период найдите следующее:

- общую сумму привлеченных средств филиалами банка;
- среднюю сумму привлеченных средств за те месяцы, когда привлекалось средств меньше заданного значения.

Вариант 8

По данным о ежемесячных затратах организации на сырье за отчетный период определите следующее:

- сколько месяцев организация расходовала сырья больше норматива;
- какова доля этих затрат в общей сумме затрат на сырье организации за анализируемый период.

Вариант 9

По данным о цеховых расходах организации за конкретный месяц определите следующее:

- количество цехов, отработавших на уровне среднецеховых расходов;
- долю расходов тех цехов организации, которые отработали «хуже» первого цеха.

Вариант 10

По данным о фонде заработной платы производственных рабочих цехов организации найдите следующее:

- среднецеховую заработную плату производственных рабочих;
- долю заработной платы рабочих тех цехов, которые имеют фонд заработной платы ниже заданного значения.

Лабораторная работа 17.
Использование флажков и групп переключателей

Цель работы: получение навыков реализации проектов, использующих флажки и переключатели.

Для указанного преподавателем варианта разработайте интерфейс проекта, использующего объекты *CheckBox* (флажок) и *RadioGroup* (группа переключателей).

При размещении переключателей в области группы используйте свойство *Items*. В этот список занесите заголовки всех требуемых переключателей.

Кроме основной процедуры, выполняющей требуемые в условии задачи расчеты, создайте процедуру очистки полей результатов при щелчке в любой области исходных данных. Для этого напишите процедуру очистки результирующих полей для щелчка по одному полю задания исходных данных, а затем сделайте ссылку на нее для события *OnClick* остальных полей ввода данных.

Вариант 1

Вся продаваемая предприятием партия продукции относится к одному из видов качества:

- со знаком качества;
- соответствующая стандарту;
- не соответствующая стандарту.

Определите сумму выручки от реализации заданного числа единиц продукции с учетом ее качества при условии, что предприятию оплачивают по 1 000 р. за каждое изделие, соответствующее стандарту, за единицу изделия со знаком качества платится надбавка в размере 15% от цены стандартного изделия, а при оплате изделий, не соответствующих стандарту, делается скидка в размере 10%. Для постоянных покупателей делается скидка в размере 5%.

Для оптовых покупателей делается скидка в размере 7%.

Вариант 2

Тарифы на телефонные разговоры с тремя населенными пунктами составляют 1 000, 900, 800 р./мин. Если телефонный разговор происходит в вечернее время, то производится скидка в размере 10%, в выходные и праздничные дни – 7%. Определите стоимость телефонного разговора заданной длительности с заданным населенным пунктом.

Вариант 3

Себестоимость строительно-монтажных работ (C) определяется по формуле

$$C = M + З + A + H ,$$

где M – стоимость используемых материалов, конструкций, электроэнергии и т. д.;

$З$ – расходы на оплату труда;

A – расходы на содержание строительных машин и механизмов;

H – накладные расходы.

На строительно-монтажные работы, выполняемые хозяйственным способом, к нормам накладных расходов применяется понижающий коэффициент 0,506. На внутренние санитарно-технические работы, выполняемые в сельских районах, к нормам накладных расходов применяется коэффициент 1,15. При реконструкции действующих предприятий к нормам накладных расходов применяется коэффициент 1,1.

Рассчитайте себестоимость строительно-монтажных работ заданного типа.

Вариант 4

Определите новую цену книги после переоценки. Для книг, изданных до 1992 г. цена увеличивается в 200 раз, в период с 1992 по 1994 г. – в 12 раз, до 2000 г. – в 5 раз. Кроме того, для художественных книг цена увеличивается еще на 15% от начисленной суммы, а при отличном состоянии книги – еще на 5%.

Вариант 5

Определите доплату к стипендии за проживание на территории, загрязненной после аварии на Чернобыльской АЭС. Если студент проживал до поступления в университет на территории, относящейся к I категории загрязнения, доплата составляет 50% от уровня минимальной заработной платы, II категории – 100, III категории – 150%. Для студентов, имеющих детей, доплата увеличивается еще на 20% от уровня минимальной заработной платы.

Вариант 6

Определите размер заработной платы рабочих как произведение минимальной зарплаты на тарифный коэффициент. Значение тарифного коэффициента равно 1,1; 1,25 и 1,4 для 1, 2 и 3 разряда рабочего соответственно. При стаже работы больше 10 лет размер заработной платы увеличивается на 10% от начисленной суммы. При стаже работы на данном предприятии больше 5 лет размер заработной платы увеличивается еще на 7% от начисленной суммы.

Вариант 7

Определите оптовую цену детали, зависящую от сортности: для 1-го сорта оптовая цена равна себестоимости детали, увеличенной на 15%, для 2-го сорта – на 7, для 3-го – на 2% от себестоимости детали. При

использовании для производства только местных материалов оптовая цена уменьшается на 4% от себестоимости. Оптовая цена неликвидных деталей уменьшается на 7% от себестоимости.

Вариант 8

Определите размер оплаты за проживание в общежитии, составляющий для одноместной комнаты 100%, двухместной – 60, трехместной – 40% от размера минимальной заработной платы. Для проживающих семей предусмотрена скидка в размере 10% от начисленной суммы. Если семья проживает в одноместной комнате, то делается дополнительная скидка 8%.

Вариант 9

Житель Республики Беларусь решил подключится к сети Velcom на один из тарифных планов – «Корпоративный» или «Социальный». Известно, что его исходящие разговоры составят m часов в месяц, из которых n часов составят разговоры в стационарной сети, k часов в сетях БелСел, МТС, БеСТ и t часов в рамках закрытой абонентской группы (корпорации), $m = n + k + t$. Тарифы за одну минуту исходящих соединений тарифных планов «Корпоративный» и «Социальный» приведены в табл. 4.

Таблица 4. **Тарифы за одну минуту исходящих соединений, р.**

Стационарная сеть		БелСел, МТС, БеСТ		Внутри корпорации
Тариф		Тариф		
«Корпоративный»	«Социальный»	«Корпоративный»	«Социальный»	
220	290	255	290	28

Кроме того, тарифный план «Корпоративный» предоставляет 2,5 ч бесплатных разговоров, а тарифный план «Социальный» – только 1 ч бесплатных разговоров. Абонентская плата составляет 5 250 р. и 4 500 р. для «Корпоративного» и «Социального» планов соответственно.

Определите, на какой тарифный план компании Velcom выгоднее подключиться.

ВОПРОСЫ ДЛЯ ПОДГОТОВКИ К ТЕСТИРОВАНИЮ И ЭКЗАМЕНУ ПО КУРСУ «АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ» ДЛЯ СТУДЕНТОВ ЗАОЧНОЙ ФОРМЫ ОБУЧЕНИЯ

1. *Основы алгоритмизации.*
1.1. Понятие алгоритма.
1.2. Свойства алгоритма.
1.3. Средства записи алгоритма.
1.4. Графические схемы алгоритмов.
1.5. Типы алгоритмов.
1.6. Линейные алгоритмы.
1.7. Разветвляющиеся алгоритмы.
1.8. Циклические алгоритмы.
2. *Понятия модульного программирования.*
2.1. Свойства модуля.
2.2. Модульная структура программных продуктов.
3. *Основные понятия объектно-ориентированного программирования.*
3.1. Объект.
3.2. Свойство.
3.3. Метод.
3.4. Событие.
3.5. Класс.
4. *Важнейшие принципы объектного подхода при проектировании программных продуктов.*
4.1. Инкапсуляция.
4.2. Наследование.
4.3. Полиморфизм.
5. *Общие сведения о системе программирования СП Delphi.*
5.1. Класс средств программирования, к которому относится СП Delphi.
5.2. Характеристики СП Delphi, благодаря которым достигается ускорение разработки программ.
5.3. Особенности разработки интерфейса программы (приложения) в СП Delphi.
5.4. Язык программирования в СП Delphi.
6. *Определение и структура проекта СП Delphi.*

6.1. Проект.
6.2. Модуль.
7. *Инструменты среды СП Delphi.*
7.1. Главное меню.
7.2. Панель инструментов.
7.3. Палитра компонентов (Component Palette).
7.4. Инспектор объектов (Object Inspector).
7.5. Окно формы.
7.6. Редактор кода (Code Editor).
8. *Этапы создания простого приложения в СП Delphi.*
8.1. Запуск системы программирования Delphi.
8.2. Открытие существующего проекта или создание нового.
8.3. Создание интерфейса приложения.
8.4. Первое сохранение проекта.
8.5. Создание процедур-обработчиков событий.
8.6. Сохранение проекта.
8.7. Выполнение (запуск) проекта.
8.8. Внесение изменений в проект.
8.9. Завершение работы Delphi.
9. *Реализация проектов в СП Delphi (базовые алгоритмы).*
9.1. Проект на основе линейного алгоритма.
9.1.1. Объект *Label* (надпись), его основные свойства.
9.1.2. Объект *Edit* (поле для ввода), его основные свойства.
9.1.3. Объект *Button* (кнопка), его основные свойства.
9.1.4. Описание переменных, назначение и примеры записи.
9.1.5. Оператор присваивания, назначение и примеры записи.
9.1.6. Пример проекта на основе линейного алгоритма – вычисление значения функции.
9.2. Проект на основе разветвляющегося алгоритма.
9.2.1. Условный оператор, назначение и примеры записи.
9.2.2. Пример разветвляющегося алгоритма.
9.3. Табулирование функции.
9.3.1. Объект *Memo* (памятка, примечание, поле для вывода текста), его основные свойства.
9.3.2. Объект *StringGrid* (текстовая таблица), его основные свойства.
9.3.3. Оператор цикла *while-do*, назначение и примеры записи.
9.3.4. Оператор цикла *for-to-do*.
9.3.5. Управление форматом представления результатов на базе функций *StrToInt, IntToStr, StrToFloat* и *FloatToStr*, примеры их использования.
9.3.6. Пример табулирования функции с выводом результатов в *Memo*.
9.3.7. Пример табулирования функции с выводом результатов в текстовую таблицу.
10. *Работа с массивом чисел.*
10.1. Определение и описание числовых массивов.
10.2. Организация ввода-вывода элементов массива.
10.3. Нахождение суммы и произведения всех элементов числового массива.
10.4. Нахождение суммы и произведения элементов массива, стоящих на четных (нечетных) местах.
10.5. Нахождение суммы и произведения элементов массива, удовлетворяющих заданному условию.

ПРИМЕРНЫЕ ТЕСТОВЫЕ ЗАДАНИЯ

1. Из каких правил состоит алгоритм?

Варианты ответа:

а) запрещающих;
б) предписывающих;
в) разрешающих;
г) любых.

2. «Понятность» – это свойство алгоритма, заключающееся в том, что …

Варианты ответа:

а) каждая команда алгоритма должна входить в систему команд исполнителя;
б) исполнителю понятно, что надо делать в данный момент;
в) каждая команда алгоритма понятна всем исполнителям;

г) каждая команда алгоритма точно определена.

3. Значения какой функции вычисляет данная программа?

```
Program Primer 1;
var
x, y, z, b: real;
begin
ввод(x, y, z);
b:=ln(sqrt(exp(x-y)+z))/ln(10.0);
вывод(x, y, z, b)
end.
```

Варианты ответа:

а) $b = \ln\sqrt{(e^{x-y} + z)}$;

б) $b = \lg\sqrt{e^{x-y+z}}$;

в) $b = \lg\sqrt{(e^{x-y} \cdot z)}$;

г) $b = \lg\sqrt{(e^{x-y} + z)}$.

4. Что вычисляет программа, приведенная ниже?

```
Program Primer 2;
var
N, i: integer;
x: array [1..50] of real;
S: real;
begin
ввод(N); ввод(массива x);
вывод("дано:", массив x);
S:=0;
i:=1;
while i<=N do
begin
S:=S+sqr(x[i]);
i:=i+2
end;
вывод(S)
end.
```

Варианты ответа:

а) сумму квадратов всех элементов массива *x*;
б) сумму квадратов элементов массива *x*, стоящих на нечетных местах;
в) сумму квадратов элементов массива *x*, стоящих на четных местах;
г) сумму элементов массива *x*, стоящих на нечетных местах.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ. 1992-01-01. – М. : Изд-во стандартов, 1992. – 28 с.

Информатика : учеб. для вузов / под ред. Н. В. Макаровой. – М. : Финансы и статистика, 2003. – 768 с.

Вычислительная техника и программирование : учеб. для техн. вузов / А. В. Петров [и др.] ; под ред. А. В. Петрова. – М. : Высш. шк., 1990. – 478 с.

Попов, В. Б. Паскаль и Дельфи : учеб. курс / В. Б. Попов. – СПб. : Питер, 2005. – 576 с.

Алексеев, В. Е. Вычислительная техника и программирование : практикум по программированию / В. Е. Алексеев, А. С. Ваулин, Г. Б. Петрова ; под ред. А. В. Петрова. – М. : Высш. шк., 1991. – 399 с.

Прикладная информатика : учеб. пособие / А. М. Морозевич [и др.] ; под ред. А. М. Морозевича. – Минск : Выш. шк., 2003. – 335 с.

Фаронов, В. В. Турбо Паскаль 7.0. Начальный курс : учеб. пособие / В. В. Фаронов. – М. : КНОРУС, 2006. – 576 с.

Фаронов, В. В. Delphi. Программирование на языке высокого уровня : учеб. для вузов / В. В. Фаронов. – СПб. : Питер, 2004. – 640 с.

Глинский, Я. В. Turbo Pascal 7.0. Delphi : учеб. пособие для вузов / Я. В. Глинский, В. Е. Анохин, В. А. Ряжская. – СПб. : ООО «ДиаСофтЮП», 2001. – 208 с.

Могилев, А. В. Информатика : учеб. пособие для вузов / А. В. Могилев, Н. И. Пак, Е. К. Хеннер ; под ред. Е. К. Хеннера. – М. : Академия, 1999. – 816 с.

Меняев, М. Ф. Информатика и основы программирования : учеб. пособие для вузов / М. Ф. Меняев. – М. : Омега-Л, 2005. – 432 с.

Основы информатики и вычислительной техники. Информатика : практикум к лабораторным работам по одноименным курсам для студентов экономических специальностей дневной и заочной форм обучения / авт.-сост. : Н. В. Водополова, В. И. Мисюткин, С. А. Чабуркина. – Гомель : ГГТУим. П. О. Сухого, 2005. – 33 с.

Основные приемы программирования в объектно ориентированных языках : практикум по выполнению лабораторных и контрольных работ по курсам «Информатика» и «Основы информатики и вычислительной техники» для студентов экономических специальностей / авт.-сост. : Н. В. Водополова, В. И. Мисюткин, С. А. Чабуркина. – Гомель : ГГТУ им. П. О. Сухого, 2006. – 41 с.

СОДЕРЖАНИЕ

Учебное издание

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Пособие
для студентов специальности 1-25 01 07
«Экономика и управление на предприятии»
специализации 1-25 01 07 02
«Экономическая информатика»,
специальности 1-26 03 01 «Управление
информационными ресурсами»

Авторы-составители:
Мовшович Семен Михайлович
Кравченко Ольга Алексеевна

Редактор О. В. Ивановская
Технический редактор Н. Н. Короедова
Компьютерная верстка Л. Ф. Кириленкова

Подписано в печать 29.02.08. Бумага типографская № 1.
Формат 60 × 84 $^{1}/_{16}$. Гарнитура Таймс. Ризография.
Усл. печ. л. 6,04. Уч.-изд. л. 6,38. Тираж 120 экз.
Заказ №

Учреждение образования
«Белорусский торгово-экономический
университет потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.
ЛИ № 02330/0056814 от 02.03.2004 г.

Отпечатано в учреждении образования
«Белорусский торгово-экономический
университет потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.