Международный консорциум «Электронный университет»

Московский государственный университет экономики, статистики и информатики

Евразийский открытый институт

Н.В. Комлева, А.А. Смирнов, Д.В. Хрипков

ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

Учебно-методический комплекс

Москва, 2008

УДК 004
ББК 32.81
К 633

Комлева Н.В., Смирнов А.А., Хрипков Д.В. **ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ**: Учебно-методический комплекс. – М.: Изд. центр ЕАОИ, 2008. – 94 с.

Пособие содержит изложение основных понятий в сфере информатики и основ программирования, а также практические примеры.

Пособие предназначено для студентов следующих специальностей:

– «Менеджмент», «Менеджмент организаций», «Управление персоналом», «Коммерция», «Маркетинг», «Мировая экономика», «Антикризисное управление», «Бухгалтерский учет, анализ и аудит», «Финансы и кредит», «Лингвистика», «Налоги и налогообложение», «Психология».

ISBN 978-5-374-00085-6

Оглавление

Сведения об авторах

Комлева Нина Викторовна, кандидат экономических наук, заведующая кафедрой МОиТП.

Смирнов Александр Алексеевич, кандидат экономических наук, профессор кафедры МОиТП.

Хрипков Денис Викторович, преподаватель кафедры МОиТП.

1. ИНФОРМАТИКА

Содержание темы

Информация и информатизация общества. Измерение и представление информации. Технические средства реализации информационных процессов. Программные средства реализации информационных процессов. Современные технологии программирования.

Цель изучения темы формулирование исходных представлений о информатике.

Изучив данную тему, студент должен:

- знать основные понятия информации;
- знать измерение и представление информации;
- уметь измерять объем данных;
- приобрести навыки работы с учебной и технической литературой.

1.1. Информация и информатизация общества

Под ***информатикой*** в широком смысле понимается совокупность разнообразных отраслей науки, техники и производства, связанных с переработкой информации. В узком смысле информатику можно представить как совокупность следующих взаимосвязанных частей:

1) технические средства (hardware);
2) программные средства (software);
3) алгоритмические средства (brainware).

Характерно, что информатику как в широком, так и в узком смысле можно рассматривать с различных позиций:

- как отрасль народного хозяйства;
- как фундаментальную науку;
- как прикладную дисциплину.

Термин «информация» происходит от латинского слова «Informatio», что означает разъяснение, осведомление, изложение. Информатика рассматривает информацию как концептуально связанные между собой сведения, данные, понятия, изменяющие наши представления о явлении или объекте окружающего мира. Наряду с информацией в информатике часто употребляется понятие данные. Данные могут рассматриваться как признаки или записанные наблюдения, которые в данный момент не используются, но хранятся. Когда данные начинают использоваться, они превращаются в информацию.

Информация может быть преобразована в знания. Информация, полученная от специалистов, только в том случае становится знаниями, если она структурирована, специальным образом представлена, тщательно протестирована и имеет способность к развитию. Знания при использовании логического вывода позволяют порождать новые знания.

Экономическая информация представляет собой совокупность сведений, отражающих социально-экономические процессы, предназначенные для управления процессами и коллективами людей в производственной и непроизводственной сфере.

Внедрение ЭВМ, современных средств переработки и передачи информации в различные сферы деятельности послужило началом нового процесса, называемого «информатизацией».

Под информатизацией общества понимается организованный социально-экономический и научно-технический процесс создания оптимальных условий для удовлетворения информационных потребностей органов управления и граждан на основе использования информационных ресурсов. Современное материальное производство и другие сферы деятельности все больше нуждаются в информационном обслуживании, переработке огромного количества информации. Общество, в котором большинство работающих занято производством, хранением и переработкой информации, называется информационным.

1.2. Измерение и представление информации

Для измерения информации могут быть использованы два параметра:

1) количество информации;
2) объем данных.

Количество информации измеряется изменением неопределенности состояния системы. После получения дополнительной информации уменьшается априорная неопределенность состояния системы.

Объем данных измеряется количеством символов (разрядов), при этом могут быть использованы следующие единицы измерения:

1) бит (двоичный разряд);
2) байт (8 двоичных разрядов);
3) дит (десятичный разряд).

Возможность и эффективность использования информации обусловливается такими основными ее потребитель-

скими показателями качества, как репрезентативность, содержательность, достаточность, доступность, актуальность, своевременность, точность, достоверность, устойчивость.

Репрезентативность информации связана с правильностью ее отбора и формирования в целях адекватного отражения свойств объекта.

Содержательность отражает семантическую емкость, равную отношению количества семантической информации в сообщении к объему обрабатываемых данных.

Достаточность (полнота) означает, что она содержит минимальный, но достаточный для принятия правильного решения набор показателей.

Доступность определяется восприятием информации пользователем. Повышение доступности предусматривает преобразование информации в доступную и удобную для восприятия пользователем форму.

Актуальность определяется степенью сохранения ценности информации в момент ее использования.

Своевременность означает ее поступление не позже заранее назначенного момента времени.

Точность определяется степенью близости получаемой информации к реальному состоянию объекта.

Достоверность определяется ее свойством отражать реально существующие объекты с необходимой точностью.

Устойчивость отражает способность реагировать на изменения исходных данных без нарушения необходимой точности.

1.3. Технические средства реализации информационных процессов

При рассмотрении технических средств реализации информационных процессов целесообразно выделить следующие аспекты:

1) представление информации в ЭВМ;
2) логические основы построения персонального компьютера;
3) архитектуру компьютера;
4) компьютерные сети.

Представление информации в вычислительных машинах осуществляется с использованием позиционных систем счисления. Под системой счисления понимается способ записи чисел с помощью заданного набора специальных знаков (цифр).

Позиция цифры в числе, записанном с использованием позиционной системы счисления, представляет собой степень, в которую должно быть возведено основание системы счисления (в которую производится перевод), при умножении на данную цифру.

Количество знаков, используемых для записи числа, определяет название системы счисления. Так, в десятичной системе счисления имеется десять цифр (0, 1, 2, … 9); в двоичной системе счисления две цифры (0, 1); в шестнадцатеричной системе 16: (0,1, 2, 3, … 9, A, B, C, D, E, F).

Перевод чисел из любой системы счисления в десятичную обеспечивается перемножением цифр числа на основание системы счисления в соответствующей степени:

Пример:

1) Перевод числа 11011 из двоичной системы счисления в десятичную:
   4 3 2 1 0 4 3 2 1 0
   11011 = 1*2 + 1*2 + 0*2 + 1*2 + 1*2 = 16 + 8 + 2 + 1 = 27.
2) Перевод числа 51 из восьмеричной системы счисления в десятичную:
   1 0 1 0
   51 = 5*8 +1*8 = 40 + 1 =41.

Перевод из десятичной системы счисления в другие системы счисления целых чисел обеспечивается делением переводимого числа на основание системы счисления (в которую производится перевод)и записью остатков, полученных от деления, в обратном порядке.

Пример. Перевод числа 25 из десятичной системы счисления в двоичную:

25 ∟ 2
24 12 ∟ 2
(1) 12 6 ∟ 2
(0) 6 3 ∟ 2
(0) 2 (1)
(1)

Результат: 11001

Для обработки в компьютере информация, как правило, кодируется в двоичной системе счисления. В вычислительных машинах применяются две формы представления двоичных чисел:

1) с фиксированной запятой;
2) с использованием нормальной формы.

Нормальной формой представления числа называется число, записанное в виде

$A = \pm m * P^n$,

где m – мантисса числа[1];

P – основание системы счисления;

n – порядок числа.

Например, число 21.34(10) может быть представлено в следующих видах:

21.34

$2134 * 10^{-2}$

2.134*10

$0.2134 * 10^2$

Как правило, при выдаче информации с использованием нормальной формы, используется специальный символ «E», который определяет порядок числа для десятичной системы счисления.

[1] Мантисса – m – правильная P-ичная дробь, у которой 1-я цифра после запятой не равна 0, т.е. $1 / p \le m < 1$, m – мантисса.

Например: число «-678.9876», при выдаче на экран в нормальной форме, будет представлено в следующем виде «-6.7898760000E+02».

Для программирования и анализа вычислительных схем используется математический аппарат **алгебры логики**. Алгебра логики оперирует логическими высказываниями. При логической обработке информации используется такое понятие, как высказывание.

Под высказыванием понимается любое замечание, сообщение или утверждение об объективно существующем мире. Высказывание может совпадать или не совпадать с действительностью, о которой оно что-то утверждает. Высказывание может быть оценено как истинное (1) или ложное (0). Истина (TRUE) и ложь (FALSE) являются логическими значениями высказываний. Пример высказывания: «На улице идет дождь».

Различают два вида высказываний:

1) простые;
2) сложные.

Простые высказывания представляют собой неделимые на части логические объекты. Из простых высказываний можно составить сложные высказывания.

Сложное высказывание состоит из двух и более простых высказываний, соединенных с помощью логических операций. Логическое значение сложного высказывания определяется логическими значениями составляющих его простых высказываний.

В алгебре логике используются такие операции, как инверсия, конъюнкция, дизъюнкция.

Операция инверсия (логическое отрицание, NOT) преобразует значение «истинно» (1) в значение «ложно» (0), а значение «ложно» в значение «истинно».

Операция конъюнкция (логическое умножение, AND) формирует новое логическое высказывание из двух логических высказываний. Результатное высказывание «истинно» лишь тогда, когда истинны оба входящие в него высказывания.

Операция дизъюнкция (логическое сложение, OR) формирует новое логическое высказывание из двух логических высказываний. Если хотя бы одно из исходных высказываний «истинно», то и результатное высказывание «истинно».

Пример.

Логическое высказывание «(2 = 2) And (3 >= 4)» будет являться ложным, т.е. будет иметь значение «False».

Архитектура компьютера, как правило, определяется структурой компьютера и функциональными возможностями машины. В структуре компьютера, прежде всего, выделяют, микропроцессор, основную память и внешнюю память.

Под ***компьютерной сетью*** понимается совокупность компьютеров и терминалов, соединенных с помощью каналов связи в единую систему. Компьютерные сети позволяют перейти к распределенной обработке данных. Под распределенной обработкой данных понимается обработка данных, выполняемая на независимых, но связанных между собой компьютерах.

1.4. Программные средства реализации информационных процессов

Под программным обеспечением понимается совокупность программ, предназначенных для реализации целей и задач, и документации на них. Программное обеспечение в соответствии с выполняемыми функциями делится на системное и прикладное.

К системному программному обеспечению относится совокупность программ описаний и инструкций, используемых для эффективного функционирования вычислительной системы, а также при разработке новых программ.

Прикладное программное обеспечение предназначено для решения конкретных задач из различных сфер применения.

Прикладное программное обеспечение по характеру применения делится на прикладные программные средства коммерческого использования (программные продукты) и индивидуально разрабатываемые программные средства (не распространяемые программы).

Программные продукты представляют собой программное обеспечение, изготовленное на продажу; они имеют программную документацию, обеспечивающую установку и эксплуатацию программ сторонними пользователями.

Индивидуально разрабатываемые программные средства обеспечивают комплексность и функциональную завершенность прикладного программного обеспечения.

Эффективность разработки и использования прикладного программного обеспечения достигается оптимальным сочетанием данных составных частей.

1.5. Технологии программирования

Технологии программирования предназначены для повышения производительности труда при разработке и сопровождении программных изделий. Внедрение современных технологий программирования (например, использование сети Internet) позволяет решать принципиально новые задачи.

Программные технологии обладают следующими характерными свойствами:

1) развитие технологий программирования происходит необычайно быстрыми темпами;
2) программные технологии развиваются в разных направлениях.

Одним из важнейших направлений развития технологий программирования являются следующие технологии:

1) структурное программирование;
2) модульное программирование;
3) объектно-ориентированное программирование.

Структурное программирование представляет собой программирование, основанное на использовании канонических структур. Аналогией структурной программы является текст на английском языке. Структурное программирование является основой при написании текстов программных элементов.

Модульное программирование – технология, обеспечивающая разбиение единой программы на совокупность программных модулей примерно одного размера. Аналогией модульной программы является дом, построенный из кирпичей одинакового размера. Технология модульного программирования используется при организации объектов в объектно-ориентированном программировании.

Объектно-ориентированное программирование представляет собой технологию, предусматривающую формирование программы на основе заранее подготовленных объектов. В объектах объединены, инкапсулированы информация и процедуры. Путем изменения свойств и добавления процедур (методов), объекты настраиваются на конкретное применение. Использование технологии объектно-ориентированного программирования основано на применении цепочки создания программного продукта вида «Base Class → User Defined Class → Object». Поэтому, при использовании объектно-ориентированной технологии требуется знать и уметь использовать большое число базовых классов, поставляемых фирмами-разработчиками. В качестве аналогии объектно-ориентированного программирования можно взять китайский язык.

Контрольные вопросы

1. Что такое информационное общество?
2. Каким образом измеряется информация?

Тесты

1. Какой термин происходит от латинского слова «Informatio», что означает разъяснение, осведомление, изложение:
   - ☐ информация;
   - ☐ команда;
   - ☐ программа;
   - ☐ инкапсуляция.

2. Информация, полученная от специалистов, только в том случае становится …, если она структурирована, специальным образом представлена, тщательно протестирована и имеет способность к развитию:
   - ☐ данными;
   - ☐ знаниями;
   - ☐ массивом;
   - ☐ программой;
   - ☐ операционной системой.

3. В узком смысле информатику можно представить как совокупность следующих взаимосвязанных частей: технические средства (hardware); … (software); алгоритмические средства (brainware):
   - ☐ финансовые средства;
   - ☐ материальные средства;
   - ☐ программные средства;
   - ☐ средства борьбы с вирусами;
   - ☐ инструментальные средства.

4. … измеряется изменением неопределенности состояния системы:
   - ☐ программный код;
   - ☐ количество информации;
   - ☐ массив;

☐ дисперсия;
☐ объем данных.

5. Для измерения объема данных могут быть использованы следующие единицы измерения: бит (двоичный разряд); байт (…); дит (десятичный разряд):
   ☐ 2 двоичных разряда;
   ☐ 4 двоичных разряда;
   ☐ 6 двоичных разрядов;
   ☐ 8 двоичных разрядов;
   ☐ 16 двоичных разрядов.

6. Результат перевода числа 10001 из двоичной системы счисления в десятичную систему счисления будет равен:
   ☐ число 9 в десятичной системе счисления;
   ☐ число 12 в десятичной системе счисления;
   ☐ число 15 в десятичной системе счисления;
   ☐ число 17 в десятичной системе счисления;
   ☐ число 21 в десятичной системе счисления;
   ☐ число 101 в десятичной системе счисления.

7. Логическое высказывание «(3>3) Or (17=17)» будет являться:
   ☐ ложным;
   ☐ истинным.

8. Программное обеспечение в соответствии с выполняемыми функциями делится на следующие виды:
   ☐ лицензионное программное обеспечение;
   ☐ системное программное обеспечение;
   ☐ прикладное программное обеспечение;
   ☐ индивидуально разрабатываемое программное обеспечение;
   ☐ свободно распространяемое программное обеспечение.

9. Одним из важнейших направлений развития технологий программирования являются следующие технологии:
   - ☐ математическое программирование;
   - ☐ структурное программирование;
   - ☐ модульное программирование;
   - ☐ системное программирование;
   - ☐ прикладное программирование;
   - ☐ объектно-ориентированное программирование.

2. АЛГОРИТМИЗАЦИЯ ПРОЦЕССОВ ОБРАБОТКИ ДАННЫХ

Содержание темы

Основные понятия и определения. Средства изображения алгоритмов. Характеристика и классификация данных. Особенности структур данных и программного обеспечения, используемого при обработке экономической информации. Современные технологии программирования.

Изучив данную тему, студент должен:

- знать основные виды представления экономической информации при обработке на вычислительной технике;
- знать средства разработки алгоритмов обработки экономической информации;
- уметь составлять алгоритмы обработки экономической информации;
- пробрести навыки работы с учебной и технической литературой.

2.1. Основные понятия и определения

При обработке информации под ***алгоритмом*** понимается точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату. Однако алгоритмы могут быть использованы не только в вычислительной технике, но и в других областях. Например, в математике, физике или в повседневной жизни. Поэтому, допускается и более широкое толкование этого понятия.

Под ***алгоритмизацией*** вычислительного процесса понимается построение алгоритма решения задачи, результатом которого является выделение этапов процесса обработки данных, формальная запись содержания этих этапов и определение порядка их выполнения.

Алгоритмы обладают следующими свойствами:

1) детерминированностью (определенностью), которое означает, что набор указаний должен быть точным и исключать неоднозначность толкования;
2) дискретностью, которое определяет возможность расчленения вычислительного процесса на отдельные элементарные операции;
3) массовостью, предполагающей, что алгоритм должен быть пригоден для решения всех задач данного типа;
4) результативностью, что означает прекращение процесса через определенное число шагов с выдачей искомых результатов или сообщения о невозможности продолжения вычислительного процесса.

Для записи вычислительных алгоритмов используются специально разработанные алгоритмические языки. Под ***алгоритмическим языком*** понимается набор символов и система правил образования и истолкования конструкций из этих символов для задания алгоритмов.

Для выполнения на вычислительной машине алгоритм представляется в виде программы (программного проекта, программного модуля).

2.2. Средства изображения алгоритмов

Можно выделить следующие средства изображения алгоритмов:

1) словесный;
2) блок-схемный;
3) языки программирования.

При ***словесном способе*** записи алгоритма содержание последовательных этапов вычислений задается в произвольной форме на естественном языке. Например, алгоритм определения наибольшего значения из двух чисел в словесной форме будет иметь следующий вид:

Если первое число больше второго, то наибольшее значение равно первому числу, иначе наибольшее значение равно второму числу.

Преимуществом словесной формы записи является возможность применения для алгоритмизации различных процессов. Недостаток данного способа – недостаточно строгая формализация, что затрудняет понимание алгоритма вычислительной техникой.

Блок-схемный способ представляет собой графическое изображение логической структуры алгоритма. Существуют различные варианты графического представления алгоритмического процесса. Графическое представление облегчает восприятие алгоритма человеком. Как правило, графические символы, используемые представления алгоритма однозначно определены стандартом. Имеются программные системы, позволяющие преобразовывать программные проекты, представленные в графической форме в программный код.

Языки программирования представляют собой изобразительные средства, предназначенные для непосредственной реализации алгоритма на вычислительной машине. С помощью языков программирования пишется программный код, организованный в виде программы, программного модуля или отдельного метода, входящего в объект. Программный

код преобразуется в машинный язык, который понимается и выполняется вычислительной машиной.

2.3. Характеристика и классификация данных

Под ***данными*** понимается информация, представленная в виде, однозначно воспринимаемом вычислительной машиной. Данные, обрабатываемые программой, могут быть различным образом представлены и организованы. В языках высокого уровня выделяют простые (элементарные) и структурные (агрегативные) данные.

Под ***простыми данными*** понимаются данные, с которыми можно выполнять элементарные действия. Например, такие как сложение, сцепление или отрицание. Простые данные представляются в виде переменных или констант. Элементарные действия могут быть числовыми, символьными или логическими.

Структурные данные представляют собой совокупность элементарных данных, между которыми существуют связи, определяемые совокупностью правил и ограничений. Можно выделить следующие виды структурных данных: массив, записи, динамические структуры, файлы, базы данных.

Под ***массивом*** понимается упорядоченная совокупность однотипных данных.

Под ***записью*** понимается структура данных, состоящая из фиксированного числа элементов различных типов.

Под ***динамической структурой*** понимаются специальным образом организованные данные, располагаемые в динамической памяти и обрабатываемые с использованием указателей (ссылок).

Под ***файлом*** понимается поименованная совокупность логически связанных между собой данных, хранящихся на внешнем носителе.

Под ***базой данных*** понимается множество наборов единообразных записей. Набор единообразных записей, относящихся к определенной теме, называется таблицей базы данных. Обычно база данных создается для хранения и доступа к данным, содержащим сведения о некоторой предметной области.

Контрольные вопросы

1. Какими свойствами обладает алгоритм?
2. Какие можно выделить средства изображения алгоритмов?
3. Чем отличаются простые данные от структурных данных?

Тесты

1. Под … понимается построение алгоритма решения задачи, результатом которого является выделение этапов процесса обработки данных, формальная запись содержания этих этапов и определение порядка их выполнения.
   ☐ разработкой базы данных;
   ☐ алгоритмизацией вычислительного процесса;
   ☐ массивом;
   ☐ построением базы знаний;
   ☐ операционной системой;
   ☐ системой программирования;
   ☐ архивацией.

2. Свойство алгоритма, которое называют …, означает, что набор указаний должен быть точным и исключать неоднозначность толкования.
   ☐ дискретностью;
   ☐ массовостью;

☐ детерминированностью (определенностью);
☐ результативностью;
☐ надежностью.

3. Свойство алгоритма, которое называют …, означает возможность расчленения вычислительного процесса на отдельные элементарные операции.
☐ дискретностью;
☐ массовостью;
☐ детерминированностью (определенностью);
☐ результативностью;
☐ надежностью.

4. Свойство алгоритма, которое называют …, означает, что алгоритм должен быть пригоден для решения всех задач данного типа.
☐ дискретностью;
☐ массовостью;
☐ детерминированностью (определенностью);
☐ результативностью;
☐ надежностью.

5. При … формируется графическое изображение логической структуры алгоритма.
☐ использовании блок-схем;
☐ словесном способе, записи алгоритма;
☐ использовании языков программирования;
☐ использовании операционных систем.

6. При … содержание последовательных этапов вычислений задается в произвольной форме на естественном языке.
☐ использовании блок-схем;
☐ словесном способе, записи алгоритма;
☐ использовании языков программирования;
☐ использовании операционных систем.

7. … представляют собой изобразительные средства, предназначенные для непосредственной реализации алгоритма на вычислительной машине.
   ☐ блок-схемы;
   ☐ словесная запись алгоритма;
   ☐ языки программирования;
   ☐ операционные системы.

8. В языках высокого уровня выделяют простые (элементарные) данные и … .
   ☐ переменные;
   ☐ массивы;
   ☐ структурные (агрегативные) данные;
   ☐ константы.

3. ОСНОВНЫЕ КОНСТРУКЦИИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PASCAL

Содержание темы

Организация выполнения программ. Программные элементы языка. Ввод данных. Управление ветвлениями. Организация циклического выполнения команд. Обработка символьной информации.

Изучив данную тему, студент должен:

- знать основные программные конструкции, используемые при создании программных продуктов;
- уметь разрабатывать программный код, обеспечивающий использование основных конструкций;
- пробрести навыки работы с учебной и технической литературой.

3.1. Основные элементы программы на языке Pascal

Каждая программа на языке Pascal записывается в следующем порядке:

1) заголовок;
2) описательная часть;
3) операторная часть.

Заголовок состоит из одной строки; начинается со слова «Program», затем следует имя программы; завершается заголовок символом «;». Например:

Program Work1; { Петров Пётр Петрович дка-101 }

Описательная часть должна содержать объявление всех используемых переменных и массивов. Раздел описаний переменных начинается со слова «Var». Элементы описания разделяются между собой символом «;». Каждый идентификатор (имя) может обозначать только один объект описания.

Uses crt, printer ;

Var a, b: byte; c1, c2: integer;

Операторная часть реализует алгоритм решения задачи. Она представляет собой заключенную в слова «Begin» и «End» последовательность операторов. После слова «End» всегда ставится точка, фиксирующая конец программы, – «End.».

При записи программы используются латинские буквы (с них запись и начинается), цифры и набор специальных знаков.

Можно: b или A2 или A23_34

Нельзя: 23A или 24_A2B

В языке Pascal различаются операции деления «/» и целочисленного деления «Div». Для нахождения остатка от деления используется операция «Mod».

5 / 2 = 2,5

5 div 2 = 2

5 mod 2 = 1[1]

[1] Можно разложить в формулу A mod B = A – ((A div B) * B).

```
Var i: byte;
i := 5 / 2;
```

В языке Pascal отсутствует символ для обозначения возведения числа в степень. Для этой цели могут использоваться функции. Например, для возведения числа «X» в степень «N», можно использовать следующую конструкцию «Exp(N*Ln(X))».

Для обозначения обрабатываемых объектов используются переменные и константы. i := 10; b = 11;

Под переменной понимается программный элемент, который имеет имя и значение. Имя переменной уникально и неизменно, а значение может меняться в процессе выполнения алгоритма.

Имя переменной (идентификатор) представляет собой символьное обозначение, которое отличает заданную переменную от других объектов программы. Имя может содержать буквы, цифры и знак подчеркивания. Первым символом имени должна быть буква. Имя переменной не должно совпадать с зарезервированными (служебными) словами языка.

Можно выделить следующие типы данных:

1) целое число (Integer). Значение переменной этого типа занимает 2 байта памяти (при реализации в среде Turbo Pascal);
   Byte целое число занимает 1 байт памяти (1 байт = 8 бит, 1 бит = 0 или 1).
   1 байт может иметь значение не больше чем = 255 (11111111);
2) десятичное число, которое может иметь как целую, так и дробную часть (Real). Значение занимает 4 байта памяти;
3) строка (String). Занимаемая память зависит от числа символов в строке;
4) логический тип (Boolean). Значение переменной этого типа занимает 1 байт памяти – либо ДА либо НЕТ.
   Все переменные описываются в разделе Var.
   Пример:

```
Var
  I, J, N, M: Integer;
  Sr: Real;      b1: byte;
  S, S1: String;
  I, FL: Boolean;
```

Константы не меняют своего значения в процессе выполнения программного кода. Числовые константы могут содержать знак числа и точку, отделяющую целую часть от дробной части. Например: -73; 35.49.

Строковые константы представляют собой последовательность символов, заключенных в апострофы. Например: 'Итоговое значение'.

3.2. Операторы языка

Для присвоения значения переменным служит ***оператор присваивания***. В операторе присваивания сначала пишется идентификатор переменной, затем символ «:=» (присвоить значение), затем выражение. Признаком конца оператора присваивания является символ точка с запятой.

Например: i:=1; Max:= A[i];

Для объединения группы операторов в единое целое предназначен ***составной оператор***. Он представляет собой один или несколько операторов, заключенных в операторные скобки «Begin» и «End». Каждый из внутренних операторов может быть любым допустимым на языке Pascal, в том числе составным.

Объединение последовательности операторов, в составной оператор, осуществляется в том случае, если необходимо совместное выполнение некоторой группы операторов. Как правило, такая необходимость возникает при организации ветвлений и циклических процессов.

```
If ….     then
```

```
begin
  write (‘’);
  write (‘’);
end;
```

For .... do begin write (‘’); write (‘’); end;

Пример записи составного оператора:

```
Begin
   A:=1; B:=0;
End;
```

Ввод и вывод информации, в простейшем случае, обеспечивается с использованием клавиатуры и экрана.

```
Readln (a, b, c);
2 3 4
a b c
Write(‘….i = ’);
Readln(i);
…. I= _
```

Для ввода информации используются операторы «***Read***» или «***Readln***». Однократным выполнением либо оператора «Read», либо оператора «Readln» можно ввести значения как в одну, так и в несколько переменных. Единственное различие между ними состоит в том, что после выполнения оператора «Readln» курсор переводится в начало следующей строки, а после выполнения оператора «Read» он остается на той же строке.

Для вывода информации на экран могут быть использованы операторы «***Write***» и «***Writeln***». Отличие заключается в том, что после выполнения оператора Writeln обеспечивается переход на новую строку.

Программный код рекомендуется сопровождать комментариями. Для выделения комментария используются фигурные скобки «{}» или две наклонные черты «//». Комментарии не влияют на ход выполнения программы.

Например:

```
{ Ввод документа }   I: = 10;
// Ввод документа      I: =10;
```

3.3. Условный оператор и его применение для организации ветвлений

Условный оператор предназначен для изменения порядка выполнения операторов при выполнении логического условия.

Синтаксис условного оператора имеет следующий вид:

IF <условие> THEN <оператор1> [ELSE <оператор2 >] ;

Если условие истинно, то выполняется <оператор1>.

Если условие ложно, то выполняется <оператор2>.

Если не задана конструкция «ELSE», то при ложном условии выполняется переход к следующему оператору.

Логические условия подразделяются на простые и сложные.

Простое условие представляет собой два выражения, между которыми помещается знак сравнения. Например: Price*Quant>=Max. If (Price*Quant>=Max)or(Quant=Max)

Сложное условие представляет собой последовательность простых условий, которые соединены между собой знаками логических операций And, OR, NOT. Простые условия, используемые при формировании сложного логического условия, обязательно выделяются скобками.

Пример:

```
If ((Day >= 22) And (Month = 12)) Or
   ((Day <= 20) And (Month = 1)) Then
    ZnakZ :='Козерог';
```

Когда в зависимости от условия требуется выполнить несколько операторов, они объединяются с помощью операторных скобок 'Begin' и 'End' в составной оператор.

Пример:

```
If A = 0 Then
Begin
  X := 4;  Y := X + 2;
  Z := X / Y;
End;
```

Программа на паскале целиком

```
Program my;
Uses crt;
Var A:byte;
Begin A:=4;
End.
```

3.4. Управление ветвлениями с помощью оператора Case

Когда необходимо разветвление на значительное число ветвей алгоритма, целесообразно использовать оператор «Case». Синтаксис оператора

Case <селекторное выражение> Of
 <значение 1>: <оператор 1>;
<значение 2>: <оператор 2>;
. . .

<значение N>: <оператор N>;
[Else <оператор>;]
End;

В простейшем случае селекторное выражение представляет собой переменную, объявленную с типом Integer. При выполнении команды «Case» поочередно просматриваются все заданные значения. Если значение переменной совпадает с заданным значением, то выполняется заданная команда. Все заданные значения должны быть уникальны.

Конструкция «Else» определяет команду, которую необходимо выполнить, если заданные значения не совпадают с имеющимся значением селекторного выражения.

Ветви оператора «Case» могут содержать команду «Case».

Пример. Ввести числовое значение дня недели и выдать на экран его символьное обозначение.

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjCase;
{$APPTYPE CONSOLE}
Uses
 SysUtils;
Var
  A: Integer;
  S: String;
Begin
 Writeln ('Input number');
 Readln (A);
 Case A Of
  1: S:='Monday';
  2: S:='Tuesday';
  3: S:='Wednesday';
  4: S:='Thursday';
  5: S:='Friday';
  6: S:='Saturday';
```

```
 7: S:='Sunday';
 Else S:='Mistake number';
End;
Writeln (S);
Readln;
End.
```

3.5. Операторы цикла

Когда какая-либо группа операторов должна выполняться неоднократно, требуется организовать циклический процесс. Имеется две разновидности операторов цикла.

Первая разновидность используется для задания циклического процесса, если число повторений не известно заранее.

Вторая разновидность используется для организации циклов с известным числом повторений.

Для организации циклов с неизвестным числом повторений используется оператор WHILE.

Синтаксис оператора

WHILE <логическое выражение> Do

<оператор>;

Пока логическое условие истинно, выполняются оператор, следующий после слова «Do». Если условие ложно, то выполняется оператор, следующий после завершения оператора «While».

Оператор WHILE является канонической конструкцией. Это значит, что с его помощью можно организовать любой циклический процесс.

Пример:

Обучающая программа, обеспечивающая усвоение материала по организации разветвлений алгоритма. После завершения процесса усвоения материала выдается анализ работы с количеством допущенных ошибок.

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}
```

```
Program PrjWhile;
{$APPTYPE CONSOLE}
Uses
  SysUtils;
Var
  Answer: String;
  Mistake: Integer;
  FL: Boolean;
Begin
  FL:=False;
  Mistake:=0;
  While FL= False Do
  Begin
    Writeln ('Kakoi operator ispolzyetsia');
    Writeln ('esli neobhodimo razvetvlenie');
    Writeln ('na znachitelnoe chislo');
    Writeln ('vetvei algoritma');
    Writeln ('Var; While; Case;');
    Writeln ('If; Begin ');
    Writeln ('Vvedite oboznachenie operatora');
    Readln(Answer);
    If Answer = 'Case' Then
      Begin
       Writeln ('Otvet veren');
       FL:=True;
      End
    Else
      Begin
       Writeln ('Otvet ne veren');
       Mistake:=Mistake +1;
      End;
  End;
  Writeln ('Kolichestvo oshibok', Mistake);
  Readln;
End.
```

Для организации циклического процесса при известном числе повторений используется управляющая структура (оператор) «For». Структура «For» имеет две разновидности.

Наиболее распространенная форма имеет вид:

For <управляющая переменная> :=<выражение1> To <выражение2> Do <оператор>;

частный случай: for i:=1 to N do write(‘’);

Оператор, указанный после слова «Do», выполняется для каждого заданного значения управляющей переменной. Выполняемый оператор может представлять собой составной оператор, т.е. группу операторов, заключенную в операторные скобки «Begin» и «End».

Переменная, задаваемая после слова «FOR», называется переменной цикла. Управляющая переменная представляет собой целое число. Первоначально управляющей переменной присваивается значение, определенное как <выражение1>. Затем управляющая переменная будет автоматически принимать очередные значения, увеличенные на единицу. Выполнение завершается после того, как управляющая переменная достигнет значения, определенного как <выражение 2>.

После выхода из цикла значение переменной цикла считается неопределенным. Менять значения переменной или выражений внутри цикла программным путем не рекомендуется.

Пример.

Программный код, обеспечивающий нахождение наибольшего общего делителя для трех чисел, имеет следующий вид:

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjNod;
{$APPTYPE CONSOLE}
Uses SysUtils;
```

```
Var
  A, B, C, Nod, I: Integer;
Begin
  Readln (A, B, C);
  For I:= 1 To A Do
    If (A Mod I = 0) And (B Mod I = 0) And (C Mod I = 0)
 Then Nod:= I;
  Writeln('Nod=',Nod);
  Readln;
End.
```

Вторая разновидность оператора «For» отличается от первой тем, что изменение управляющей переменной идет сверху вниз от большего значения к меньшему значению. Синтаксис второй разновидности:

For <управляющая переменная> :=<выражение1> To downto <выражение2> Do <оператор>;

Правильный вариант:

```
For i:=1 to 3 do          {i=   1,  2,   3 }
For i:=1 downto -1 do   {i=   1,  0,  -1}
```

Неправильный вариант:

```
For i:=3 to 1 do
For i:=1 downto 3 do
```

Первоначально управляющей переменной присваивается значение, определенное как <выражение1>. Затем управляющая переменная будет уменьшаться на единицу. Выполнение завершается после того, как управляющая переменная достигнет значения, определенного как <выражение 2>.

Пример. Программный код обеспечивает выдачу заданного количества чисел натурального ряда, предшествующих введенному числу.

{ для PASCAL первые 3 строки заменяются на Uses Crt;}

```
Program PrjDownTo;
{$APPTYPE CONSOLE}
Uses SysUtils;
Var
  N, Q, L, I: Integer;
Begin
  Readln(N, Q);
  L:=N-Q;
  For I:= N-1 DownTo L Do
    Writeln (I);
  Readln;
End.
```

3.6. Обработка символьной информации

Представление информации в виде символьных строк широко используется при решении экономических задач. Используемые символьные переменные обязательно должны быть объявлены с типом «String».

Обработка символьной информации обеспечивается при помощи совокупности специальных функций, процедур и команд. Наиболее применимыми функциями являются следующие:

- Length (<символьная строка>), которая определяет количество символов в строке;
- Copy (<строка>, <начальная позиция>, <количество символов>), которая выделяет из заданной строки, начиная с заданной позиции, требуемое количество символов;
- Pos (<подстрока>, <строка>), которая определяет местоположение подстроки в строке;
- операция «сцепление» (+), которая позволяет объединить две символьные строки в единое целое.

Нужно помнить, что строка – это набор символов, организованных в виде массивов (подробнее об этом в пункте 4.1).

Пример.

Имеется список сотрудников. Он представлен в виде символьной строки. Каждый сотрудник обозначен фамилией и инициалами. Фамилия и инициалы разделяются символами подчеркивания и точками. Пробелы между фамилией и инициалами не допускаются. Фамилии различных сотрудников разделены запятыми и пробелами.

Например: Александров_А.И., Петров_Н.П., Сергеев_В.С.

Требуется проверить список на наличие заданного сотрудника. По результатам проверки выдается либо сообщение «Сотрудник есть в списке», либо сообщение «Сотрудника нет в списке».

Программный код, обеспечивающий выполнение заданного алгоритма, имеет следующий вид:

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjString;
{$APPTYPE CONSOLE}
Uses
  SysUtils;
Var
   St,Sc,S: String;
   I,J,Ln,Ns,R: Integer;
   FL: Boolean;
Begin
  Writeln('Vvedite stroku');
  Readln(St);
  Writeln('Vvedite familiu sotrudnika');
  Readln(Sc);
  Ln:= Length(St); I:= 1;
  While I<= Ln  Do
    If (St[I]<>' ') And (St[I]<>',')Then
     Begin
      Ns:= I;
      While (St[I]<>' ')And (St[I]<>',') And (I<=Ln)  Do
        I:= I+1;
```

```
    R:= I-Ns; S:= Copy (St,Ns,R);
    IF S = Sc Then FL:= True;
   End
  Else
   I:= I+1;
 If FL=True Then Writeln ('Da')
  Else Writeln ('Net');
 Readln;
End.
```

3.7. Организация выполнения программы в среде DELPHI

Консольным приложением называется программа, в которой ввод и вывод информации обеспечивается в специально созданном окне (аналогично программам для DOS). Для вывода информации в консольное окно используется команда WRITE. Для ввода информации из консольного окна используется команда READ.

Создание консольного приложения в Delphi определяет директива {$APPTYPE CONSOLE}.

Когда запускается консольное приложение, Windows создает окно текстового ввода/вывода, через которое пользователь взаимодействует с приложением. Стандартные устройства ввода/вывода автоматически связываются с окном консоли.

Для создания нового консольного приложения используется команда File\New\Console application. При выполнении этой команды обеспечивается автоматический переход в текстовый редактор Delphi. В текстовом редакторе автоматически генерируются следующие конструкции:

1) заголовок процедуры;
2) опция компилятора, определяющая консольное приложение;

3) конструкция USES, обеспечивающая подключение системной библиотеки;
4) конструкции, определяющие начало и конец операторной части.

Для сохранения программы выбирается команда «File\Save Project As». Для выполнения набранного программного кода выбирается команда «Run» из меню «Run».

Следует учитывать, что система Delphi предназначена для объектно-ориентированного программирования и использование консольного режима имеет вспомогательный характер. Поэтому для того чтобы сформированное окно ввода/вывода сохранилось на экране, рекомендуется добавить в программный код команду фиктивного ввода.

Кроме того, следует учитывать, что при выполнении консольные приложения используют коды ASCII, которые используются в операционной системе MsDos. В операционной системе Windows используются коды ANSI. Данное обстоятельство может привести к появлению символов псевдографики в выдаваемом тексте. Поэтому в тексте программного кода, при написании выводимой информации, целесообразно использовать латинские буквы.

Пример. Программа, обеспечивающая нахождение суммы двух чисел. Данная программа представлена в виде консольного приложения.

```
Program DELPH_01;
{$APPTYPE CONSOLE}
 {Лекционный пример}
Uses
  SysUtils;
Var A,B,C: Integer;
Begin
  Writeln('Input, please');
  ReadLn(A,B);
  C:=A+B;
```

```
  Writeln ('Result ',C);
        Readln;
End.
```

При использовании консольного режима система Delphi формирует одновременно несколько файлов, в частности:

1) файл «приложение», который может использоваться для выполнения программного кода;
2) файл «Delphi project», который может быть использован для загрузки проекта в систему Delphi;
3) файл модуля, содержащий код консольного приложения. Данный файл имеет расширение «.Dpr».

Для восстановления программы достаточно сохранить файл проекта или файл модуля. Если сохраняется файл модуля «.DPR», то после открытия файла **«.DPR»** требуется обеспечить повторную компиляцию программы. Повторную компиляцию программы можно обеспечить либо выбором пункта COMPILE в меню PROJECT, либо нажатием кнопок CTRL+F9.

Контрольные вопросы

1. В какой последовательности записывается программный код на языке Pascal?
2. Какие конструкции используются для организации ветвлений?
3. Какие конструкции используются для организации циклических процессов?

Задачи

Выполните разработку следующих программ на языке Pascal.

1. Ввести два числа. Если первое число окажется больше второго, то обменять их значения. Выдать сообщение «Был выполнен обмен» или «Обмена не было». Выдать новые значения чисел.
2. Ввести три числа.
   Если они все равны между собой, то выдать сообщение: «Все числа равны между собой».
   Если среди них только два числа равны между собой, то выдать сообщение: «Два числа равны между собой».
   Если среди них нет чисел равных между собой, то выдать сообщение: «Все числа различны».
   Предусмотреть выдачу только одного сообщения.
3. Ввести три различных числа.
   Самое большое из трех чисел уменьшить на пять.

Тесты

1. Каждая программа на языке Pascal записывается в следующем порядке:
   заголовок;
   описательная часть и … .
   ☐ процедуры;
   ☐ функции;
   ☐ переменные;
   ☐ массивы;
   ☐ операторная часть;
   ☐ константы.

2. Программа на языке Pascal завершается конструкцией … .
   ☐ FOR;
   ☐ IF;

☐ WHILE;
☐ BEGIN;
☐ END;
☐ CASE.

3. Укажите соответствие между обозначением операции и ее назначением:

а)	DIV	1) нахождение остатка от деления
б)	“/”	2) целочисленное деление
в)	MOD	3) операция деления чисел, имеющих дробную часть.

4. Для возведения числа X в степень N необходимо записать конструкцию … .
☐ X**N
☐ X ^ N
☐ X↑N
☐ Exp(N*Ln(X))
☐ X^^N

5. Укажите соответствие между обозначением оператора и его назначением:

а)	IF	1) для объединения группы операторов в единое целое
б)	WHILE	2) для разветвления на значительное число ветвей алгоритма
в)	FOR	3) для изменения порядка выполнения операторов при выполнении логического условия
г)	CASE	4) для задания циклического процесса, если число повторений не известно заранее
д)	BEGIN END	5) для организации циклов с известным числом повторений.

6. Чему будет равно значение переменной «S» после выполнения следующего алгоритма:
   S:=2; For I:= 1 To 4 Do S:=S+I;?
   ☐ 4
   ☐ 10
   ☐ 12
   ☐ 20
   ☐ 22

4. ПРОГРАММНАЯ ОБРАБОТКА СТРУКТУРНЫХ ТИПОВ

Содержание темы

Организация информации в виде массива. Программная обработка экономической информации, представленной в виде записей. Особенности обработки экономической информации, организованной в виде массива записей.

Изучив данную тему, студент должен:

- знать особенности использования структурных типов для решения экономических задач;
- уметь разрабатывать программный код, обеспечивающий использование структурных типов для решения экономических задач;
- пробрести навыки работы с учебной и технической литературой.

4.1. Организация информации в виде массивов

Под ***массивом*** понимается упорядоченная совокупность однотипных элементов, расположенных в оперативной памяти. Все элементы массива имеют одно имя. Обращение к элементам массива осуществляется с помощью индексов. Индексы могут задаваться в виде констант, переменных или арифметических выражений. Каждому элементу массива ставится в соответствие один или несколько индексов. Если указывается несколько индексов, то индексы разделяются запятой.

Например:

```
A[1] := 1;
B [I, J] := A [J];
C[J+1] := W;
```

Если для обращения к элементу массива используется один индекс, то массив называется одномерным. Если используется два индекса – двумерным.

Тип массива может быть определен в разделе Type. При задании типа массива определен диапазон значений и тип элементов массива. Диапазон значений индекса содержит минимальное и максимальное значения индекса, разделенные двумя точками. Границы диапазона задаются в виде целочисленных констант. Использование переменных для задания границ диапазона не допускается. Допускается задание отрицательного значения индекса.

Например:

```
Type
    TA2 = Array [1..40, 1..40] Of Integer;
    TA1 = Array [-3..20] Of Real;
```

Для выделения оперативной памяти для элементов массива требуется задать описание массива в разделе «Var». Опи-

сание массива выполняется в соответствии со следующим синтаксисом:

<Список имен массивов>: <тип>;

Для задания типа может быть использовано описание типа, определенное в разделе «Type».

Например,

```
Var
   A,B:TA2;
  C,D: Array [1..30] Of Real;
  S: Array[1..50] Of String;
```

В процессе обработки можно использовать значения индекса, не выходящие за границы диапазона.

При использовании консольного режима поступление информации в массив обеспечивается при помощи команды Read.

Например,

```
Read (A[I]);
```

Пример.

Программный код, обеспечивающий определение наиболее часто встречающегося элемента в одномерном массиве.

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program MaxFreq;
{$APPTYPE CONSOLE}
Uses
 SysUtils;
Type
 TA1= Array[1..30] Of Integer;
Var
 A:TA1;
 I,J,N,Max,Q,Amax: Integer;
begin
```

```
  Writeln('Input N');
  Readln (N);
  Writeln('Input array');
  For I:= 1 To N Do
    Read( A[I]);
  Max:= - MaxInt;
  For I:=1 To N Do
  Begin
    Q:=0;
    For J:= 1 To N Do
     If A[I] = A[J] Then Q:=Q+1;
    If Q>Max Then
    Begin
      Max:=Q;
      Amax:=A[I];
    End;
  End;
  Writeln (Amax);
  { Фиктивная команда ввода.
   Задается любое число}
  Readln;
  Readln;
End.
```

4.2. Организация информации в виде записей

Под ***записью*** понимается структура данных, состоящая из фиксированного числа элементов. Каждый элемент записи имеет свой идентификатор и тип. Идентификатор в пределах записи должен быть уникальным. Для обращения к отдельным элементам записи указываются составные имена. Составное имя состоит из имени записи, после которого ставится точка и записывается идентификатор элемента записи. Например, «PotrM.Date». Использование записей позволяет объ-

единять в единое целое совокупность логически связанных реквизитов документа.

В программной среде Pascal или Delphi для описания структурных взаимосвязей различных реквизитов предназначено утверждение «Record», которое описывается в разделе «Type».

Конструкция «Record» записывается в соответствии со следующим синтаксисом:

```
<имя структуры> = Record
        <имя элемента>: <тип>;
        [<имя элемента>:<тип>;]
End;
```

Выделение памяти для заданной структуры выполняется в разделе «Var» при описании переменных указанного типа.

Например:

```
Type
  TPotrM = Record
    Date: String [4];
    ShCzeh: String [6];
    NameMat: String [20];
    Potr: Real;
  End;
Var
 PotrM: TPotrM;
```

При обработке документов, представленных в виде записей, можно использовать имена записей.

Например,

```
W := PotrM;
```

Пример.

Имеется документ, который содержит информацию по потребности материалов на текущий период. Документ содержит следующие реквизиты:

1) дата, представленная в виде четырех символов. Первые два символа определяют месяц. Третий и четвертый символ определяют год;
2) код цеха, представленный в виде шести символов;
3) код материала, представленный в виде десяти символов;
4) потребность в материале, представленная в виде 8 цифр, две из которых определяют дробную часть.

Программный код, обеспечивающий определение квартала, к которому относится документ, имеет следующий вид:

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjPotrM;
{$APPTYPE CONSOLE}
Uses
 SysUtils;
Type
 TPotrM = Record
   Date: String [4];
   ShCzeh: String [6];
   NameMat: String [20];
   Potr: Real;
 End;
Var
 PotrM: TPotrM;
 Kv, Mes, Rc: Integer;
 SMes: String;
Begin
// Ввод записи
 Writeln ('Vvedite datu');
 Readln (PotrM.Date);
 Writeln ('Kod Czeha');
 Readln (PotrM.ShCzeh);
 Writeln ('Kod materiala');
 Readln (PotrM.NameMat);
 Writeln('Potrebnoct v materiale');
```

```
  Readln (PotrM.Potr);
// Определение квартала
  SMes := Copy(PotrM.Date, 1, 2);
  Val(SMes,Mes,Rc); Kv:=0;
  Case Mes Of
   1..3: Kv:= 1;
   4..6: Kv:= 2;
   7..9: Kv:= 3;
   10,11,12: Kv:= 4;
  Else
   Writeln('Oshibka v zadanii nomera mesjacza');
  End;
  Writeln ('Kvartal',Kv);
  Readln;
End.
```

4.3. Организация информации в виде множества

Под ***множеством*** понимается набор однотипных данных.

Тип множества определяется следующей синтаксической конструкцией:

<идентификатор множества> = Set Of<тип множества>.

Над множеством могут выполняться следующие действия:

1) проверка принадлежности элемента множеству
   <элемент> IN <множество>;
2) сложение множеств
   <множество> + <множество>;
3) разность множеств
   <множество> - <множество>;
4) пересечение множеств или определение общих элементов
   <множество> * <множество>;
5) проверка вхождения одного множества в другое

<множество> <= <множество>;

6) проверка эквивалентности
<множество> = <множество>.

Для проверки наличия элементов в множестве можно использовать пустое множество, которое обозначается двумя квадратными скобками «[]».

Например,
IF OperA=[ ] Then Writeln ('Пустое множество');

Множества не могут задаваться, в качестве параметров, при выдаче на печать в процедуре Write. Поэтому печать элементов множества обеспечивается путем использования специальных алгоритмов. Например, для выдачи символов, входящих в множество, можно использовать следующий алгоритм:

```
Var
 C: Char;
  . . .
For C := 'A' To 'Z' Do
  If C In SetA Then Writeln (C);
```

Обработка информации, представленной в виде множества перечислимого типа, имеет следующие особенности:

1. Определение набора элементов, которые могут входить в множество, может быть выполнено в разделе «'Type». Например, SpisOper = Set Of (Oper1, Oper2, Oper3, Oper4); . Имена, заданные в списке, не могут быть использованы для обозначения других переменных.
2. В разделе Var определяются переменные, имеющие заданный перечислимый тип. Например, OperA,OperB: SpisOper; . При объявлении переменной информация, заданная в разделе Type, не заносится. В разделе Var рекомендуется определить специальную переменную, которая может быть использована для обращения к элементам множества. Переменная должна быть описана с типом «Byte». При использовании переменной типа

«Integer» или числовой константы выдается сообщение об ошибке.

3. Для занесения информации в переменную, определенную как перечислимый тип, может быть использован оператор присваивания вида

   <Множество>: = <элемент>;

   Например,
   OperA: = Oper1;
   OperA: = OperA + Oper1;

4. Для занесения информации в переменную, определенную как перечислимый тип, может быть использовано обращение к элементам множества, с помощью специальной переменной, имеющий тип «Byte». Следует обратить внимание, что специальная переменная определяет не элемент множества, а подмножество элементов, задаваемых в соответствии с двоичной маской. Например, значение специальной переменной, равное 6, обеспечивает одновременное занесение второго и третьего элемента. (Двоичный код 6 равен «110».) Для занесения отдельных переменных необходимо использовать значения специальной переменной, равные числу 2, в соответствующей степени.

   Например, совокупность операторов

   W:= 8;
   OperA:=SpisOper(W);

   обеспечит занесение четвертого элемента множества.
5. При выполнении операции «IN» для проверки множества, не могут быть использованы другие значения, кроме значений, определенных в конструкции «Set Of».
6. При выполнении команды «IN» можно использовать элемент, заданный в перечислимом типе. Например, «Oper1 In OperA». Использовать конструкцию, опреде-

ляющую данный элемент с использованием индекса, не допускается. Например, при использовании конструкции «SpisOper(w) In OperA» выдается сообщение об ошибке.

7. Для выдачи на экран элементов множества перечислимого типа может быть использована совокупность операторов вида

   If Oper1 In OperA Then Writeln('Oper1');

8. Следует различать множества на перечислимом типе и перечислимый тип. Индексация элементов и операции, выполняемые над перечислимым типом, отличны от индексации и операций над множеством перечислимого типа.

Особенности обработки экономической информации, организованной в виде массива записей

Представление информации в виде массива записей используется для организации экономической информации в оперативной памяти. В этом случае совокупность реквизитов, составляющих отдельный документ, представляется в виде отдельного элемента массива.

При определении типа массива записей в разделе «Type», размерность массива записей определяется максимально возможным количеством обрабатываемых документов. В многострочных документах в виде отдельного элемента массива записей представляется каждая строка документа.

Фактически используемое количество элементов массива будет соответствовать числу введенных документов или введенных строк многострочного документа.

Обращении к конкретным реквизитам документов, организованных в виде массива записей, выполняется в соответствии со следующим синтаксисом:

<имя массива> [значение индекса]. <имя реквизита>

Например, конструкция для обращения к реквизиту «Date», второго элемента массива «Potrm» имеет следующий вид:

PotrM[2].Date

Пример.

Имеется многострочный документ, содержащий информацию о продаже черных металлов. Документ содержит следующие реквизиты:

1) дата, представленная в виде четырех символов. Первые два символа определяют месяц. Третий и четвертый символы – год;
2) код покупателя, представленный в виде десяти символов;
3) всего продано на сумму[1];
4) чугун;
5) прокат;
6) прочие металлы[2].

Программный код обеспечивает определение процента проданного проката в общей сумме продаж.

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjChMet;
{$APPTYPE CONSOLE}
Uses
  SysUtils;
Type
  Tchmet = Record
    Date: String [4];
```

[1] Реквизит 3 строки – это сумма 4, 5 и 6.

[2] Строки 2–6 – реквизит представлен в виде 10 цифр, две из которых определяют дробную часть.

```
    ShPokyp: String [10];
    Prod: Real;
    Chyg: Real;
    Prok: Real;
    Proch: Real;
  End;
  T_Array = Array [1..40] Of Tchmet;
Var
 ChMet: T_Array;
 N,I: Integer;
 SProd, SProk, PerCent:Real;
Begin
  Writeln ('Vvedite kolichestvo zapisei');
  Readln (N);
  Writeln ('Vvedite document');
  For I:= 1 To N Do
  Begin
    Writeln ('Vvedite Daty');
    Readln ( ChMet[I].Date);
    Writeln ('Kod pokupatelja');
    Readln (ChMet[I].ShPokyp);
    Writeln ('Vsego prodano');
    Readln (ChMet[I].Prod);
    Writeln ('V tom chisle chugun');
    Readln (ChMet[I].Chyg);
    Writeln ('V tom chisle prokat');
    Readln (ChMet[I].Prok);
    Writeln ('V tom chisle prochie');
    Readln (ChMet[I].Proch);
   End;
 SProd:= 0; SProk:= 0;
 For I:= 1 To N Do
 Begin
   SProd:= SProd + ChMet[I].Prod;
   SProk:= SProk + ChMet[I].Prok;
 End;
```

```
PerCent := (SProk / SProd) * 100;
Writeln ('Procent prokata', PerCent :7:3);
Readln;
End.
```

Контрольные вопросы

1. Какие конструкции используются для описания экономической информации?
2. Каким образом обеспечивается ввод совокупности документов в оперативную память?

Задачи

Выполните разработку следующих программ на языке Pascal.

1. Имеется документ «Товарно-транспортная накладная». Документ содержит следующие реквизиты:
   1) поставщик. Реквизит представлен в виде символьной строки длиной в 20 символов;
   2) пункт назначения. Реквизит представлен в виде символьной строки длиной в 20 символов;
   3) название товара. Реквизит представлен в виде символьной строки длиной в 20 символов;
   4) количество. Реквизит представлен в виде числа, не имеющего дробной части;
   5) цена. Реквизит представлен в виде числа, имеющего дробную часть.

   Требуется разработать программный код, обеспечивающий ввод информации, формирование массива записей и определение общей суммы товара, отправляемого в заданный пункт назначения.

2. Имеется документ, содержащий информацию о стоимости поставки определенного вида продукции от поставщиков к потребителям. Строка документа содержит следующие реквизиты:

1) код продукции. Реквизит представлен в виде символьной строки длиной в 10 символов;
2) код поставщика продукции. Реквизит представлен в виде символьной строки длиной в 20 символов;
3) код потребителя продукции. Реквизит представлен в виде символьной строки длиной в 20 символов;
4) сумма поставленной продукции. Реквизит представлен в виде числа, имеющего дробную часть.

Требуется разработать программный код, обеспечивающий ввод информации, формирование массива записей и определение поставщика, по которому наблюдается максимальное значение реквизита «сумма поставленной продукции».

3. Имеется документ, содержащий информацию об экономических показателях различных стран. Строка документа содержит следующие реквизиты:
   1) «Название региона» – 20 символов;
   2) «Название государства» – 30 символов;
   3) «Численность населения» – число, имеющее дробную часть;
   4) «Валовой внутренний продукт» – число, имеющее дробную часть;
   5) «Продукция промышленности» – число, имеющее дробную часть.

   Необходимо:
   а) Разработать программный код, который обеспечит выдачу информации о численности населения для заданного государства;
   б) Разработать программный код, который обеспечит определение региона, имеющего максимальный суммарный валовой внутренний продукт;
   в) Разработать программный код, который обеспечит формирование массива из записей, относящихся к заданному региону.

г) Разработать программный код, который обеспечит определение доли валового внутреннего продукта заданного государства в суммарном объеме валового внутреннего продукта, региона, к которому относится данное государство. Название региона, к которому относится государство, определяется по строке документа, относящейся к заданному государству;
д) Разработать программный код, который обеспечит упорядочивание строк документа по названию региона;
е) Разработать программный код, который обеспечит формирование документа, содержащего итоговые значения по регионам. Сформированный документ должен быть представлен в виде массива записей. Итоговый документ содержит следующие реквизиты:
   1) «Название региона» – 20 символов;
   2) «Численность населения в регионе» – число, имеющее дробную часть;
   3) «Валовой внутренний продукт региона» – число, имеющее дробную часть;
   4) «Продукция промышленности региона» – число, имеющее дробную часть.

Примечание: Предполагается, что строки исходного документа упорядочены по названию региона.

Тесты

1. Тип массива может быть определен в разделе … .
   ☐ FOR;
   ☐ IF;
   ☐ WHILE;
   ☐ BEGIN;
   ☐ TYPE;
   ☐ CASE.

2. Допускается ли задание отрицательного значения границы диапазона индекса массива?
   ☐ да;
   ☐ нет.
3. Для описания структурных взаимосвязей различных реквизитов документа предназначено утверждение:
   ☐ FOR;
   ☐ IF;
   ☐ WHILE;
   ☐ BEGIN;
   ☐ RECORD;
   ☐ CASE.

4. Допускается ли объединять записи в массив записей?
   ☐ да;
   ☐ нет.

5. Конструкция для обращения к реквизиту «Date» второго элемента массива «Potrm» имеет вид:
   ☐ Potrm[Date[2]];
   ☐ PotrM[2].Date;
   ☐ Date[2]:=Potrm.

5. МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ

Содержание темы

Организация модульной структуры программы. Особенности реализации модульной структуры при обработке экономической информации. Использование процедур. Использование функций. Процедуры и функции без параметров. Организация внешних модулей.

Изучив данную тему, студент должен:

- знать особенности использования современных технологий программирования для решения экономических задач;
- уметь разрабатывать программный код, обеспечивающий использование технологии модульного программирования для решения экономических задач;
- пробрести навыки работы с учебной и технической литературой.

5.1. Организация модульной структуры программы

Delphi (Pascal) позволяет эффективно реализовывать технологию модульного программирования. В соответствии с концепцией модульного программирования программа должна представлять собой не единый программный код, а совокупность модулей. При использовании модульного подхода рекомендуется единый алгоритм создавать из совокупности модулей, примерно одного размера, аналогично тому как дом строится из кирпичей.

Модульный подход позволяет разбить алгоритм любой сложности на совокупность алгоритмов, сложность каждого из которых значительно меньше. Отдельный модуль легче написать и отладить автономно. После автономной отладки модули собираются в единую программу.

В процессе декомпозиции (разбиения) основного алгоритма выявляются неоднократно повторяющиеся участки алгоритма. Выделение этих участков в отдельные модули позволяет многократно использовать однажды разработанный модуль.

Модульность подразумевает подразделение всей системы обработки информации на независимые функциональные элементы. Модуль при выполнении заданной функции образует определенные связи с другими модулями, входящими в программу. Эти связи характеризуют зависимость конкретного модуля от других частей программы.

Другой вид связей характеризует взаимосвязи различных элементов, составляющих модуль. Теснота этих связей характеризует функциональную однородность модуля.

Соотношение связей обоих видов определяет свойства модулей. Качество разрабатываемого модуля характеризуется двумя свойствами модулей:

1) связностью;
2) прочностью.

Программа обладает хорошей модульностью в том случае, если она характеризуется минимальной связностью модулей. Модули, входящие в программу, должны обладать максимальной прочностью.

Связность модульных программ характеризует независимость модулей друг от друга. Независимость модуля означает, что его можно модифицировать, не вызывая каких-либо последствий в других модулях.

Связность модулей определяется формой передачи данных. Передача данных в модуль может обеспечиваться либо через параметры, либо через глобальные переменные.

Параметры определяют состав и порядок расположения как входной для модуля информации, так и передаваемых из модуля значений. Все остальные переменные, используемые для обработки информации, объявляются локальными для разрабатываемого модуля. Таким образом, при использовании параметров обеспечивается минимальная связность модулей.

Передача данных через глобальные переменные предусматривает использование различными модулями одних и тех же переменных или массивов. Для задания глобальных переменных или массивов в программной среде Delphi (Pascal) используется раздел «Var» основного программного кода. При использовании глобальных переменных результат выполнения модуля в значительной степени зависит от того, каким образом меняются глобальные переменные в других модулях. Некорректное использование глобальной переменной в одном из модулей может вызвать ошибку в другом модуле, использующем ту же самую переменную. Таким образом, необоснованное использование глобальных переменных повышает связность модулей и снижает качество программы.

Под ***прочностью*** модуля понимается мера сцепления его элементов. Сильное сцепление предполагает, что исключение любого из элементов нарушает цепь преобразований и делает невозможным выполнение функции.

Модульный принцип реализован в программной среде Delphi (Pascal) на различных уровнях. В частности, предусматривается использование процедур и функций.

5.2. Использование процедур

Под ***процедурой*** понимается поименованная совокупность операторов, вычисляющих некоторое число результатов в зависимости от заданных аргументов.

Программный код процедуры определяется в разделе описаний. Описание процедуры имеет такую же структуру, что и основная программа, т.е. состоит из заголовка, описательной части и выполняемой части. Заголовок процедуры записывается в соответствии со следующим синтаксисом:

Procedure <имя процедуры> [(<список формальных параметров>)];

Конструкция <имя процедуры> определяет уникальный идентификатор процедуры, используемый для ее вызова.

Конструкция <список формальных параметров> задает переменные, которые формально определены как входные и выходные параметры. Входным параметром называется переменная, значения которой используются для работы процедуры. Значения входных параметров должны быть установлены до начала работы процедуры. Выходным параметром называется переменная, которая получает свое значение в результате работы процедуры.

Элементы списка параметров разделяются запятыми. При определении параметров задается их тип. В том случае если передается массив элементов, то используется тип массива, определенный в разделе «Type» основной программы.

Перед элементом списка может быть задан признак способа передачи параметра «Var». Признак «Var», показывает, что аргумент передается по ссылке, т.е. «By Reference». По

умолчанию данные передаются по значению, т.е. «By Value». Все выходные параметры должны передаваться по ссылке.

Пример, задания заголовка процедуры
Procedure (A: Ta1; N: Integer; Var S: Real);

Обращение к процедуре обеспечивается с помощью оператора процедуры, который записывается в соответствии со следующим синтаксисом:

<имя процедуры> [(<список фактических параметров>)];

Пример. Программный код, содержащий процедуру нахождения среднего арифметического элементов одномерного массива, имеет следующий вид:

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjSred;
{$APPTYPE CONSOLE}
Uses
  SysUtils;
Type
   TA1 = Array[1..40] Of Integer;
Var
  A:TA1;
  Sr:Real;
  I,N:Integer;
Procedure SredAr(A:TA1; N:Integer; Var Sr:Real);
Var
  I,S: Integer;
Begin
  S:=0;
  For I:= 1 To N Do
    S:=S + A[I];
  SR:= S/N;
End;
Begin
```

```
 Writeln('Vvedite razmernost');
 Readln(N);
 Writeln ('Massiv');
 For I:= 1 To N Do
   Read(A[I]);
 SredAr(A,N,Sr);
 Writeln('Sr=',Sr:6:2);
 Readln;
 Readln;
End.
```

При обработке экономической информации в качестве параметра может использоваться массив записей.

Пример. Имеется информация, содержащая данные об объеме производства по месяцам года. Информация представлена в виде массива записей. Каждый элемент массива содержит следующие два реквизита:

1) название месяца (реквизит представлен в символьном виде);
2) объем произведенной продукции (реквизит представлен в числовом виде).

Программный код обеспечивает выдачу на экран названий месяцев в порядке убывания объемов произведенной продукции.

Программный код включает процедуру, обеспечивающую сортировку массива по убыванию реквизита «объем произведенной продукции».

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjSortMetMB;
{$APPTYPE CONSOLE}
Uses
  SysUtils;
Type
 TProdM = Record
```

```
   Month: String [10];
   Prod: Real;
  End;
  TAProdm = Array[1..40] Of TProdm;
Var
  ProdM:TAProdm;
  I,N:Integer;
Procedure SortMetMB (Var Prodm:TAProdm; N:Integer);
Var I,J: Integer;
     W:TProdM;
Begin
  For I:= 1 To N-1 Do
    For J:= I+1 To N Do
    If Prodm[I].Prod < Prodm[J].Prod Then
    Begin
      W:=Prodm[I]; Prodm[I]:= Prodm[J];
      Prodm[J]:=W;
    End;
End;
Begin
 Writeln('Vvedite chislo mesjacev');
 Readln(N);
 For I:= 1 To N Do
 Begin
   Writeln ('Mesjac');
   Readln(Prodm[I].Month);
   Writeln ('Produkcija');
   Readln(Prodm[I].Prod);
 End;
 SortMetMB(ProdM,N);
 For I:= 1 To N Do
   Writeln(Prodm[I].Month);
 Readln;
End.
```

5.3. Использование функций

Под ***функцией*** понимается поименованная совокупность операторов, обеспечивающая вычисление единственного результата. Функция отличается от процедуры тем, что у нее нет выходных параметров. Входные параметры функции называются аргументами.

Заголовок функции записывается в соответствии со следующим синтаксисом:

Function <имя функции> [(<список формальных параметров>)]: <тип результата>;

Конструкция <имя функции> определяет уникальный идентификатор функции, используемый для ее вызова.

Конструкция <список формальных параметров> задает переменные, которые являются входными параметрами.

Конструкция <тип результата> определяет тип результатного значения функции.

Функция может включать любое количество выполняемых операторов. Однако в их состав обязательно должен включаться оператор, пересылающий результатное значение в точку вызова функции. Данный оператор записывается в соответствии со следующим форматом:

<имя функции> := <результатное значение>;

Вызов функции обеспечивается включением имени функции в состав выражения.

Пример. Программный код обеспечивает вычисление индекса массы тела и определение недостающего или избыточного веса.

Программный код включает две функции.

Функция «IndexMas» определяет индекс массы тела, который вычисляется как вес в килограммах, деленный на квадрат роста в метрах.

Функция «Wmaxmin» определяет максимально допустимый либо минимально допустимый вес, соответствующий заданному росту.

{ для PASCAL первые 4 строки заменяются на Uses Crt;}

```
Program PrjFuncMas;
{$APPTYPE CONSOLE}
Uses
  SysUtils;
Var
  Height,Weight,Wmax,Wmin, Wdif,IndexM:Real;
  FL: Boolean;
// Функция определения индекса массы тела
Function IndexMas(Height:Real; Weight:Real): Real;
Begin
  IndexMas := Weight / (Height * Height);
End;  {Function}
//  Функция определения максимально
// (минимально) допустимого веса
Function Wmaxmin(Height: Real; FL: Boolean): Real;
Begin
 If FL = True Then
   Wmaxmin := 24.9 * (Height * Height)
 Else
   Wmaxmin := 18.5 * (Height * Height);
End; {Function}
Begin
  Writeln ('Ukazite rost(metr)');
  Readln (Height);
  Writeln ('Ukazite ves');
  Readln (Weight);
  IndexM := IndexMas(Height, Weight);
  Writeln ('Index massi =', IndexM:6:2);
  If IndexM < 18.5 Then
  Begin
```

```
    FL:= False;
    Wmin:= Wmaxmin(Height, FL);
    Wdif:= Wmin - Weight;
    Writeln ('Nedostatok vesa=', Wdif:6:2)
  End;
  If (IndexM > 18.5) And (IndexM < 24.9) Then
    Writeln ('Normalniy ves');
  If IndexM > 24.9 Then
  Begin
     FL:= True;
     Wmax := Wmaxmin(Height, FL);
     Wdif := Weight - Wmax;
     Writeln ('Izbitochniy ves =', Wdif:6:2);
  End;
  Readln;
End.
```

При использовании функций для обработки экономической информации в качестве входных параметров могут использоваться массивы записей.

Пример. Имеется документ, который содержит информацию об остатках предметов. Документ содержит следующие реквизиты:

1) дата. Реквизит представлен в виде символьной строки длиной 6 символов (два символа определяют месяц, четыре символа определяют год);
2) код цеха. Реквизит представлен в виде символьной строки длиной 10 символов;
3) название предмета. Реквизит представлен в виде символьной строки длиной 20 символов;
4) цена предмета. Реквизит представлен в виде числа, имеющего дробную часть;
5) количество на конец периода. Реквизит представлен в виде числа, не имеющего дробной части;

6) сумма на конец периода. Реквизит представлен в виде числа, имеющего дробную часть.

Программный код включает функцию, которая обеспечивает определение статистической характеристики дисперсия для реквизита «Сумма на конец периода». Для расчета дисперсии используется следующая зависимость:

$$D = \frac{\Sigma(X - \overline{X})^2}{N},$$

где X – текущее значение реквизита «сумма на конец периода»;

$\overline{X}$ – среднее арифметическое значение реквизита «сумма на конец периода»;

N – количество строк документа.

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjFuncDisp;
{$APPTYPE CONSOLE}
Uses
  SysUtils;
Type
  TRest = Record
    Date:String[6];
    CodC:String[10];
    Name:String[20];
    Price:Real;
    Quant:Integer;
    Sum:Real;
  End;
  TARest=Array [1..40] Of TRest;
Var
  Rest:TARest;
  I,N:Integer;
  Disp:Real;
Function StatDisp(Rest:TArest; N:Integer): Real;
```

```
Var
  I:Integer;
  S, D, SR: Real;
Begin
  S:=0;
  For I:= 1 To N Do
    S:= S + Rest[I].Sum;
  SR:= S / N;
  D:= 0;
  For I:= 1 To N Do
    D:= D + (Rest[I].Sum - SR)*(Rest[I].Sum - SR);
  StatDisp:= D / N;
End; {Function}
Begin
    Writeln('Vvedite chislo zapisei');
 Readln(N);
 For I:= 1 To N Do
 Begin
   Writeln ('Data');
   Readln(Rest[I].Date);
   Writeln ('Kod cexa');
   Readln(Rest[I].CodC);
   Writeln ('Nazvanie predmeta');
   Readln(Rest[I].Name);
   Writeln ('Cena predmeta');
   Readln(Rest[I].Price);
   Writeln ('Kolichestvo');
   Readln(Rest[I].Quant);
   Writeln ('Summa');
   Readln(Rest[I].Sum);
 End;
 Disp:= StatDisp(Rest, N);
 Writeln('Dispersia =',Disp:7:2);
 Readln;
End.
```

5.4. Процедуры и функции без параметров

Как процедуры, так и функции могут не иметь входных параметров. Если параметры отсутствуют, то либо информация в процедуру не передается, либо передается через глобальные переменные.

При использовании глобальных переменных обрабатывается информация, заданная в разделе «Var», основной программы. В этом случае процедура или функция используются всегда для обработки одних и тех же массивов. Обратиться к процедуре для обработки других массивов нельзя.

Пример использования процедуры без параметров.

Представленная процедура обеспечивает выдачу справочной информации по кафедре «Математического обеспечения и технологий программирования».

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjProcSprav;
{$APPTYPE CONSOLE}
Uses
  SysUtils;
Procedure SpraV;
 Begin
  Writeln ('**************');
  Writeln ('* KAFEDRA    *');
  Writeln ('* MO & TP    *');
  Writeln ('* Komnata    *');
  Writeln ('*   348      *');
  Writeln ('* Telefon    *');
  Writeln ('* 442-80-98  *');
  Writeln ('* www.moitp.mesi.ru  *');
  Writeln ('**************');
  Readln;
End;
Begin
```

```
 Sprav;
End.
```

Пример использования функции без параметров.

Имеется документ, содержащий информацию о реализованной продукции. Документ содержит следующие реквизиты:

1) Шифр плательщика. Реквизит представлен в виде символьной строки длиной 10 символов;
2) Наименование изделия. Реквизит представлен в виде символьной строки длиной 20 символов;
3) Дата оплаты. Реквизит представлен в виде символьной строки длиной 8 символов (два символа определяют день, два символа определяют месяц, четыре символа определяют год);
4) Количество оплаченных изделий. Реквизит представлен в виде числа, не имеющего дробной части;
5) Цена изделия. Реквизит представлен в виде числа, имеющего дробную часть;
6) Сумма оплаты. Реквизит представлен в виде числа, имеющего дробную часть.

Программный код, включает функцию, которая обеспечивает тестирование введенного документа на корректность занесения значения суммы. Значение суммы сравнивается с произведением количества на цену (проверка на совпадение). Функция не имеет входных параметров.

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjFuncTest;
{$APPTYPE CONSOLE}
Uses
 SysUtils;
Type
 TRealiz = Record
```

```
    ShPlat:String[10];
    NIzd:String[20];
    Date:String[8];
    Quant:Integer;
    Price:Real;
    Sum:Real;
  End;
  TARealiz = Array [1..40] Of TRealiz;
Var
  Realiz:TARealiz;
  I,N:Integer;
Function Test:Boolean;
Var
  I:Integer;
Begin
  Test:=True;
  For I:= 1 To N Do
    If Realiz[I].Sum <> Realiz[I].Quant*Realiz[I].Price Then
      Test:=False;
End; {Function}
Begin
 Writeln('Vvedite chislo zapisei');
 Readln(N);
 For I:= 1 To N Do
 Begin
   Writeln ('Shifr platelshika');
   Readln(Realiz[I].SHPlat);
   Writeln ('Nazvanie izdelia');
   Readln(Realiz[I].Nizd);
   Writeln ('Data');
   Readln (Realiz[I].Date);
   Writeln ('Kolichestvo');
   Readln(Realiz[I].Quant);
   Writeln ('Cena izdelia');
   Readln(Realiz[I].Price);
```

```
  Writeln ('Symma oplati');
  Readln(Realiz[I].Sum);
 End;
 If Test = False Then Writeln ('Document ne veren')
          Else writeln ('Document veren');
 Readln;
End.
```

Процедура является более универсальной конструкцией, чем функция. Любая функция может быть преобразована в процедуру. Однако процедура может быть преобразована в функцию, исключительно в том случае, если имеет единственный результатный параметр.

5.5. Организация внешних модулей

Модуль (unit) представляет собой программную единицу, текст которой компилируется отдельно. Текст модуля пишется в соответствии со структурой, отличной от структуры программы на языке Pascal.

При использовании модулей программный код записывается в нескольких программных файлах. Один программный файл представляет собой программный модуль. Другие программные файлы представляют собой программный код, обеспечивающий обращение к процедурам, включенным в программный модуль.

Программный код модуля включает следующие части:

1) заголовок модуля, который записывается в соответствии со следующим синтаксисом:
   Unit <имя модуля>;
2) интерфейсная часть, которая начинается зарезервированным словом «Interface». В интерфейсной секции помещаются объявления типов данных, переменных, процедур, функций, которые должны быть доступны для программного кода, использующего данный модуль;

3) исполнительную часть, которая начинается зарезервированным словом «Implementation». В секцию «Implementation» включаются описания процедур и функций.

Для создания программного модуля в среде «Delphi» выбирается команда «New» из меню «File». В окне «NewItem» на странице «New» выбирается пиктограмма «Unit». При выполнении этой команды обеспечивается автоматический переход в текстовый редактор Delphi. В текстовом редакторе автоматически генерируются следующие конструкции:

1) заголовок модуля;
2) предложение «Interface», определяющее начало интерфейсной секции;
3) предложение «Implementation», определяющее начало исполнительной части;
4) предложение «End.», определяющее конец модуля.

После набора программного кода модуль сохраняется при помощи команды «Save As». При сохранении модуля выбирается тип сохраняемого файла «Delphi unit».

Программный код, обеспечивающий обращение к модулю, может быть организован как консольное предложение. В программный код консольного предложения добавляется конструкция, определяющая подключаемый модуль.

Например:

```
Uses
SysUtils,
UnitSort in 'Unitsort.pas';
```

Пример. Организация сортировки заданного фрагмента массива в виде модуля.

```
unit UnitSort;
Interface
 Type
  TArray=Array[1..40] Of Integer;
  Procedure SortMb(Var A:Tarray; N:Integer;
```

```
            N1:Integer;N2:Integer);
Implementation
 Procedure SortMb;
 Var
   I,J,W:Integer;
 Begin
  For I := N1 To N2-1 Do
    For J := I+1 To N2 Do
    If A[I]>A[J] Then
    Begin
      W:= A[I]; A[I]:=A[J]; A[J]:=W;
    End;
 End;
End.
```

Для вызова данной сортировки необходимо разработать программу, обеспечивающую вызов модуля. Программа имеет следующий вид:

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjSort;
{$APPTYPE CONSOLE}
uses
  SysUtils, UnitSort in 'Unitsort.pas';
Var
   A:TArray;
  I,N,N1,N2: Integer;
Begin
  Writeln ('Input N');
  Readln (N);
  Writeln ('Input Array');
  For I:=1 To N Do
     Read(A[I]);
  Writeln ('Input N1');    Readln (N1);
  Writeln ('Input N2');    Readln (N2);
    SortMB(A,N,N1,N2);
```

```
 For I :=1 To N Do
    Writeln(A[I]);
  Readln;
End.
```

Пример обработки документов с использованием внешнего программного модуля.

Имеется многострочный документ, содержащий учетную информацию об обработанном рабочем времени. Документ содержит следующие реквизиты:

1) наименование подразделения. Реквизит представлен в виде 30 символов;
2) фамилия, имя и отчество сотрудника. Реквизит представлен в виде 40 символов;
3) число отработанных часов. Реквизит представлен в виде пяти цифр;
4) число потерянных часов. Реквизит представлен в виде пяти цифр.

Требуется определить значение статистической характеристики «медиана» для реквизита «число отработанных часов». ***Медианой*** называется значение признака, приходящееся на середину упорядоченной совокупности. Если в совокупности четное число единиц, то медиана равна средней арифметической из двух средних значений вариантов.

Программный код организован в виде двух программных файлов.

Первый программный файл организован в виде модуля. Модуль содержит процедуру, обеспечивающую определения значения статистической характеристики медиана для совокупности значений, организованных в виде целочисленного массива.

Программный код первого программного файла имеет следующий вид:

```
Unit UnitMediana;
Interface
```

```
 Type
   TArray=Array[1..40] Of Integer;
   Procedure Mediana(A:Tarray; N:Integer;
          Var Rezult:Integer);
Implementation
Procedure Mediana;
Var
   I,J,W,N1:Integer;
Begin
  For I := 1 To N-1 Do
    For J := I+1 To N Do
    If A[I]>A[J] Then
    Begin
      W:= A[I]; A[I]:=A[J]; A[J]:=W;
    End;
 N1 := N div 2;
 If N Mod 2 = 0 Then
  Rezult := (A[N1] + A[N1 + 1]) div 2
 Else
  Rezult := A[N1 + 1];
End;
End.
```

Программный код второго программного файла обеспечивает ввод информации, обращение к процедуре нахождения медианы и выдачу результата.

```
{ для PASCAL первые 4 строки заменяются на Uses Crt;}

Program PrjMediana;
{$APPTYPE CONSOLE}
Uses
  SysUtils, UnitMediana in 'UnitMediana.pas';
Type
 TYchet = Record
  Podr: String [30];
```

```
  Fio: String [40];
  Otr: Integer;
  Pot: Integer;
 End;
  TArrayYchet = Array [1..100] Of TYchet;
Var
 Ychet: TArrayYchet;
 A:TArray;
 I,N,Med: Integer;
Begin
 Writeln('Vvedite chislo zapisei');
 Readln(N);
 For I:= 1 To N Do
 Begin
   Writeln ('Naimenovanie podrazdelenija');
   Readln(Ychet[I].Podr);
   Writeln ('Familija,Imia,Otchestvo sotrudnika');
   Readln(Ychet[I].Fio);
   Writeln ('Chislo otrabotannih chasov');
   Readln(Ychet[I].Otr);
   Writeln ('Chislo poterjanih chasov');
   Readln(Ychet[I].Pot);
 End;
 For I:= 1 To N Do
   A[I] := Ychet[I].Otr;
   Mediana(A, N, Med);
   Writeln ('Mediana ravna',Med);
   Readln;
End.
```

Контрольные вопросы

1. Каким образом организуются процедуры?
2. Каким образом обеспечивается вызов функции?

Задачи

Выполните разработку следующих программ на языке Pascal.

1. Имеется документ, содержащий информацию о выбытии работников по группе предприятий. Информация содержит следующие реквизиты:
   1) шифр предприятия – 20 символов;
   2) код причины выбытия – число, не имеющее дробной части;
   3) количество выбывших работников – число, не имеющее дробной части.

   Требуется: разработать программный код, включающий процедуру-подпрограмму, которая обеспечивает формирование нового массива, из записей, относящихся к заданному предприятию.

2. Имеется документ, содержащий информацию о выбытии работников по группе предприятий. Информация содержит следующие реквизиты:
   1) шифр предприятия – 20 символов;
   2) код причины выбытия – 4 цифры;
   3) количество выбывших работников – 4 цифры.

   Требуется: разработать программный код, включающий основной программный текст и следующие процедуры-подпрограммы:
   1) процедуру, которая обеспечивает определение общего количества выбывших работников по заданному предприятию;
   2) процедуру, обеспечивающую формирование нового массива, в котором информация будет упорядочена по кодам причины выбытия.

3. Имеется документ «Классификатор ценник», содержащий следующие реквизиты:
   1) наименование предмета – 40 символов;

2) единица измерения – 5 символов;
3) размер – 20 символов (строка может содержать цифровые символы, специальные символы и прочие виды символов);
4) цена – число, имеющее дробную часть.

Требуется: разработать программный код, включающий функцию. Функция должна обеспечить определение статистической характеристики «Вариационный размах» для реквизита «Цена». «Вариационный размах» представляет собой разность между наибольшими и наименьшими значениями.

Тесты

1. … позволяет разбить алгоритм любой сложности на совокупность алгоритмов, сложность каждого из которых значительно меньше.
   ☐ структурное программирование;
   ☐ словесная запись алгоритма;
   ☐ модульный подход;
   ☐ операционные системы.

2. Под … понимается поименованная совокупность операторов, вычисляющих некоторое число результатов в зависимости от заданных аргументов.
   ☐ переменной;
   ☐ базой данных;
   ☐ процедурой;
   ☐ операционной системой.

3. Перед элементом списка может быть задан признак способа передачи параметра:
   ☐ IF;
   ☐ WHILE;
   ☐ BEGIN;

☐ VAR;
☐ CASE.

4. Обращение к процедуре обеспечивается с помощью:
   ☐ оператора присваивания;
   ☐ условного оператора;
   ☐ оператора процедуры;
   ☐ оператора цикла.

5. Функция отличается от процедуры тем, что у нее нет:
   ☐ операторная часть;
   ☐ описательной части;
   ☐ переменных;
   ☐ массивов;
   ☐ выходных параметров;
   ☐ оператора присваивания;
   ☐ условного оператора.

6. Как процедуры, так и функции … не иметь входных параметров.
   ☐ могут;
   ☐ не могут.

Темы лабораторных (семестровых) работ

Темы 1, 2, 3 – Основные конструкции языка программирования

Тема 4 – Программная обработка структурных типов

Тема 5 – Модульное программирование

Глоссарий

Алгоритм – точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату.

Алгоритмизация – вычислительный процесс построения алгоритма решения задачи, результатом которого является выделение этапов процесса обработки данных, формальная запись содержания этих этапов и определение порядка их выполнения.

Алгоритмический язык – набор символов и система правил образования и истолкования конструкций из этих символов для задания алгоритмов.

База данных – множество наборов единообразных записей.

Динамическая структура – специальным образом организованные данные, располагаемые в динамической памяти и обрабатываемые с использованием указателей (ссылок).

Запись – структура данных, состоящая из фиксированного числа элементов.

Инкапсуляция – основополагающий принцип объектно-ориентированного программирования, который предусматри-

вает объединение данных и процедур обработки в единый тип, называемый объектом.

Информатизация общества – организованный социально-экономический и научно-технический процесс создания оптимальных условий для удовлетворения информационных потребностей органов управления и граждан на основе использования информационных ресурсов.

Информатика – совокупность разнообразных отраслей науки, техники и производства, связанных с переработкой информации.

Класс – образец, предназначенный для создания работающих объектов определенного типа.

Массив – упорядоченная совокупность однотипных элементов, расположенных в оперативной памяти.

Метод – функция или процедура, которая будет управлять работой объекта.

Множество – набор однотипных данных.

Модуль (unit) – представляет собой программную единицу, текст которой компилируется отдельно.

Наследование – основополагающий принцип объектно-ориентированного программирования, который определяет способность порожденного класса сохранять характеристики, присущие родительскому классу.

Полиморфизм – основополагающий принцип объектно-ориентированного программирования, под которым понимается присвоение действию единого имени при допустимости различных вариантов его реализации.

Прикладное приложение – программное приложение, обеспечивающее выполнение прикладной задачи. Как правило, представляет собой совокупность файлов различных типов, которые объединяются в единую программу.

Программное обеспечение – совокупность программ, предназначенных для реализации целей и задач, и документации на них.

Программные продукты – представляют собой программное обеспечение, изготовленное на продажу; они имеют программную документацию, обеспечивающую установку и эксплуатацию программ сторонними пользователями.

Процедура – поименованная совокупность операторов, вычисляющих некоторое

число результатов в зависимости от заданных аргументов.

Свойство – характеристика, с помощью которой описывается внешний вид и функционирование объекта.

Событие – действие, связанное с объектом. Событие может быть инициировано пользователем, вызвано программой или определено системой.

Технологии программирования – предназначены для повышения производительности труда при разработке и сопровождении программных изделий. Выделяют:
1) структурное программирование;
2) модульное программирование;
3) объектно-ориентированное программирование.

Файл – поименованная совокупность логически связанных между собой данных, хранящихся на внешнем носителе.

Функция – поименованная совокупность операторов, обеспечивающая вычисление единственного результата.

Язык программирования – изобразительные средства, предназначенные для непосредственной реализации алгоритма на вычислительной машине.

Список рекомендуемой литературы

Основная

1. Информатика: Учебник / Под ред. Н.В. Макаровой. – М.: Финансы и статистика, 2005.
2. Грибанов В.П., Калмыкова О.В., Сорока Р.И. Основы алгоритмизации и программирования. – М.: МЭСИ, 2005.

Дополнительная

1. Архангельский А.Я. Программирование в Delphi 6. – М.: Бином, 2002.
2. Иванова Г.С. Основы программирования. – М.: Изд-во МПУ, 2001.
3. Информатика: Практикум / Под ред. Н.В. Макаровой. – М.: Финансы и статистика, 2003.
4. Симонович С.В. и др. Информатика. Базовый курс. – СПб.: Питер, 2000.
5. Смирнов А.А. Основы программирования в среде Visual FoxPro. – М.: МЭСИ, 1999.

Интернет-ресурсы

1. WWW.PROGRAMMERS.RU
2. WWW.MICROSOFT.COM
3. WWW.DELPHIKINGDOM.COM
4. WWW.SUN.COM
5. WWW.CITFORUM.RU