Федеральное агентство по образованию

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.А. Иванков Г.К. Измайлов
В.Е. Клавдиев М.Е. Фролов

ИНФОРМАТИКА

КРАТКИЙ КУРС

Учебное пособие

Санкт-Петербург
Издательство Политехнического университета
2007

УДК 004(07)
ББК 32.81я7

Иванков А.А., Измайлов Г.К., Клавдиев В.Е., Фролов М.Е. **Информатика. Краткий курс**: Учебное пособие / Под общей ред. проф. В.В. Глухова. СПб.: Изд-во Политехн. ун-та, 2007. 63 с.

Содержание курса соответствует государственному образовательному стандарту дисциплины «Информатика» направления подготовки бакалавров 010500 «Прикладная математика и информатика» и других направлений подготовки бакалавров.

В кратком изложении рассмотрены базовые понятия, относящиеся к изучаемой дисциплине: информация, моделирование, состав и работа компьютерной системы, системное и прикладное программное обеспечение, базы данных, алгоритм и программирование, сети и защита информации.

Пособие предназначено для студентов СПбГПУ, других университетов, и может быть полезно преподавателям информатики при составлении авторских курсов лекций и упражнений.

Табл. 5 Ил. 4 Библиогр.: 3 назв.

Печатается по решению редакционно-издательского совета Санкт-Петербургского государственного политехнического университета.

ВВЕДЕНИЕ

Информатика входит в число предметов, которые формируют фундамент знаний и навыков технического специалиста. Инженерная деятельность предполагает широкое использование информационных процессов, поскольку без них немыслима практически ни одна современная технология.

Предлагаемое пособие написано авторами по итогам анализа результатов тестирования студентов СПбГПУ, которое явилось этапом подготовки к интернет-экзамену в сфере профессионального образования. Большинство разделов дополнено небольшим количеством вопросов, идентичных тем, из которых состоял сам тест. Читатель может использовать пособие при подготовке к сдаче зачетов и экзаменов по курсу информатики, для самообразования, и с целью обобщить имеющиеся у него знания в этой предметной области. Следует особо подчеркнуть, что изложенный материал не заменит полноценный лекционный курс, и уж тем более – курс лабораторных работ.

В ходе работы мы пользовались помощью и поддержкой целого ряда наших коллег по преподавательской деятельности на кафедре «Прикладная математика» СПбГПУ, а также студентов этой кафедры. Особую признательность авторы выражают Головину Олегу Андреевичу, Берёзину Сергею, Валынец Елене, Кожевникову Михаилу, Помелову Алексею, Рыбалкину Михаилу, Шолухе Павлу. Их помощь при подготовке этого издания невозможно переоценить.

ИНФОРМАЦИЯ И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ

Предметом науки информатики являются методы получения, хранения, обработки, представления и передачи информации. Существует несколько различных точек зрения относительно определения фундаментального понятия **информация**. Общепринятой является трактовка информации как отрицательной энтропии (негэнтропии) материальной системы или, другими словами, количественной меры неоднородности распределения вещества (энергии) материальной системы в пространстве и во времени.

Человек постоянно имеет дело с информационными процессами в ходе своей эволюции. Любая деятельность, любое взаимодействие индивида с окружающим миром требует информационного обеспечения (информационных процессов). Информация используется для устранения неопределенности принятия решения относительно любого действия, поступка.

Исторически и биологически процессы накопления информации человеком протекали посредством выработки рефлексов (впрочем, как и у всех живых организмов). Таким образом, информация накапливалась нервной системой. На качественно новый уровень накопления и использования информации перешли с появлением второй сигнальной системы − речи. С этого момента информацию могли хранить и, что самое главное, обрабатывать не только отдельные индивиды, но и социум в целом. Следующий этап в эволюции человека и используемых им информационных процессов − появление письменности, т.е. знаковой системы (пиктограммы, иероглифы, клинопись, линейное письмо А, линейное письмо Б). Информация начала накапливаться, обрабатываться в форме записей на определенных материальных носителях: глиняных табличках, папирусе и т.п. И, тем самым, как любой материальный объект, стала характеризоваться формой и содержанием. Форма − это способ хранения, т.е. используемая система знаков и синтаксис. Содержание, или **семантический аспект** хранимой информации − это смысл записи, отражающий ссылкой на что, на какой материальный объект, процесс, служит запись в той или иной форме. Что касается возможности использовать хранимую информацию, **прагматический аспект**, то она тесно связана с такой характеристикой информации как **полезность.**

В конечном итоге произошла увязка второй сигнальной системы и письменности, т.е. знаки стали служить для обозначения различаемых в речи фонем, а композиции знаков соответствовать семантически нагруженным, осмысленным морфемам – словам устной речи, которыми

именовали конкретные материальные объекты, процессы. Так у славян появилась кириллица, у западноевропейцев – их национальные алфавиты на основе латиницы, у японцев – катакана, хирагана.

В качестве примера информационного процесса рассмотрим хрестоматийную задачу – решение квадратного уравнения. Такого вида уравнения отражают фундаментальные законы в целом ряде предметных областей. В микроэкономике, в частности, это математическая модель (представленное на языке математики описание реально протекающих процессов или свойств материальных систем и связей между ними), характеризующая зависимость дохода от объема выпускаемой предприятием продукции. Прежде всего, необходимо получить исходные данные, значения коэффициентов A, B, C. Отсутствие хотя бы одного из них приведет к тому, что решение окажется невозможно получить – исходная информация неполна. Располагая полной исходной информацией, порождаем информационный процесс – получение новой, искомой информации. На первом этапе требуется определить множество, которому принадлежит искомое решение: поле комплексных чисел или поле вещественных чисел. Чтобы устранить возникшую неопределенность (**энтропию процесса**) относительно выбора множества возможных решений требуется дополнительная информация! Энтропия устраняется информацией! Эту дополнительную информацию мы можем получить либо извне, либо самостоятельно на основе исходных данных (тех, которыми уже располагаем). Из школьного курса математики хорошо известно, что дополнительная информация, требуемая для устранения неопределенности с выбором множества возможных решений, легко может быть получена на основе исходных данных (путем вычисления дискриминанта). На следующем этапе получаем решение и затем организуем его вывод пользователю.

Вернемся к предмету науки информатики. Само название, которое мы позаимствовали с французского (l'Information + l'Automatique), указывает, что собственно исследования начались с появлением ЭВМ – чрезвычайно мощного инструмента для реализации процессов получения, хранения, обработки и передачи информации. Сохраняя преемственность с теми формами информационных процессов, которые имели место еще в докомпьютерную эпоху, а также, принимая во внимание физические особенности электронных носителей информации, были приняты изложенные далее стандарты. Поскольку атомарные элементы электронных носителей информации допускали лишь два возможных состояния, то элементарную порцию информации можно было кодировать с помощью элементов множества, состоящего из двух символов (знаков). Традиционно для этих целей использовали 0 и 1 (отсюда **бит** – BInary

uniT), т.е. вели записи в двоичной системе счисления. Задачу переноса информации с традиционных носителей (бумаги) на электронные одновременно решали несколько независимых групп специалистов в нескольких странах. В конечном счете, решения, предложенные ими, свелись к использованию последовательностей из 7, 8 или 16 бит для кодирования отдельных символов традиционных алфавитов. Исторически сложилось так, что расширенный ASCII-код (кодировка 8-разрядными двоичными числами − **байтами**), это наиболее распространенная форма. Сегодня, в связи с бурным развитием сетевых технологий и возникшей необходимостью обмена информацией на разных языках мира, активно используется т.н. UNICODE (кодировка с помощью 16-разрядных двоичных чисел) и ее модификации.

Для сбора, хранения, обработки и передачи информации нетекстового характера используется очень большое количество различных систем кодирования. Например, графические изображения кодируют одним из двух способов:

- разбивают их на множества неделимых далее элементов (растр точек или растровое изображение);
- описывают его с помощью семейства отрезков, кривых и других графических примитивов, получая векторное изображение.

Обычно отдельный документ хранится на отдельном листе (в отдельной папке), отдельное изображение на отдельном холсте. По аналогии с обычным способом хранения текстовых документов, изображений, звукозаписей, при переносе этой информации на электронные носители, коды, например, символов текста документа, упорядочивают в соответствии с порядком их следования в оригинале (документе) и записывают на носитель в форме логической единицы информации − набора данных или **файла**. Множество файлов, которые хранят в отдельно взятой информационно-вычислительной системе, упорядочивают с помощью иерархической системы каталогов (аналог книжного шкафа, имеющего систему вложенных ящиков, каждый из которых в качестве вложения может иметь ящики меньшего размера и файлы).

Вопросы

1. Возможна ли ситуация, когда хранимая на электронных носителях информация (набор данных) доступна пользователю, но он не может этой информацией воспользоваться, так как она ему непонятна?
Комментарий. ► Да, если на этапе представления информации она была неверно интерпретирована, скажем, при интерпретации документа

ошибочно была использована кодовая таблица CP1251, а на самом деле текст в файле хранили в кодировке KOI-8. Для решения возникшей проблемы нужно всего лишь указать программе-интерпретатору верную кодировку.

2. Прагматический аспект информации
1) определяет значение символа естественного алфавита;
2) определяет отношение между единицами информации;
3) дает возможность раскрыть её содержание и показать отношение между смысловыми значениями её элементов;
4) рассматривает информацию с точки зрения её практической полезности для получателя.
Комментарий. ► Правильный ответ: рассматривает информацию с точки зрения ее практической полезности для получателя.

3. Свойство информации, заключающееся в достаточности данных для принятия решений, есть...
1) объективность; 2) содержательность; 3) полнота; 4) достоверность.
Комментарий. ► Правильный ответ: полнота.

СИСТЕМЫ СЧИСЛЕНИЯ

Системой счисления называется совокупность символов (цифр) и правил их использования для представления чисел. Существуют **позиционные** и **непозиционные** системы счисления. В непозиционных системах вес цифры (т. е. тот вклад, который она вносит в значение числа) не зависит от ее позиции в записи числа. Так, в римской системе счисления в числе XXXII (тридцать два) вес цифры X в любой позиции равен десяти. В позиционных системах счисления вес каждой цифры изменяется в зависимости от ее положения (позиции) в последовательности цифр, изображающих число. Например, в десятичном числе 757,7 первая семерка означает 7 сотен, вторая – 7 единиц, а третья – 7 десятых долей единицы. Позиция цифры называется разрядом. Разряд числа возрастает справа налево. **Основание** позиционной системы счисления равно количеству цифр, используемых ею, и определяет, во сколько раз различаются значения цифр соседних разрядов числа. Любое число, записанное в позиционной системе счисления с произвольным основанием, можно записать в виде полинома (многочлена):

$$A_{(S)} = a_n S^n + a_{n-1} S^{n-1} + \ldots + a_1 S^1 + a_0 S^0 + a_{-1} S^{-1} + \ldots + a_m S^m,$$

где S – основание системы счисления, а степень соответствует разряду цифры a в числе $A_{(S)}$. Например, число 757,7 в десятичной системе счисления запишется в виде

$$757{,}7_{10} = 7\cdot10^2+5\cdot10^1+7\cdot10^0+7\cdot10^{-1}.$$

Для компьютеров оказалась весьма удобной двоичная система счисления с двумя цифрами: 0 и 1, так как ее реализация основана на технических устройствах с двумя устойчивыми состояниями (есть ток – нет тока, намагничен – не намагничен и т.п.). Перевод чисел из десятичной системы в двоичную, и наоборот, выполняет машина. Однако профессиональное использование компьютера требует понимания слов машины.

В восьмеричной системе восемь цифр: 0, 1, 2, 3, 4, 5, 6, 7, число восемь обозначается 10 (единица второго разряда), 64_{10} – не что иное, как 100_8.

В шестнадцатеричной системе для чисел до девяти используются цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, а для следующих чисел – от десяти до пятнадцати – в качестве цифр используются символы *A, B, C, D, E, F*. Так как при десятичном сложении пяти и шести получается одиннадцать, то в шестнадцатеричной системе это цифра *B*.

В каждой системе счисления цифры упорядочены в соответствии с их значениями: 1 больше 0, 2 больше 1 и т. д. **Продвижением** цифры называют замену ее следующей по величине. Продвинуть цифру 1 – это значит заменить ее на 2, продвинуть цифру 2 – значит заменить ее на 3 и т.д. Продвижение старшей цифры (например, цифры 9 в десятичной системе) означает замену ее на ноль. В двоичной системе, использующей только две цифры, продвижение 0 означает замену его на 1, а продвижение 1 – замену ее на ноль. Целые числа в любой системе счисления порождаются с помощью **правила счета**: для образования целого числа, следующего за любым данным целым числом, нужно продвинуть самую правую цифру числа; если какая-либо цифра после продвижения стала нулем, то нужно продвинуть цифру, стоящую слева от нее. В соответствии с этим правилом первые десять целых чисел в двоичной системе будут: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001.

Чтобы перевести число из произвольной системы в десятичную, его следует записать в виде приведенного выше полинома и вычислить его значение. Например, переведем двоичное число 111101_2 в десятичную систему счисления:

$$111101_2 = 1\cdot2^5+1\cdot2^4+1\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0 = 32+16+8+4+1 = 61_{10}.$$

Для перевода целого десятичного числа в систему счисления с основанием S надо последовательно делить его на это основание и записывать остатки, не забывая нулевые. Выписав все остатки, начиная с последнего, получим

представление числа в новой системе счисления. Этот алгоритм основан на том, что остаток от целочисленного деления дает количество единиц в соответствующем разряде. Так, остаток от первого деления дает число единиц в новом представлении числа, остаток от второго – число десятков и т. д. Для примера переведем число 13 в двоичную форму: 13:2=6, остаток 1 (количество единиц); 6:2=3, остаток 0; 3:2=1, остаток 1; результат последнего деления (а это 1) на 2 уже не делится, и эта цифра будет старшей цифрой нашего числа. Начиная с нее, записываем в обратном порядке все полученные остатки и получаем двоичное представление числа: $13_{10}= 1101_2$.

Сложение чисел в любой позиционной системе производится по тем же правилам, что и сложение десятичных чисел, то есть вычисляется сумма значений одноименных разрядов и перенос в следующий разряд при превышении значения основания системы. Так, сложение двоичных чисел производится по следующим правилам: 0+0=0, 0+1=1, 1+0=1, 1+1=10 (перенос единицы в следующий старший разряд).

Вычитание чисел производится также поразрядно, при этом необходимо помнить, что, выполняя вычитание большего значения, следует занять единицу из старшего значащего разряда. Вычитание двоичных чисел производится по следующим правилам: 0–0=0, 1–0=1, 1–1=0, 10–1=1 (т. е. вычитая из нуля единицу, занимаем единицу из старшего значащего разряда).

Вопросы

1. Чему равна разность двоичных чисел 10101_2 и 1000_2 в двоичной системе счисления?

Комментарий. ► Вычитаемое число состоит из четырех разрядов, а в четвертом разряде уменьшаемого числа стоит 0, поэтому занимаем единицу из следующего разряда и получаем в разности на этом месте 1. Остальные цифры просто переносятся в разность из уменьшаемого числа без изменения, так как в этих разрядах вычитаемого числа стоят нули. В результате получаем число 1101_2.

2 Чему равна разность двоичных чисел 1001_2 и 101_2 в десятичной системе счисления?

Комментарий. ► В двоичной системе разность равна 100_2, т.е. 4 – в десятичной.

3. Чему равна сумма двоичных чисел 11001_2 и 1010_2 в двоичной системе счисления?

Комментарий. ► Вычисляем суммы значений одноименных разрядов, начиная справа налево, и в четвертом разряде получаем значение 2 следовательно требуется осуществить перенос из четвертого разряда в пятый и далее в шестой. В итоге 100011_2.

4. Чему равна последняя цифра суммы чисел 55_{16} *и* 56_{16} *в шестнадцатеричной системе счисления?*
Комментарий. ► В шестнадцатеричной системе счисления цифры после девятки обозначаются латинскими буквами *A, B, C, D, E, F*. Так как при десятичном сложении пяти и шести получается одиннадцать, то в шестнадцатеричной системе это цифра *B*.

5. Какая из приведенных последовательностей значений:
1) $55_{16}\ 55_8\ 55_7$; 2) $55_7\ 55_8\ 55_{16}$; 3) $55_8\ 55_{16}\ 55_7$; 4) $55_8\ 55_7\ 55_{16}$
является упорядоченной по возрастанию?
Комментарий. ► Число единиц во всех приведенных системах счисления одинаково, тогда как вес одного “десятка” тем больше, чем больше основание системы. Поскольку число десятков во всех числах одинаковое, то по возрастанию упорядочена последовательность $55_{16}\ 55_8\ 55_7$.

ЛОГИЧЕСКИЕ ВЫСКАЗЫВАНИЯ И ОПЕРАЦИИ НАД НИМИ

В данном параграфе рассматривается такой важный раздел математической логики как **исчисление высказываний**. Синонимами высказывания являются утверждение или предложение. Основные логические операции определяются на простых высказываниях, которые принимают только два значения 1 или 0, т.е. **«истина»** (англ. **TRUE**) или **«ложь»** (англ. **FALSE**). Примером **простого истинного** высказывания служит высказывание «Дуб – дерево», а примером **простого ложного** – «Кирпич – дерево». Простые высказывания также называют **атомарными**.

Существуют 3 базовые операции связывания простых высказываний в составные сложные – это следующие операции:

- **конъюнкция** (логическое И, AND, &, $\wedge$);
- **дизъюнкция** (логическое ИЛИ, OR, ||, $\vee$);
- **инверсия** или **отрицание** (логическое НЕ, NOT, $\neg$, $\sim$, $\bar{}$).

Логическая операция инверсии применяется к одному высказыванию, отрицая его. Если исходное высказывание было истинно, то результат

будет ложным и, наоборот. Две другие операции определены для двух и более высказываний. **Конъюнкция** нескольких высказываний истинна тогда и только тогда, когда истинны все высказывания. Если среди них есть хоть одно ложное, то конъюнкция ложна. **Дизъюнкция** нескольких высказываний, наоборот, ложна тогда и только тогда, когда все высказывания ложны, а если среди них есть хоть одно истинное, то результат тоже будет истинным. Примером **составного высказывания** служит высказывание «Доцент тут и аппаратура при нем». Оно состоит из двух высказываний «Доцент тут» и «Аппаратура при нем», объединенных операцией конъюнкции. Результирующее составное высказывание является истинным тогда и только тогда, когда оба простых высказывания истинны.

Для упрощения рассмотрения различных ситуаций все случаи сведены в так называемую **таблицу истинности,** являющуюся базовой при проведении логических вычислений.

Утверждение A	Утверждение B	Отрицание не A	Конъюнкция A и B	Дизъюнкция A или B
0	0	1	0	0
0	1	-	0	1
1	0	0	0	1
1	1	-	1	1

Хотя любое логическое выражение может быть представлено в виде простых выражений, связанных этими тремя базовыми операциями, для большей наглядности используют еще две операции. Эти логические связки носят названия

- **импликация** (→, THEN), и
- **эквивалентность** (≡, ↔, EQV).

Импликация связывает два утверждения отношением логического следования:

ЕСЛИ A, ТО B.

Импликация истинна всегда, кроме того случая, когда утверждение A истинно, а утверждение B – ложно. Примером служит высказывание «Если пойдет дождь, мы не поедем на дачу». Оно ложно только в том случае, если дождь пошел, а на дачу мы все-таки поехали. Эквивалентность выражает такую связь двух утверждений, при которой они либо одновременно истинны, либо одновременно ложны:

A ТОГДА И ТОЛЬКО ТОГДА, КОГДА B.

Таблица истинности для этих операций имеет вид

Утверждение A	Утверждение B	Импликация $A \rightarrow B$	Эквивалентность $A \equiv B$
0	0	1	1
0	1	1	0
1	0	0	0
1	1	1	1

Заметим в заключение, что существует определенная последовательность выполнения операций при преобразовании сложных высказываний (если она не определена явно наличием скобок). Эти операции выполняются в соответствии с **приоритетом**. **Наивысший приоритет** имеет **инверсия**, а общий порядок следующий:

инверсия, конъюнкция, дизъюнкция, импликация и эквивалентность.

Для решения приведенных ниже контрольных заданий и других заданий такого типа необходимым является знание логических операции, их таблиц истинности и приоритетов выполнения.

Вопросы

1. Какое логическое выражение соответствует высказыванию «A совпадает с B и не является min(A, B, C)»

1) (A = B) или (C < A);
2) ((A < B) или (A > C)) и ((A = C) или (A > B));
3) (A = B) и (C < A).

Комментарий. ► Приведенное высказывание на самом деле эквивалентно следующему: A=B и это число не является наименьшим из тройки A, B, C. Поэтому наименьшим является число C, т.е. C < A. Следовательно, правильный ответ: (A = B) и (C < A). Условия в первом варианте ответа не подходят, т.к. фактически они не ограничивают значение C, поскольку, если условие A=B выполнено, то логическое выражение истинно вне зависимости от выполнения второго условия. Во втором варианте ответа необходимо учитывать то, что при наличии скобок логические операции выполняются не в соответствии с приоритетом, а согласно расстановке скобок. Условия в этом варианте также не подходят, т.к. это логическое выражение истинно, например, в том случае, когда A >C и A>B.

2. Вычислить таблицу истинности выражения не A или B. Какой операции она соответствует?

Комментарий. ► Правильный ответ: импликация. Выражение не А или В есть на самом деле импликация А → В, записанная при помощи базовых операций. Таблица истинности этого выражения соответствует этой операции (см. выше).

3. Как упрощается выражение В или не В и А?
Комментарий. ► Выражение В или не В тождественно истинно, поэтому значение логического выражения определяется только значением А. Следовательно, В или не В и А эквивалентно А.

СОСТАВ И РАБОТА КОМПЬЮТЕРНОЙ СИСТЕМЫ

Компьютер представляет собой программируемое электронное устройство для обработки данных. Существуют два основных класса компьютеров: **цифровые**, обрабатывающие данные в виде числовых двоичных кодов, и **аналоговые**, обрабатывающие непрерывно меняющиеся физические величины. В любом компьютере выделяют следующие главные устройства: **память** (запоминающее устройство – ЗУ), состоящую из перенумерованных ячеек; **процессор**, включающий **устройство управления** (УУ) и **арифметико-логическое устройство** (АЛУ); **устройство ввода; устройство вывода**. Эти устройства соединены **каналами связи**, по которым передается информация. В составе процессора имеется ряд специализированных ячеек памяти, называемых **регистрами**. Регистр выполняет функцию кратковременного хранения числа или команды. Некоторые регистры имеют свои названия: **сумматор, счетчик команд, регистр команд.**

В основу построения подавляющего большинства компьютеров положены принципы, сформулированные американским ученым Джоном фон Нейманом.

- **Принцип программного управления**. Процессор выполняет программу автоматически, без вмешательства человека. Программа состоит из набора команд, выполняемых в определенной последовательности, пока не встретится команда «стоп».
- **Принцип однородности памяти.** Программы и данные хранятся в одной и той же памяти, поэтому компьютер не различает, что хранится в данной ячейке памяти – число, текст или команда.

- **Принцип адресности.** Структурно основная память состоит из перенумерованных ячеек. Процессору в произвольный момент времени доступна любая ячейка.

Компьютеры, построенные на этих принципах, называются **фон-неймановские**.

Архитектурой компьютера называется его описание на некотором общем уровне, включающее описание системы команд, системы адресации, организации памяти и т. д.

Структура компьютера – это совокупность его функциональных элементов и связей между ними. Элементами выступают самые различные устройства – от основных логических узлов компьютера до простейших схем.

Классическая архитектура (архитектура фон Неймана) – одно АЛУ, через которое проходит поток данных, и одно УУ, через которое проходит поток команд – программа. Это **однопроцессорный компьютер**. К этому типу архитектуры относится и архитектура персонального компьютера с **общей шиной**, называемой также **системной магистралью**. Физически **магистраль** представляет собой многопроводную линию с гнездами для подключения электронных схем. Совокупность проводов магистрали разделяется на отдельные группы: шину адреса, шину данных и шину управления. Периферийные устройства подключаются к компьютеру через специальные **контроллеры** – устройства управления периферийными устройствами. Контроллеры освобождают процессор от непосредственного управления функционированием данного оборудования.

Основной рабочий компонент компьютера – **центральный процессор** (CPU). Он выполняет арифметические и логические операции, задаваемые программой, управляет вычислительным процессом и координирует работу всех устройств компьютера. В общем случае центральный процессор содержит: АЛУ, шины данных и шины адресов, регистры, счетчики команд, кэш – очень быструю память малого объема, математический сопроцессор. Современные процессоры выполняются в виде **микропроцессора.** Физически микропроцессор представляет собой **интегральную схему** – тонкую пластинку кристаллического кремния прямоугольной формы площадью несколько квадратных миллиметров. На этой пластинке размещены схемы, реализующие все функции процессора.

Память компьютера построена из двоичных запоминающих элементов – **бит**, объединенных в группы по 8 бит, которые называются **байтами**. Все байты пронумерованы. Номер байта называется его **адресом**. Байты могут объединяться в ячейки, которые называются также **словами**. Для каждого компьютера характерна определенная длина слова –

чаще всего четыре байта. Как правило, в одном машинном слове может быть представлено либо одно целое число, либо одна команда. Широко используются и более крупные единицы объема памяти: Килобайт, Мегабайт, Гигабайт, Терабайт, Петабайт.

Современные компьютеры имеют много разнообразных запоминающих устройств. Различают два вида памяти – **внутреннюю** и **внешнюю**.

В состав внутренней памяти входят: оперативная память, кэш-память, специальная память.

Оперативная память (ОЗУ, англ. RAM) – это быстрое запоминающее устройство не очень большого объема, которое непосредственно связано с процессором. ОЗУ используется только для временного хранения данных и программ, т.к. когда машина выключается, все, что находилось в ОЗУ, пропадает. Доступ к оперативной памяти прямой – это означает, что каждый байт памяти имеет свой индивидуальный адрес.

Кэш-память или **сверхоперативная память** – очень быстрое запоминающее устройство небольшого объема, которое используется при обмене данными между микропроцессором и ОЗУ для компенсации разницы в скорости обработки информации между ними.

Специальная память. К ней относятся постоянная память (ПЗУ, англ. ROM), перепрограммируемая постоянная память (Flash Memory), память CMOS RAM – разновидность ПЗУ, видеопамять и ряд других.

Постоянная память – энергонезависимая память, используемая для хранения данных, которые никогда не потребуют изменения. Содержимое памяти специальным образом «зашивается» в устройстве при его изготовлении для постоянного хранения. Из ПЗУ можно только читать.

Перепрограммируемая постоянная память – энергонезависимая память, допускающая многократную перезапись своего содержимого с дискеты.

CMOS RAM – это память с невысоким быстродействием и минимальным энергопотреблением от батарейки. Используется для хранения информации о конфигурации и составе оборудования компьютера. Содержимое CMOS изменяется специальной программой установки Setup.

Внешняя память предназначена для длительного хранения программ и данных, и целостность ее содержимого не зависит от того, включен или выключен компьютер. В отличие от оперативной памяти внешняя память не имеет прямой связи с процессором. В состав внешней памяти входят различные накопители – на **жестких** дисках (англ. **HDD**), на **гибких** дисках (**дискеты**), на **компакт**-дисках (**CD-ROM**), на

магнитооптических компакт-дисках, на **магнитной ленте** (**стримеры**) и др.

Видеосистема компьютера состоит из трех компонент: монитора (называемого также дисплеем), видеоадаптера, и программного обеспечения (драйверов видеосистемы). **Монитор** – устройство визуального отображения информации. Мониторы бывают на базе **электронно-лучевой трубки**, **жидкокристаллические** и **сенсорные**, когда общение с компьютером осуществляется путем прикосновения пальца к определенному месту чувствительного экрана.

К компьютеру подсоединяются **периферийные** (внешние) устройства: **принтер** – печатающее устройство; **плоттер** – графопостроитель; **сканер** – устройство для ввода в компьютер графических изображений; **модем** – устройство для передачи компьютерных данных на большие расстояния по телефонным линиям связи, различные **манипуляторы – мышь, джойстик, трекбол, дигитайзер**. Для соединения друг с другом различных устройств они должны иметь одинаковый **интерфейс.**

Интерфейс – это средство сопряжения двух устройств, в котором все физические и логические параметры согласуются между собой. Каждый из функциональных элементов связан с шиной определенного типа – адресной, управляющей или шиной данных. Для согласования интерфейсов периферийные устройства подключаются к шине не напрямую, а через свои **контроллеры** (**адаптеры**) и **порты.** Контроллеры осуществляют непосредственное управление периферийными устройствами по запросам микропроцессора. Порты устройств представляют собой некие электронные схемы, содержащие один или несколько регистров ввода-вывода и позволяющие подключаться к внешним шинам микропроцессора.

Вопросы

1. Является ли дисковая память памятью прямого доступа?
Комментарий. ► Информация на дисках записывается по **дорожкам**, которые делятся на **секторы**. Для считывания данных необходимо, чтобы диск вращался, поэтому дисковая память является памятью последовательного доступа.

2. Для чего в первую очередь предназначен джойстик?
Комментарий. ► Джойстик – это манипулятор в игровых приставках.

3. Что обеспечивают COM – порты компьютера?

Комментарий. ► COM – порты компьютера обеспечивают синхронную и асинхронную передачу данных.

4. По какому типу классифицируются принтеры?
Комментарий. ► Принтеры классифицируются в соответствии с механизмом выполнения печати: **матричные, лазерные** и **струйные.** Матричные используют комбинации маленьких штырьков, которые бьют по красящей ленте. Лазерные принтеры работают подобно копировальным аппаратам. Струйные принтеры генерируют символы в виде последовательности чернильных точек, которые через крошечные сопла выбрызгиваются на бумагу.

5. Являются ли регистры устройствами памяти?
Комментарий. ► Регистры необходимы для временного хранения данных или команд. Основным элементом регистра является электронная схема, называемая **триггером**, которая способна хранить одну двоичную цифру (разряд). Регистр представляет собой совокупность триггеров, связанных друг с другом общей системой управления.

6. Какие виды памяти используется для временного хранения данных?
Комментарий. ► Для временного хранения данных используется оперативная память и связанная с ней кэш-память. При выключении компьютера данные в них пропадают.

7. Для чего используется сканер?
Комментарий. ► Если принтеры выводят информацию из компьютера, то сканеры, наоборот, переносят информацию с бумажных документов в компьютер, который воспринимает ее как картинку, даже если это текст. Чтобы преобразовать такой графический текст в обычный символьный формат, требуются специальные программы оптического распознавания образов.

8. Для чего предназначены драйверы?
Комментарий. ► Драйверы – это специальные программы для согласования работы внешних и внутренних устройств компьютера. Так, например драйвер принтера способен переводить стандартные команды печати компьютера в специальные команды принтера.

9. Что такое компакт-диск (CD)?
Комментарий. ► Компакт-диск – это оптический диск, информация с которого считывается лазерным лучом. В отличие от магнитных дисков,

компакт-диски имеют не множество кольцевых дорожек, а одну – спиральную.

10. Для чего предназначен процессор?
Комментарий. ► Процессор предназначен для обработки всех видов информации. Он выполняет универсальные инструкции, называемые машинными командами.

11. Как называется устройство, обеспечивающее сохранение формы и амплитуды сигнала при передаче его на большее, чем предусмотрено данным типом физической передающей среды расстояние?
Комментарий. ► Шлюзом.

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Системное программное обеспечение (**СПО**) – это обязательный компонент каждой работоспособной информационно-вычислительной системы (**ИВС**). Без СПО компьютер превращается просто в обычную груду железа. Задачи, которые решаются с помощью СПО, можно условно разделить на две группы:

- управление аппаратным обеспечением;
- обеспечение эффективного взаимодействия пользователя с ИВС.

Какие задачи попадают в первую группу? Вспомним, что до сих пор ни одному пользователю никогда не приходилось вертеть пальцем электронные носители информации в приводах, будь-то дискета, компакт-диск или носители в винчестере (жесткий диск), позиционируя их для того, чтобы прочесть хранящуюся там информацию или записать новую. Работы по управлению аппаратным обеспечением выполняют компоненты СПО. В первую группу задач попадают также работы по тестированию аппаратного обеспечения и устранению ряда технических неисправностей компьютера (пользователю должно быть известно, что сбойные участки на магнитных поверхностях носителей в жестких дисках и дискетах можно устранить, «изъять из обращения», выполнив ряд операций с помощью соответствующего ПО). Программные средства для тестирования и устранения подобных проблем традиционно называют **утилитами** (англ. utility), что можно трактовать по-русски как «полезные».

Вторая группа задач – это обеспечение одной из стандартных форм взаимодействия пользователя с компьютером. Под взаимодействием

подразумевается, что пользователю необходимо каким-то образом, в какой-то форме, передать ИВС на исполнение команды и получить от нее результаты по итогам исполнения этих команд. Стандартами *де-факто* стали три следующих варианта реализации взаимодействия пользователь-компьютер:

- **режим командной строки;**
- **режим меню**;
- **графический объектно-ориентированный интерфейс**.

В первом случае взаимодействие осуществляется посредством команд. Пользователь должен знать команды, необходимые для выполнения работы, и указать их компьютеру с помощью такого устройства ввода как клавиатура. Вводимые пользователем команды отображаются на экране монитора, чтобы вовремя обнаружить и устранить допущенные в них ошибки. Результаты исполнения команд ИВС также отображает на экране монитора (как правило, в виде текста).

Во втором случае ИВС отображает на экране монитора для пользователя меню, т.е. иерархически упорядоченные списки команд, и ему остается только выбрать необходимую команду. Выбор осуществляется либо с помощью клавиатуры, либо с помощью мыши или трекбола. По итогам исполнения команды ИВС отображает результаты и снова возвращает пользователя к меню.

В третьем случае ИВС организует общение посредством графических, интуитивно понятных пользователю образов. Пользователь выполняет с помощью устройств ввода (клавиатуры, мыши, трекбола и т.п.) некоторые типовые операции над этими образами. ИВС, интерпретируя эти операции, выполняет предписанную ему работу и предоставляет на экране монитора результаты в той или иной форме.

Состав СПО достаточно разнообразен. Это связано с существенными различиями т.н. компьютерных платформ. Однако на сегодняшний день большинство разработчиков СПО ориентировано на машины клона IBM PC-совместимых. Даже консервативная Apple Macintosh создала версию своей MacOS для работы на PC с Intel-процессорами.

Ядро СПО – **операционная система** (ОС). Это главная, необходимая компонента СПО. Она загружается в ОЗУ при включении компьютера и управляет им с этого момента и до выключения машины, выполняя базовый, стандартный набор задач из обеих вышеуказанных групп. Следуя требованиям структурного программирования, ОС реализованы по модульному принципу. Скажем, в DR-OpenDOS (Caldera DOS) аппаратным обеспечением управляет IBMBIO.COM, в MS-DOS – Io.sys, а взаимодействие с пользователем реализовано с помощью **командных интерпретаторов** COMMAND.COM. В MS Windows 9X-NT

управление аппаратным обеспечением реализовано с помощью нескольких программных модулей, включая user32.exe, drv-модули (**драйверы устройств**). Взаимодействие с пользователем реализует (в стандартной конфигурации) программа explorer.exe.

Дополнительные компоненты СПО устанавливаются либо в составе ОС (дефрагментаторы, программы восстановления системы и т.д.), либо добавляются при установке аппаратного обеспечения или прикладных программ. Например, создавая системы одновременного хранения информации на нескольких жестких дисках (т.н. RAID-массивы) нужно установить такую компоненту СПО как драйверы этих дисков. Много новых компонент СПО приходится добавлять в ходе настройки ноутбуков. В качестве примера добавления новых компонент СПО при установке прикладных программ можно указать Nero Burning ROM. Эта программа устанавливается с собственным ASPI-драйвером, что позволяет более эффективно работать с CD/DVD дисками.

В завершение раздела ответим на вопрос: «Существуют ли альтернативы ОС MS Windows?» Да, во многих странах мира и даже на государственном уровне осуществляется перевод пользователей на БЕСПЛАТНЫЕ UNIX-оидные операционные системы (Linux, FreeBSD, Open Solaris и т.п.)

ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

В настоящее время существует множество прикладных программ со сходными функциями и возможностями. Выбор между ними – это дело вкуса каждого. Например, для загрузки страниц сети Internet можно использовать как программу Microsoft Internet Explorer, так и, например, Mozilla Firefox. Редактор Microsoft Office Word может быть заменен более мощной, но и более сложной для освоения, издательской системой TeX и ее многочисленными модификациями. Вообще, хорошей альтернативой всему пакету программ Microsoft Office является набор программ Open Office, который распространяется свободно. Для загрузки, чтения, редактирования и отправки электронной почты также можно использовать разные средства (Microsoft Office Outlook, The Bat и др.). В общем, практически у каждого программного продукта существует альтернатива.

Как показывает опыт последних лет, современный студент достаточно хорошо владеет наиболее распространенным прикладным программным обеспечением и ориентируется в постоянно появляющихся полезных новинках. Уровень его подготовки и необходимых практических

навыков достаточен для быстрого освоения любого программного продукта и ориентации в широких возможностях наиболее распространенных из них. По этой причине, мы позволили себе не описывать подробно работу с различными программами, тем более что это не является нашей основной задачей и не может быть предметом обсуждения в методическом пособии относительно небольшого объема. Остановимся лишь на некоторых общих вопросах.

Microsoft Office Excel

Табличный процессор Excel ориентирован на создание и обработку электронных таблиц. Рассмотрим простую таблицу, приведенную ниже.

	A	B	C
1	3,5	2	7
2		5	1
3	1	2	
4	4	=Функция(аргумент)	3

Ячейка B4 содержит результат вычисления некоторой функции, явный вид которой конкретизируется далее. В качестве аргумента функции выступает набор ячеек. Диапазон и перечисление ячеек задаются при помощи знаков **:** и **;**. Например, команда A1**:**C3 задает все ячейки таблицы, находящиеся в столбцах от A до C с номерами строк от 1 до 3. В этом диапазоне три столбца и три строки, следовательно – 9 ячеек. Команда A1**:**A2**;** C3 выделяет блок ячеек A1**:**A2 и ячейку C3.

В таблице, расположенной ниже, приведен список некоторых функций с описанием их действия и получающегося в ячейке B4 результата (вместо десятичной точки Excel использует запятую в качестве разделителя).

Функция	Описание	Рез-т
=МАКС(A1:B2;B2+C1)	Функция возвращает максимальное значение из списка аргументов, игнорируя логические и текстовые значения.	12
=СУММ(A1:B2;B2+C4)	Функция суммирует значения из всех указанных в параметрах ячеек, включая диапазоны, игнорируя пустые ячейки.	18,50
=ПРОИЗВЕД(A1;B2;A2)	Функция вычисляет произведение, игнорируя пустые ячейки (A2).	17,50

=СЧЁТ(A1:A2; A1; C3)	Функция подсчитывает число ячеек, в которых содержатся только числовые значения. Пустые ячейки не учитываются (A2 и C3). Если ячейка встречается дважды – она входит в сумму дважды.	2
=СРЗНАЧ(A1:A2; B2; C4)	Функция возвращает среднее арифметическое своих аргументов, которые могут быть числами, массивами или ссылками на ячейки с числами.	3,83
=ОСТАТ(B2;A1)	Функция возвращает остаток от деления первого аргумента на второй.	1,50
=ЗНАК(A1)	Функция возвращает 1, если аргумент положителен, -1 – если он отрицателен, и 0 – если он равен нулю.	1
=НЕЧЁТ(A1)	Функция округляет числовой аргумент до ближайшего нечетного целого значения.	5,00
=ЦЕЛОЕ(A1)	Функция округляет числовой аргумент до ближайшего меньшего целого значения.	3,00
=СТЕПЕНЬ(A2;2)	Функция возводит первый аргумент в степень, задаваемую вторым аргументом. Если первый аргумент содержит ссылку на пустую ячейку, результат будет равен нулю.	0
=НЕ(A1>B1)	Логическая функция возвращает отрицание значения аргумента, являющегося логическим выражением (т.к. A1>B1, то условие истинно, а его отрицание ложно).	Ложь
=И(A1>B1; B1>B2/2)	Логическая функция содержит два параметра, являющихся логическими выражениями. Ее значением может быть либо ложь, либо истина. Если оба выражения истинны – значение истинно, иначе – ложно.	Ложь
=ИЛИ(A1>B1; B1>B2/2)	Логическая функция содержит два параметра, являющихся логическими выражениями. Ее значением может быть либо ложь, либо истина. Если оба выражения ложны – значение ложно, иначе – истинно.	Истина
=ЕСЛИ(И(A1>B1;СРЗНАЧ(A1:B1)> B2/2); 1; 0)	Логическая функция ЕСЛИ содержит три параметра, первым из которых является логическое выражение. Если оно истинно, то в качестве результата	1

	выступает второй параметр (в данном примере это единица), иначе – третий параметр (в данном примере это ноль). Здесь логическое выражение представляет собой два условия, связанные функцией “и”, причем оба условия истинны, что и дает в результате единицу.	

Следует помнить, что если какое-либо из значений, используемых в формуле ячейки, имеет **некорректный тип данных**, то она будет содержать следующую запись ***«#ЗНАЧ!».***

Microsoft Office Word

Текстовый процессор Word предназначен для создания документов и обладает широкими возможностями и набором инструментов. Как показала практика преподавания курсов информатики, уровень владения инструментами программы Microsoft Office Word у студентов достаточно высок. Поэтому мы позволим себе не описывать работу с этой программой, а предложить простое упражнение, выполнение которого быстро восстановит необходимые навыки.

1. Создайте документ, состоящий из нескольких абзацев. Установите в каждом абзаце свой шрифт и размер, величину абзацного отступа и **межстрочный интервал** (меню Формат/Абзац). Второй абзац наберите *курсивом* с подчеркиванием. Третий абзац наберите **жирным** шрифтом. Установите выравнивание по левому краю, по правому краю, по центру или по ширине.

2. Между вторым и третьим абзацами установите интервал 1.3 см. Поместите один из абзацев в рамку с заливкой (меню Формат/Границы и Заливка).

3. Создайте **маркированный список**

- Иванов;
- Петров;
- Сидоров.

4. Создайте **нумерованный список**

a) Иванов;

b) Петров;

c) Сидоров.

5. Создайте **многоуровневый список**

(a) Информатика
   1. Понятие информации
   2. Логические выражения

(b) Математика
   1. Алгебра
   2. Геометрия

6. Разбейте текст на несколько разделов.

7. Начните второй раздел с новой страницы (меню Вставка/Разрыв).

8. Установите параметры страницы (меню Файл/Параметры страницы):

I. Размер страницы: 13.5 см на 19 см;

II. Книжную ориентацию;

III. Поля страницы: слева и справа – 25 мм, сверху и снизу – 10 мм;

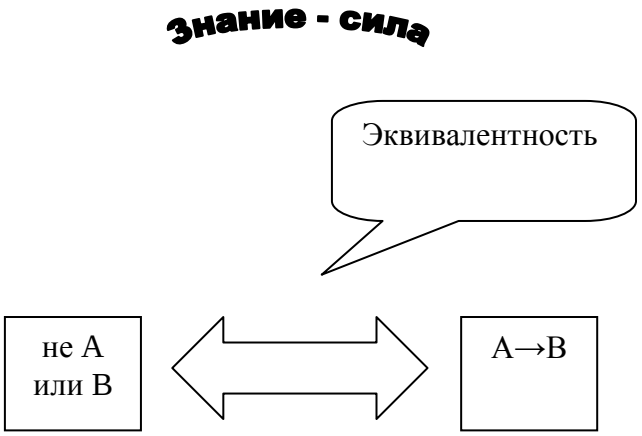
IV. Поле переплета слева размером 9 мм;

9. Задайте разные для четных и нечетных страниц верхний и нижний **колонтитулы** размером 1.2 см (меню Файл/Параметры страницы).

10. Сделайте **обычные сноски** и **концевые сноски** на разных страницах (текст обычной находиться внизу текущей страницы, а концевой – в конце всего документа).

11. Одну из страниц разбейте на 3 **колонки** разной ширины, с разделительной линией межу колонками (меню Формат/Колонки);

12. Вставьте объект из стандартного набора WordArt (меню Вставка/Рисунок/Объект WordArt) и добавьте произвольную схему, нарисованную с помощью панели инструментов «Рисование» (установите одинаковые размеры прямоугольников и введите текст командой «Добавить текст», центрируйте его).



13. Вставьте в текст объект **Microsoft Equation** – формулу, приведенную ниже (меню Вставка/Объект). Щелкните дважды на формуле, активировав **редактор формул**, и отредактируйте ее

$$Q_0^* = \left\{ q_0^* \in L_2(\Omega, R^2) \middle| \int_\Omega q_0^* \cdot \nabla w dx = 0, \forall w \in V_0 \right\}.$$

14. Вставьте в текст таблицу, состоящую из 2 строк и 2 столбцов (меню Таблица/Добавить). В этой же вкладке добавьте в таблицу столбец справа от последнего столбца, еще одну строку между первой и второй. Установите приведенное ниже выравнивание текста в ячейках. Объедините ячейки второго и третьего столбцов в одну. Нажав на правую кнопку мыши и выбрав свойства таблицы, установите длину получившейся ячейки равной 11.2 см, выравнивание по нижнему краю и введите текст.

Информатика	
Химия	
Математика	Необходимо знать не только эти предметы

Microsoft Office PowerPoint

Программа Microsoft Office PowerPoint служит для создания современных презентаций, включающих в себя широкий спектр элементов – текст, графику, диаграммы, видео и прочее. Основным элементом презентации является **слайд**. Презентация включает в себя не только непосредственно слайды, но и заметки докладчика, и раздаточный материал для слушателей. PowerPoint обладает мощными средствами разработки и оформления презентации с широким набором стандартных стилей оформления слайдов – задания структуры и расположения информации, фонового рисунка, цветовой гаммы и т.п.

Запуск программы осуществляется абсолютно аналогично другим приложениям пакета Microsoft Office. Мы кратко опишем работу с версией 2003 года. Другие версии могут отличаться от нее некоторыми деталями реализации, общие принципы работы, тем не менее, сохраняются. В целом набор меню этой программы стандартен и мы не будем на нем останавливаться. Существует, однако, ряд уникальных опции программы, отражающих ее назначение в пакете Microsoft Office. По умолчанию программа открывается в Обычном режиме. При этом на экране появляется панель, содержащая вкладки Структура/Слайды, а также панель Заметки. Первая панель позволяет переключать стиль отображения общей структуры презентации со списка на показ уменьшенных копий изображении каждого слайда в том виде, в котором они появятся в самой презентации. Вторая панель служит для ввода заметок. При этом в центре будет отображаться текущий выбранный для работы слайд.

Создать новую презентацию можно через меню Файл. При этом справа появится область задач, в которой выбирается разметка слайда, общий дизайн (шаблон оформления) и прочие установки. Пользователю будет предложено выбрать один из множества стандартных вариантов оформления, определяющих положение и объем текста, наличие других элементов (таблицы, диаграммы, видео, изображения). Далее можно добавлять слайды в презентацию при помощи кнопки Новый слайд. Обычно каждый слайд содержит заголовок, элемент презентации и текст комментария к нему. Текст комментария можно вводить как при помощи панели Заметки, лежащей ниже изображения текущего слайда, так и в специальном режиме через пункт меню Вид/Страницы заметок.

Для просмотра презентации с целью внесения изменений удобно использовать находящиеся слева вкладки Структура/Слайды, указывая стрелкой на конкретный слайд в списке или на его уменьшенное изображение и нажимая левую кнопку мыши для выбора. Перемещаться по слайдам презентации также можно при помощи клавиш PgUp/PgDn или при помощи кнопок с изображением стрелок Вниз/Вверх на панели инструментов. Обратите внимание на то, что в строке состояния в нижнем левом углу можно увидеть надпись Слайд «Номер» из «Общее количество».

Для изменения порядка следования слайдов удобен не Обычный режим, а специальный режим Сортировщика слайдов (кнопка для переключения режима находится в левом нижнем углу, можно также использовать меню Вид/Сортировщик слайдов). При его выборе все слайды отображаются в одном окне. Их можно перемещать, указывая на них стрелкой, нажимая и удерживая левую кнопку мыши.

Для полноэкранного показа уже готовой и полностью отредактированной презентации существует несколько способов. Начать показ можно нажатием клавиши F5, выбором пункта основного меню Показ слайдов/Начать показ или нажатием на кнопку Показ слайдов с текущего слайда (Shift+F5). Перемещение по презентации в разных направлениях осуществляется при помощи клавиш PgUp/PgDn, стрелок ↑/↓, стрелок ←/→ или колеса прокрутки мыши. Для перехода от предыдущего слайда к следующему можно использовать также клавишу Enter. Чтобы завершить демонстрацию слайдов необходимо нажать клавишу Esc.

Сохранить презентацию можно при помощи пункта меню Файл/Сохранить. Созданный **файл презентации** PowerPoint имеет расширение **ppt.**

Архиваторы

Архиваторы – это программы, предназначенные для сжатия и компактной упаковки файлов в один файл архива. Процесс создания архива называют также **архивацией**. Основным результатом архивации является уменьшение суммарного объема памяти, требующегося для хранения файлов, что, в свою очередь, ускоряет и упрощает процесс их передачи. Такой эффект достигается путем обработки данных (выделения в них повторяющихся последовательностей и т.п.) и сохранения в специальном формате.

Важными характеристиками эффективности процесса сжатия являются степень сжатия (в процентном отношении к исходному объему) и время, которое затрачивается на упаковку и распаковку архива. Естественно, предпочтительнее тот **архиватор**, который обрабатывает архив за меньшее время и дает результат меньшего объема. Существует множество различных форматов сжатия файлов и версий предназначенных для этого программ-архиваторов. Как правило, одна программа поддерживает сразу несколько форматов записи архивов. В некоторых из них реализованы функции перекодировки из одного формата в другой. Важной особенностью всех этих форматов является то, что при распаковке данные должны полностью восстанавливаться в оригинальной форме. Таким образом, это должно быть **сжатие без потерь** информации.

Качество сжатия зависит от нескольких факторов. Во-первых, важен тот метод, который был использован при создании архива и та программа-архиватор, в которой этот метод был реализован. Помимо метода сжатия на его эффективность существенно влияет тип архивируемых данных. Легко проверить, что текстовые файлы сжимаются намного лучше, чем,

например, исполняемые. Естественно, что файлы, создание которых само по себе основано на обработке информации, и ее представлении в особом формате с потерями малозначимой информации (jpg, avi, pdf, djvu и др.), сжимаются хуже. Качество их сжатия зависит от эффективности алгоритмов, заложенных в сам метод представления данных.

Наиболее распространенными на данный момент архиваторами являются **WinRAR** и **WinZIP**. Остановимся подробнее на возможностях программы WinRAR, поскольку, по нашему мнению, эта программа значительно превосходит по своей эффективности вторую. В стандартном режиме она создает архивные файлы с расширением **rar.** В случае, когда это необходимо программа позволяет создавать и **zip**-архивы. Она позволяет распаковывать файлы, запакованные в собственном формате (если использованная при сжатии версия не новее той версии, которая запускается для распаковки), а также в ряде других употребительных форматов (например, **zip**, **arj**, **cab**, **tar** и др.). Чтобы избежать возможных проблем, связанных с отсутствием программы-архиватора на компьютере, на котором будет производиться распаковка данных, можно создать так называемый самораспаковывающийся **SFX**-архив. Он имеет расширение исполняемого файла **exe**, содержит дополнительную запись и распаковывается автоматически при запуске файла на исполнение. Из полезных опций защиты информации отметим возможность установить пароль на открытие архива и сделать содержание архива невидимым. WinRAR позволяет протестировать существующий архив на наличие ошибок, связанных с потерями данных, т.е. определить степень целостности данных (Test archived files). В случае необходимости можно сделать попытку восстановления данных поврежденного архива (Repair damaged archive). Имеется также ряд важных опции, позволяющих при создании архива уменьшить возможные потери данных при повреждении целостности архива. Например, это опция включения в архив специальной записи для его восстановления (Put recovery record).

Вопросы

1. Всегда ли обязательным является наличие версии программы-архиватора, которой производилось сжатие архива на компьютере, на котором будет происходить его распаковка?
Комментарий. ► Нет, не всегда. Во-первых, если архив самораспаковывающийся, то ему не требуется программа архиватор. Во-вторых, многие программы позволяют распаковать архивы, записанные в «чужих» форматах. Возможно, та программа, которая установлена на Вашем компьютере, способна успешно распаковать предложенный архив.

2. Для чего предназначена программа Microsoft PowerPoint?
1) Для просмотра электронной почты.
2) Для подготовки презентаций.
3) Только для сортировки уже готовых слайдов.
Комментарий. ► Для подготовки презентаций.

3. Максимальное количество стилей оформления текста, которые могут быть использованы одновременно в документе равно…
Варианты:
1) количеству страниц в документе;
2) не ограничено;
3) не более 128;
Комментарий. ► Не ограничено.

БАЗЫ ДАННЫХ

Информационные технологии используются практически во всех областях человеческой деятельности. Это чрезвычайно мощный инструментарий для поддержки процессов принятия решений, будь-то управление какими-то сложными системами, исследования или проектирование новых систем.

Сложность объектов, с которыми приходится иметь дело, требует, согласно принципу Эшби, сопоставимых по уровню сложности инструментов и огромных объемов исходной информации. Поэтому ядром современных информационных технологий являются **системы управления базами данных** (**СУБД**). Системы, основанные на использовании БД, позволяют решать поставленные задачи с минимальными потерями, как времени, так и других расходуемых ресурсов, например, привлекаемых технических средств информационно-вычислительных систем (внутренняя и внешняя память).

Следует заметить, что конкретная БД проектируется и используется согласно информационным потребностям ее возможных пользователей. В одной или нескольких БД невозможно хранить все обо всем! Создатели БД разрабатывают ее как хранилище сведений об объектах конкретной предметной области, согласно требованиям будущих пользователей. Идеи, положенные в основу технологий, использующих БД, вкратце можно сформулировать одним предложением: «Информация должна быть структурирована!». Такой подход к накоплению и хранению больших

объемов информации человечество использовало задолго до появления информационно-вычислительных систем. Вспомним, вместо текстов, содержащих описание некоторого множества однотипных объектов, обычно составляются таблицы. К примеру, вместо «Студент Иванов Иван сын Иванов родился в 1990 году от Р.Х., а Петров Петр Петрович 1991 г.р. ...» составляют таблицу:

Фамилия	Имя	Отчество	
Иванов	Иван	Иванович	1990
Петров	Петр	Петрович	1991
...			

Это существенно упрощает как сбор, так и последующее использование накопленных сведений, например поиск.

Создание БД всегда начинается с определения множества объектов конкретной предметной области, сведения о которых планируется хранить в будущей БД. Например, для управления ходом учебного процесса в вузе понадобятся данные о студентах, преподавателях, студенческих группах, кафедрах, факультетах. Таким образом, формируется список **информационных объектов** (**ИО**): абстрактный студент, абстрактный преподаватель, абстрактная студенческая группа, и т.д. Далее, в контексте будущих решаемых с помощью БД задач анализируется каждый из ИО:

- взятый в отдельности;
- во взаимосвязи с другими объектами их этого множества.

Цель подобного анализа − выявление подлежащих хранению данных о каждом из них. Другими словами, выделяются атрибуты (именованные характеристики) ИО, которые необходимы для последующей работы.

В нашем примере у ИО «абстрактный студент» выделим такие атрибуты как «Фамилия», «Имя», «Отчество», «Год рождения». Его связь с ИО «абстрактная студенческая группа» предполагает хранение атрибута «Номер студенческой группы студента», и т.д. В то же время такой атрибут как «Размер головного убора студента» или «Размер обуви студента» едва ли стоит принимать во внимание, т.к. эти характеристики вряд ли влияют на ход учебного процесса. Затем, по аналогии с обычными таблицами на традиционном носителе (бумаге), формируем электронные таблицы. Таблицы − это основа любой БД. Каждая из них будет хранить данные об экземплярах ИО определенного типа. В нашем примере одна таблица будет хранить сведения о студентах, другая − студенческих группах и т.д.

При создании таблицы указывают ее имя − это первая характеристика (обычно имя таблицы семантически, т.е. по смыслу связано с хранящейся в ней информацией). Кроме имени таблица имеет

еще две характеристики: структуру (или форму хранения) и содержимое (содержание).

Форма хранения (структура таблицы) определяется тем набором атрибутов ИО, которые мы выделили на предыдущем этапе (это ее графы или столбцы, используя аналогию с обычной таблицей), и порядком их следования.

Содержание таблицы − это конкретные сведения, данные о конкретных экземплярах ИО, т.е. в нашем примере − о студентах Иванове, Петрове и т.д. Содержание таблицы − это то, что мы храним в ней (это аналог всей совокупности строк обычной таблицы).

Записью в общепринятой терминологии называется аналог строки обычной таблицы, которая содержит сведения об отдельном экземпляре ИО. Запись в свою очередь состоит из отдельных далее уже неделимых порций информации − значений атрибутов этого экземпляра ИО (аналог содержимого графы или столбца в этой строке обычной таблицы). Эти атомарные порции информации принято называть **полями записи** таблицы. Для каждого поля записи таблицы БД на этапе ее создания требуется указать две из трех характеристик: имя поля и форму хранения данных в этом поле (тип и размер).

В нашем примере для поля записи таблицы «Студенты», в котором будем хранить фамилию студента, укажем имя поля «Фамилия», а в качестве формы хранения выберем текстовый тип данных длиной, положим, в 25 символов. А третья характеристика − содержимое поля записи, будет указана в ходе заполнения таблицы данными, когда мы завершим проектирование БД, и станем добавлять в таблицы записи. Скажем, внося в таблицу «Студенты» сведения о студенте Иванове И.И. 1990 г.р., мы в поле «Фамилия» записи об этом студенте, внесем значение «Иванов». Внося в таблицу запись о Петрове в поле «Фамилия» внесем значение «Петров».

Каждая по отдельности таблица содержит сведения об экземплярах ИО одного определенного типа. Но, кроме того, в БД должна быть сохранена информация и о существующих в данной предметной области отношениях между ИО, о тех связях между объектами, которые важны в контексте решаемых с помощью БД задач. В нашем примере существует отношение между ИО «абстрактный студент» и ИО «абстрактная студенческая группа», которое важно для управления учебным процессом. Для хранения в БД информации о таких отношениях между ИО предметной области, на этапе проектирования следует выбрать одну их трех стандартных абстрактных моделей хранения этих связей (по сути, это выбор системы управления базой данных):

- **иерархическую**;

- **сетевую**;
- **реляционную**.

Иерархическая модель − это, в определенном смысле, наиболее жесткий вариант выразить отношения между ИО: все ИО должны быть выстроены иерархически, так что у каждого из них может быть только одна связь, которую охарактеризуем как связь между «родителем данного ИО» и, собственно, самим ИО. Но при этом допустимо наличие любого количества связей, в которых данный ИО сам выступает как «родитель», а связанные с ним – как его «потомки». Все «потомки» одного ИО должны располагаться на одном и том же уровне иерархии. В нашем примере в качестве самого верхнего уровня иерархии можно выбрать ИО «ВУЗ», тогда его потомками на следующем уровне иерархии будут ИО «факультеты», а они в свою очередь порождают уровень иерархии «кафедры», и т.д.

Сетевая модель хранения связей между ИО допускает произвольное число связей между любыми двумя экземплярами двух разных ИО. В нашем примере, используя иерархическую модель, было бы крайне тяжело организовать связь между ИО «абстрактный студент» и ИО «абстрактный преподаватель», если по одной из этих дисциплин студенческая группа досталась бы сразу нескольким преподавателям (такое случается при выполнении практических работ, курсовых проектов и т.п.). А сетевая модель в этой ситуации предоставляет достаточную свободу.

Наиболее распространенный вариант модели данных − **реляционная модель**. Это некоторое промежуточное решение (по сравнению с двумя вышеперечисленными). Связи организуют между таблицами БД на основе совпадающих значений атрибутов, хранимых в полях записей обеих связываемых таблиц. В нашем примере организация связи между таблицами «Студенты» и «Группы» возможна в том случае, если запись таблицы «Группы» имеет поле «Порядковый номер группы», а запись таблицы «Студенты» имеет, соответственно, поле «Номер группы, в которой учится студент». Пусть у студента Петрова запись содержит «1» в качестве значения поля «Номер группы, в которой учится студент». Тогда в таблице «Группы» мы найдем среди ее записей ту, что содержит значение «1» в поле «Порядковый номер группы» и выяснится, что это группа «3057/1», мы получим сведения о специальности, по которой готовят студентов этой группы, включая Петрова, и еще много другой важной информации. Естественно, все это возможно лишь в том случае, когда каждая запись в таблице «Группы» содержит уникальное значение в поле «Порядковый номер группы». В связи с этим полезно указать сложившуюся терминологию.

Первичным ключом к записям таблицы БД является одно поле (**простой первичный ключ**) или совокупность полей (**составной первичный ключ**) записи этой таблицы, позволяющие однозначно определить (идентифицировать) в этой таблице отдельную запись о конкретном экземпляре ИО. Так, в нашем примере, простым первичным ключом записей таблицы «Группы» будет поле «Порядковый номер группы», хранящее порядковый номер группы в этой таблице. Составным ключом могла бы служить совокупность полей «Название группы» и «Специальность», если допустить существование студенческих групп с одинаковым названием.

Внешним ключом к записи таблицы БД называется поле записи, по значению которого организована связь записей этой таблицы с записями другой таблицы БД. Можно трактовать это поле как «ключ», с помощью которого мы в нашем примере (поле «Номер группы, в которой учится студент» записи таблицы «Студенты») «открыли дверь» в другую таблицу «Группы» и получили там релевантную информацию о группе, в которой учится студент Петров. А типы отношений между таблицами, которые образуются при их связывании, можно определить как **«один − ко многим»** (записи таблицы «Группы» к записям таблицы «Студенты»), **«многие − к одному»** (записи таблицы «Студенты» к записям таблицы «Группы») − т.е. это две различных точки зрения на одну и ту же связь, и **«многие − ко многим»** (этого типа отношений между таблицами реляционной БД лучше избегать).

Наконец, по окончании проектирования БД, т.е. после определения структур записей таблиц и связей между таблицами, можно приступать к заполнению БД информацией. При этом следуют правилу: сначала заполняют информацией таблицы, которые во всех связях участвуют со стороны «один». Таблицы с внешними ключами заполняют информацией, когда уже можно разрешить ссылки на связанные записи другой таблицы. В нашем примере вначале заполняют таблицу «Группы», а затем уже в таблицу «Студенты» можно вносить сведения о Петрове, т.к. ссылка на группу, в которой учится Петров, (с помощью «1») уже разрешима (в таблице «Группы» уже есть запись с таким значением в поле «Порядковый номер группы»).

Определение структуры таблиц БД и связей между ними обычно осуществляется с помощью языка программирования **DDL** (Data Definition Language). Если Вы используете при проектировании своей БД графический интерфейс, предоставляемый СУБД, скажем **QBE** (Query By Example) в **MS Access**, то нужно понимать, что за графическим интерфейсом разработчики СУБД скрыли от вас один из диалектов DDL (для реляционных БД это чаще всего один из диалектов языка **SQL**).

В ходе эксплуатации БД возникает необходимость выполнить над данными таблиц БД такие операции как:

- **выборка** некоторого подмножества записей из одной или нескольких связанных между собой таблиц согласно указанным условиям (критериям отбора информации);
- **добавление записей** в одну или несколько связанных между собой таблиц;
- **редактирование записей** в одной или нескольких связанных между собой таблицах;

Эти задачи обычно решают с помощью небольших по объему исходного текста программ, написанных на языке программирования **DML** (Data Manipulation Language). Для реляционных баз данных это один из диалектов SQL. Операторы (или их еще часто называют предложениями) SQL реализуют вышеперечисленные операции, если они допустимы с точки зрения сохранения целостности хранимой информации. В нашем примере, запрос на выборку записей о мужчинах (критерий отбора − значение «муж» в поле «Пол» записей таблицы «Студенты») будет выполнен в помощью оператора (предложения) SELECT. Результат запроса − динамическая таблица (ее часто называют **выборкой**), созданная по итогам выполнения оператора SELECT. Она будет содержать подмножество записей таблицы «Студенты», удовлетворяющих указанным условиям. А содержимое таблицы «Студенты» при этом никак не меняется! Заметим, что в некоторых руководствах такую операцию (запрос на выборку) еще называют **фильтрацией записей.**

Современные СУБД предоставляют пользователям ряд полезных инструментальных средств, предназначенных для работы с данными таблиц БД посредством диалоговых окон (**формы**) и для произвольного форматирования подмножеств записей, получаемых по итогам выполнения запросов (**отчеты**).

Вопросы

1. Какие характеристики таблицы БД изменяются, когда мы:

- *добавляем или удаляем записи;*
- *добавляем или удаляем поле.*

Комментарий. ► Добавляя или удаляя записи таблицы (аналог добавления или удаления строк обычной таблицы) мы изменяем содержимое этой таблицы, не меняя ее структуру. А вот удаление или добавление поля таблицы (аналог удаления одной из граф или добавление новой графы в обычную таблицу) приводит к изменениям не только содержимого, но и формы хранения (структуры таблицы).

2. Языком запросов к реляционным базам данных является…
1) SSH; 2) SQL; 3) Pascal; 4) C#.
Комментарий. ► Правильный ответ: SQL. Аббревиатура SQL расшифровывается Structured Query Language, что переводится как Структурный Язык Запросов.

3. Для получения таблицы из совокупности связанных таблиц путём выбора полей, удовлетворяющих заданным условиям, используется...
1) запросы; 2) отчёты; 3) формы; 4) схемы.
Комментарий. ► Правильный ответ: запросы.

4. Представление реляционной модели данных в СУБД реализуется в виде...
1) таблиц; 2) сети; 3) деревьев; 4) предикатов.
Комментарий. ► Большинство современных систем обработки данных ориентировано на реляционную модель, когда данные организованы при помощи набора связанных таблиц.

ТЕХНОЛОГИИ ОБРАБОТКИ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

Различают два основных вида компьютерной графики: **растровая** и **векторная**. Их отличие заключается в принципе представления и хранения изображения.

Растровое изображение формируется в прямоугольной области из точек (**пикселей**), каждой из которых приписывается код цвета. При хорошем качестве изображения человеческий глаз не в состоянии воспринять такую детализацию изображения, и оно сливается в единую картину, в которой каждый пиксель невидим. Основной недостаток растровой графики проявляется при изменении размеров пиксельного рисунка. При увеличении рисунка возникает проблема определения цвета «новых» пикселей. Это делается усреднением. При сжатии изображения соответствующие пиксели удаляются, и происходит потеря визуальной информации. Поэтому качество изображения при его **масштабировании**, как правило, ухудшается. Оно также определяется разрешением исходного изображения – количеством точек на единицу длины, а также глубиной цветопередачи.

Векторная графика представляет изображение посредством

описанных математическими формулами **геометрических примитивов** (точек, линий, многогранников и т.п.). Основное преимущество такого представления заключается в сохранении качества изображения при масштабировании, т.е. при изменении масштаба изображения. Как бы ни преобразовывалась, например, линия, меняются только ее параметры, хранящиеся в ячейках памяти, поэтому векторную графику иногда называют вычисляемой графикой. Не повлияют на качество изображения и другие операции – сдвиг и вращение. В случае если изображение очень простое, оно будет качественно описано в векторном формате, и графический файл будет иметь небольшой размер. Однако, если изображение слишком сложное, то для его детализации необходимо большое количество таких примитивов, что влечет за собой существенное увеличение объема графического файла. Теоретически векторная графика, как и растровая позволяет воспроизвести изображение практически любой сложности, на практике же объем получившегося выходного файла будет слишком велик.

Таким образом, анализируя все сказанное выше, можно сделать вывод, что растровая графика хорошо подходит для представления фотографий и других графических объектов с большим разнообразием цветовых переходов, большим количеством мелких деталей сложной формы. При помощи векторной графики хорошо представляются простые объекты – схемы, логотипы и т.п.

Одним из наиболее распространенных редакторов с широким набором инструментов, позволяющих работать с растровой графикой, является Adobe Photoshop. Один из простейших редакторов растровых изображений – это стандартная программа Microsoft Paint. С векторными изображениями удобно работать в графических пакетах Adobe Illustrator и CorelDRAW.

Графические форматы

К числу распространенных форматов хранения растровых изображений относятся

GIF, TIFF, PNG, JPG, BMP.

Опишем их подробнее.

Формат **GIF** (англ. Graphics Interchange Format) отличается использованием режима 256 цветов. Его используют, в основном, для хранения простых элементов с минимальным количеством цветов, предназначенных для оформления web-страниц. Формат поддерживает анимацию – вывод последовательности кадров.

Формат **TIFF** (англ. Tagged Image File Format) поддерживает очень

большие изображения и разнообразные методы сжатия, и используется в случаях, когда требуется изображение наилучшего качества. Он стал популярным форматом для хранения изображений с большой глубиной цвета, используется при сканировании, отправке факсов, распознавании текста, в полиграфии, широко поддерживается графическими приложениями.

Формат **PNG** (англ. Portable Network Graphics) был создан сообществом независимых разработчиков как альтернатива формату GIF после перехода последнего в разряд коммерческих продуктов. Подходит для использования в сети Internet и публикации там высококачественной растровой графики. Сжатие информации в этом формате происходит без потерь. Меньше подходит для хранения небольших элементов оформления web-страниц.

Формат **JPEG** (англ. Joint Photographic Experts Group) является результатом работы экспертной группы по фотографии. Он использует одноименный алгоритм сжатия с «потерями», причем теряется та графическая информация, которая обычно не проявляется в реальных изображениях. В результате получается эффективный метод сжатия файлов, при использовании которого, как правило, но далеко не всегда, отбрасывается только незначимая и незаметная человеческому глазу визуальная информация.

Формат **BMP** (англ. Bitmap) – это собственный графический формат системы Windows, отличающийся предельной простотой. В этом формате доступно большое количество изображений, однако он поддерживает лишь самые простые методы сжатия. Это делает его пригодным для быстрого чтения и записи небольших изображений. Поэтому он используется в основном самой системой Windows, и почти не используется в Internet.

К числу форматов хранения векторных изображений относятся PDF, EPS, AI, CDR, WMF.

Формат **PDF** (англ. Portable Document Format) разработан фирмой Adobe для создания переносимой на разные платформы электронной документации, презентаций и т.п. Сейчас он является открытым для использования стандартом. Формат удобен, поскольку он обладает широким набором алгоритмов сжатия данных различных типов, входящих в документ. Открыть файл, сохраненный в формате PDF можно, например, при помощи программ Adobe Acrobat Reader, Acrobat Distiller или Acrobat Professional.

Формат **EPS** (англ. Encapsulated PostScript) создан фирмой Adobe. Файлы этого типа практически представляют собой программу с командами для устройства вывода. Формат Encapsulated PostScript считается специалистами одним из самых надежных и универсальных. Он

обладает массой настроек, задаваемых при помощи параметров. Возможность сохранения файла в таком формате поддерживается большинством программ, работающих с графикой.

Формат **AI** – это собственный формат программы Adobe Illustrator.

Формат **CDR** – стандартный формат, являющийся основным для пакета CorelDRAW.

Формат **WMF** является собственным векторным форматом системы Windows.

МОДЕЛИРОВАНИЕ

В основе любой инженерной деятельности лежит методология моделирования. Модель и **моделирование** – очень широкие понятия. Например, всякое познание – это уже моделирование, так как в коре головного мозга в идеальном виде отображается данный объект, т.е. мы имеем здесь дело с моделью объекта.

Под **моделью** понимают некоторое упрощенное отображение (подобие) реального объекта, с помощью которого воспроизводятся его существенные признаки. Отображаться могут как реальные или абстрактные объекты и процессы, так и связи между ними и их свойствами. Всё многообразие моделей можно разделить на две категории: **предметные** и **информационные**. Предметные модели, которые иногда называют материальными, – это физическое подобие объекта моделирования. Например, с помощью модели самолета, уменьшенной по сравнению с оригиналом во много раз, проверяют его аэродинамические свойства. Информационные модели могут быть словесные (или вербальные), графические, математические или табличные. Словесная модель описывает объект моделирования на разговорном языке. Графические модели представляют собой чертежи, графики или схемы реальных объектов. В математической модели объект описывается с помощью математических формул и уравнений, табличные модели представляют собой совокупность данных, расположенных в прямоугольной таблице или таблицах. Периодическая таблица Менделеева – это информационная модель. Большое значение моделирование имеет для изучения объектов и процессов, которые недоступны прямому наблюдению – вспомним знаменитую модель атома Нильса Бора, реальный прототип которой никто не видел.

Среди информационных моделей выделяют **формализованные** модели. Под формализацией понимают представление описания модели в

виде цепочки четких и машинно-воспроизводимых алгоритмов. Примером строго формализованных моделей являются математические модели.

Модели могут различаться в зависимости от степени их соответствия реальным объектам. Модель может воспроизводить только функциональные характеристики процесса или явления, в то же время структура и внутренние процессы объекта и модели совершенно различаются. В этом случае говорят, что моделируется «**черный ящик**». «Черный ящик» раскрывается, если построенная модель наряду с функциями имитирует и протекающие внутри объекта процессы. Такие модели называют **имитационными.**

Главное в моделировании – избежать излишнего усложнения или упрощения. Для этого в каждом конкретном случае необходимо проводить тщательный и всесторонний анализ процесса моделирования, степени соответствия модели и исследуемого процесса или явления.

Построение любых моделей так или иначе связано с процессом научного познания. **Познавательный процесс** заключается в активном взаимодействии человека (субъекта) с реальной действительностью (объектом) и носит целенаправленный характер. В основе этого процесса всегда лежит **задача**, которую решает человек для достижения своих целей. Можно выделить несколько этапов в этом процессе.

На первом этапе у человека появляется осознание необходимости в познании или изменении существующего положения вещей и определение цели как желательного результата будущих действий. Этот этап связан с построением и анализом проблемной ситуации.

На втором этапе формулируется задача, подлежащая решению. При этом уточняется цель с учетом свойств объекта и имеющихся в распоряжении ресурсов и возможностей.

На следующем этапе происходит выбор из множества уже известных, познанных свойств и характеристик объекта наиболее существенных. Формируется конкретный идеальный образ объекта, предназначенный для решения данной задачи.

И, наконец, на последнем этапе реализуется процесс поиска решения задачи. Причем в зависимости от получаемых результатов производится корректировка допущений, принятых на предыдущих этапах. Другими словами, весь процесс достижения цели осуществляется методом последовательных приближений (итераций).

Важно, что центральным моментом этого процесса является задача, так как только она позволяет ограничить практически бесконечное множество признаков, свойств и характеристик реального объекта и перейти к ограниченному по свойствам его идеальному образу. Вводить

понятие модели без четкого указания задачи, ради решения которой она создается, не имеет смысла.

Математическая модель – это система математических соотношений: формул, уравнений, неравенств и т.д., отражающих **существенные свойства объекта** или явления. Наиболее эффективно математическую модель можно реализовать на компьютере в виде алгоритмической модели – так называемого **вычислительного эксперимента.**

Рассмотрим классификацию объектов моделирования в соответствии с их попарно противоположными свойствами, влияющими на выбор методов моделирования и соответствующего математического аппарата. **Непрерывными** (континуальными) считаются объекты, выходные переменные которых, являются непрерывными. **Дискретные** объекты имеют выходные переменные, которые могут принимать некоторое конечное число известных значений. К таким объектам принадлежит, например, автомат для продажи билетов. Модели непрерывных объектов, полученные с помощью дискретных методов (например, методом конечных элементов) называют иногда непрерывными дискретизированными в отличие от дискретных моделей дискретных объектов. Свойства **стационарности-нестационарности** характеризует степень изменчивости объекта во времени (**статические – динамические модели**). Весьма существенно деление объектов на **линейные** и **нелинейные**. Различие между ними заключается в том, что для первых справедлив принцип суперпозиции (наложения), когда каждый из выходов объекта характеризуется линейной зависимостью от соответствующих входных переменных. Одним из важнейших признаков, определяющих возможные методы описания, является деление объектов на **детерменированные** и **стохастические** (случайные). Определение «детерменированные» означает лишь тот факт, что по условиям решаемой задачи и применительно к свойствам конкретного объекта случайными факторами в данном конкретном случае можно пренебречь. Для всех реально существующих объектов объективно присуще свойство стохастичности. И весьма вероятно, что при другой постановке задачи «детерменированный» объект придется рассматривать как «стохастический».

В последнее время появилось **визуально-натурное моделирование**. Оно обеспечивается за счет отображения явления средствами машинной графики, т.е. перед исследователем демонстрируется своеобразный «компьютерный мультфильм», снимаемый в реальном масштабе времени.

Вопросы

1. К моделированию нецелесообразно прибегать в случае, если...
1) создание объекта чрезвычайно дорого;
2) процесс очень быстрый;
3) не определены существенные свойства исследуемого объекта.
Комментарий. ► Правильным является последний ответ. Если не определены существенные свойства исследуемого объекта, то существуют большие сомнения в достоверности результатов моделирования.

2. В каком отношении находятся понятия «фрукт – апельсин» ...
1) «общее – частное»;
2) «процесс – результат»;
3) «целое – часть»;
4) «модель – объект»;
5) «объект – субъект».
Комментарий. ► Правильный ответ: «общее – частное». Апельсин является разновидностью фрукта. В отношении «процесс – результат» находятся, например, понятия «обучение – аттестат», «целое – часть» – понятия «учебник – раздел», «модель – объект» – понятия «карта – местность», «объект – субъект» – понятия «автомобиль – водитель».

3. В отношении «общее – частное» ***НЕ*** *находятся понятия*
1) «механизм – весы»;
2) «самолет – МИГ29»;
3) «книжный шкаф – книга».
Комментарий. ► Правильный ответ: «книжный шкаф – книга», поскольку книга не является разновидностью книжного шкафа.

4. Метод Монте-Карло относится к методам _____ моделирования.
1) графического;
2) логического;
3) аналитического;
4) статистического.
Комментарий. ► Правильный ответ: статистического.

5. К предметным моделям относятся ...
а) схема эвакуации при пожаре;
б) таблица значений давления газа при изменении температуры;
в) авиамодель истребителя;
г) полоса препятствий.

Комментарий. ► Правильный ответ: в, г.

6. *Примером неформализованного описания модели служит…*
1) фотография объекта;
2) уравнение 3-го закона Ньютона;
3) запись алгоритма в виде блок-схемы;
4) инструкция пилота самолета.
Комментарий. ► Правильный ответ: фотография объекта.

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Приступая к разработке программ, разработчик руководствуется своими собственными соображениями о способах проектирования. В силу большой сложности решаемых задач к работе обычно привлекаются целые коллективы программистов и специалистов в смежных областях. Подобная ситуация с необходимостью требует следовать определенным стандартам.

Общепринятым на сегодняшний день является **принцип проектирования «сверху-вниз»**, когда вся задача в целом расчленяется, разбивается на ряд подзадач (более простых), которые подлежат решению в определенном порядке или параллельно. Если требуется, подзадачи подвергаются аналогичному анализу и разделению на составляющие. В конечном счете, разработчики должны получить некоторое множество элементарных задач (элементарных в том смысле, что для их решения уже имеются стандартные алгоритмы). Правила составления алгоритма решения задачи в целом (синтез, композиция) можно свести к следующим:

- **линейный порядок исполнения кода** (операторов) можно изменять, в случае необходимости, с помощью управляющих структур;
- управляющие структуры могут реализовать **точки ветвления** (если в какой-то момент по ходу решения задачи возможно его продолжение по более чем одному пути) и **циклы** (если какую-то линейную последовательность операторов требуется выполнить многократно, например при поэлементной обработке каких-то структурированных данных или порождая итерационный процесс).

Технология **структурного программирования** запрещает использовать в алгоритмах, программах такие анахронизмы, как операторы безусловного перехода (это оператор перехода по метке, **GOTO** в большинстве языков программирования высокого уровня). В то же время

она требует разрабатывать в форме отдельных **модулей** те элементарные алгоритмы, которые приходится многократно использовать в ходе решения всей задачи в целом.

Элементарные алгоритмы обычно оформляются в виде отдельных **подпрограмм** (функций на языке Си, функций и процедур на языке Паскаль). Затем подпрограммы объединяют в **модули**, которые называют библиотеками подпрограмм. При таком объединении обычно руководствуются следующими принципами: объединяемые в одну библиотеку подпрограммы либо позволяют решать схожие по содержание задачи, либо служат для обработки данных одного и того же типа.

Строгое следование вышеописанным принципам структурного программирования позволяет существенно сократить время разработки программ и многократно их использовать в целом ряде проектов. Следует отметить, что сегодня не только прикладное, но и системное программное обеспечение организовано по модульному принципу, т.е. в виде набора библиотек подпрограмм.

В последние десятилетия широкое применение находят технологии **объектно-ориентированного программирования** (**ООП**). Идеи, положенные в основу этих технологий, можно вкратце обобщить так: процессы, предметы, относящиеся к какой-то конкретной области человеческой деятельности, в программах должны быть представлены в виде абстракций, именуемых **объектами**. Эту абстракцию в программном коде воплотили, расширив понятие **структуры** (язык Си) или **записи** (язык Паскаль). Структуры или записи, как подмножество пользовательских типов данных, в рамках структурного программирования использовались для хранения агрегатов данных, связанных между собой по смыслу. Например, учетные данные о студенте: Фамилия, Имя, Отчество, год рождения и т.д. Они могли быть представлены в форме агрегатов из строк текста, целых и вещественных чисел. Переменные типа структура или запись позволяли выполнять в программе типовые операции над такими агрегатами, не рискуя нарушить целостность хранимых данных.

В технологиях ООП пользовательский тип данных (его назвали **классом**) расширили, допуская хранение в рамках это типа не только данных (они называются **свойствами объекта**), но и программного кода, подпрограмм, обрабатывающих эти данные (такие подпрограммы назвали **методами объекта**). При этом на этапе проектирования можно указать, какие методы и свойства объекта будут доступны из кода программы, а какие свойства и методы – только из кода подпрограмм-методов этого же объекта (последние как бы защищены от стороннего доступа).

Методы, доступные из кода программы, могут быть вызваны в программном коде методов других классов (объектов). Так абстрактно реализованные связи между объектами (их называют «посылкой объектами сообщений друг другу») отражают существующие связи между натурными прототипами этих объектов (процессами, предметами).

Включение в описание программного объекта не только данных, но и кода, называется **инкапсуляцией** (первое ключевое понятие технологии ООП).

Один класс как элементарный тип данных можно использовать в описании (определении) другого класса как одну из его составляющих. Тогда класс, включаемый как составляющяя, называют родительским классом по отношению к классу, его включающему. Так реализовано еще одно концептуальное новшество ООП – **наследование.**

Наконец, имеется возможность в рамках класса-наследника изменить программный код некоторых методов, унаследованных от класса-родителя. Так реализовано третье концептуальное новшество ООП – **полиморфизм.**

Для инициализации переменных любого из классов (такие инициализированные переменные обычно называют **экземплярами класса**) в числе методов класса должен быть указан метод-**конструктор** (его задача – инициализировать свойства порождаемого экземпляра класса). По окончании работы с экземпляром класса «правилом хорошего тона ООП» является вызов еще одного обязательного метода – **деструктора** (его роль подобна роли добросовестного мусорщика).

АЛГОРИТМЫ И ФОРМЫ ИХ ПРЕДСТАВЛЕНИЯ

Перед написанием программы требуется разработать алгоритм решения определенной задачи. **Алгоритм** – одно из основных понятий математики, которое не имеет строгого определения. Близкими по смыслу словами являются слова «рецепт», «метод», «программа». С понятием «алгоритм» тесно связано понятие «исполнитель» алгоритма. В качестве такового может выступать человек, робот, компьютер, станок и т.д. Исполнитель «знает» и «умеет» выполнять определенный набор команд, называемый **системой команд исполнителя – СКИ**. Исполнитель действует формально, в рамках своей системы команд. Дадим следующее определение: алгоритм – понятная и точная совокупность действий, предписанная исполнителю и направленная на достижение поставленной

цели. Другими словами, алгоритм – это набор инструкций, который описывает, как может быть выполнено некоторое задание.

Алгоритм состоит из трех частей: ввод исходных данных, обработка и вывод полученных результатов. К основным свойствам алгоритма относятся

- **понятность** для исполнителя;
- **дискретность** – представление процесса выполнения в виде простых ранее определенных шагов;
- **определенность** – каждый шаг алгоритма должен быть четким и однозначным;
- **выполнимость** – каждая инструкция содержит такие операции и в таком количестве, что их можно выполнить за конечное время;
- **результативность или конечность** – алгоритм должен приводить к решению задачи за конечное число шагов;
- **массовость** – алгоритм должен быть применим для некоторого класса задач, различающихся лишь исходными данными;
- **эффективность** – время работы алгоритма в определенном смысле должно быть минимальным.

Алгоритмы представляются в следующих видах:

- **словесном** – запись на естественном языке;
- **графическом** – изображение с помощью графических блоков и символов;
- **псевдокод** – полуформализованное описание алгоритма на условном языке, включающем как элементы языка программирования, так и фразы естественного языка;
- **программы** – тексты на языках программирования.

При **словесном способе** записи алгоритм задается в произвольном изложении на естественном языке. Рассмотрим описание типового алгоритма умножения двух целых положительных чисел A и N, в результате чего получается результат M. Предполагается, что исполнитель знает только действие сложение и умеет сравнивать два числа.

Шаг 1. Задать два целых положительных числа A и N,

Шаг 2. Присвоить ноль переменной-счетчику I.

Шаг 3. Присвоить ноль переменной накопления P.

Шаг 4. Если N равно нулю или A равно нулю, то перейти к шагу 8.

Шаг 5. Присвоить P значение суммы $P+A$.

Шаг 6. Присвоить I значение суммы $I+1$.

Шаг 7. Если счетчик I меньше числа N, то перейти к шагу 5.

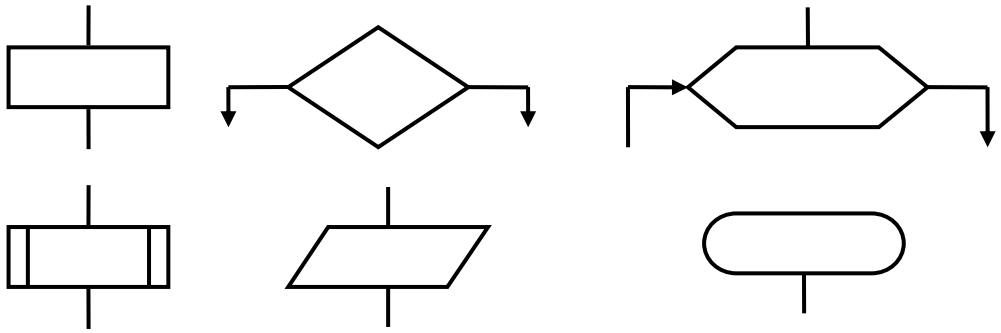
Шаг 8. Присвоить переменной M накопленное значение P.

Шаг 9. Конец.

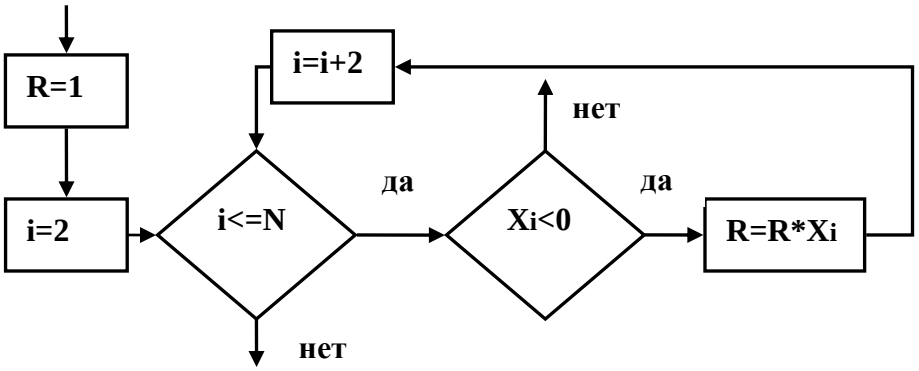
Приведенный алгоритм демонстрирует классический пример цикла: задание начальных значений – шаги 2,3; накопление – шаг 5; продвижение цикла – шаг 6; проверка условия окончания цикла – шаг 7.

При **графическом представлении** алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий. Такое представление называется **блок-схемой**. В блок-схеме каждому типу действий – вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий и т.п. – соответствует геометрическая фигура в виде блочного символа.

Приведем часто используемые блочные символы:



Так, блок «процесс», изображаемый прямоугольником, применяется для обозначения действия или последовательности действий, изменяющих значение. Блок «развилка», изображаемый ромбом, используется для обозначения переходов в алгоритме по условию. Блок «модификация» в виде растянутого шестиугольника используется для обозначения начала циклов. Блок «предопределенный процесс» с двойными боковыми линиями показывает вычисления по подпрограмме. Для обозначения ввода-вывода используется блок в виде параллелограмма. Последний из приведенных блоков применяется для обозначения начала и конца алгоритма. Блочные символы соединяются линиями переходов, определяющими очередность выполнения действий.



В качестве примера разберем блок-схему еще одного типового алгоритма. В переменной *R* накапливается произведение элементов массива *X*. Управляющая циклом переменная *i* принимает только четные значения: начинается с 2 и изменяется с шагом 2. Первое проверяемое условие (первый ромб) обеспечивает прекращение вычислений при превышении переменной *i* заданного количества элементов массива *N*, а второе условие обеспечивает выбор только отрицательных элементов. Таким образом, алгоритм представляет собой вычисление произведения отрицательных элементов одномерного массива с чётными номерами.

Псевдокод занимает промежуточное место между естественным и формальным языками. В псевдокод вводятся некоторые конструкции, присущие формальным языкам, а также **служебные слова**, смысл которых определен раз и навсегда. Рассмотрим пример алгоритма, написанного на псевдокоде:

Процедура ABCD;
начать
  писать (‘введите значение A, B, C, D’);
  читать (A, B, C, D);
  если A=B **то**
    если C<D **то** X:=1 **иначе** X:=2
  иначе X:=3
конец

Как аргументы задаются четыре значения, которые определяют величину переменной X. Первое условие проверяет равенство значений *A* и *B*. Одного этого равенства оказывается недостаточно для определения значения *X*, т.к. далее сравниваются между собой *C* и *D*: если *C*<*D*, то переменной *X* присваиваивается значение 1, в противном случае – 2. Наконец, *X* получает значение 3 в случае альтернативы первому условию, т.е. когда *A*≠*B* при любом соотношении между *C* и *D*. Таким образом, приведенный алгоритм реализует вычисление выражения вида

$$X= \begin{cases} \textbf{1, если A=B и C<D;} \\ \textbf{2, если A=B и C}\geq\textbf{D;} \\ \textbf{3, если A} \neq \textbf{B.} \end{cases}$$

Часто для краткости служебные слова сокращают. Вместо слова **конец** пишут **кон,** вместо **начало цикла** – **нц**; вместо **конец цикла** – **кц** и т. д.

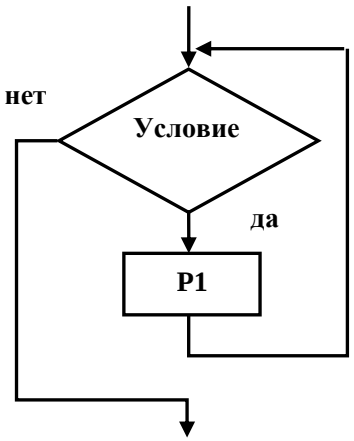
На практике в качестве исполнителей алгоритмов используются компьютеры. Алгоритмы, предназначенные для исполнения на компьютере, должны быть записаны на понятном ему языке. Такой язык называют **языком программирования**, а запись алгоритмов на этом языке – **программой** для компьютера. Приведем фрагмент программы на языке Паскаль:

```
X := 12;   Z := 7;
write ('X= ',  X=Z,  ' X= ',  X,  Z+X);
```

Здесь всего три оператора – два оператора присваивания переменным числовых значений и один оператор вывода. Выясним, что будет выведено на печать после выполнения компьютером этого фрагмента.

Сразу заметим, что оба выводимых строчных выражения (в апострофах) совершенно одинаковы – '*X*= '. Как первое значение выводится результат сравнения переменных *X* и *Z*, а т.к. они разные, то будет выведено логическое значение *FALSE*. После второго набора символов ('*X*= ') будут выведены два числовых значения, ничем не отделенные друг от друга – это значение *X* и величина суммы *Z*+*X*, т.е. числа 12 и 19. В результате на печать будет выведена строка: *X*= *FALSE X*= 1219.

Логическая структура любого алгоритма может быть представлена комбинацией четырех базовых структур: **следование** (линейная структура), **ветвление** (разветвляющаяся структура), **выбор**, **цикл** (циклическая структура с предусловием или постусловием)**.** Характерной особенностью базовых структур является наличие в них одного входа и одного выхода. Например, цикл **пока** с предусловием может быть представлен структурой, изображенной на рисунке.



Здесь сначала проверяется условие окончания цикла и только потом выполняется рабочая часть *P1*. Существует структура цикла **пока** с постусловием, когда условие проверяется после рабочей части *P1*, такой цикл всегда выполняется хотя бы один раз.

Вопросы

1. Чему будет равно значение d после выполнения алгоритма?

```
b:= 12     d:=46
нц пока d >= b
|          d:=d-b
кц
```

Комментарий. ► Здесь приведен цикл с предусловием, в теле которого переменная *d* каждый раз уменьшается на величину *b*, то есть переменная *d* последовательно принимает значения 34, 22, 10. Последнее значение преобразует условие окончания цикла – оно становится ложным, и цикл прекращается, при этом *d*=10.

2. Чему будет равно значение d после выполнения алгоритма?

```
k:=40
выбор
        при div (k, 12)=3:  d:=k;
        при mod (k, 12)<5:  d:=2;
        при mod (k, 12)>9:  d:=3;
        иначе  d:=1;
все
```

Комментарий. ► Правильный ответ: 40. Операция mod (*x, y*) – получение остатка целочисленного деления *x* на *y*, div (*x, y*) – целочисленное деление *x* на *y*. Первое же условие операции «выбор» истинно и значение *k* присваивается переменной *d*, т. е. она станет равной 40.

3. Если элементы массива D[1..5] равны соответственно 3, 4, 5, 1, 2, то чему равно значение выражения D[D[4]] – D[D[3]]?

Комментарий. ► Здесь индексами выступают сами элементы заданного массива. Так как D[4] = 1, а D[3]=5, то разность D[1] – D[5] = 3 – 2 = 1.

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Язык программирования – язык, позволяющий записать алгоритм решения интересующей нас задачи. В принципе, для этой цели годится и

любой естественный язык, но на языки программирования наложено одно существенное ограничение – записанный на этом языке текст должен легко преобразовываться в текст на внутреннем языке той машины, на которой он будет обработан. А это, кроме всего, означает, что язык реализован, если на компьютере установлена программа-переводчик с этого языка программирования в **коды компьютера – транслятор**. Транслятор обязательно должен входить в состав средств программирования высокого уровня. Современные трансляторы, это часто не только программа-переводчик, а довольно мощная среда, обеспечивающая весь цикл обработки программы, что было совсем не характерно для ситуации еще десятилетней давности. В то время программист должен был, например, сам позаботиться о связи своей программы с большим количеством вспомогательных программных продуктов.

Транслятор может работать в разных режимах – компиляции или интерпретации, и пользователь не имеет возможности выбрать тот или другой режим перевода. **Компиляция** – этап перевода модулей программы в машинные коды и сборка их в единое целое с привязкой к определенному адресному пространству. Создание исполняемого файла из исходного текста программы предполагает выполнение процессов компиляции и компоновки. При **интерпретации** – перевод программы осуществляется покомандно, переведенная команда выполняется, после этого перевод продолжается.

Существует несколько классификаций языков программирования. Одна из них определяет степень близости языка к машинным кодам. В этой классификации нижний уровень – коды машины, выше – **ассемблер** и т.д. **Ассемблер** – машинно-ориентированный язык. Чем ниже уровень языка, тем меньшая адресуемая единица памяти оказывается доступной (либо это будет бит для машинных кодов, либо идентификатор для языка высокого уровня), и тем большие возможности Вам предоставлены. **Системное программирование** требует доступа к минимальной адресуемой единице. Для него наиболее подходят языки C, C++ и Ассемблер. Однако, при всей прелести языков низкого уровня, композиция программы на языках низкого уровня просматривается с трудом, использование этих языков требует специальных навыков, требовать которых от специалистов в другой области неразумно. В соответствие с этим создано, и продолжает создаваться довольно большое количество языков программирования, классифицированных уже по степени их выразительности для описания разных классов задач – по степени их возможности выразить семантику описываемого объекта. Понятие семантики имеет строгое определение, но вот как звучит данное Дейкстрой не вполне строгое, но простое и красивое объяснение этого понятия:

«…если я знаю, что этот объект способен для меня сделать, значит, я знаю его семантику». Этим объяснением можно пользоваться и определяя семантику языка программирования – если я понимаю, что эта команда языка программирования способна сделать, я знаю ее семантику. В таком контексте синонимичность «семантика команды» и «смысл команды» допустима.

Вопросы

1. Система программирования предоставляет программисту возможность
1) автоматического построения математической модели, исходя из постановки задачи;
2) анализа существующих программных продуктов по соответствующей тематике;
3) автоматической сборки разработанных модулей в единый проект;
4) выбора языка программирования.
Комментарий. ► Система программирования (или языковая среда) дает возможность собрать в единый проект все разработанные модули.

2. Языками программирования являются:
1) C++; 2) MPI; 3) Pascal; 4) Far; 5) MATLAB; 6) MathCAD.
Комментарий. ► Среди перечисленных выше, собственно языками программирования являются C++ и Pascal, хотя математические пакеты (MATLAB и MathCAD) также обладают встроенными средствами создания программ.

3. Деятельность, направленная на исправление ошибок в программной системе называется…
1) отладка; 2) рефакторинг; 3) тестирование; 4) демонстрация.
Комментарий. ► Задачей **тестирования** является поиск ошибки, задачей **отладки** – проверка правильности взаимодействия программных модулей. И те, и другие действия направлены на исправление ошибок в программной системе.

КОМПЬЮТЕРНЫЕ СЕТИ

Сетевые информационные технологии востребованы в случаях, когда пользователю недостаточно ресурсов его машины, и требуется

получить эти ресурсы на другом, удаленном компьютере. В качестве таких ресурсов могут быть рассмотрены элементы аппаратного обеспечения (принтер, модем и т.д.), программного обеспечения (программы, которых нет у пользователя или которые невозможно выполнять на его машине) и, наконец, информация.

Общепринятая терминология: компьютер, получающий доступ к ресурсам другой машины, называют **клиентом** (включая и программы, с помощью которых осуществляется этот доступ), компьютер, предоставляющий свои ресурсы другим компьютерам сети, называют **сервером** (включая и программы, с помощью которых предоставляется подобная услуга). Кроме того, между сервером и его клиентами могут оказаться в качестве посредников еще целый ряд устройств, о них речь пойдет дальше.

Специалисты в области компьютерных информационных технологий разработали стандарт, согласно которому должны проектироваться и работать современные компьютерные сети. И хотя реализация сетей нового поколения, работающих по этим стандартам, дело будущего, по крайней мере, в нашей стране, имеет смысл познакомиться с ним. Тогда станет понятным общий принцип их построения и работы.

Международный стандарт взаимодействия компьютеров в сети определен как эталонная **семиуровневая модель** ISO/OSI. В качестве ответа на вопрос «Почему семь уровней, а не больше?», можно предложить такой вариант: традиционно число семь означало «очень много», вспомните русский фольклор – «Семь раз отмерь, один раз отрежь», «Волк и семеро козлят». Модель удобно представить себе с помощью следующей метафоры. Пусть имеются два пользователя: отправитель и адресат (получатель). Представим себе обоих сидящими на крышах разных семиэтажек. Семиэтажки – это метафора программно-аппаратных комплексов, которые в рамках отдельно взятого компьютера должны обеспечивать возможность его работы в сети. На каждом из этажей размещены абстрактные исполнители (программы либо технические средства, либо и то, и другое), которые могут получать задания только с вышестоящего этажа (от обитающих там исполнителей) и в свою очередь могут озадачить только нижестоящий этаж, точнее исполнителей, обитающих на нем.

Итак, пользователь-отправитель (клиент) подготовил **запрос** на получение файла от своего соседа с другой семиэтажки (сервера). Он опускает этот запрос на седьмой этаж своей семиэтажки и требует отправить его. На седьмом этаже принимают этот запрос и выясняют, кто из исполнителей-обитателей этого этажа возьмется за работу. Дело в том, что в зависимости от поставленной задачи она может быть решена

разными исполнителями. К примеру, либо исполнителем по доставке гипертекстовых документов, если требуется получить файл с сервера, хранящего такие документы, либо исполнителем по доставке почты, если необходимо получить почтовое сообщение. Взявшийся за выполнение этой работы исполнитель должным образом готовит запрос (скажем, снабдит его сопровождающей информацией), так, чтобы его коллега на седьмом этаже семиэтажки адресата (сервера) смог этот запрос впоследствии обработать. Правила, по которым они готовят запросы друг к другу, назовем **протоколом взаимодействия**. После этого с седьмого этажа запрос с сопровождающей информацией опускается на нижележащий (шестой), где его обитатели-исполнители в свою очередь добавят к нему свою порцию сопровождающих сведений, скажем, от кого исходит этот запрос. Запрос спускают на пятый этаж, где к нему прикладывают руку его обитатели.

Запрос, по мере опускания с верхних этажей на первый, обрастает сопроводительной, вспомогательной информацией, которая добавлена в соответствии с соглашениями о правилах общения исполнителей соответствующих этажей. Иными словами – согласно протоколам их общения, т.е. обитателей пятого этажа на стороне клиента с обитателями пятого этажа на стороне сервера, обитателей четвертого этажа на стороне клиента с обитателями четвертого этажа на стороне сервера, и т.д. По мере опускания запроса с верхних этажей на первый произойдут еще следующие важные преобразования: информация, которая составляет запрос, будет разрезана на отдельные фрагменты, пакеты, и к каждому такому пакету будет добавлена вспомогательная информация в виде различных контролек, бандеролей и, наконец, на каждом пакете будет указан адрес получателя. На первом этаже каждый из пакетов попадает в канал связи. По каналам связи пакеты могут перемещаться в обоих направлениях. Когда пакеты доберутся по каналу связи до первого этажа семиэтажки адресата, то будут взяты оттуда согласно указанному на них адресу. По контролькам на пакетах, обитающие на нижних этажах адресата исполнители проверят целостность полученных данных, соберут из пакета запрос, т.е. выполнят операции, обратные тем, что выполнялись на стороне клиента, и передадут на вышестоящие этажи. Если на шестом этаже адресата (сервера) обнаружат, что указанный клиент имеет право беспокоить сервер с подобным запросом, то этот запрос передают выше, на седьмой этаж. Там, в свою очередь, выясняют, кто из исполнителей готов его обслужить. Взявшийся за дело на седьмом этаже адресата исполнитель начнет готовить затребованный файл к отправке (если, конечно, этот файл на сервере имеется). Причем процедура подготовки ответа к отправке аналогична: отсылаемый ответ разбивается на пакеты, которые

снабжаются контрольками, адресом клиента, и пакеты опускаются на первый этаж, откуда попадают в канал связи (т.е. информация принимает форму совокупности сигналов, распространяемых по каналу связи), а по нему – на первый этаж семиэтажки клиента.

Из этого метафорического изложения должно быть понятно, что компьютерные сети – это сети с коммутацией пакетов. Ни одна из общающихся пар «сервер – клиент» не забирает в монопольное пользование канал связи, т.е. не лишает остальных возможности общаться друг с другом. Каналы связи используются совместно всеми общающимися, и практически одновременно. В этом причина огромной производительности сетей, их высокая пропускная способность. Общение обитателей-исполнителей соответствующих этажей на стороне клиента и сервера организовано по строго определенным правилам, которые называются **протоколами взаимодействия**. Причем у этих исполнителей «создается впечатление», что они общаются друг с другом напрямую, т.е. не замечают влияния на процесс общения нижележащих этажей. Обитатели-исполнители верхних этажей – это программы, чем ниже расположен этаж, тем больше работы берут на себя технические средства (контроллеры, адаптеры и пр.) и тем меньше остается на нем «работников»-программ.

Глобальная сеть Internet возникла задолго до появления вышеописанного эталона. Поэтому сеть Internet спроектирована скорее как «многоэтажка из четырех этажей» (некоторые этажи эталона «склеены»). Работает она на основе семейства протоколов TCP/IP. Это собирательное название для иерархически упорядоченного семейства программных средств, реализующих стандарты этого семейства протоколов. На самом верхнем уровне размещаются программные средства, реализующие протоколы так называемого прикладного уровня (пользователь работает непосредственно с ними) :

FTP – протокол для получения пользователем файлов с удаленного компьютера, на стороне клиента такая программа обычно называется FTP-клиентом, а на стороне сервера FTP-сервером;

HTTP – протокол для доступа к гипертекстовым документам, размещаемым на машинах в так называемой «всемирной паутине» (www-world wide web), на стороне клиента такая программа обычно называется HTTP-клиентом, а на стороне сервера HTTP-сервером;

SMTP, POP3 – протоколы для отсылки и получения почтовых сообщений, на стороне клиента такая программа обычно называется почтовым клиентом, на стороне сервера – сервером электронной почты;

NNTP – протокол для пересылки сетевых новостей;

Telnet – протокол, предоставляющий возможность клиенту пользоваться всем аппаратным и программным обеспечением сервера (как если бы пользователь работал непосредственно с клавиатурой сервера), на стороне клиента такая программа обычно называется Telnet-клиент, а на стороне сервера Telnet-сервер или Telnet-сервис.

Это далеко не полный перечень протоколов прикладного уровня (нами указаны протоколы, используемые чаще всего). Следующий уровень занимает протокол **TCP** и его аналоги. С их помощью решается задача формирования пакетов из информационного потока, поступающего сверху, и сборка в информационный поток пакетов, идущих снизу-вверх.

Еще ниже в иерархии расположен сетевой протокол **IP**, в задачи которого входит решать проблему адресации пакетов. Здесь уместно остановиться на особенностях систем адресации в сети Internet. Первоначально компьютеры, работающие в этой сети, адресовали путем их нумерации. Только эти номера выглядели как набор из четырех целых чисел, каждое из которых в диапазоне от 0 до 255, и эти четыре числа разделяли точками (127.0.0.1, 192.168.0.1 и т.п.). Но по мере роста количества компьютеров в сети выяснилось, что пользователям не очень удобно запоминать такие адреса. Решили перейти к иерархически упорядоченной системе символических имен, выделив компьютеры, имеющие отношение к коммерческим организациям в отдельное семейство (домен) com, далее в пределах этого семейства отдельному компьютеру присваивали уникальное (в этом семействе) имя, например google.com, microsoft.com и т.п. Либо в таком домене верхнего уровня формировали подмножества, домены второго уровня, именовали эти домены, а уже внутри них именовали уникальными именами отдельные компьютеры, например, Mycomp.bigdomain.com. По мере «расползания» сети с американского на другие континенты были созданы домены верхнего уровня по географическому принципу su, ru, by, ua. Однако и по сегодняшний день никто не гарантирует, что компьютер с именем mycomp из домена ru (mycomp.ru) размещен и работает на территории России. Используются доменные имена для формирования запроса на информационные ресурсы, хранящиеся на серверах сети Internet, следующим образом. Программе-клиенту нужно указать т.н. **URL** (Uniform Remote Locator), имеющий следующий формат:

имя_протокола_прикладного_уровня://имя_сервера.имя_домена_второго_уровня.имя_домена_верхнего_уровня/путь_к_ресурсу_в_пределах_сервера/собственно_имя_ресурса

К примеру

http://niceserver.nicedomain.info/public/nicefile.html

расшифровывается как попытка добраться к http-серверу, размещенному и работающему на компьютере niceserver в домене второго уровня nicedomain, который в свою очередь расположен в домене верхнего уровня info. На этом сервере нас интересует файл nicefile.html, хранящийся в каталоге public этого сервера. Далее HTTP-клиент (скажем, браузер Mozilla Firefox) инициирует рассылку по сети запроса к соседним машинам примерно следующего содержания: «Не знает ли кто IP-адрес сервера с именем niceserver.nicedomain.info?». Если кто-либо ответит на этот запрос положительно и укажет искомый IP-адрес, то по нему и начнется отсылка запроса на файл nicefile.html. Если же никто не ответит, то программа-клиент с прискорбием сообщит пользователю, что сервер niceserver.nicedomain.info не найден. В последнем случае услугу могут оказать т.н. **поисковые машины** (search engines). Это серверы с распределенными системами баз данных, хранящие информацию о наличии ресурсов в **WWW**-сети, на FTP-серверах. Наиболее популярные среди русскоязычных пользователей размещены на yandex.ru, rambler.ru.

На самом нижнем уровне находятся канальные протоколы, которые осуществляют управление пересылкой сформированных пакетов по каналам связи. Следует заметить, что существует достаточно много различных типов каналов связи. Это беспроводные **Wi-Fi**, **IrDA**, радиоканалы, проводные **Ethernet** (которая может быть построена как на основе витой пары, аналога телефонной линии, так и коаксиала), **FDDI** – оптоволоконные каналы связи, обладающие наряду со спутниковыми каналами наименьшими временами задержки потока информации. При значительной удаленности клиента и сервера в канале связи между ними размещают дополнительные устройства – **повторители, концентраторы**, которые служат для усиления сигнала и обеспечения целостности передаваемой информации, если передающая сигнал сторона не в состоянии это обеспечить. **Топология сети**, т.е. принцип или способ, руководствуясь которым проложили каналы связи между компьютерами этой сети, также влияет на выбор протоколов канального уровня, поэтому они, как правило, реализованы с помощью технических средств (т.е. как элементы аппаратного, а не программного обеспечения).

В заключение, нужно сказать несколько слов о проблеме безопасности при работе в сети. **Абсолютную защиту своего компьютера от атак со стороны злоумышленников, обитающих в сети, вы можете обеспечить, только отключив его от сети**. Это крайность. Другая крайность – это предоставление полного доступа к вашей машине (по протоколу Telnet). Ни то, ни другое рядовому пользователю компьютерных сетей не нужно. По большей части опасность заразить машину вирусами или поселить в ОС своей машины сетевых червей – это бездумно получать электронные

письма и столь же бездумно открывать вложенные в них файлы. Какие вложения можно получить в электронном письме? Ответ: любые наборы данных (файлы). В электронное письмо можно вкладывать текстовые документы, рисунки, видео ролики, а также программы, маскируя их, например, под файлы с фотографиями.

Второй по степени опасности представляется просмотр гипертекстовых документов, в безопасности которых вы не уверены. Сам по себе гипертекстовый документ подготовлен на языке **HTML**, который не предоставляет серьезных лазеек для злоумышленников. Но современные гипертекстовые документы снабжаются еще и программами на языке **JavaScript** или **VBS**, которые могут таить в себе опасные фрагменты кода, приводящие к заражению компьютера.

ЗАЩИТА ИНФОРМАЦИИ

Понятие **защиты информации** в сети, защиты элементов самой сети, изначально связано с понятием удаленной атаки (мы не будем обсуждать грубого механического разрушения носителей информации). История удаленных атак на узлы глобальной компьютерной сети началась 2 ноября 1988 года, когда выпускник Корнельского университета Роберт Таппан Моррис запустил свою программу, которая вышла из-под контроля автора и начала быстро перемещаться по сети. В короткий срок вирус-червь заполнил многие узлы Internet, загружая операционные системы своими копиями, вызывая отказы в обслуживании. В результате этого инцидента мир получил представление о возможности нарушения безопасности компьютерной сети Internet. **Сетевые черви** – так на программистском сленге называют программы, копирующие сами себя в сети («расползающиеся» по сети). Они проникают в операционную среду и пользовательские файлы пораженного компьютера; могут переноситься на внешних (переносимых) хранилищах информации, очень часто – в виде вложенных в письмо файлов. Наиболее уязвимая среда (как это ни обидно, хотя и легко объяснимо широкой распространенностью) – Windows. Эффективные способы борьбы – использование **антивирусных программ** (DrWeb, продукты лаборатории Касперского, Norton Antivirus и т.п.), регулярное копирование наиболее важной информации (**Backup**).

Атаки классифицируются в зависимости от цели, которую преследует хакер, способа исполнения атаки, характера воздействия на объект атаки. В результате успешной атаки может быть нарушена целостность, или полностью разрушена информация пользователя

компьютера, разрушена операционная среда компьютера, искажено адресное пространство сети, прерван диалог хостов сети, подменены субъекты диалога, «подслушана» передаваемая по сети информация.

В соответствии с такой классификацией создаются, или подбираются, подходящие средства защиты. Распространенными инструментами, позволяющим «спрятаться», или скрыть свой локальный сегмент сети являются **межсетевые экраны** и серверы, работающие в режиме **Firewall** (**Брандмауэр**) – «огненной стены». Еще один, достаточно широко распространенный способ защиты – использование **электронной цифровой подписи**, позволяющей удостовериться, что сообщение получено от известного участнику диалога абонента. Но гарантируется (с некоторой долей неуверенности!) только этот факт. Подпись говорит об истинности отправителя документа, а, следовательно, о режиме доступа к нему. И только косвенно – о подлинности документа. Для создания электронно-цифровой подписи используется сам текст сообщения, но обработанный специальным образом, и сгенерированный по этому тексту ключ, передаваемый получателю сообщения. Если ключ позволяет обратное преобразование текста к исходному виду, считается, что отправитель идентифицирован. Более надежный способ передать информацию без искажения и не позволить прочитать ее непосвященным – применить один из способов **шифрования**. Следует заметить, что почти все способы защиты носят либо «запретительный» характер, что привносит некоторый дискомфорт при работе в сети, либо пассивны – дают лишь оценку уже имевшей место ситуации после ее возникновения. Активные средства, позволяющие обнаружить и пресечь атаку еще на этапе ее исполнения, крайне редки. Достаточно эффективным средством защиты от целого класса атак – вирусов, можно считать широко известные **антивирусные программы.**

Вместе с тем, следует признать, что абсолютно **надежных средств защиты** компьютера, интегрированного в сеть и информации, в нем хранящейся, **нет** – в этой борьбе щита и меча, меч всегда начинает первым. Основным средством **антивирусной защиты** является периодическая проверка компьютера с помощью антивирусного программного обеспечения.

Вопросы

*1. В чем принципиальное отличие **межсетевых экранов** от **систем обнаружения атак**?*

Комментарий. ► Задача межсетевого экрана – не допустить атаку. Часто это достигается сокрытием адресного пространства защищаемого

фрагмента сети и, даже, – своего собственного адреса. Межсетевой экран может иметь или не иметь средства обнаружения атаки. Системы обнаружения атак лишь регистрируют этот факт, причем не всегда в момент исполнения атаки.

2. Программными средствами для защиты информации в компьютерной сети из списка являются:
1) Firewall;
2) Brandmauer;
3) Sniffer;
4) Backup.
Комментарий. ► Это проверка знания программистского сленга, или, скорее – сленга системного администратора. **Брандмауэр** (**Firewall**) – непреодолимая защита, огненная стена. **Sniffer** – инструмент, позволяющий обнаружить факт сканирования портов защищаемого сервера, т.е. факт возможной подготовки удаленной атаки. К сожалению, этот инструмент также используется для сканирования портов и подготовки к проведению самой атаки. **Backup** – периодическое копирование наиболее важной информации. Это мера, скорее организационная. Итак, среди приведенных вариантов следует выбрать первые два.

ЛИТЕРАТУРА

1. **Михеева Е.В.** Практикум по информатике: учебное пособие.— М.: «Академия», 2004.— 192 с.
2. **Симонович С.В.** Информатика. Базовый курс: учебное пособие для втузов / под ред. С.В. Симоновича.— 2-е изд.— М. [и др.]: Питер, 2007.— 639 с.
3. **Шелест В.Д.** Программирование: учебное пособие.— СПб.: БХВ, 2002.— 584 с.

Предметный указатель

Содержание