Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Пермский национальный исследовательский
политехнический университет»

И.Н. Щапова, В.А. Щапов

ИНФОРМАТИКА

Утверждено
Редакционно-издательским советом университета
в качестве учебного пособия

Издательство
Пермского национального исследовательского
политехнического университета
2016

УДК 004(075.8)
Щ23

Рецензенты:
канд. техн. наук, проф. *Р.А. Сажин*
(Пермский национальный исследовательский
политехнический университет);
канд. техн. наук *И.Я. Сальников*
(ЗАО «Энергосервис», г. Пермь)

Щапова, И.Н.

Щ23 Информатика: учеб. пособие / И.Н. Щапова, В.А. Щапов – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2016. – 154 с.

ISBN 978-5-398-01556-0

Рассмотрены основные вопросы информатики – технические и программные средства реализации информационных процессов, технологии компьютерных сетей, алгоритмизация и программирование на языке PascalABC.NET, технологии программирования.

Предназначено для студентов, изучающих дисциплину «Информатика» в рамках основной образовательной программы подготовки бакалавров и специалистов по направлениям ВПО в ПНИПУ.

Главы 1–11 учебного пособия – для студентов, обучающихся по направлениям подготовки бакалавров и специалистов, для которых предусмотрена трудоемкость дисциплины «Информатика» – 4/5 ЗЕ (144/180 часов).

Учебное пособие в полном объеме – для студентов, обучающихся по направлениям подготовки бакалавров и специалистов, для которых предусмотрена трудоемкость дисциплины «Информатика» – 7 ЗЕ (252 часа).

УДК 004(075.8)

ISBN 978-5-398-01556-0

ОГЛАВЛЕНИЕ

1. Основные понятия теории информации 5
1.1. Информатика: предмет и задачи 5
1.2. Понятие информации. Свойства информации 6
1.3. Данные. Операции с данными 7
1.4. Основные структуры данных 8
1.5. Системы счисления 8
1.6. Кодирование данных двоичным кодом 9
2. Технические средства реализации информационных процессов 14
2.1. История развития ЭВМ. Поколения ЭВМ 14
2.2. Классификация компьютеров 19
2.3. Состав вычислительной системы 20
2.4. Базовая аппаратная конфигурация персонального компьютера 23
2.5. Внутренние устройства системного блока 23
2.6. Системы, расположенные на материнской плате 27
2.7. Периферийные устройства персонального компьютера 31
3. Программные средства реализации информационных процессов 32
3.1. Программное обеспечение. Классификация программного обеспечения компьютера 32
3.2. Классификация операционных систем 36
3.3. Функции операционных систем персональных компьютеров 40
4. Текстовый процессор 46
5. Электронные таблицы 46
6. Алгоритмы и алгоритмизация 47
6.1. Алгоритм и его свойства 47
6.2. Основные структуры алгоритмов 48

6.3. Алгоритмы линейной, разветвляющейся и циклической структуры ... 50
6.4. Параллельные алгоритмы ... 56
7. Программные средства реализации алгоритмов ... 59
7.1. Языки программирования ... 59
7.2. Программирование на языке Pascal ... 62
8. Пакеты прикладных программ ... 82
8.1. Математический пакет Mathcad ... 82
8.2. Графические пакеты прикладных программ ... 89
9. Базы данных ... 89
10. Телекоммуникации. Локальные и глобальные компьютерные сети ... 90
10.1. Основные понятия компьютерных сетей ... 90
10.2. Сетевые протоколы ... 93
10.3. Основные службы Интернета ... 95
11. Методы и средства защиты информации ... 100
11.1. Вопросы компьютерной безопасности ... 100
11.2. Защита информации в Интернете. Понятие о шифровании данных ... 103
Вопросы по дисциплине «Информатика» (главы 1–11) ... 104
12. Технологии программирования. Структурное и объектно-ориентированное программирование ... 106
12.1. Жизненный цикл программного обеспечения ... 106
12.2. Структурное программирование ... 109
12.3. Объектно-ориентированное программирование ... 136
13. Современные информационные технологии ... 138
Вопросы по дисциплине «Информатика» (главы 12–13) ... 139
Список литературы ... 140
Приложение 1. Контрольная работа для студентов заочного отделения ... 141
Приложение 2. Требования к оформлению контрольной работы ... 150
Приложение 3. Образец оформления титульного листа ... 151
Приложение 4. Единый формат оформления библиографических списков ... 152

1. ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ ИНФОРМАЦИИ

1.1. Информатика: предмет и задачи

Информатика – это техническая наука, систематизирующая приемы создания, хранения, воспроизведения, обработки и передачи данных средствами вычислительной техники, а также принципы функционирования этих средств и методы управления ими [1].

Предмет информатики составляют:

– аппаратное обеспечение средств вычислительной техники;

– программное обеспечение средств вычислительной техники;

– средства взаимодействия аппаратного и программного обеспечения (аппаратно-программный интерфейс);

– средства взаимодействия человека с аппаратными и программными средствами (пользовательский интерфейс).

Особое внимание в информатике уделяется вопросам взаимодействия – *интерфейсам*. Кроме указанных выше, есть еще аппаратные и программные интерфейсы.

Основной задачей информатики является систематизация приемов и методов работы с аппаратными и программными средствами вычислительной техники.

Основная задача информатики включает следующие направления для практических приложений:

– архитектура вычислительных систем (приемы и методы построения систем, предназначенных для обработки данных);

– интерфейсы вычислительных систем (приемы и методы управления аппаратным и программным обеспечением);

– программирование (приемы, методы и средства разработки компьютерных программ);

– преобразование данных (приемы и методы преобразования структур данных);

– защита информации (обобщение приемов, разработка методов и средств защиты данных);

– автоматизация (функционирование программно-аппаратных средств без участия человека);

– стандартизация (обеспечение совместимости между аппаратными и программными средствами, а также форматами представления данных, относящихся к различным типам вычислительных систем).

1.2. Понятие информации. Свойства информации

Информация – это продукт взаимодействия данных и адекватных им методов.

Для определения количества информации в случае различных вероятностей событий используется формула, предложенная в 1948 г. американским инженером и математиком К. Шенноном, в виде

$$I = -\sum_{i=1}^{N} p_i \log_2 p_i ,$$

где I – количество информации; N – количество возможных событий; p_i – вероятность i-го события.

С точки зрения информатики наиболее важными являются следующие свойства информации:

– *объективность* (более объективной принято считать ту информацию, в которую методы вносят меньший субъективный элемент);

– *полнота* (означает, что информация содержит минимальный, но достаточный для принятия правильного решения набор показателей);

– *достоверность*;

– *адекватность* (определенный уровень соответствия создаваемого с помощью полученной информации образа реальному объекту, процессу, явлению и т.п.);

– *доступность*;

– *актуальность* (степень соответствия информации текущему моменту времени).

1.3. Данные. Операции с данными

Данные – составная часть информации, представляющая собой зарегистрированные сигналы. При этом физический метод регистрации может быть любым (изменение электрических, магнитных, оптических характеристик). В соответствии с методом регистрации данные могут храниться и транспортироваться на носителях различных видов (бумага, магнитные и оптические диски и т.п.) [1].

Характеристиками носителей являются параметр *разрешающей способности* (количество данных, записанных в принятой для носителя единице измерения) и *динамический диапазон* (логарифмическое отношение интенсивности амплитуд максимального и минимального регистрируемых сигналов). От этих свойств носителя зависят такие свойства информации, как полнота, доступность и достоверность.

В ходе информационного процесса данные преобразуются из одного вида в другой с помощью методов. Обработка данных включает в себя следующие операции:

– *сбор* данных;

– *формализация* данных (приведение данных, поступающих из разных источников, к одинаковой форме, чтобы сделать их сопоставимыми между собой, т.е. повысить уровень их доступности);

– *фильтрация* данных (отсеивание данных, в которых нет необходимости для принятия решения);

– *сортировка* данных (упорядочение данных по заданному признаку для удобства их использования);

– *архивация* данных (организация хранения данных в удобной и доступной форме);

– *защита* данных;

– *транспортировка* данных;

– *преобразование* данных (перевод данных из одной формы в другую или из одной структуры в другую).

1.4. Основные структуры данных

Работа с большими наборами данных автоматизируется проще, когда данные упорядочены, т.е. образуют заданную структуру. Существует три основных типа структур данных:

– *линейная* – список данных – это упорядоченная структура, в которой адрес элемента однозначно определяется его номером;

– *табличная* – матрица – это упорядоченная структура, в которой адрес элемента определяется номером строки и номером столбца, на пересечении которых находится ячейка, содержащая искомый элемент;

– *иерархическая* – это структура, в которой адрес каждого элемента определяется путем доступа (маршрутом), ведущим от вершины структуры к данному элементу.

В качестве единицы хранения данных принят объект переменной длины, называемый *файлом*. Файл – это последовательность произвольного числа байтов, обладающая уникальным собственным именем (тип данных определяет тип файла). Совокупность файлов образует *файловую структуру*, которая, как правило, относится к иерархическому типу.

1.5. Системы счисления

Система счисления – способ наименования и обозначения чисел.

Различают два типа систем счисления: *непозиционные* (римская система счисления) и *позиционные* (десятичная, двоичная, восьмеричная, шестнадцатеричная системы счисления). В позиционных системах счисления один и тот же числовой знак (цифра) в записи числа имеет различные значения в зависимости от того места (разряда), где он расположен.

Для записи числа в позиционной системе счисления с основанием p используется p различных символов, которые обозначают числа от 0 до p–1 включительно. Два соседних разряда отличаются друг от друга в p раз. Так, в восьмеричной системе счисления используются цифры 0–7, в шестнадцатеричной – цифры 0–9 и латинские буквы *A*, *B*, *C*, *D*, *E*, *F*.

1.6. Кодирование данных двоичным кодом

Для автоматизации работы с данными, относящимися к различным типам, необходимо унифицировать их форму представления. С этой целью используется прием кодирования, т.е. выражение данных одного типа через данные другого типа [1].

Двоичное кодирование, используемое в вычислительной технике, основано на представлении данных последовательностью всего двух знаков: 0 и 1. Эти знаки называются *двоичными цифрами*, по-английски – *binary digit* или сокращенно bit (бит).

Одним битом могут быть выражены два понятия: 0 и 1 (да или нет, истина или ложь и т.п.). Если количество битов увеличить до двух, то уже можно выразить четыре различных понятия:

00 01 10 11

Тремя битами можно закодировать восемь различных значений:

000 001 010 011 100 101 110 111

Увеличивая на единицу количество разрядов в системе двоичного кодирования, мы увеличиваем в два раза количество значений, которое может быть выражено в данной системе, т.е. общая формула имеет вид

$$N = 2^m,$$

где N – количество независимых кодируемых значений; m – разрядность двоичного кодирования, принятая в данной системе.

Наименьшей единицей измерения данных является *байт* – минимально адресуемая последовательность фиксированного числа битов. Производные единицы измерения данных:

1 килобайт (Кбайт) = 1024 байт = 2^{10} байт;
1 мегабайт (Мбайт) = 1024 Кбайт = 2^{20} байт;
1 гигабайт (Гбайт) = 1024 Мбайт = 2^{30} байт;
1 терабайт (Тбайт) = 1024 Гбайт = 2^{40} байт;
1 петабайт (Пбайт) = 1024 Тбайт = 2^{50} байт.

Кодирование числовых данных

Кодирование целых чисел осуществляется следующим образом: число делится пополам до тех пор, пока в остатке не образуется ноль или единица. Совокупность частного и всех остатков от каждого деления, записанная в обратном порядке, и образует двоичный аналог десятичного числа.

Пример. Перевести $14_{10} \rightarrow X_2$.

14 : 2 = 7 (остаток **0**), 7 : 2 = 3 (остаток **1**), 3 : 2 = **1** (остаток **1**).

Таким образом, $14_{10} \rightarrow 1110_2$.

Для перевода из двоичной системы счисления в десятичную нужно представить число в виде многочлена и вычислить его значение.

Пример. Перевести $1110_2 \rightarrow X_{10}$.

$1110_2 \rightarrow 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 4 + 2 = 14$.

Таким образом, $1110_2 \rightarrow 14_{10}$.

8 разрядов двоичного кода (8 бит) позволяют закодировать целые числа от 0 до 255; 16 бит – целые числа от 0 до 65535, а 24 бита – более 16,5 миллионов различных значений.

Для кодирования действительных чисел используют 64-разрядное кодирование. При этом число предварительно преобразуется в нормализованную форму: $1{,}2534671 = 0{,}12534671 \cdot 10^1$.

Первая часть числа называется мантиссой, а вторая – характеристикой.

Кодирование текстовых данных

При кодировании текстовых данных каждому символу алфавита сопоставляется определенное целое число (например, порядковый номер). Восьми двоичных разрядов достаточно для

кодирования 256 различных символов. Для того чтобы все одинаково кодировали текстовые данные, нужны единые таблицы кодирования [1].

В 1963 г. была разработана и стандартизована в США система кодирования ASCII (American Standard Code for Information Interchange – *стандартный код информационного обмена США*). В системе ASCII закреплены две таблицы кодирования – *базовая* и *расширенная*.

Базовая таблица закрепляет значения кодов от 0 до 127, а расширенная относится к символам с номерами от 128 до 255.

Коды базовой таблицы с 0 по 31 – это так называемые *управляющие коды*, которым не соответствуют никакие символы. Они отданы производителям аппаратных средств (в первую очередь производителям компьютеров и печатающих устройств).

Символы английского алфавита, знаков препинания, цифр, арифметических действий и некоторых вспомогательных символов размещены, начиная с кода 32 по код 127.

Расширенная часть системы кодирования, определяющая значения кодов со 128 по 255, – это национальная система кодирования. Отсутствие единого стандарта в этой области привело к множественности одновременно действующих кодировок. Только в России можно указать три действующих стандарта кодировки: windows-1251, КОИ-8 (код обмена информацией, восьмизначный), ISO.

Разнообразие кодировок вызвано ограниченным набором кодов (256) при восьмиразрядном кодировании. Универсальная система кодирования UNICODE, основанная на 16-разрядном кодировании символов, позволяет обеспечить уникальные коды для 65 536 различных символов – этого достаточно для размещения в одной таблице символов большинства языков планеты. В системе кодирования UNICODE все текстовые документы автоматически становятся вдвое длиннее по сравнению с 8-разрядным кодированием.

Коды в стандарте UNICODE разделены на несколько областей. Область с кодами от U+0000 до U+007F содержит символы набора ASCII с соответствующими кодами. Далее расположены

области знаков различных письменностей, знаки пунктуации и технические символы. Часть кодов зарезервирована для использования в будущем.

В настоящий момент идет массовый переход к использованию семейства кодировок стандарта UNICODE, а однобайтовые кодировки, такие как windows-1251, поддерживаются в основном для совместимости с документами, созданными ранее.

Кодирование графических данных

В зависимости от способа формирования изображений компьютерную графику подразделяют:

– на *растровую* (графический объект представлен в виде комбинации точек, образующих растр и обладающих свойствами яркости и цвета);

– *векторную* (элементарным объектом является не точка, а линия);

– *фрактальную* (базовым элементом является математическая формула).

Трехмерная (*3D*) *графика* сочетает в себе векторный и растровый способ формирования изображений.

Вследствие того, что линейные координаты и индивидуальные свойства каждой точки (яркость) можно выразить с помощью целых чисел, растровое кодирование позволяет использовать двоичный код для представления графических данных. Кодирование черно-белых изображений осуществляется восьмиразрядным кодированием, что позволяет отобразить 256 оттенков серого цвета.

Для кодирования цветных графических изображений применяется *принцип декомпозиции* (разложения) произвольного цвета на основные составляющие. В качестве таких составляющих используют три основных цвета: красный (Red, R), зеленый (Green, G) и синий (Blue, B). Такая система кодирования называется системой RGB. Если для кодирования яркости каждой из основных составляющих использовать по 256 значений (8 двоичных разрядов), то на кодирование цвета одной точки надо затратить 24 разряда, что обеспечивает определение 16,5 млн различных цветов. Такой режим называется *полноцветным* (*True Color*).

Каждому из основных цветов можно поставить в соответствие дополнительный цвет, т.е. цвет, дополняющий основной цвет до белого. Для любого из основных цветов дополнительным будет цвет, образованный суммой пары остальных основных цветов. Соответственно, дополнительными цветами являются: голубой (Cyan, C), пурпурный (Magenta, M) и желтый (Yellow, Y). Принцип декомпозиции произвольного цвета на составляющие компоненты применяется не только для основных цветов, но и для дополнительных, т.е. любой цвет можно представить в виде суммы голубой, пурпурной и желтой составляющей. Такой метод кодирования цвета с использованием четвертой краски – черной (Black, K) – принят в полиграфии. Данная система кодирования обозначается четырьмя буквами CMYK и для представления цветной графики в этой системе используется 32 двоичных разряда. Такой режим тоже называется *полноцветным* (*True Color*).

Кодирование звуковой информации

В кодировании звуковой информации можно выделить два основных направления.

Метод FM (Frequency Modulation) основан на том, что теоретически любой сложный звук можно разложить на последовательность простейших гармонических сигналов разных частот, каждый из которых представляет собой правильную синусоиду и, следовательно, может быть описан числовыми параметрами, т.е. кодом. Разложение звуковых сигналов в гармонические ряды и представление их в виде дискретных цифровых сигналов выполняют специальные устройства – аналого-цифровые преобразователи (АЦП). Обратное преобразование для воспроизведения звука, закодированного числовым кодом, выполняют цифро-аналоговые преобразователи (ЦАП). При таких преобразованиях неизбежны потери информации, поэтому качество звукозаписи не вполне удовлетворительное. В то же время данный метод кодирования обеспечивает весьма компактный код.

Метод таблично-волнового (Wave-Table) *синтеза* лучше соответствует современному уровню развития техники. Если говорить упрощенно, то можно сказать, что где-то в заранее подготовленных таблицах хранятся образцы звуков (сэмплы) для множества различных музыкальных инструментов. Числовые коды выражают тип инструмента, номер его модели, высоту тона, продолжительность и интенсивность звука, динамику его изменения и др. Звук, полученный в результате синтеза, получается достаточно высокого качества.

2. ТЕХНИЧЕСКИЕ СРЕДСТВА РЕАЛИЗАЦИИ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ

2.1. История развития ЭВМ. Поколения ЭВМ

Более трех тысяч лет назад в Средиземноморье было распространено простейшее приспособление для счета: доска, разделенная на полосы, где перемещались камешки или кости. Такая счетная дощечка называлась абак и использовалась для ручного счета. Абак позволял лишь запоминать результат, а все арифметические действия должен был выполнять человек.

Первая механическая машина была построена немецким ученым Вильгельмом Шиккардом (предположительно в 1623 г.). Машина была реализована в единственном экземпляре и предназначалась для выполнения арифметических операций. Из-за недостаточной известности машины Шиккарда более 300 лет считалось, что первую суммирующую машину сконструировал французский математик и физик Блез Паскаль.

Блез Паскаль в 1642 г. изобрел механическую счетную машину, выполнявшую сложение, а в 1673 г. немецкий философ, математик и физик Готфрид Лейбниц расширил возможности машины Паскаля, добавив операции умножения, деления и извлечения квадратного корня. Специально для своей машины Лейбниц применил систему счисления, использующую вместо привычных для человека десяти цифр две: 1 и 0. Двоичная система счисления широко используется в современных ЭВМ.

Ни одна из этих машин не была автоматической и требовала непрерывного вмешательства человека. В 1834 г. английский математик и изобретатель Чарльз Бэббидж первым разработал подробный проект автоматической вычислительной машины. Он так и не построил свою машину, так как в то время невозможно было достичь требуемой точности изготовления ее узлов. Особенностью его машины стало то, что здесь впервые был реализован принцип разделения информации на команды и данные. И сегодня в вычислительной технике принцип раздельного рассмотрения программ и данных учитывается и при разработке архитектур современных компьютеров, и при разработке компьютерных программ.

Ч. Бэббидж выделял в своей машине следующие составные части:

– «склад» для хранения чисел (в современной терминологии – память);

– «мельницу» для производства арифметических действий (арифметическое устройство, процессор);

– устройство, управляющее последовательностью выполнения операций (устройство управления);

– устройства ввода и вывода данных.

В качестве источника энергии для приведения в действие механизмов машины Ч. Бэббидж предполагал использовать паровой двигатель. Бэббидж предложил управлять своей машиной с помощью перфорированных карт, содержащих коды команд, подобно тому как использовались перфокарты в ткацких станках Жаккара. На этих картах было представлено то, что сегодня мы назвали бы программой.

Ч. Бэббидж довольно подробно рассматривал вопросы, связанные с программированием. В частности, им была разработана весьма важная для программирования идея «условной передачи управления». Идеи Бэббиджа заложили фундамент, на котором со временем были построены ЭВМ. Первые программы для вычислительной машины Бэббиджа создавала Ада Лавлейс – дочь известного поэта Джорджа Байрона, в честь которой впоследствии был назван один из языков программирования.

Теоретические основы современных цифровых вычислительных машин заложил английский математик Джордж Буль (1815–1864). Он разработал алгебру логики, ввел в обиход логические операторы И, ИЛИ и НЕ.

В 1888 г. американским инженером немецкого происхождения Германом Холлеритом была сконструирована первая электромеханическая машина для сортировки и подсчета перфокарт. Эта машина, названная табулятором, содержала реле, счетчики, сортировочный ящик. Изобретение Холлерита было использовано при подведении итогов переписи населения в США. Успех вычислительных машин с перфокартами был феноменален. То, чем за десять лет до этого занимались 500 сотрудников в течение семи лет, Холлерит сделал с 43 помощниками на 43 вычислительных машинах за 4 недели. В 1896 г. Герман Холлерит основал фирму Computing Tabulation Company. Спустя несколько лет это предприятие переименовали в известнейшую сейчас фирму International Business Machine Corporation (IBM).

Первая электронная вычислительная машина ENIAC была разработана в США в 1943–1946 гг. Она состояла из 18 тысяч электронных ламп, 1,5 тысячи реле, имела вес более 30 тонн, потребляла мощность более 150 кВт.

Первоначально ENIAC программировалась путем соединения проводами соответствующих гнезд на коммутационной панели, что делало составление программы очень медленным и утомительным занятием.

Американский математик и физик венгерского происхождения Джон фон Нейман (1903–1957) предложил хранить программу – последовательность команд управления ЭВМ – в памяти ЭВМ, что позволяло оперировать с программой так же, как с данными.

Последующие ЭВМ строились с большим объемом памяти, с учетом того, что там будет храниться программа. В докладе фон Неймана, посвященном описанию ЭВМ, выделено пять базовых элементов компьютера:

– арифметико-логическое устройство (АЛУ);
– устройство управления (УУ);
– запоминающее устройство (ЗУ);
– система ввода информации;
– система вывода информации.

Описанную структуру ЭВМ принято называть архитектурой фон Неймана (рис. 1).

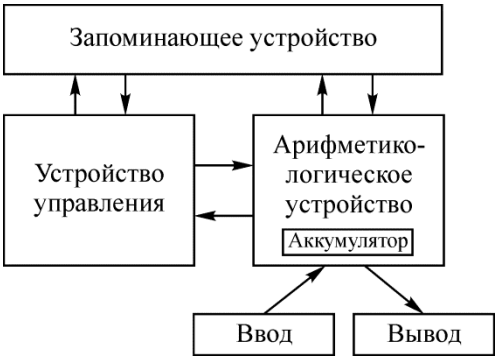


Рис. 1. Архитектура фон Неймана

Поколения ЭВМ

ЭВМ первого поколения в качестве элементной базы использовали электронные лампы и реле, оперативная память – на электронно-лучевых трубках и ферритовых сердечниках, быстродействие до 20000 оп/с, однопрограммность.

Изобретение в 1948 г. транзисторов и запоминающих устройств на магнитных сердечниках оказало глубокое воздействие на вычислительную технику. Ненадежные вакуумные лампы, которые требовали большой мощности для нагревания катода, заменялись небольшими германиевыми (впоследствии кремниевыми) транзисторами.

ЭВМ второго поколения (50-е гг. и начало 60-х гг.) в качестве элементной базы использовали полупроводниковые транзисторы, быстродействие 10^4–10^5 оп/с. Благодаря транзисторам

и печатным платам было достигнуто значительное уменьшение размеров и объемов потребляемой энергии, а также повышение надежности. Программирование велось на алгоритмических языках Фортран, Алгол, Кобол.

В 1958 г. американский инженер Д. Килби разработал первую интегральную микросхему.

ЭВМ третьего поколения (с середины 60-х гг.) – основу их элементной базы составляли микросхемы малой и средней степени интеграции – интегральные схемы (ИС), быстродействие 10^6–10^7 оп/с. Были снижены габариты и энергопотребление ЭВМ. Оперативная память строилась на ИС и достигала объема 10^5–10^6 байт. Появился широкий выбор языков программирования. Стали использоваться операционные системы, позволяющие резко повысить производительность и организовать многопрограммный и терминальный режимы

В 1971 г. американский инженер Маршиан Эдвард Хофф объединил основные элементы компьютера в один небольшой кремниевый чип (кристалл), который он назвал микропроцессором. Первый микропроцессор получил маркировку Intel 4004.

ЭВМ четвертого поколения строятся на интегральных микросхемах с большой степенью интеграции. На одном кристалле размещается целая микроЭВМ. Переход от третьего поколения ЭВМ к четвертому не был революционным. Отличия коснулись не столько принципов построения ЭВМ, сколько плотности упаковки элементов в микросхемах.

Элементная база ЭВМ четвертого поколения – большие и сверхбольшие ИС (БИС и СБИС). Быстродействие 10^7–10^8 оп/с. Формируются два направления – многопроцессорные и персональные ЭВМ. Появляются компьютерные сети. Разрабатывается специализированное программное обеспечение, позволяющее оперативно программировать решение задач определенного класса (например, в таких областях, как статистика, инженерная графика, научно-технические расчеты и т.д.).

Развитие ЭВМ идет по пути непрерывного повышения быстродействия, надежности, расширения функциональных возможностей, уменьшения габаритов и потребляемой мощности,

упрощения правил работы на компьютере. Среди ЭВМ четвертого поколения появились персональные компьютеры (ПК или ПЭВМ), которые позволяют индивидуально работать каждому пользователю.

Первой ПЭВМ можно считать компьютер Altair-8800, созданный в 1974 г. Э. Робертсом. Для этого компьютера П. Аллен и Б. Гейтс в 1975 г. создали транслятор с популярного языка Basic. Впоследствии П. Аллен и Б. Гейтс создали известную компанию Microsoft.

После изобретения интегральной схемы развитие компьютерной техники резко ускорилось. Эмпирический факт, замеченный в 1965 г. соучредителем компании Intel Гордоном Е. Муром, назвали Законом Мура, согласно которому (в современной формулировке) количество транзисторов, размещаемых на кристалле интегральной схемы, удваивается каждые 24 месяца. В 2007 г. Мур заявил, что закон, очевидно, скоро перестанет действовать из-за атомарной природы вещества и ограничения скорости света.

2.2. Классификация компьютеров

Существуют различные классификации компьютеров:

- по этапам развития (по поколениям);
- по классу выполняемых задач:
  - универсальные;
  - специализированные;
  - микроконтроллеры;
- по виду вычислительного процесса:
  - аналоговые вычислительные машины (АВМ);
  - гибридные вычислительные системы (машины) (ГВС, ГВМ);
  - цифровые вычислительные машины:
    - двоичные;
    - троичные;
    - десятичные;

- по виду рабочей среды:
  - квантовый компьютер;
  - механический компьютер;
    - пневматический компьютер;
    - гидравлический компьютер;
  - оптический компьютер;
  - электронный компьютер;
  - биологический компьютер;
- по назначению:
  - сервер;
  - рабочая станция;
  - персональный компьютер;
- в зависимости от размера:
  - мейнфрейм;
  - мини-компьютер;
  - карманный персональный компьютер (КПК);
  - ноутбук;
  - нанокомпьютер;
- по другим признакам:
  - встраиваемая система;
  - параллельные вычислительные системы;
  - компьютер для операций с функциями;
  - суперкомпьютер;
  - нейрокомпьютер;
  - биокомпьютер;
  - молекулярный компьютер;
  - ДНК-компьютер.

2.3. Состав вычислительной системы

Принцип действия компьютера. В основе любого современного компьютера лежит *тактовый генератор*, вырабатывающий через равные интервалы времени электрические сигналы, которые используются для приведения в действие всех устройств компьютерной системы. В персональном компьютере

тактовые импульсы задает одна из микросхем микропроцессорного комплекта (чипсета), расположенного на материнской плате. Чем выше частота тактов, поступающих на процессор, тем больше команд он может исполнить в единицу времени (исполнение каждой команды занимает определенное количество тактов), тем выше его производительность [1].

Управление компьютером фактически сводится к управлению распределением сигналов между устройствами. Такое управление может производиться автоматически (*программное управление*). В современных компьютерах внешнее управление в значительной степени автоматизировано с помощью специальных аппаратно-логических интерфейсов, к которым подключаются устройства управления и ввода данных (клавиатура, мышь, джойстик и др.). Такое управление называется *интерактивным*.

Состав вычислительной системы называется *конфигурацией*, состоящей из *аппаратной конфигурации* (аппаратного обеспечения) и *программной конфигурации* (программного обеспечения).

Несмотря на то, что программное и аппаратное обеспечение рассматриваются отдельно, в компьютере они работают в неразрывной связи и в непрерывном взаимодействии.

Аппаратная конфигурация представляет собой блочно-модульную конструкцию, т.е. состоящую из готовых узлов и блоков.

По способу расположения устройств относительно *центрального процессорного устройства* (*ЦПУ*, *CPU*) различают *внутренние* и *внешние* устройства. *Внешними*, как правило, являются большинство устройств ввода-вывода данных (периферийные устройства) и некоторые устройства длительного хранения данных.

Согласование между отдельными узлами и блоками выполняют с помощью переходных аппаратно-логических устройств, называемых *аппаратными интерфейсами*. Стандарты на аппаратные интерфейсы называют *протоколами*. Таким образом, *протокол* – это совокупность технических условий, которые должны быть обеспечены разработчиками устройств для успешного согласования их работы с другими устройствами.

Интерфейсы в архитектуре вычислительной системы можно условно разделить на две большие группы: *последовательные* и *параллельные*. Через последовательный интерфейс данные передаются, бит за битом, а через параллельный – одновременно группами битов (количество битов, участвующих в одной посылке, определяется разрядностью интерфейса). Параллельные интерфейсы обычно имеют более сложное устройство, чем последовательные, и теоретически способны обеспечить большую скорость передачи данных. Устройство последовательных интерфейсов проще. Как правило, для них не надо синхронизировать работу передающего и принимающего устройств (поэтому их часто называют асинхронными интерфейсами).

На первый взгляд, кажется, что последовательное соединение должно проигрывать параллельному, так как оно может передавать меньше данных за такт. Однако зачастую последовательные соединения могут функционировать значительно быстрее по сравнению с параллельными соединениями, и за счёт этого достигать более высокой скорости передачи данных. Среди факторов, позволяющих последовательному соединению работать быстрее параллельного, выделяют:

– отсутствие расфазировки синхронизирующих импульсов между различными каналами (для нескоростных асинхронных последовательных соединений).

– последовательное соединение требует меньше соединительных кабелей (проводов/волокон) и поэтому занимает меньше пространства. Дополнительное пространство позволяет использовать лучшую изоляцию канала от влияния окружающей среды.

– перекрестные помехи существенно ниже, так как там меньше расположенных рядом проводников.

Во многих случаях, последовательное соединение является лучшим выбором, так как оно дешевле в реализации.

В связи с этим большинство современных высокоскоростных шин передачи данных являются последовательными.

2.4. Базовая аппаратная конфигурация персонального компьютера

Конфигурацию (состав оборудования) персонального компьютера можно гибко изменять по мере необходимости. Тем не менее существует понятие *базовой аппаратной конфигурации*, которую считают типовой:

– системный блок (в случае моноблочной компоновки объединен в один корпус с монитором);

– монитор;

– клавиатура;

– мышь.

Основные потребительские параметры монитора:

– размер экрана – определяется длиной диагонали, измеряется в дюймах;

– разрешающая способность монитора, dpi (количество точек на дюйм);

– время отклика пикселя (время, за которое пиксель меняет свой цвет);

– яркость и контрастность изображения;

– углы обзора по горизонтали и вертикали (из-за конструкции ЖК-мониторов при взгляде под углом уменьшается контрастность и яркость изображения, углы обзора показывают предельные параметры, при которых картинка выводится без цветовых искажений);

– максимальная частота регенерации изображения (показывает, сколько раз в течение секунды монитор может полностью сменить изображение), измеряется в герцах (Гц).

2.5. Внутренние устройства системного блока

Материнская плата

На материнской плате размещаются:

– *процессор* – основная микросхема, выполняющая большинство математических и логических операций;

– *микропроцессорный комплект – чипсет* – набор микросхем, управляющих работой внутренних устройств компьютера

и определяющих основные функциональные возможности материнской платы;

– *шины* – наборы проводников, по которым происходит обмен сигналами между внутренними устройствами компьютера;

– *оперативная память (оперативное запоминающее устройство, ОЗУ)* – набор микросхем, предназначенных для временного хранения данных, когда компьютер включен;

– *ПЗУ (постоянное запоминающее устройство)* – микросхема, предназначенная для длительного хранения данных;

– разъемы для подключения дополнительных устройств (*слоты*).

Жесткий диск

Это основное устройство для долговременного хранения больших объемов данных и программ, в том числе и при отсутствии питания, т.е. является энергонезависимым. Вся информация сохраняется на металлических магнитных дисках, расположенных друг над другом, покрытых слоем ферромагнитного материала, откуда считывается и записывается подвижной головкой. В данном устройстве носитель информации совмещен в корпусе с приводом и блоком управляющей электроники.

Управление работой жесткого диска выполняет специальное аппаратно-логическое устройство – контроллер жесткого диска. В настоящее время в ПК функции контроллеров дисков выполняют микросхемы, входящие в микропроцессорный комплект (чипсет). В то же время в серверах контроллер жестких дисков может быть выполнен в виде отдельной платы, подключаемой к материнской плате по шине PCIe.

К основным параметрам жестких дисков относятся *емкость* (зависит от технологии изготовления) и *производительность* (зависит от характеристик интерфейса, с помощью которого они связаны с материнской платой). С производительностью диска напрямую связан параметр *среднего времени доступа*. Он определяет интервал времени, необходимый для поиска нужных данных, и зависит от скорости вращения диска.

Отдельной разновидностью жестких дисков являются так называемые твердотельные диски, использующие для хранения данных микросхемы, построенные по технологии электрически перепрограммируемой памяти (по этой технологии в том числе изготавливаются и флеш-накопители). Основным преимуществом твердотельных дисков является высокая скорость чтения/записи данных и очень низкое время доступа. Недостаток – высокая цена.

Оптический привод

Оптический (лазерный) привод – это устройство, имеющее механическую составляющую и предназначенное для считывания/записи информации с оптических носителей информации. Принцип действия этого устройства состоит в считывании числовых данных с помощью лазерного луча, отражающегося от поверхности диска.

Название «оптический привод» включает в себя несколько типов приводов, предназначенных для чтения различных типов дисков: CD – компакт-диски, объем около 700 Мбайт; DVD – цифровые универсальные диски, объем более 4,7 Гбайт и Blu-ray, объем 25–128 Гбайт и выше. Основные отличия заключаются в объеме данных, внешне они идентичны.

Диски с дополнительным обозначением – R/RW – позволяют записывать/перезаписывать на них данные. Для этого необходимо наличие привода, поддерживающего запись дисков.

Основные характеристики – объем накопителя и скорость чтения/записи.

Скорость чтения/записи измеряется в кратных долях. За единицу измерения принята скорость чтения музыкальных компакт-дисков, составляющая 150 Кбайт/с. Таким образом, дисковод с удвоенной скоростью обеспечивает производительность 300 Кбайт/с и т.д.

Видеокарта (видеоадаптер)

Совместно с монитором видеокарта образует видеоподсистему персонального компьютера. Физически видеоадаптер выполнен в виде отдельной дочерней платы, которая вставляется в

один из слотов материнской платы и называется видеокартой. Видеоадаптер выполняет функции *видеопроцессора* (отвечает за формирование изображения в видеопамяти и осуществляет обработку запросов центрального процессора) и *видеопамяти*.

Основные параметры видеоподсистемы:

– *разрешение экрана* (количество точек по горизонтали и вертикали). Для каждого размера монитора существует свое оптимальное разрешение экрана, которое должен обеспечивать видеоадаптер;

– *цветовое разрешение (глубина цвета)* определяет количество различных оттенков, которые может принимать отдельная точка экрана. Максимально возможное цветовое разрешение зависит от свойств видеоадаптера, и в первую очередь от количества установленной на нем видеопамяти. Кроме того, оно зависит и от установленного разрешения экрана;

– *поддержка технологий видеоускорения* – одно из свойств видеоадаптера, которое заключается в том, что часть операций по построению изображений может происходить без выполнения математических вычислений в основном процессоре компьютера, чисто аппаратным путем – преобразованием данных в микросхемах видеоускорителя (технологии DirectX, OpenGL).

Звуковая карта

Звуковая карта устанавливается в один из разъемов материнской платы в виде дочерней карты и выполняет вычислительные операции, связанные с обработкой звука, речи, музыки.

Основным параметром звуковой карты является *разрядность*, определяющая количество битов, используемых при преобразовании сигналов из аналоговой в цифровую форму и наоборот. Чем выше разрядность, тем меньше погрешность, связанная с оцифровкой, тем выше качество звучания.

При отсутствии повышенных требований к качеству звука используются интегрированные звуковые системы, в которых функции обработки звука выполняются центральным процессором и микросхемами материнской платы.

2.6. Системы, расположенные на материнской плате

Оперативная память

Оперативная память (*Random Access Memory, RAM,* память с произвольным доступом) – это энергозависимая часть системы компьютерной памяти, в которой во время работы компьютера хранится выполняемый машинный код (программы), а также входные, выходные и промежуточные данные, обрабатываемые процессором.

С точки зрения физического принципа действия различают *динамическую память (DRAM)* и *статическую память (SRAM)*.

Динамическая память (DRAM) хранит бит данных в виде наличия (или отсутствия) заряда *конденсатора*.

Это наиболее распространенный и экономически доступный вид памяти. Вследствие высокой плотности DRAM занимает меньшую площадь на кристалле по сравнению с SRAM.

Недостатки этого типа связаны, во-первых, с тем, что для изменения состояния конденсатора его нужно зарядить или разрядить, т.е. неизбежны переходные процессы, поэтому запись данных происходит сравнительно медленно. Второй существенный недостаток связан с тем, что конденсаторы со временем разряжаются (именно из-за того, что заряд конденсатора динамически уменьшается во времени, память на конденсаторах получила свое название DRAM – динамическая память). Поэтому, чтобы не потерять содержимое памяти, заряд конденсаторов периодически восстанавливается (регенерируется) через определенное время, называемое циклом регенерации (обычно 2 мс). При этом обращение к памяти периодически приостанавливается, что снижает среднюю скорость обмена с DRAM.

В *статической памяти (SRAM)* бит данных хранится в виде состояния (включен/выключен) *триггера*, представляющего собой несколько транзисторов. Этот тип памяти обеспечивает более высокое быстродействие, так как изменение управляющего напряжения на входе триггера очень быстро изменяет его состояние, и меньшее энергопотребление, чем DRAM, хотя технологически он сложнее и, соответственно, дороже.

Микросхемы динамической памяти используют в качестве основной оперативной памяти компьютера. Микросхемы статической памяти используют в качестве вспомогательной памяти (кэш-памяти), предназначенной для оптимизации работы процессора, а именно для согласования работы сравнительно медленных устройств (в этом случае это DRAM) со сравнительно быстрым микропроцессором.

У каждой ячейки памяти есть свой адрес, который выражается числом. Максимальный размер поля оперативной памяти, установленной в компьютере, определяется микропроцессорным комплектом (чипсетом) материнской платы.

Оперативная память в компьютере размещается на стандартных панельках, называемых *модулями*, которые вставляют в соответствующие разъемы на материнской плате.

Основными характеристиками оперативной памяти являются *объем памяти* и *скорость передачи данных*. Скорость передачи данных определяет максимальную пропускную способность памяти (Гбайт/с) в оптимальном режиме доступа. Одинаковые по объему модули могут иметь разные скоростные характеристики. Также характеристикой памяти является *время доступа*, измеряется в миллиардных долях секунды (*наносекундах, нс*).

Процессор

Процессор – основная микросхема компьютера, в которой производятся все вычисления. Конструктивно процессор состоит из ячеек, в которых данные могут не только храниться, но и изменяться заданным образом. Внутренние ячейки памяти в процессоре называют *регистрами*. Таким образом, любой процессор состоит из набора регистров различного назначения (данные, адресные данные, команды), которые определенным образом связаны между собой и обрабатываются в соответствии с некоторой системой правил. Исполнение программ основано на управлении засылкой данных в разные регистры процессора и таким образом управлении обработкой данных.

К обязательным компонентам процессора относятся *арифметико-логическое устройство* и *устройство управления*. Выполнение процессором команды предусматривает: арифметические действия, логические операции, передачу управления, перемещение данных из одного места памяти в другое и координацию взаимодействия различных устройств компьютера. Выделяют четыре этапа обработки команды процессором: выборка команды из ОЗУ, декодирование, выполнение и запись результата.

Основные параметры процессоров. Основными параметрами процессоров являются: *разрядность, рабочая тактовая частота, коэффициент внутреннего умножения тактовой частоты*, *размер кэш-памяти и набор поддерживаемых инструкций*.

Параметр, который имеет немаловажное значение для производительности процессора, – это его разрядность. *Разрядность процессора* показывает, сколько бит данных он может принять и обработать в своих регистрах за один такт.

Исполнение каждой команды процессором занимает определенное количество тактов. Чем выше частота тактов, поступающих на процессор, тем больше команд он может исполнить в единицу времени, тем выше его производительность. В настоящее время *рабочие частоты* процессоров превосходят миллиард тактов в секунду (1 ГГц).

Тактовые сигналы процессор получает от материнской платы, которая представляет собой большой набор проводников и микросхем и по чисто физическим причинам не может работать со столь высокими частотами, как процессор. Для получения более высоких частот в процессоре происходит *внутреннее умножение частоты* на коэффициент.

Обмен данными внутри процессора происходит в несколько раз быстрее, чем обмен с другими устройствами, например с оперативной памятью. Для того чтобы уменьшить количество обращений к оперативной памяти, внутри процессора создают

буферную область – *кэш-память*. Когда процессору нужны данные, он сначала обращается в кэш-память, и только если там нужных данных нет, обращается в оперативную память. Принимая блок данных из оперативной памяти, процессор заносит его одновременно и в кэш-память.

Расширенные наборы инструкций, позволяющие процессору обрабатывать сразу несколько элементов данных за один такт, ускоряют вычисления в таких областях, как обработка графики, кодирование видеоданных и выполнение сложных математических расчетов.

С остальными устройствами компьютера, и в первую очередь с оперативной памятью, процессор связан несколькими группами проводников, называемых *шинами*. Основных шин три: *шина данных, адресная шина* и *шина управления*.

Адресная шина. 64-разрядная адресная шина состоит из 64 параллельных линий. В зависимости от того, есть напряжение на какой-то из линий или нет, говорят, что на этой линии выставлена единица или ноль. Комбинация из 64 нулей и единиц образует 64-разрядный адрес, указывающий на одну из ячеек оперативной памяти, к которой и подключается процессор для копирования данных из ячейки в один из своих регистров.

Шина данных. По этой шине происходит копирование данных из оперативной памяти в регистры процессора и обратно. По 64-разрядной шине данных за один раз на обработку поступают сразу 8 байт.

Шина управления. По этой шине передаются сигналы, определяющие характер обмена информацией по магистрали. Сигналы управления определяют, какую операцию (считывание или запись информации из памяти) нужно производить, синхронизируют обмен информацией между устройствами и т.д. Эта шина не имеет такой же четкой структуры, как шина данных или шина адреса. В шину управления условно объединяют набор линий, передающих различные управляющие сигналы от процессора на все периферийные устройства и обратно.

Так как связь между всеми собственными и подключаемыми устройствами материнской платы выполняют ее шины и логические устройства, размещенные в микросхемах микропроцессорного комплекта (чипсета), от архитектуры этих элементов во многом зависит производительность компьютера.

Вычислительная мощность (производительность) суперкомпьютера

Эта количественная характеристика скорости выполнения определенных операций на суперкомпьютере измеряется во флопсах (количество операций с плавающей запятой в секунду), а также производными от нее: мега-, гига-, тера-, петафлопсах (10^{12}). В связи с неоднозначностью определения данного параметра выделяют *пиковую вычислительную мощность* – гипотетически максимально возможное количество операций над числами с плавающей запятой в секунду, которое способен произвести данный суперкомпьютер.

2.7. Периферийные устройства персонального компьютера

Периферийные устройства персонального компьютера подключаются к его интерфейсам и предназначены для выполнения вспомогательных операций. Благодаря им компьютерная система приобретает гибкость и универсальность.

По назначению периферийные устройства можно подразделить:

– на устройства ввода данных (сканеры, цифровые фотокамеры, специальные манипуляторы);

– устройства вывода данных (принтеры);

– устройства хранения данных (магнитооптические устройства, флеш-накопители и др.);

– устройства обмена данными (сетевые карты).

3. ПРОГРАММНЫЕ СРЕДСТВА РЕАЛИЗАЦИИ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ

3.1. Программное обеспечение. Классификация программного обеспечения компьютера

Программы – это упорядоченные последовательности команд. Конечная цель любой компьютерной программы – управление аппаратными средствами.

Состав программного обеспечения вычислительной системы называют *программной конфигурацией*. Между программами, как и между физическими узлами и блоками, существует взаимосвязь – многие программы работают, опираясь на другие программы более низкого уровня (*межпрограммный интерфейс*). Таким образом, программное обеспечение можно распределить на несколько взаимодействующих между собой уровней, представляющих пирамидальную конструкцию. Каждый следующий уровень опирается на программное обеспечение предшествующих уровней, и каждый вышележащий уровень повышает функциональность всей системы [1].

Базовый уровень. Отвечает за взаимодействие с базовыми аппаратными средствами. Как правило, базовые программные средства непосредственно входят в состав базового оборудования и хранятся в специальных микросхемах, называемых *постоянными запоминающими устройствами (ПЗУ – Read Only Memory, ROM)*. Программы и данные записываются («прошиваются») в микросхемы ПЗУ на этапе производства и не могут быть изменены в процессе эксплуатации.

Если во время эксплуатации изменение базовых программных средств является технически целесообразным, то вместо

микросхем ПЗУ применяют *перепрограммируемые постоянные запоминающие устройства.*

Системный уровень. Программы, работающие на этом уровне, обеспечивают взаимодействие программ компьютерной системы с программами базового уровня и с аппаратным обеспечением.

В состав программного обеспечения системного уровня входят программы, отвечающие за взаимодействие с конкретными устройствами, называемые *драйверами устройств*.

Другой класс программ системного уровня отвечает за взаимодействие с пользователем (средства обеспечения пользовательского интерфейса).

Совокупность программного обеспечения системного уровня образует *ядро операционной системы* компьютера.

Служебный уровень. Основное назначение служебных программ – *утилит* – состоит в автоматизации работ по проверке, наладке и настройке компьютерной системы. Служебные программы в большинстве случаев используются для расширения или улучшения функций системных программ.

Классификация служебных программных средств:

– *диспетчеры файлов (файловые менеджеры)*. С их помощью выполняется большинство операций, связанных с обслуживанием файловой структуры: копирование, перемещение и переименование файлов, создание и удаление файлов и каталогов, навигация по файловой структуре (например, программа *Проводник* – стандартное приложение операционной системы Windows);

– *средства сжатия данных (архиваторы)*. Предназначены для создания архивов с целью повышения эффективности использования носителя за счет того, что архивные файлы обычно имеют повышенную плотность записи информации;

– *средства просмотра и воспроизведения*;

– *средства диагностики*. Предназначены для автоматизации процессов диагностики программного и аппаратного обеспечения;

– *средства контроля (мониторинга)*. Позволяют следить за процессами, происходящими в компьютерной системе;

– *мониторы установки*. Предназначены для контроля над установкой программного обеспечения (для протоколирования образования связей между различными категориями программного обеспечения);

– *средства коммуникации*. Позволяют устанавливать соединения с удаленными компьютерами, обслуживают передачу сообщений электронной почты, работу с телеконференциями (группами новостей) и др.;

– *средства обеспечения компьютерной безопасности*. К ним относятся средства пассивной защиты (программы для резервного копирования) и активной защиты (антивирусное программное обеспечение, например программы NOD32, Dr.Web, Антивирус Касперского), а также средства защиты от несанкционированного доступа, просмотра и изменения данных.

Прикладной уровень. Представляет собой комплекс прикладных программ, с помощью которых выполняются конкретные задания.

Классификация прикладных программных средств:

– *текстовые редакторы*. Обеспечивают ввод и редактирование текстовых данных;

– *текстовые процессоры*. Позволяют не только вводить и редактировать текст, но и форматировать его, т.е. оформлять (например, Microsoft Word);

– *графические редакторы*. Предназначены для создания и обработки графических изображений. Различают следующие категории: *растровые редакторы* (например, Adobe Photoshop), *векторные редакторы* (например, CorelDraw) и программные средства для создания и обработки трехмерной графики (*3D-редакторы*, например 3D Studio Max).

Растровые редакторы применяют в тех случаях, когда графический объект представлен в виде комбинации точек, образующий растр и обладающих свойствами яркости и цвета. Такой

подход эффективен в тех случаях, когда графическое изображение имеет много полутонов и информация о цвете элементов, составляющих объект, важнее, чем информация об их форме. Растровые редакторы широко применяются для обработки изображений, их ретуши, создания фотоэффектов.

Векторные редакторы отличаются от растровых способом представления данных об изображении. Элементарным объектом векторного изображения является линия. Такой подход характерен для чертежно-графических работ, в которых форма линий имеет большее значение, чем информация о цвете отдельных точек, составляющих ее;

– *электронные таблицы*. Предоставляют комплексные средства для хранения различных типов данных и их обработки (например, Microsoft Excel). Основное свойство электронных таблиц состоит в том, что при изменении содержания любых ячеек таблицы происходит автоматическое изменение содержания во всех прочих ячейках, связанных с измененными математическими или логическими выражениями (формулами). Электронные таблицы применяются там, где необходимо автоматизировать регулярно повторяющиеся вычисления достаточно больших объемов числовых данных;

– *системы управления базами данных (СУБД)*. Базами данных называют массивы данных, организованных в табличные структуры. Основными функциями СУБД (например, Microsoft Access) являются:

1) создание структуры базы данных (создание таблиц и определение перечня полей, из которых она должна состоять, их типов и размеров);

2) предоставление средств ее заполнения или импорта данных из таблиц другой базы;

3) обеспечение возможности доступа к данным и предоставление средств поиска и фильтрации;

– *системы автоматизированного проектирования (CAD-системы)*. Предназначены для автоматизации проектно-конструкторских работ. Кроме чертежно-графических работ эти системы

позволяют производить простейшие расчеты (например, расчеты прочности деталей) и выбор готовых конструктивных элементов из обширных баз данных;

– *специальные пакеты (математические, статистические и др.)*. Обеспечивают решение задач в определенной предметной области, например в математике, статистике, банковском деле и т.д.;

– *экспертные системы*. Предназначены для анализа данных, содержащихся в базах знаний, и выдачи рекомендаций по запросу пользователя. Такие системы применяют в тех случаях, когда исходные данные хорошо формализуются, но для принятия решений требуются обширные специальные знания (юриспруденция, медицина, химия);

– *web-редакторы*. Предназначены для создания и редактирования web-документов (web-страниц Интернета). Объединяют в себе свойства текстовых и графических редакторов;

– *браузеры (обозреватели, средства просмотра web)*. Программные средства, предназначенные для просмотра электронных документов, выполненных в формате HTML (например, Mozilla Firefox);

– *интегрированные системы делопроизводства*. Представляют собой программные средства автоматизации рабочего места руководителя;

– *геоинформационные системы (ГИС)*. Предназначены для автоматизации картографических и геодезических работ на основе информации, полученной топографическими или аэрокосмическими методами.

3.2. Классификация операционных систем

Операционная система (ОС) представляет собой комплекс системных и служебных программных средств, предназначенных для управления ресурсами вычислительного устройства и организации взаимодействия с пользователем. *Приложениями операционной системы* называют программы, предназначенные для работы под управлением данной системы.

Существует множество параметров, по которым можно классифицировать операционные системы. К ним могут относиться особенности алгоритмов управления ресурсами компьютера, особенности архитектуры операционной системы, типы поддерживаемых аппаратных платформ, области применения и многое другое.

Ниже приведена классификация операционных систем по некоторым из этих признаков.

С точки зрения особенностей алгоритмов управления ресурсами можно выделить следующие классификации.

- По числу одновременно выполняющихся процессов (задач):
  - однозадачные (MS-DOS);
  - многозадачные (Windows, UNIX):
    - невытесняющая многозадачность;
    - вытесняющая многозадачность.
- По количеству одновременно работающих пользователей:
  - однопользовательские (MS-DOS, Windows до NT);
  - многопользовательские (Windows NT, UNIX).

Однозадачные ОС включают средства управления периферийными устройствами, средства управления файлами, средства общения с пользователем.

Многозадачные ОС, кроме вышеперечисленных функций, управляют разделением совместно используемых ресурсов, таких как процессор, оперативная память, файлы и внешние устройства. Многозадачные системы могут быть двух видов. В системах с невытесняющей многозадачностью переключение между процессами происходит по решению процесса, который выполняется в текущий момент времени. В системах с вытесняющей многозадачностью переключение между задачами происходит по командам операционной системы, чаще всего в результате срабатывания аппаратного прерывания от системного таймера.

Главным отличием многопользовательских систем от однопользовательских является наличие средств защиты информации каждого пользователя от несанкционированного доступа других

пользователей. Следует заметить, что не всякая многозадачная система является многопользовательской, и не всякая однопользовательская ОС является однозадачной.

С точки зрения особенностей аппаратных платформ различают операционные системы персональных компьютеров, миникомпьютеров, мейнфреймов, кластеров и сетей ЭВМ. Среди перечисленных типов компьютеров могут встречаться как однопроцессорные варианты, так и многопроцессорные. В любом случае специфика аппаратных средств, как правило, отражается на специфике операционных систем.

Сетевая ОС имеет в своем составе средства передачи сообщений между компьютерами по линиям связи, которые совершенно не нужны в автономной ОС. На основе этих сообщений сетевая ОС поддерживает разделение ресурсов компьютера между удаленными пользователями, подключенными к сети.

Многопроцессорные системы требуют от операционной системы особой организации, с помощью которой сама операционная система, а также поддерживаемые ею приложения могли бы выполняться параллельно отдельными процессорами системы. Параллельная работа отдельных частей ОС создает дополнительные проблемы для разработчиков ОС, так как в этом случае гораздо сложнее обеспечить согласованный доступ отдельных процессов к общим системным таблицам, исключить эффект гонок и прочие нежелательные последствия асинхронного выполнения работ.

С точки зрения особенностей областей использования многозадачные операционные системы разделяют на три типа в зависимости от критерия эффективности, которому должен удовлетворять механизм работы многозадачности:

- системы пакетной обработки;
- системы разделения времени (UNIX, VMS);
- системы реального времени (QNX, RT/11).

Системы пакетной обработки рассчитаны на ситуацию, когда не требуется быстрого получения результатов, а главной целью является пропускная способность – т.е. количество решенных за-

дач в единицу времени. Для достижения этих целей каждая задача в операционной системе содержит метаинформацию о том, какие ресурсы и в каком количестве ей необходимы. На основе этой метаинформации операционная система планирует очередность запуска задач таким образом, чтобы обеспечить максимальную пропускную способность системы и эффективно загрузить все компоненты компьютера, минимизировав их простои.

Основной идеей систем разделения времени является предоставление пользователям системы возможности запускать свои задачи в любое время и интерактивно с ними взаимодействовать. В системах разделения времени каждой задаче выделяется только квант процессорного времени, ни одна задача не занимает процессор надолго, и время ответа оказывается приемлемым. Если квант выбран достаточно небольшим, то у всех пользователей, одновременно работающих на одной и той же машине, складывается впечатление, что каждый из них единолично использует машину. Критерием эффективности систем разделения времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя.

Системы реального времени применяются для управления различными техническими объектами, например станками, спутниками, научным оборудованием, технологическими процессами и т.п. Во всех этих случаях существует предельно допустимое время, за которое программа должна дать свой ответ на изменение входных данных. В случае если нарушение этого условия приведет к аварии, то систему называют системой жесткого реального времени, если к ухудшению качества работы – то системой мягкого реального времени, а если требуется, чтобы этот критерий выполнялся в среднем с заданной вероятностью, то системой квазиреального времени. Таким образом, критерием эффективности для систем реального времени является их способность выдерживать заранее заданные интервалы времени между запуском программы и получением результата (управляющего воздействия).

Некоторые операционные системы могут совмещать в себе свойства систем разных типов. Например, часть задач может выполняться в режиме пакетной обработки, а часть – в режиме реального времени или в режиме разделения времени. В таких случаях режим пакетной обработки часто называют фоновым режимом.

По особенностям архитектуры операционной системы они делятся на монолитные и микроядерные.

В операционных системах, построенных по монолитной архитектуре, ядро является одним исполняемым модулем, в который включены все необходимые компоненты, исполняющиеся в едином адресном пространстве. Достоинством этого решения является производительность, недостатком – низкая устойчивость к ошибкам, так как ошибка в одном компоненте может привести к повреждению памяти и нарушению работы других компонентов.

Напротив, в микроядерной архитектуре ядро состоит из набора взаимодействующих процессов, каждый из которых отвечает за свою задачу. Например, один процесс может заниматься работой с видеокартой, в то время как другой управляет дисковым вводом/выводом. Данный тип архитектуры повышает надежность системы, так как в случае сбоя одного из процессов чаще всего он может быть перезапущен без серьезных последствий для работающей операционной системы. Недостатком является сложность разработки такого типа операционных систем и более низкая производительность по сравнению с монолитной архитектурой.

3.3. Функции операционных систем персональных компьютеров

Основная функция операционных систем – посредническая [1]. Она заключается в обеспечении нескольких видов интерфейса:

– интерфейса между пользователем и программно-аппаратными средствами компьютера (*интерфейс пользователя*);

– интерфейса между программным и аппаратным обеспечением (*аппаратно-программный интерфейс*);

– интерфейса между разными видами программного обеспечения (*программный интерфейс*).

Обеспечение интерфейса пользователя

По реализации интерфейса пользователя различают *неграфические* и *графические* операционные системы.

Неграфические операционные системы (например, MS-DOS) реализуют *интерфейс командной строки*. Основным устройством управления в этом случае является клавиатура.

Работа с графической операционной системой основана на взаимодействии активных (указатель мыши) и пассивных (строки меню, экранные кнопки, значки, раскрывающиеся списки и т.д.) экранных элементов управления.

Обеспечение автоматического запуска

Все операционные системы обеспечивают свой автоматический запуск. Для дисковых операционных систем в специальной (системной) области диска создается запись программного кода, обращение к которому выполняют программы, находящиеся в базовой системе ввода-вывода (BIOS).

Недисковые операционные системы характерны для специализированных вычислительных систем, в частности для компьютеризированных устройств автоматического управления. Математическое обеспечение, содержащееся в микросхемах ПЗУ таких компьютеров, можно условно рассматривать как аналог операционной системы. Ее автоматический запуск осуществляется аппаратно. При подаче питания процессор обращается к фиксированному физическому адресу ПЗУ, с которого начинается запись программы инициализации операционной системы.

Организация файловой системы

Все современные дисковые операционные системы обеспечивают создание файловой системы, предназначенной для хранения данных на дисках и обеспечения доступа к ним. Принцип

организации файловой системы – табличный. Поверхность жесткого диска рассматривается как трехмерная матрица, измерениями которой являются номера поверхности, цилиндра и сектора. Под цилиндром понимается совокупность всех дорожек, принадлежащих разным поверхностям и находящихся на равном удалении от оси вращения. Данные о том, в каком месте диска записан тот или иной файл, хранятся в системной области диска в специальных *таблицах размещения файлов* (FAT-таблицах, File Allocation Table).

Наименьшей физической единицей хранения данных является сектор. Размер сектора равен 512 байт или 4 Кбайт. Поскольку размер FAT-таблицы ограничен, то для дисков, размер которых превышает 32 Мбайт, обеспечить адресацию к каждому отдельному сектору не представляется возможным. В связи с этим группы секторов условно объединяются в кластеры. Кластер является наименьшей единицей адресации к данным. Размер кластера, в отличие от размера сектора, не фиксирован и зависит от емкости диска.

Файловая система NTFS заменила файловую систему FAT, использовавшуюся в операционных системах MS-DOS и Windows.

NTFS (New Technology File System – файловая система новой технологии) – стандартная файловая система для семейства операционных систем Windows NT фирмы Microsoft.

NTFS поддерживает хранение метаданных. С целью улучшения производительности, надёжности и эффективности использования дискового пространства для хранения информации о файлах в NTFS используются специализированные структуры данных. Информация о файлах хранится в главной файловой таблице – Master File Table (MFT). NTFS поддерживает разграничение доступа к данным для различных пользователей и групп пользователей (списки контроля доступа – access control lists, ACL), а также позволяет назначать дисковые квоты (ограничения на максимальный объём дискового пространства, занимаемый файлами тех или иных пользователей). Для повышения надёжности файловой системы в NTFS используется система жур-

налирования. Для NTFS размер кластера по умолчанию составляет от 512 байт до 64 Кбайт в зависимости от размера диска и версии ОС.

Обслуживание файловой структуры

Несмотря на то, что данные о местоположении файлов хранятся в табличной структуре, пользователю они представляются в виде иерархической структуры – файловой структуры. К функции обслуживания файловой структуры относятся следующие операции, осуществляющиеся под управлением операционной системы:

– создание файлов и присвоение им имен;

– создание каталогов (папок) и присвоение им имен;

– переименование файлов и каталогов (папок);

– копирование и перемещение файлов;

– удаление файлов и каталогов (папок) (существует как минимум три режима удаления данных: удаление, уничтожение и стирание);

– навигация по файловой структуре с целью доступа к файлу, каталогу (папке);

– управление атрибутами файлов (кроме имени и расширения имени файла, операционная система хранит для каждого файла дату его создания или изменения и несколько величин, называемых атрибутами файла, например, «только для чтения», «скрытый», «системный», «архивный»). Операционная система позволяет их контролировать и изменять; состояние атрибутов учитывается при проведении автоматических операций с файлами.

Управление установкой, исполнением и удалением приложений

Работа с приложениями составляет наиболее важную часть работы операционной системы, так как основная функция операционной системы состоит в обеспечении интерфейса приложений с аппаратными и программными средствами вычисли-

тельной системы, а также с пользователем. С точки зрения управления исполнением приложений различают *однозадачные* и *многозадачные* операционные системы.

Большинство современных графических операционных систем – многозадачные. Они управляют распределением ресурсов вычислительной системы между задачами и обеспечивают:

– возможность одновременной или поочередной работы нескольких приложений;

– возможность обмена данными между приложениями;

– возможность совместного использования программных, аппаратных, сетевых и прочих ресурсов вычислительной системы несколькими приложениями.

Для правильной работы приложений на компьютере они должны пройти операцию, называемую *установкой*. Необходимость в установке связана с тем, что разработчики программного обеспечения не могут заранее предвидеть особенности аппаратной и программной конфигурации вычислительной системы, на которой предстоит работать их программам. Таким образом, *дистрибутивный комплект* (*установочный пакет*) программного обеспечения, как правило, представляет собой незаконченный программный продукт, из которого в процессе установки на компьютере формируется полноценное рабочее приложение. При этом осуществляется привязка приложения к существующей аппаратно-программной среде и его настройка на работу именно в этой среде. Управляют установкой приложений операционные системы. Они управляют распределением ресурсов вычислительной системы между приложениями, обеспечивают доступ устанавливаемых приложений к драйверам устройств, формируют общие ресурсы, которые могут использоваться разными приложениями, выполняют регистрацию установленных приложений и выделенных им ресурсов.

Процесс удаления приложений имеет свои особенности и происходит под строгим контролем операционной системы. Нельзя допустить, чтобы при удалении одного приложения были

удалены ресурсы, на которые опираются другие приложения, даже если эти ресурсы были установлены вместе с удаляемым приложением.

Обеспечение взаимодействия с аппаратными средствами

Средства аппаратного обеспечения вычислительной техники отличаются огромным многообразием. Гибкость аппаратных и программных конфигураций вычислительных систем поддерживается за счет того, что каждый разработчик оборудования прикладывает к нему специальные программные средства управления – драйверы. Драйверы имеют точки входа для взаимодействия с прикладными программами, а диспетчеризация обращений прикладных программ к драйверам устройств – это одна из функций операционной системы. Операционная система выполняет все функции по установке драйверов устройств и передаче им управления от приложений. Во многих случаях операционная система даже не нуждается в драйверах, полученных от разработчиков устройств, а использует драйверы из собственной базы данных.

Современные операционные системы позволяют управлять не только установкой и регистрацией программных драйверов устройств, но и процессом аппаратно-логического подключения, при этом реализуется принцип динамического распределения ресурсов операционной системой, который называется *plug-and-play*, а устройства, удовлетворяющие этому принципу, называются *самоустанавливающимися*.

Обслуживание компьютера

Предоставление основных средств обслуживания компьютера – одна из функций операционной системы. Обычно она решается внешним образом – включением в базовый состав операционной системы первоочередных служебных приложений.

4. ТЕКСТОВЫЙ ПРОЦЕССОР

Общее название программных средств, предназначенных для создания, редактирования и форматирования простых и комплексных текстовых документов, – *текстовые процессоры* [1].

К базовым приемам работы с текстами в текстовом процессоре Microsoft Word относятся следующие:

– создание документа;

– ввод текста;

– редактирование текста;

– рецензирование текста (редактирование текста с регистрацией изменений и комментирование текста, т.е. создание примечаний);

– форматирование текста (выбор и изменение гарнитуры шрифта, размера шрифта, начертания и цвета шрифта, метода выравнивания; управление параметрами абзаца);

– сохранение документа;

– печать документа.

Текстовый процессор Microsoft Word обладает развитой функциональностью по работе с объектами нетекстовой природы – формулами, таблицами, диаграммами, художественными заголовками, растровыми и векторными иллюстрациями, а также объектами мультимедиа.

Основные возможности и приемы работы с текстовым процессором Microsoft Word приведены в работе [2].

5. ЭЛЕКТРОННЫЕ ТАБЛИЦЫ

Для представления данных в удобном виде используют таблицы. Компьютер позволяет представлять их в электронной форме, а это дает возможность не только отображать, но и обрабатывать данные. Класс программ, используемых для этой цели, называется *электронными таблицами* (Microsoft Excel, LibreOffice Calc и др.) [1].

Особенность электронных таблиц заключается в возможности применения формул для описания связи между значениями различных ячеек. Расчет по заданным формулам выполняется автоматически. Изменение содержимого какой-либо ячейки приводит к пересчету значений всех ячеек, которые с ней связаны формулами.

Электронные таблицы эффективно используются:

– для проведения однотипных расчетов над большими наборами данных;

– автоматизации итоговых вычислений;

– решения задач путем подбора значений параметров;

– обработки результатов экспериментов;

– проведения поиска оптимальных значений параметров;

– подготовки табличных документов;

– построения диаграмм и графиков по имеющимся данным.

Основные возможности и приемы работы с Microsoft Excel приведены в работе [3].

6. АЛГОРИТМЫ И АЛГОРИТМИЗАЦИЯ

6.1. Алгоритм и его свойства

Под *алгоритмизацией* понимают сведение задачи к последовательности этапов, выполняемых друг за другом, так что результаты предыдущих этапов используются при выполнении последующих. *Алгоритм* – это четкое описание последовательности действий, которые необходимо выполнить для решения задачи.

Алгоритм обладает следующими свойствами: дискретностью, определенностью, результативностью, массовостью.

Дискретность. Процесс преобразования исходных данных в результат осуществляется дискретно, так что значения величин в каждый последующий момент времени получаются по определенным правилам из значений величин в предшествующий момент времени.

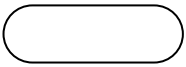
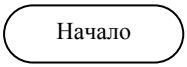
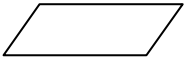
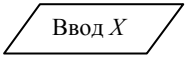
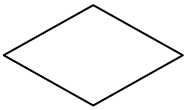
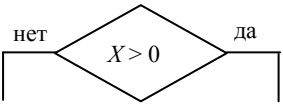
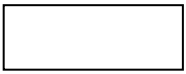
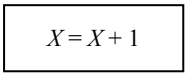
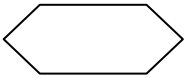
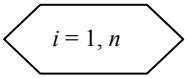
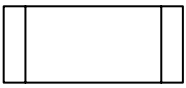
Определенность. Каждое правило алгоритма должно быть четким и однозначным, не допускающим двусмысленного толкования.

Результативность. Алгоритм должен приводить к результату за конечное число шагов.

Массовость. Алгоритм решения задачи разрабатывается в общем виде так, чтобы его можно было применить для класса задач, различающихся лишь исходными данными.

Программа – это окончательный вариант решения задачи на языке программирования.

Алгоритм решения задачи может быть представлен *графически* – в виде блок-схем. Блок-схема алгоритма представляет собой совокупность блоков, соединенных между собой линиями связи.

Символ	Описание	Пример
	Начало и окончание алгоритма	
	Ввод и вывод данных	
	Операция, определяющая выбор направления выполнения алгоритма	
	Обозначение операций присваивания	
	Обозначение заголовка цикла с параметром	
	Обозначение подпрограмм	

6.2. Основные структуры алгоритмов

Теория структурного программирования (гл. 12) доказывает, что алгоритм любой степени сложности можно построить с помощью основного базового набора структур [4].

К основным (базовым) структурам алгоритмов относятся: следование, разветвление, цикл, каждая из которых имеет один вход и один выход.

Следование – это последовательное размещение блоков или групп блоков (рис. 2, *а*). Например, два блока *S*1 и *S*2 могут быть размещены друг за другом, при этом каждый из них, в свою очередь, может быть любым из базовых структур.

Разветвление состоит из логического блока с проверкой некоторого условия *P* и блоков *S*1, *S*2. Разветвление может быть двух видов: полная условная конструкция (рис. 2, *б*) и неполная условная конструкция – обход (рис. 2, *в*). Полная условная конструкция применяется, когда в зависимости от условия *P* нужно выполнить либо *S*1, либо *S*2. Для структуры обход блок *S* или выполняется, или не выполняется.

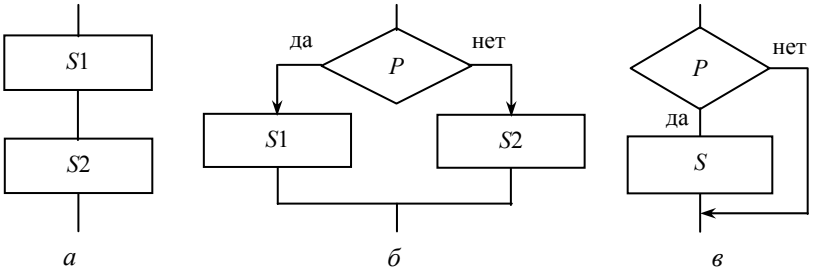


Рис. 2. Основные структуры алгоритмов:
а – следование; *б*, *в* – разветвление

Циклическими называются алгоритмы, у которых выполнение некоторых операторов (групп операторов) осуществляется многократно с одними и теми же или модифицированными данными.

В зависимости от способа организации числа повторений различают три типа циклов: цикл с заданным условием продолжения работы (цикл-ПОКА, рис. 3, *а*), цикл с заданным условием окончания работы (цикл-ДО, рис. 3, *б*) и цикл с заданным числом повторений (цикл с параметром, рис. 3, *в*).

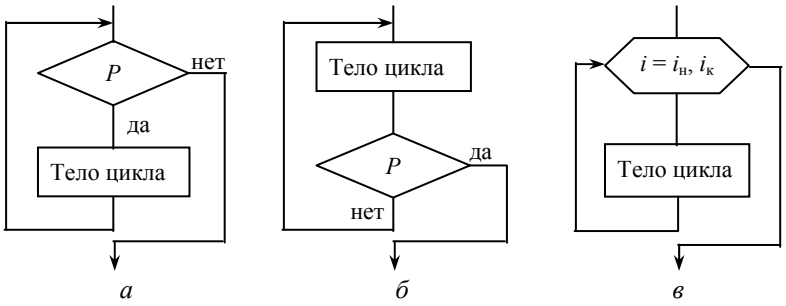


Рис. 3. Типы циклов: *а* – цикл-ПОКА (цикл с предусловием); *б* – цикл-ДО (цикл с постусловием); *в* – цикл с параметром

Тело цикла может включать в себя группу операторов любой степени сложности.

Для цикла-ПОКА (рис. 3, *а*) при выполнении условия *P* выполняется тело цикла; если же условие не выполняется, то работа циклической структуры заканчивается и начинает выполняться следующая структура основного алгоритма. В этом случае *P* – условие на продолжение цикла.

Структура цикла-ПОКА предусматривает вариант, когда тело цикла не выполняется ни разу. Такое возможно, если условие, стоящее в начале цикла, сразу же не выполняется. Когда при решении задач возникает необходимость использовать структуру, у которой тело цикла выполняется хотя бы один раз, то в этом случае применяется структура цикла-ДО (рис. 3, *б*). В этом случае *P* – условие окончания цикла, т.е. выход из цикла-ДО осуществляется при выполнении условия.

Выполнение цикла с параметром (рис. 3, *в*) осуществляется следующим образом: при изменении параметра *i* от начального значения $i_н$ до конечного значения $i_к$ повторяется тело цикла.

6.3. Алгоритмы линейной, разветвляющейся и циклической структуры

Алгоритмы линейной структуры. *Линейный* вычислительный процесс – это такой процесс, все вычисления которого выполняются последовательно одно за другим в естественном порядке.

Пример. Даны катеты прямоугольного треугольника a, b. Найти его гипотенузу c и площадь s. Блок-схема алгоритма приведена на рис. 4.

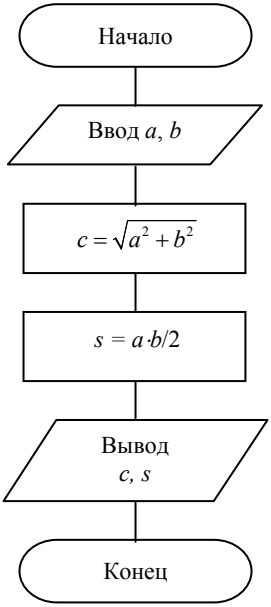


Рис. 4. Пример линейного алгоритма

Алгоритмы разветвляющейся структуры. *Разветвляющимся* (ветвящимся) называется алгоритм, который в зависимости от исходных данных или промежуточных результатов вычисления реализуется по одному из нескольких заранее предусмотренных направлений. Такие направления называются *ветвями вычислений*. Выбор той или иной ветви осуществляется в зависимости от результата проверки условия. В каждом конкретном случае алгоритм реализуется только по одной ветви, а выполнение остальных исключается.

В качестве примера использования ветвлений рассмотрим составление алгоритма для вычисления значения функции $f(x)$ в зависимости от конкретных значений x, a, b:

$$f = \begin{cases} x + a^2 & \text{при } x \le 1, \\ x - a \cdot b & \text{при } 1 < x < 4, \\ x + b^2 & \text{при } x \ge 4. \end{cases}$$

Блок-схема алгоритма решения этой задачи приведена на рис. 5.

Алгоритмы циклической структуры. Алгоритмы, в которых отдельные действия многократно повторяются, называются *циклическими*. Типовые алгоритмические структуры, реализующие циклический вычислительный процесс, приведены на рис. 3. Циклический алгоритм состоит из подготовки цикла, тела цикла и условия повторения цикла.

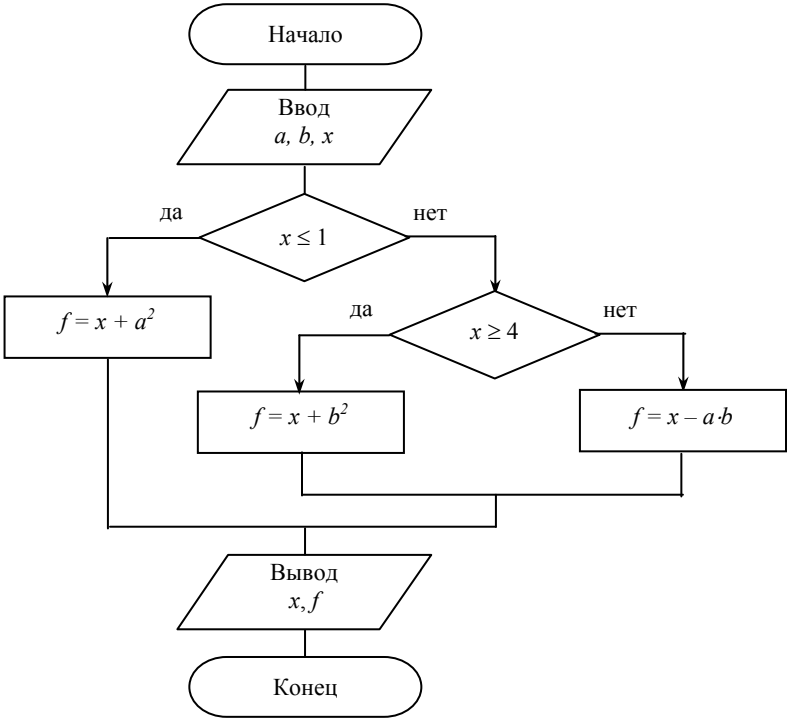


Рис. 5. Пример разветвляющегося алгоритма

Характерной для этого класса вычислительных процессов является задача табулирования функции, т.е. вычисления значений функции $y = f(x)$ на отрезке $[x_н, x_k]$ с шагом Δx (рис. 6).

Примером алгоритма циклической структуры является задача вычисления суммы и произведения.

Если необходимо вычислить сумму значений некоторой функции $y = f(x)$ при различных значениях аргумента, то целесообразно организовать цикл, в котором не только вычисляются текущие значения функции, но и накапливается их сумма путем прибавления полученного слагаемого к сумме предыдущих. Формула для вычисления суммы имеет следующий вид:

$$S_i = S_{i-1} + y_i .$$

При первом выполнении цикла вычисляется значение

$$S_1 = S_0 + y_1,$$

которое должно быть равно y_1. Поэтому перед циклом начальному значению суммы следует присвоить значение ноль, т.е. $S_0 = 0$.

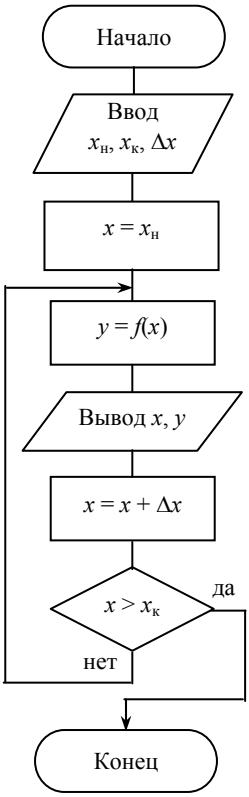


Рис. 6. Блок-схема алгоритма табулирования функции (циклический алгоритм)

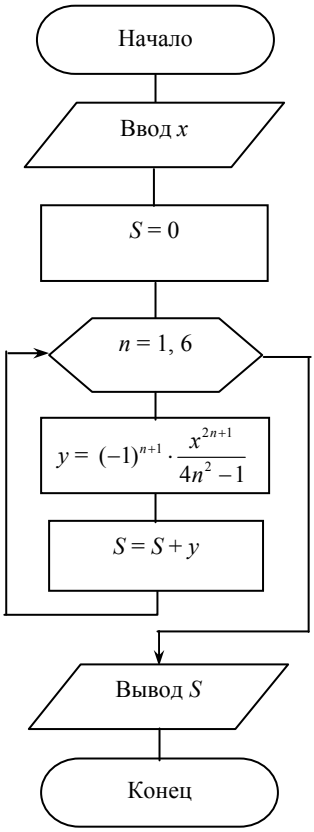


Рис. 7. Блок-схема алгоритма вычисления суммы членов ряда (циклический алгоритм)

Аналогично вычисляется и произведение, с той лишь разницей, что для его накопления используется формула

$$P_i = P_{i-1} \cdot y_i ,$$

а начальное значение произведения должно быть равно единице, т.е. $P_0 = 1$.

Пример. Вычислить $S = \quad = \sum_{n=1}^{6} (-1)^{n+1} \frac{x^{2n+1}}{4n^2 - 1}$, $x = 0,3$. Блок-схема алгоритма решения этой задачи представлена на рис. 7. Так как результат решения представляет собой одно число, то блок вывода результата стоит за циклом и результаты печатаются один раз.

Циклические алгоритмы с неизвестным числом повторений. Циклические процессы, в которых заранее неизвестно число повторений, а проверка выхода из цикла ведется по достижении требуемой точности, называются *итерационными*. Большинство численных (приближенных) методов решения многих математических задач являются итерационными: вычисления проводятся до тех пор, пока не будет достигнута заданная точность, при этом заранее не известно, сколько необходимо для этого совершить циклических операций. Аналогичная задача возникает при вычислении сумм рядов с бесконечным числом слагаемых.

Пример. Вычислить сумму членов сходящегося ряда $S = \sum_{k=1}^{\infty} \frac{x^k}{k^k}$, $x = 1,1$ с заданной точностью $\varepsilon = 10^{-3}$.

Вычисление суммы прекращается, если очередной член ряда y_k становится меньше заданной точности ε, т.е. выполнится условие $\left| \frac{x^k}{k^k} \right| < \varepsilon$. Блок-схема алгоритма решения этой задачи представлена на рис. 8. Блок вывода результата содержит также переменную k – количество членов ряда, вошедших в сумму.

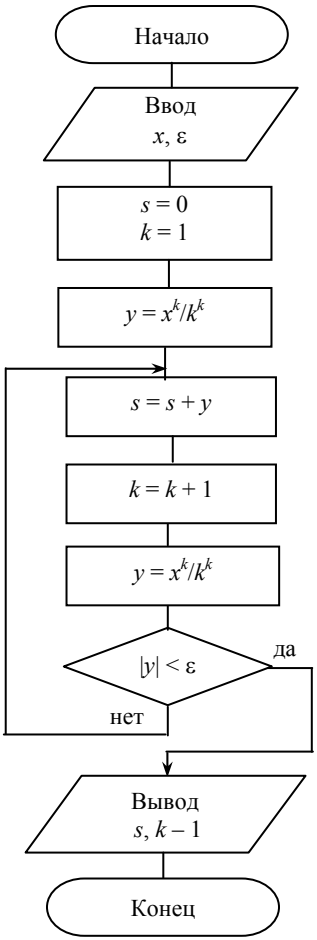


Рис. 8. Блок-схема итерационного
алгоритма вычисления суммы сходящегося ряда

Вложенные циклы. Существует возможность организовать цикл внутри тела другого цикла. Такой цикл называется *вложенным циклом*.

При этом цикл, содержащий в себе другой, называют *внешним*, а цикл, находящийся в теле первого, – *внутренним* (*вложенным*). Внутрь вложенного цикла, в свою очередь, может быть вложен еще один цикл, образуя следующий уровень вложенности и т.д.

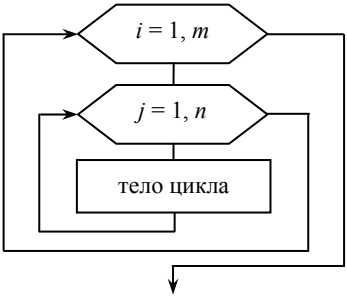


Рис. 9. Блок-схема вложенных циклов

Параметры внешнего и внутреннего циклов разные и изменяются не одновременно: при одном значении параметра внешнего цикла параметр внутреннего цикла принимает поочередно все значения (рис. 9). Затем значение параметра внешнего цикла изменяется на единицу, и опять от начала и до конца выполняется вложенный цикл. И так до тех пор, пока значение параметра внешнего цикла не станет больше своего конечного значения.

При программировании вложенных циклов необходимо соблюдать следующее дополнительное условие: все операторы внутреннего цикла должны полностью располагаться в теле внешнего цикла.

6.4. Параллельные алгоритмы

Параллельный алгоритм, противопоставляемый традиционным последовательным алгоритмам, – алгоритм, который может быть реализован по частям на множестве различных вычислительных устройств (процессоров) с последующим объединением полученных результатов.

Некоторые алгоритмы достаточно просто поддаются разбиению на независимо выполняемые фрагменты. Например, распределение работы по проверке всех чисел от 1 до 100000 на предмет того, какие из них являются простыми, может быть выполнено путем назначения каждому доступному процессору некоторого подмножества чисел с последующим объединением полученных множеств простых чисел.

С другой стороны, большинство известных алгоритмов вычисления значения числа π не допускают разбиения на параллельно выполняемые части, так как требуют результата предыдущей итерации выполнения алгоритма. Итерационные числен-

ные методы также являются сугубо последовательными алгоритмами. Некоторые примеры рекурсивных алгоритмов достаточно сложно поддаются распараллеливанию.

К 2005 г. производительность одного процессора достигла некоторого предела, в котором дальнейший рост производительности ограничивался физическими законами. Появление многоядерных систем позволило преодолеть этот барьер, однако это потребовало новой парадигмы написания приложений и алгоритмов – начали активно развиваться и применяться параллельные алгоритмы.

Сложность последовательных алгоритмов выражается в объеме используемой памяти и времени (числе тактов процессора), необходимых для выполнения алгоритма. Параллельные алгоритмы требуют учета использования ещё одного ресурса: подсистемы связей между различными процессорами. Существует два способа обмена между процессорами: использование общей памяти и системы передачи сообщений.

Системы с общей памятью требуют введения дополнительных блокировок для обрабатываемых данных, налагая определенные ограничения при использовании дополнительных процессоров.

Системы передачи сообщений используют понятия каналов и блоков сообщений, что создает дополнительный трафик на шине и требует дополнительных затрат памяти для организации очередей сообщений.

Еще одной проблемой, связанной с использованием параллельных алгоритмов, является балансировка нагрузки. Например, поиск простых чисел в диапазоне от 1 до 100000 легко распределить между имеющимися процессорами, однако некоторые процессоры могут получить больший объем работы, в то время как другие закончат обработку раньше, и будут простаивать. Проблема балансировки нагрузки ещё больше усугубляется при использовании гетерогенных вычислительных сред, в которых вычислительные элементы существенно отличаются по производительности и доступности.

Разновидность параллельных алгоритмов, называемая распределенными алгоритмами, специально разрабатывается для

применения на кластерах и в распределенных вычислительных системах с учетом ряда особенностей подобной обработки.

Основной стратегией развития вычислительной техники является создание многопроцессорных вычислителей. Получается, что параллельные алгоритмы – единственный перспективный способ повышения производительности при решении задач.

На рис. 10 приведен пример одной из моделей написания параллельных алгоритмов для систем с общей памятью – так называемый fork-join-параллелизм. Суть этой модели заключается в том, что внутри последовательной программы выделяются секции, в рамках которых происходит создание некоторого количества параллельных потоков исполнения. Операция fork в данном подходе отвечает за создание параллельных потоков и за распределение по ним работы. Операция join ожидает завершения всех рабочих потоков и при необходимости объединяет результаты работы отдельных потоков в общий результат параллельной секции. Примером технологии, которая реализует модель fork-join-параллелизма, является OpenMP – открытый стандарт для распараллеливания программ на языках C, C++, Фортран. Он дает описание совокупности директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для программирования многопоточных приложений на многопроцессорных системах с общей памятью.

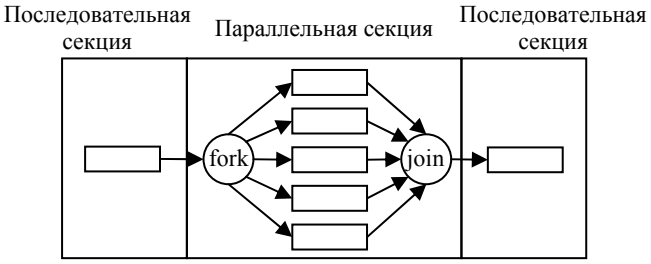


Рис. 10. Модель fork-join-параллелизма

Message Passing Interface (MPI, интерфейс передачи сообщений) – программный интерфейс (API) для передачи информа-

ции, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу.

MPI является наиболее распространённым стандартом интерфейса обмена данными в параллельном программировании, существуют его реализации для большого числа компьютерных платформ. Используется при разработке программ для кластеров и суперкомпьютеров. Основным средством коммуникации между процессами в MPI является передача сообщений друг другу.

Стандартизацией MPI занимается MPI Forum. В стандарте MPI описан интерфейс передачи сообщений, который должен поддерживаться как на платформе, так и в приложениях пользователя. В настоящее время существует большое количество бесплатных и коммерческих реализаций MPI. Существуют реализации для языков Фортран 77/90, Java, C и C++.

В первую очередь MPI ориентирован на системы с распределенной памятью, т.е. когда затраты на передачу данных велики, а OpenMP ориентирован на системы с общей памятью. Обе технологии могут использоваться совместно, чтобы оптимально использовать в кластере многоядерные системы.

7. ПРОГРАММНЫЕ СРЕДСТВА РЕАЛИЗАЦИИ АЛГОРИТМОВ

7.1. Языки программирования

Язык программирования – искусственный язык, который имеет ограниченное число «слов», значение которых понятно транслятору, и очень строгие правила записи команд (операторов).

Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности (разные типы процессоров имеют разные наборы команд), то он называется *языком программирования низкого уровня*. Таким языком является *язык ассемблера*. Подобные языки обычно применяют для написания небольших системных приложений, драйверов устройств, модулей стыковки с нестандартным оборудованием, когда одним из важнейших требований становится возможность прямого доступа к аппаратным ресурсам.

Языки программирования высокого уровня значительно ближе и понятнее человеку, нежели компьютеру. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому создаваемые программы на уровне исходных текстов легко переносимы на другие платформы, для которых создан транслятор этого языка.

Языками программирования высокого уровня являются: Фортран, Кобол, Алгол, Pascal, Basic, C, C++, Java, C#.

Языки программирования, используемые в базах данных, предназначены для работы с хранилищами данных, со встроенной функциональностью для хранения и обработки данных. Для управления большими базами данных и их эффективной обработки разработаны СУБД (системы управления базами данных). Основным языком для работы с данными в СУБД является *структурированный язык запросов SQL*.

Однако в промышленных СУБД, кроме поддержки языка SQL, есть поддержка собственных языков, расширяющих возможности SQL. Примерами таких языков могут служить в СУБД Oracle: PL/SQL, в СУБД PostgreSQL: PL/pgSQL.

Алфавит, синтаксис и семантика

Любой язык программирования имеет три основные составляющие: алфавит, синтаксис и семантику [5].

Алфавит языка программирования – набор допустимых символов, которые можно использовать при записи программ (буквы, цифры и специальные символы).

Синтаксис языка программирования – совокупность правил, определяющих допустимые конструкции языка, его форму. Пример синтаксической ошибки – неправильно записанный, например, оператор цикла.

Семантика языка – совокупность правил, определяющих смысл синтаксически корректных конструкций языка, его содержание. Пример семантической ошибки – неправильно записанное на языке программирования арифметическое выражение, что приводит к неправильному порядку выполнения в нем арифметических операций.

Трансляция, интерпретация и компиляция программ

С помощью языка программирования создается текст программы, описывающий ранее разработанный алгоритм решения задачи. Затем этот текст программы специальными служебными приложениями (*трансляторами*) либо сначала переводится в машинный код (для этого служат программы-*компиляторы*), который затем используется отдельно от исходного текста программы, либо сразу покомандно исполняется (этим занимаются программы-*интерпретаторы*).

Интерпретатор берет очередной оператор языка из текста программы, анализирует его структуру и затем сразу исполняет. Только после того, как текущий оператор успешно выполнен, интерпретатор перейдет к следующему.

Компиляторы полностью обрабатывают весь текст программы. Они просматривают его в поисках синтаксических ошибок, выполняют определенный смысловой анализ и затем автоматически переводят (транслируют) на машинный язык – генерируют машинный код, который затем исполняется непосредственно процессором.

Системы программирования

Для создания программы на выбранном языке программирования необходимо иметь следующие компоненты:

– текстовый редактор;
– компилятор;
– редактор связей;
– библиотеки функций;
– отладчик (пошаговое выполнение).

Все это представляет собой интегрированную систему программирования.

Этапы решения задачи на компьютере

1. Формализация данных.
2. Создание математической модели.
3. Детальное описание алгоритма (блок-схема).

4. Реализация на языке программирования.
5. Отладка программы.
6. Тестирование программы.
7. Анализ результатов работы программы.

7.2. Программирование на языке Pascal

PascalABC.NET – это язык программирования Pascal нового поколения, включающий классический Pascal, большинство возможностей языка Delphi, а также ряд собственных расширений. Он реализован на платформе Microsoft.NET и содержит все современные языковые средства: классы, перегрузку операций, интерфейсы, обработку исключений, обобщенные классы и подпрограммы, средства параллельного программирования [6].

PascalABC.NET – это также простая и мощная интегрированная среда разработки, содержащая средства автоформатирования, встроенный отладчик и встроенный дизайнер форм.

Операторы языка программирования

Служат для описания некоторого законченного этапа обработки данных. Операторы определяют действия над объектами программы. Операторы могут быть простыми и структурированными, т.е. состоящими из других операторов. Операторы, следующие в программе друг за другом, разделяются точкой с запятой (`;`).

Идентификаторы

Выполняют роль имен, которые даются различным программным объектам – константам, типам, переменным величинам, функциям и т. п., чтобы программисту было удобно ссылаться на эти объекты. Идентификатор может начинаться только с буквы или знака подчеркивания. Пробелы и другие специальные символы недопустимы в идентификаторах. Компилятор не делает различия между прописными и строчными буквами.

Типы данных

Каждая константа, переменная, элемент массива принадлежат к определенному типу данных. Тип определяет форму внутреннего представления, множество принимаемых значений и множество допустимых операций. Тип констант распознается компилятором автоматически. Тип переменных должен быть описан в разделе описаний (в разделе переменных, начинающемся со специального слова `var`). PascalABC.NET характеризуется разветвленной структурой типов данных (табл. 1). В табл. 1 представлена информация о простых типах данных. Для вещественных типов в скобках указано количество сохраняемых значащих цифр мантиссы в десятичном представлении числа.

Таблица 1

Типы данных языка программирования PascalABC.NET

Идентификатор	Длина, байт	Диапазон (множество) значений
Целые типы		
`shortint`	1	–128…127
`smallint`	2	–32768…32767
`integer, longint`	4	–2147483648…2147483647
`byte`	1	0…255
`word`	2	0…65535
Вещественные типы		
`real, double`	8	$-1{,}8\cdot10^{308}\ldots1{,}8\cdot10^{308}$ (15–16)
`singl`	4	$-3{,}4\cdot10^{38}\ldots3{,}4\cdot10^{38}$ (7–8)
Логический тип		
`boolean`	1	true (истина), false (ложь)
Символьный тип		
`char`	2	Unicode-символ

Арифметическое выражение

Задает порядок выполнения действий над элементами данных и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций. Выражение

должно содержать данные одного типа, при этом значение выражения получается того же типа. Однако допускается использование в одном выражении данных целого и вещественного типов, результат в этом случае получается вещественного типа.

Арифметические операции приведены в табл. 2. Операции выполняются в соответствии с их приоритетом. Операции с равным приоритетом выполняются слева направо. В случае необходимости изменения порядка выполнения операции используются круглые скобки.

Таблица 2

Арифметические операции языка программирования PascalABC.NET

Знак операции	Содержание	Выражение	Тип операндов	Тип результата
`+`	сложение	`A + B`	целый, вещественный	целый, вещественный
`-`	вычитание	`A - B`	целый, вещественный	целый, вещественный
`*`	умножение	`A * B`	целый, вещественный	целый, вещественный
`/`	деление	`A/B`	целый, вещественный	целый, вещественный
`div`	целочисленное деление	`A div B`	целый	целый
`mod`	остаток	`A mod B`	целый	целый

Операция `div` возвращает целую часть частного, дробная часть отбрасывается. Операция `mod` восстанавливает остаток, полученный при выполнении целочисленного деления. Например, значение выражения `10 div 3` будет равно 3, а значение выражения `10 mod 3` будет равно 1.

Стандартные функции

В языке Pascal существуют заранее разработанные подпрограммы-функции (табл. 3), которые могут использоваться в программах. Аргументы функций записываются в круглых скобках.

Таблица 3

Стандартные функции языка программирования PascalABC.NET

Обозначение	Тип аргумента	Тип результата	Функция
`Pi`	-	`real`	число π = 3,1415926536
`abs(x)`	`integer/real`	`integer/real`	модуль аргумента
`arctan(x)`	`integer/real`	`real`	арктангенс (радианы)
`cos(x)`	`integer/real`	`real`	косинус (*x* в радианах)
`sin(x)`	`integer/real`	`real`	синус (*x* в радианах)
`exp(x)`	`integer/real`	`real`	экспонента e^x
`frac(x)`	`integer/real`	`real`	дробная часть *x*
`int(x)`	`integer/real`	`real`	целая часть *x*
`ln(x)`	`integer/real`	`real`	натуральный логарифм
`random`		`real`	псевдослучайное число из диапазона [0,1)
`random(x)`	`integer`	`integer`	псевдослучайное число из диапазона [0,*x*)
`round(x)`	`real`	`integer`	округление до ближайшего целого
`sqr(x)`	`integer/real`	`integer/real`	квадрат *x*
`sqrt(x)`	`integer/real`	`real`	корень квадратный
`trunc(x)`	`real`	`integer`	ближайшее целое, не превышающее *x* по модулю

Для выполнения операции возведения в степень a^b используется следующая математическая формула: $a^b = e^{b\cdot\ln(a)}$. На языке Pascal это будет выглядеть так: `exp(b*ln(a))`.

Можно также воспользоваться функцией *`function Power(const Base, Exponent: Extended): Extended`*, которая возводит число *A* в степень *B*. Например, *`Power(A, B);`*

Вычисление tg(*x*) производится с помощью выражения `sin(x)/cos(x)`.

Вычисление $\log_b a$ производится с помощью выражения: `ln(a)/ln(b)`.

Например, запишем по правилам языка Pascal математическое выражение: $\frac{a+b\cdot x}{\sqrt{x^2+1}}$.

На Pascal это выглядит так: `(a+b*x)/sqrt(x*x+1)`.

Структура программы на языке Pascal

Программа на языке Pascal состоит из *заголовка* программы и *тела* программы (*блока*), оканчивающегося точкой. В свою очередь, блок содержит *разделы описаний* и *раздел операторов*:

<table>
<tr><th colspan="3">Структура программы</th><th>Пример</th></tr>
<tr><td colspan="3">Заголовок программы</td><td>program Z1;</td></tr>
<tr><td colspan="3">Оператор uses</td><td>uses GraphABC;</td></tr>
<tr><td rowspan="6">Тело программы (блок)</td><td rowspan="5">Описание данных</td><td>1. Раздел описания меток (label)</td><td>label 10, 20;</td></tr>
<tr><td>2. Раздел описания констант (const)</td><td>const n=10;</td></tr>
<tr><td>3. Раздел описания типов (type)</td><td>type vector=array[1..n] of real;</td></tr>
<tr><td>4. Раздел описания переменных (var)</td><td>var x, y: real;
i, j: integer;
a, b: vector;</td></tr>
<tr><td>5. Раздел описания процедур и функций (procedure, function)</td><td>procedure Tab;
.........</td></tr>
<tr><td>Описание действий</td><td>6. Раздел операторов</td><td>begin
оператор 1;
оператор 2;
....
оператор N;
end.</td></tr>
</table>

Раздел операторов является основным и присутствует в любой программе. Раздел `uses` и разделы описаний (все или часть) могут отсутствовать. Разделы `const`, `type`, `var` могут следовать друг за другом в любом порядке.

В `{ }` или `(* *)` записываются комментарии к программе. Комментарий не определяет никаких действий программы и является лишь пояснительным текстом. Комментарием также считается любая последовательность символов после символов `//` и до конца строки.

Операторы ввода/вывода

Ввод данных – это передача информации от внешних устройств в оперативную память. Вводятся, как правило, исходные

данные решаемой задачи. Вывод – обратный процесс, когда данные (результаты решения задачи) передаются из оперативной памяти на внешние устройства.

Оператор ввода имеет следующий формат:

```
read(список ввода);
```

Список ввода – это последовательность имен переменных, разделенных запятыми. При выполнении этого оператора происходит остановка работы компьютера, пользователь должен ввести с клавиатуры необходимые значения переменных в том порядке, в каком они записаны в списке ввода, разделяя их пробелами. При этом вводимые значения отображаются на экране. Заканчивается ввод нажатием клавиши ENTER.

Например,

```
read(a,b,c);
```

Другой вариант оператора ввода имеет вид

```
readln(список ввода);
```

Этот оператор отличается тем, что после считывания последнего значения в списке ввода для одного оператора `readln` данные для следующего оператора ввода будут считываться с начала новой строки.

Оператор вывода имеет следующий формат:

```
write(список вывода);
```

Список вывода представляет собой выражения различных типов, разделенные запятыми.

Например,

`write(25);`	выводится целая константа
`write('Результаты');`	выводится строковая константа
`write(2*a+1);`	выводится значение выражения
`write('сумма=', s);`	выводится список, состоящий из строковой константы и значения переменной *s*
`write(a1, '  ',a2, '  ',a3);`	выводятся значения переменных

Другая форма оператора вывода

```
writeln(список вывода);
```

В этом случае после вывода всех значений из списка вывода происходит переход на новую строку. Оператор `writeln` без параметров означает переход на новую строку.

Рекомендуется ввод данных оформлять, например, следующим образом:

```
write('введите коэффициенты квадратного уравнения');
readln(a,b,c);
```

Оператор присваивания имеет следующий вид:

```
переменная:=выражение;
```

Оператор присваивания заменяет текущее значение переменной значением выражения.

Переменная и выражение должны быть одного типа. Исключение составляет случай, когда выражение имеет целый тип, а переменная – вещественный.

Пример. Поменять местами значения переменных *a* и *b* (при этом используется дополнительная переменная *p*).

```
p:=a;    a:=b;    b:=p;
```

Программирование алгоритмов разветвляющейся и циклической структуры

Условный оператор

В языке Pascal управляющая структура языка, проверяющая выполнение некоторого условия и позволяющая в случае выполнения или невыполнения этого условия произвести ветвление алгоритма, реализуется с помощью условного оператора, полная форма которого следующая:

```
if условие then оператор1 else оператор2;
```

Здесь `if` – если; `then` – то; `else` – иначе. Операторы 1 и 2 могут быть как простыми, так и составными, представляющими собой

определенную последовательность операторов. Составной оператор следует заключать в операторные скобки из слов begin и end.

В качестве условия указывается некоторое логическое выражение. Если условие оказывается истинным, то выполняется оператор1, в противном случае выполняется оператор 2.

Пример вычисления наибольшего (t1) и наименьшего (t2) значения из двух чисел a и b:

```
if a>b then
begin
    t1:=a; t2:=b
end
else
begin
    t1:=b; t2:=a
end;
```

Краткая форма условного оператора

```
if условие then оператор;
```

В этом случае, если условие истинно, выполняется оператор, в противном случае происходит переход к следующему оператору программы.

Операторы цикла

В языке Pascal циклические алгоритмы могут быть записаны с помощью следующих операторов.

Цикл с параметром (с шагом +1)

```
for параметр:=начальное значение to конечное значение do оператор;
```

Здесь for, to, *do* – служебные слова. В начале выполнения цикла параметру присваивается начальное значение. Затем значение параметра цикла сравнивается с конечным значением. Если параметр цикла меньше или равен этому значению, то выполняется тело цикла (после чего значение па-

раметра увеличивается на единицу), в противном случае выполнение цикла заканчивается.

Если тело цикла – составной оператор, т.е. содержит более одного оператора, то оно оформляется с использованием операторных скобок `begin...end`.

Цикл с параметром (с шагом –1)

`for` *параметр*`:=`*начальное значение* `downto` *конечное значение* `do` *оператор*`;`

В этом случае *параметр* цикла изменяется по убыванию, т.е. при каждом повторении цикла *параметр* уменьшает свое значение на единицу.

Цикл с предусловием (цикл-ПОКА)

```
while условие do оператор;
```

Здесь `while` (пока); `do` (делать) – служебные слова. `Условие` представляет собой выражение логического типа, а оператор после `do` является телом цикла. Если условие истинно, то выполняется тело цикла, в противном случае происходит выход из цикла.

Для того чтобы цикл не стал бесконечным (не зациклился), необходимо наличие в теле цикла оператора, влияющего на значение логического выражения.

Если тело цикла – составной оператор, т.е. содержит более одного оператора, то оно оформляется с использованием операторных скобок `begin...end`.

Цикл с постусловием (цикл-ДО)

```
repeat оператор until условие;
```

Здесь `repeat` (повторять); `until` (до) – служебные слова. Если условие истинно, то происходит выход из цикла, т.е. `оператор` (тело цикла) выполняется до тех пор, пока `условие` ложно.

В этом случае операторы тела цикла выполняются по крайней мере один раз.

В теле цикла можно указывать несколько операторов без использования операторных скобок.

Пример использования операторов цикла при программировании циклических алгоритмов (табл. 4).

Пусть требуется вычислить сумму первых *n* натуральных чисел

$$S = \sum_{i=1}^{n} i .$$

Переменная цикла *i* будет выполнять две функции: номер очередного слагаемого и одновременно его значение.

Таблица 4

Пример использования операторов цикла для решения задачи

Цикл с параметром	Цикл с предусловием	Цикл с постусловием

program Z1;
var S, i, n: integer;
begin
writeln('задайте кол-во слагаемых');
readln(n);
S:=0;
for i:=1 to n do
S:=S+i;
writeln('сумма= ', S);
end.
	program Z2;
	var S, i, n: integer;
	begin
	readln(n);
	S:=0;
	i:=1;
	while i <= n do
	begin
	S:=S+i;
	i:=i+1
	end;
	writeln('сумма= ', S);
	end.
		program Z3;
		var S, i, n: integer;
		begin
		readln(n);
		S:=0;
		i:=1;
		repeat
		S:=S+i;
		i:=i+1
		until i > n
		writeln('сумма= ', S);
		end.

Массивы

Массивы представляют собой упорядоченную совокупность данных, имеющую одно имя. Каждому элементу массива соответствует выражение порядкового типа (чаще целое число), определяющее место этого элемента в массиве, которое называется *индексом*. *Размерность* массива – количество индексов, необходимое

для однозначного доступа к элементу массива. Если для определения места элемента в массиве используется один индекс, то массив называют *одномерным* (вектором), два – *двумерным* (матрицей). В языке Pascal индекс заключается в квадратные скобки. Индекс может быть константой – `a[5], b[1,1]`; переменной – `a[i], b[i,j]`; выражением – `a[i+3], b[i+1,j+1]`.

Одномерные массивы

Массивы описываются в разделе описания переменных в следующей форме:

```
var имя_массива: array[тип_индекса] of тип_элементов;
```

В качестве `типа индекса` чаще всего используется тип-диапазон – `[n..k]`.

Количество элементов в диапазоне определяется следующим образом: `k-n+1`.

Пример:

```
var a: array[1..10] of integer;
```

Здесь `a` – имя массива, элементы которого имеют базовый тип `integer`, первый элемент имеет индекс 1, индекс последнего элемента 10, всего 10 элементов.

В языке Pascal есть возможность создавать свои типы данных, которые должны быть описаны в специальном разделе описания типов `type`.

Пример:

```
const n=10;
type vector=array[1..n] of real;
                        {тип vector объединяет
                        в себе все одномерные
                        массивы, состоящие из
                        n действительных
                        элементов}
var a, b: vector;
```

Ввод и вывод массивов в языке Pascal осуществляется поэлементно, для чего необходимо организовать цикл.

Примеры ввода элементов одномерного массива (с клавиатуры и с помощью датчика случайных чисел).

```
for i:=1 to n do
  read(a[i]);
```

```
randomize;
for i:=1 to n do
begin
  a[i]:=1+random(100);
  write(a[i]:5,'  ');
end;
```

Пример вывода элементов одномерного массива:

```
for i:=1 to n do
    write(a[i], '  ');
```

Перебор элементов одномерного массива

Элементы массива можно обрабатывать, двигаясь от начала массива к его концу или в обратном направлении:

```
for i:=1 to n do
  {обработка a[i]}
```

```
for i:=n downto 1 do
  {обработка a[i]}
```

Можно обрабатывать элементы массива сразу по два элемента, двигаясь одновременно с обеих сторон:

```
i:=1;  {задание нижней границы индекса}
j:=n;  {задание верхней границы индекса}
while i<=j do
begin
  {обработка a[i] и a[j]};
  i:=i+1;    {движение слева направо, индекс
              увеличивается}
  j:=j-1;    {движение справа налево, индекс
              уменьшается}
end;
```

Если необходимо перебирать только элементы массива с четным номером, то это может быть реализовано следующим образом:

Вариант 1.

```
i:=2;          {индекс начинает изменяться с
                четного числа 2}
while i<=n do
begin
  {обработка a[i]};
  i:=i+2;      {величина шага, равная двум,
                обеспечивает сохранение четности
                индекса}
end;
```

Вариант 2.

```
for i:=1 to n do
  if i mod 2 = 0 then {обработка a[i]};
{внутрь цикла перебора индекса вложен опера-
тор, проверяющий его четность}
```

Вариант 3.

```
for i:=1 to n div 2 do
{обработка a[2*i]}; {используется формула чет-
                     ного числа. Поскольку эле-
                     ментов с четным индексом –
                     половина от всего количе-
                     ства, то параметр цикла i
                     изменяется до n div 2}
```

Двумерные массивы (матрицы)

Для работы с элементами двумерного массива нужно организовать два цикла. Каждый из них отвечает за перебор значений соответствующего индекса, т.е. получается структура вложенных циклов. Параметры внешнего и внутреннего циклов

разные и изменяются не одновременно: при одном значении параметра внешнего цикла параметр внутреннего цикла принимает поочередно все значения.

Двумерный массив `A(m,n)` с количеством строк, равным `m`, и количеством столбцов, равным `n`, содержащий элементы целого типа (`integer`), описывается в разделе описания переменных в следующей форме:

```
var a: array[1..m,1..n] of integer;
```

К элементу двумерного массива обращаются следующим образом: `a[i,j]`, где `i` – номер строки; `j` – номер столбца. Обрабатывать элементы двумерного массива можно как по строкам, так и по столбцам:

Обработка по строкам	Обработка по столбцам

`for i:=1 to m do {перебор строк}`
`  for j:=1 to n do {перебор столбцов}`
`  {обработка a[i,j]}`
	`for j:=1 to n do {перебор столбцов}`
	`  for i:=1 to m do {перебор строк}`
	`  {обработка a[i,j]}`

Примеры ввода элементов двумерного массива (с клавиатуры и с помощью датчика случайных чисел):

```
for i:=1 to m do
 for j:=1 to n do
   read(a[i,j]);
```

```
randomize;
for i:=1 to m do
begin
 for j:=1 to n do
 begin
  a[i,j]:=1+random(100);
  write(a[i,j]:5,'  ');
 end;
 writeln;
end;
```

Пример вывода элементов двумерного массива в виде матрицы:

```
for i:=1 to m do
begin
    for j:=1 to n do
        write(a[i,j]:5, '  ');
    writeln;
end;
```

Для двумерного массива можно использовать те же схемы перебора, что и для одномерного, но комбинаций здесь будет в два раза больше.

Более подробная информация по теме «Массивы» представлена в работе [7].

Процедуры и функции (подпрограммы)

Кроме стандартных функций и процедур, язык Pascal позволяет пользователю формировать свои процедуры и функции. Их целесообразно создавать, если при решении задачи возникает необходимость проводить вычисления по одним и тем же формулам (алгоритмам) многократно, например находить корни квадратного уравнения $ax^2 + bx + c = 0$ при различных значениях коэффициентов a, b и c.

В языке Pascal определяются два типа подпрограмм – процедуры и функции. Любая подпрограмма обладает той же структурой, которой обладает и вся программа.

При вызове подпрограммы выполнение основной программы приостанавливается и управление передается в подпрограмму. По окончании работы подпрограммы управление возвращается основной программе.

Основное различие между процедурой и функцией состоит в том, что процедура только выполняет какую-либо законченную последовательность действий, не возвращая результата работы в основную программу, а функция и выполняет действия, и возвращает результат.

Все переменные, которые использует подпрограмма, могут быть либо *глобальные*, т.е. объявленные в основной программе и доступные как программе, так и всем ее подпрограммам, либо *локальные*, объявленные внутри подпрограммы и доступные только ей самой.

При использовании процедур (функций) необходимо различать:

– *описание* процедуры (функции);

– *оператор вызова* процедуры (функции).

Описание процедуры (функции) дается в разделе описания процедур и функций, который должен располагаться после раздела описания переменных (см. п. «Структура программы на языке Pascal»).

Оператор вызова процедуры (функции) ставится в основной программе (в разделе операторов) и служит для активизации процедуры (функции).

Параметры, указываемые при описании подпрограммы, называются *формальными*. Параметры, указываемые при вызове подпрограммы, называются *фактическими*.

Формальные параметры – это наименования переменных, через которые передается информация из программы в процедуру либо из процедуры в программу.

Фактические параметры – это наименования переменных, значения которых при обращении к процедуре присваиваются соответствующим формальным параметрам.

Между фактическими и формальными параметрами должно быть взаимно однозначное соответствие по количеству, порядку следования и типу.

Если формальный параметр описан с предваряющим ключевым словом `var` или `const`, то его называют *параметром-переменной* и говорят, что он передается по ссылке. Если же параметр описан без слов `var` или `const`, то его называют *параметром-значением* и говорят, что он передается по значению.

Если параметр передается по значению, то при вызове подпрограммы значения фактических параметров присваиваются соответствующим формальным параметрам.

Если параметр передается по ссылке, то при вызове подпрограммы фактический параметр заменяет собой в теле процедуры соответствующий ему формальный параметр. В итоге любые изменения формального параметра-переменной внутри процедуры приводят к соответствующим изменениям фактического параметра. При передаче параметра по ссылке в подпрограмму передается адрес фактического параметра. Поэтому если параметр занимает много памяти (массив, запись, строка), то обычно он также передается по ссылке. В результате в процедуру передается не сам параметр, а его адрес, что экономит память и время работы. При этом если параметр меняется внутри подпрограммы, то он передается с ключевым словом `var`, если не меняется – с ключевым словом `const`.

Описание процедуры имеет вид

```
procedure имя (список формальных параметров);
раздел описаний
begin
   операторы
end;
```

Список формальных параметров вместе с окружающими скобками может отсутствовать.

Подпрограмма один раз описывается и может быть многократно вызвана. Для вызова процедуры используется `оператор вызова` – имя процедуры и в круглых скобках список фактических параметров:

```
имя процедуры (список фактических параметров);
```

Пример. Написать процедуру ввода элементов матрицы A.

```
program Z1;
const u=100;
var n, m: integer;
       a: array[1..u, 1..u] of integer;
procedure TAB(var g, k: integer);
var i, j: integer;
begin
   for i:=1 to g do
       for j:=1 to k do
            read (a[i, j];
end;
begin
   write('Введите размер матрицы n×m');
   read(n, m);
   TAB(n, m);
end.
```

Часть алгоритма решения задачи можно оформить как функцию, если в качестве результата выполнения функции необходимо получить одно единственное число. Поскольку функция предназначена для вычисления какого-либо значения, необходимо указать тип результата, возвращаемого функцией вызывающей программе.

Описание функции имеет вид

```
function имя (список формальных параметров):
тип возвращаемого значения;
раздел описаний
begin
   операторы
end;
```

Результаты выполнения функции записываются в ячейку памяти, имя которой совпадает с именем функции. Поэтому в разделе операторов функции должен обязательно присутствовать оператор вида:

```
имя функции:=результат;
```

Для вызова функции необходимо указать ее имя со списком фактических параметров в любом выражении вызывающей программы.

Пример. Написать функцию, которая из двух целых чисел выбирает наименьшее число.

```
program Z2;
var i, j: integer;
function MIN(i, j: integer): integer;
   begin
       if  i<j  then MIN:=i  else  MIN:=j;
   end;
begin
   readln(i, j);
   writeln('MIN=', MIN(i,j));
end.
```

Модуль GraphABC

Модуль GraphABC представляет собой простую графическую библиотеку и предназначен для создания графических и анимационных программ. Рисование осуществляется в специальном *графическом окне*.

В модуле GraphABC определен ряд констант, типов, процедур, функций и классов для рисования в графическом окне. Они подразделяются на следующие группы:

– графические примитивы;

– функции для работы с цветом;

– цветовые константы;

– действия с пером: процедуры и функции;

– действия с кистью: процедуры и функции;

– стили штриховки кисти;

– действия со шрифтом: процедуры и функции;

– стили шрифта.

Графические примитивы представляют собой процедуры, осуществляющие рисование в графическом окне. Рисование осуществляется текущим пером (линии), текущей кистью (заливка замкнутых областей) и текущим шрифтом (вывод строк).

Некоторые процедуры модуля GraphABC, с помощью которых осуществляется рисование графических примитивов:

`procedure SetPixel(x,y:integer; c:Color);` – закрашивает пиксел с координатами (*x,y*) цветом *c*;

`procedure PutPixel(x,y:integer; c:Color);` – закрашивает пиксел с координатами (*x,y*) цветом *c*;

`function GetPixel(x,y:integer): Color;` – возвращает цвет пиксела с координатами (*x,y*);

`procedure MoveTo(x,y:integer);` – устанавливает текущую позицию рисования в точку (*x,y*);

`procedure LineTo(x,y:integer);` – рисует отрезок от текущей позиции до точки (*x,y*). Текущая позиция переносится в точку (*x,y*);

`procedure LineTo(x,y:integer; c:Color);` – рисует отрезок от текущей позиции до точки (*x,y*) цветом *c*. Текущая позиция переносится в точку (*x,y*);

`procedure Line(x1,y1,x2,y2:integer);` – рисует отрезок от точки (*x1,y1*) до точки (*x2,y2*);

`procedure Line(x1,y1,x2,y2:integer; c:Color);` – рисует отрезок от точки (*x1,y1*) до точки (*x2,y2*) цветом *c*;

`procedure DrawCircle(x,y,r:integer);` – рисует окружность с центром (*x,y*) и радиусом *r*;

`procedure DrawRectangle(x1,y1,x2,y2:integer);` – рисует границу прямоугольника, заданного координатами противоположных вершин (*x1,y1*) и (*x2,y2*);

`procedure Circle(x,y,r:integer);` – рисует заполненную окружность с центром (*x,y*) и радиусом *r*;

`procedure Rectangle(x1,y1,x2,y2:integer);` – рисует заполненный прямоугольник, заданный координатами противоположных вершин (*x1,y1*) и (*x2,y2*);

`procedure TextOut(x,y:integer; s:string);` – выводит строку *s* в прямоугольник с координатами левого верхнего угла (*x,y*);

`procedure TextOut(x,y:integer; n:integer);` – выводит целое *n* в прямоугольник с координатами левого верхнего угла (*x,y*);

`procedure TextOut(x,y:integer; r:real);` – выводит вещественное *r* в прямоугольник с координатами левого верхнего угла (*x,y*).

8. ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ

8.1. Математический пакет Mathcad

Интерфейс Mathcad аналогичен интерфейсу других Windows-приложений (рис. 11).

Mathcad предназначен для решения следующих задач [8]:

– вычисление результатов математических операций, в которых участвуют числовые константы, переменные и размерные физические величины;

– операции с векторами и матрицами;

– решение уравнений и систем уравнений;

– статистические расчеты и анализ данных;

– построение графиков;

– тождественные преобразования выражений, аналитическое решение уравнений и систем;

– дифференцирование и интегрирование (аналитическое и численное);

– решение дифференциальных уравнений и др.

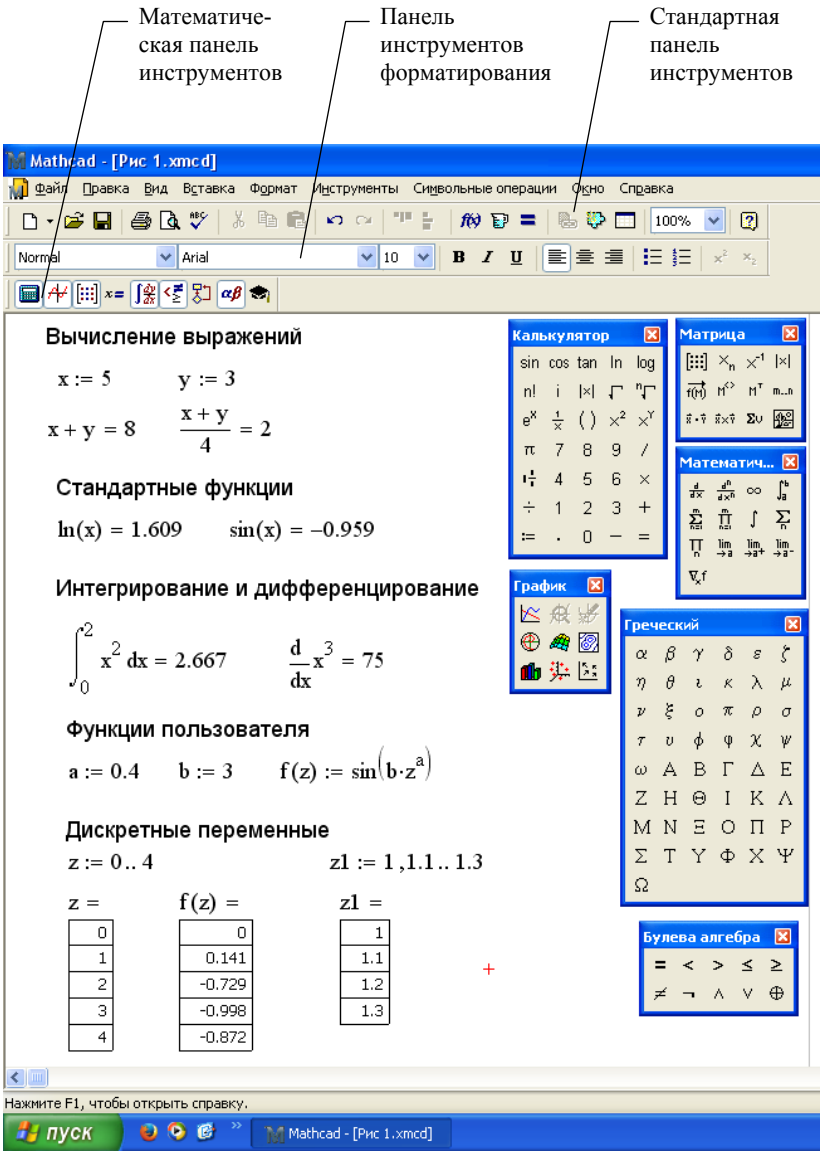


Рис. 11. Вычисления в программе Mathcad

Для ввода элементов формул предназначены дополнительные панели математической панели инструментов (см. рис. 11):

График – шаблоны графиков.

Матрица – шаблоны матриц и матричных операций.

Вычисление – операторы присваивания значений и вывода результатов расчета.

Калькулятор – шаблоны основных математических операций.

Математический анализ – шаблоны дифференцирования, интегрирования, суммирования.

Булева алгебра – логические (булевы) операторы.

Греческие символы.

Программирование – операторы для создания программных модулей.

Символьные преобразования – операторы символьных вычислений.

Документ программы Mathcad содержит два вида объектов: *формулы* и *текстовые блоки*. В ходе расчетов формулы обрабатываются последовательно, слева направо и сверху вниз, а текстовые блоки игнорируются.

Ввод информации осуществляется в месте расположения курсора. Программа Mathcad использует три вида курсоров. Если ни один объект не выбран, используется красный *крестообразный курсор*, определяющий место создания следующего объекта. При вводе формул используется синий *уголковый курсор*, указывающий текущий элемент выражения. Чтобы выделить элементы формулы, которые в рамках операции должны рассматриваться как единое целое, используют клавишу ПРОБЕЛ. При каждом ее нажатии уголковый курсор расширяется, охватывая элементы формулы, примыкающие к текущему элементу. При вводе данных в текстовый блок применяется *текстовый курсор* в виде вертикальной черты.

Редактирование введенных выражений производится обычным для всех Windows-приложений способом.

Mathcad воспринимает прописные и строчные буквы как разные идентификаторы.

При расчетах по формулам используются следующие операторы:

– *оператор присваивания* (:=) (см. рис. 11);

– *оператор вычисления* (=) (см. рис. 11);

– *оператор аналитического (символьного) вычисления* (→) (рис. 12);

– *глобальный оператор присваивания* (≡);

– *знак логического равенства* (**=**).

В Mathcad можно использовать *стандартные встроенные функции* (кнопка *f*(*x*) на стандартной панели инструментов), а также *функции*, *определенные пользователем* (см. рис. 11). Функция пользователя определяется следующим образом: слева указывается название функции, а справа после оператора присваивания (:=) – вычисляемое выражение. Переменные величины, входящие в правую часть, должны быть записаны в качестве параметров в скобках после имени функции. Все величины из правой части, не входящие в параметры левой части, должны быть заданы численно левее и выше функции пользователя.

Для получения таблицы значений функции или ее графика используются *дискретные переменные*, определяющие ряд значений. С помощью дискретной переменной можно задавать как целые, так и дробные значения переменной, но обязательно равноотстоящие друг от друга (*z* и *z*1 на рис. 11). Дискретная переменная – диапазон – определяется первым, вторым и последним элементами: $x := a, a + \frac{b-a}{n} .. b$, соответственно задает ряд чисел, где *a* – первое, $a + \frac{b-a}{n}$ – второе, *b* – последнее число, *n* – число интервалов, на которые разбит отрезок от *a* до *b*. Если интервал между числами равен 1, то второй элемент отсутствует. Дискретная переменная задается с помощью кнопки $\boxed{m..n}$ – *Переменная-диапазон* панели инструментов *Матрица*.

Для построения двухмерного графика функции надо выполнить следующие действия (рис. 12):

1) установить крестообразный курсор в то место, где должен быть построен график;

2) на панели инструментов *График* щелкнуть на кнопке *График X-Y*;

3) в появившемся на месте курсора шаблоне двухмерного графика ввести на оси абсцисс имя аргумента, на оси ординат – имя функции;

4) щелкнуть мышью вне шаблона графика – для заданного диапазона изменения аргумента график будет построен.

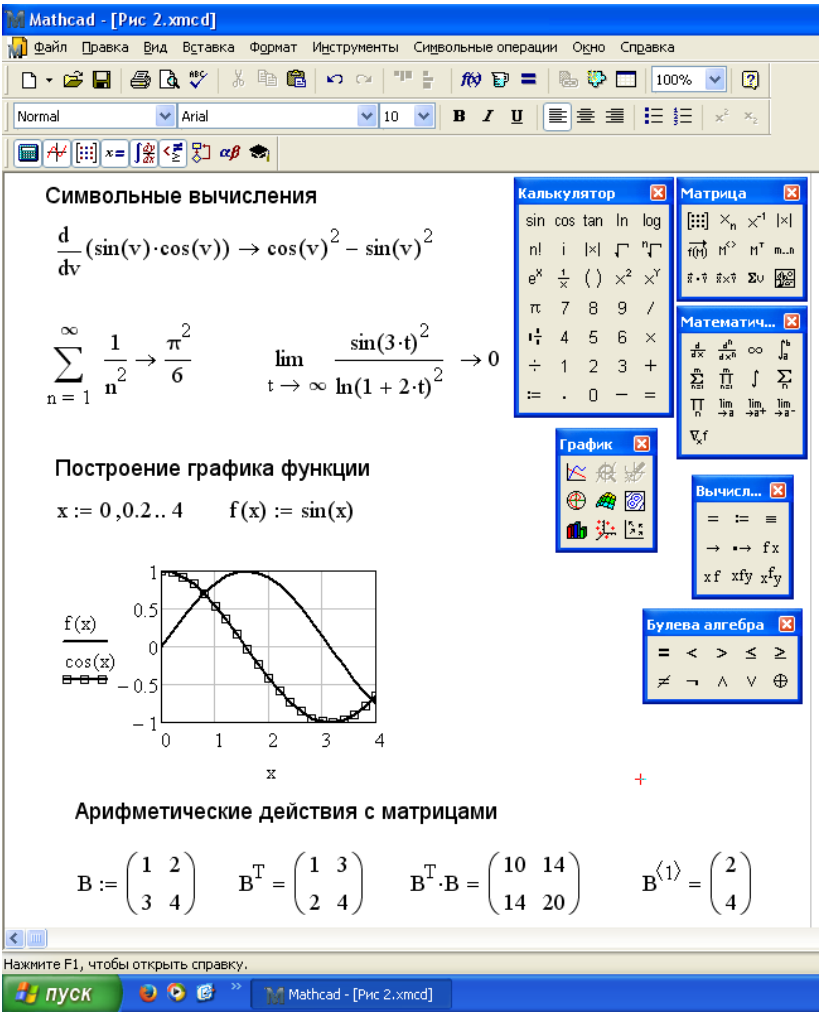


Рис. 12. Символьные вычисления, построение графика функции и арифметические действия с матрицами

Если диапазон значений аргумента не задан, по умолчанию график строится в диапазоне значений аргумента от –10 до 10.

Чтобы в одном шаблоне разместить несколько графиков, следует, набрав на оси ординат имя первой функции, нажать клавишу [,] (запятая) и в появившемся месте ввода (черном квадратике) вписать имя второй функции и т.д. Если функции имеют разные аргументы, то на оси абсцисс надо ввести также через запятую имена аргументов функций.

Для форматирования графика необходимо дважды щелкнуть мышью в области графика – откроется диалоговое окно форматирования графика.

Определить вектор или матрицу можно следующим образом (см. рис. 12):

1) введите имя матрицы и оператор присваивания (:=);

2) на панели *Матрица* щелкните на кнопке *Матрица или вектор*. Откроется диалоговое окно, в котором введите число строк и число столбцов матрицы и щелкните на кнопке ОК. На экране появится шаблон матрицы;

3) каждое место ввода в шаблоне заполните числами.

Нумерация элементов массива (вектора или матрицы) может начинаться с 0, 1 или с любого другого числа. Порядком нумерации элементов массива управляет встроенная переменная ORIGIN. По умолчанию ORIGIN=0. Это означает, что первый элемент вектора имеет номер 0. Чтобы нумерация элементов векторов и матриц начиналась с 1, надо перед вводом матрицы набрать следующую строку: ORIGIN:=1.

Доступ к любому элементу матрицы можно получить через имя матрицы с двумя индексами (например, $B_{1,1}$). Первый индекс обозначает номер строки, второй – номер столбца. Произвольный элемент вектора задается одним индексом. Для набора нижнего индекса используется кнопка *Индекс* панели *Матрица*.

Чтобы из матрицы выделить вектор (один из столбцов матрицы), используется верхний индекс – номер столбца в угловых

скобках, например, $B^{<1>}$ (см. рис. 12). Для выполнения этой операции необходимо ввести имя матрицы и щелкнуть на кнопке $M^{<>}$ – *Столбец матрицы* панели инструментов *Матрица*.

Mathcad позволяет выполнять с матрицами основные арифметические действия, включая сложение, вычитание и умножение, а также операции транспонирования, обращения, вычисления определителя матрицы, нахождения собственных чисел и собственных векторов и т.д. (см. рис. 12).

Для численного поиска корней уравнений вида $f(x) = 0$ в программе Mathcad используется функция $root(f(x), x)$. Перед вызовом функции *root* необходимо присвоить искомой переменной x начальное значение. Например,

$x:=1$

$root(2sin(x) - x, x) = 1{,}895.$

Если уравнение имеет несколько корней, то результат зависит от выбранного начального приближения.

Для решения системы уравнений (неравенств) используют *блок решения*, который начинается с ключевого слова *given* (дано) и заканчивается вызовом функции *find* (найти). Между ними располагают уравнения (неравенства), входящие в систему. При этом между левой и правой частями уравнений должен стоять знак логического равенства (**=**) с панели инструментов *Булева алгебра*. Перед решением системы уравнений необходимо задать начальные приближения для всех неизвестных. Например,

$x:=0 \qquad y:=0$

given

$x + y = 1$

$x^2 + y^2 = 4$

$$find(x, y) = \begin{pmatrix} 1{,}823 \\ -0{,}823 \end{pmatrix}.$$

Функция *find* возвращает вектор, содержащий вычисленные значения неизвестных.

8.2. Графические пакеты прикладных программ

Графический редактор – программа (или пакет программ), позволяющая создавать, просматривать, обрабатывать и редактировать цифровые изображения (рисунки, картинки, фотографии) на компьютере.

Графические редакторы делятся на несколько групп по типу кодирования обрабатываемых изображений. Существуют следующие виды редакторов.

Растровые графические редакторы. Наиболее популярные профессиональные растровые графические редакторы: коммерческий Adobe Photoshop и бесплатный с открытым исходным кодом GIMP.

Векторные графические редакторы. Наиболее популярные профессиональные векторные графические редакторы: коммерческие Adobe Illustrator и Corel Draw, а также свободно распространяемый и бесплатный графический редактор Inkscape.

Гибридные графические редакторы. Комбинируют внутри себя возможности работы с обоими видами графики. К этому классу также можно отнести редактор растровой графики Adobe Photoshop, в котором есть поддержка векторных слоев и векторные редакторы Adobe Illustrator и Corel Draw, которые поддерживают некоторые функции для работы с растровой графикой.

9. БАЗЫ ДАННЫХ

База данных – это организованная структура, предназначенная для хранения информации. С понятием базы данных тесно связано понятие система управления базами данных. Это комплекс программных средств, необходимых для создания структуры новой базы, наполнения ее содержимым, редактирования содержимого и организации поиска необходимой информации. Среди подобных программных продуктов весьма популярна система управления базами данных Microsoft Access, входящая в состав интегрированного пакета Microsoft Office.

База данных Microsoft Access представляет собой единый большой объект, который объединяет такие составляющие, как таблицы, запросы, формы, отчеты, макросы и модули [9].

Основные этапы разработки базы данных в среде Microsoft Access:

1. Определение цели создания базы данных.
2. Определение таблиц, которые должна содержать база данных.
3. Определение необходимых в таблице полей.
4. Задание индивидуального значения каждому полю.
5. Определение связей между таблицами.
6. Обновление структуры базы данных.
7. Добавление данных и создание других объектов базы данных.
8. Использование средств анализа в Microsoft Access.

Основные возможности и приемы работы с системой управления базами данных Microsoft Access приведены в работе [9].

10. ТЕЛЕКОММУНИКАЦИИ. ЛОКАЛЬНЫЕ И ГЛОБАЛЬНЫЕ КОМПЬЮТЕРНЫЕ СЕТИ

10.1. Основные понятия компьютерных сетей

При физическом соединении двух или более компьютеров образуется компьютерная сеть. В общем случае для создания компьютерных сетей необходимо специальное аппаратное обеспечение (сетевое оборудование) и специальное программное обеспечение (сетевые программные средства) [1].

Основной задачей при создании компьютерных сетей является обеспечение совместимости оборудования по электрическим и механическим характеристикам и обеспечение совместимости информационного обеспечения (программ и данных) по системе кодирования и формату данных. Решение этой задачи относится к области стандартизации и основано на так называе-

мой модели OSI (модель взаимодействия открытых систем). Она создана на основе технических предложений Международного института стандартизации ISO.

Согласно модели ISO/OSI архитектуру компьютерных сетей следует рассматривать на разных уровнях (общее число уровней – до семи). Самый верхний уровень – *прикладной*. На этом уровне пользователь взаимодействует с вычислительной системой. Самый нижний уровень – *физический*. Он обеспечивает обмен сигналами между устройствами. Обмен данными в системах связи происходит путем их перемещения с верхнего уровня на нижний, затем транспортировки и, наконец, обратным воспроизведением на компьютере клиента в результате перемещения с нижнего уровня на верхний.

Для обеспечения необходимой совместимости на каждом из семи возможных уровней действуют специальные стандарты, называемые *протоколами*. Они определяют характер аппаратного взаимодействия компонентов сети (*аппаратные протоколы*) и характер взаимодействия программ и данных (*программные протоколы*).

В соответствии с используемыми протоколами компьютерные сети подразделяют на *локальные* (*LAN – Local Area Network*) и *глобальные* (*WAN – Wide Area Network*). Компьютеры локальной сети преимущественно используют единый комплект протоколов для всех участников. По территориальному признаку локальные сети отличаются компактностью. Глобальные сети имеют, как правило, увеличенные географические размеры. Они могут объединять как отдельные компьютеры, так и отдельные локальные сети, в том числе и использующие различные протоколы.

Назначение всех видов компьютерных сетей определяется двумя функциями:

– обеспечение *совместного использования* аппаратных и программных ресурсов сети;

– обеспечение *совместного доступа* к ресурсам данных.

Уровни модели ISO/OSI (рис. 13):

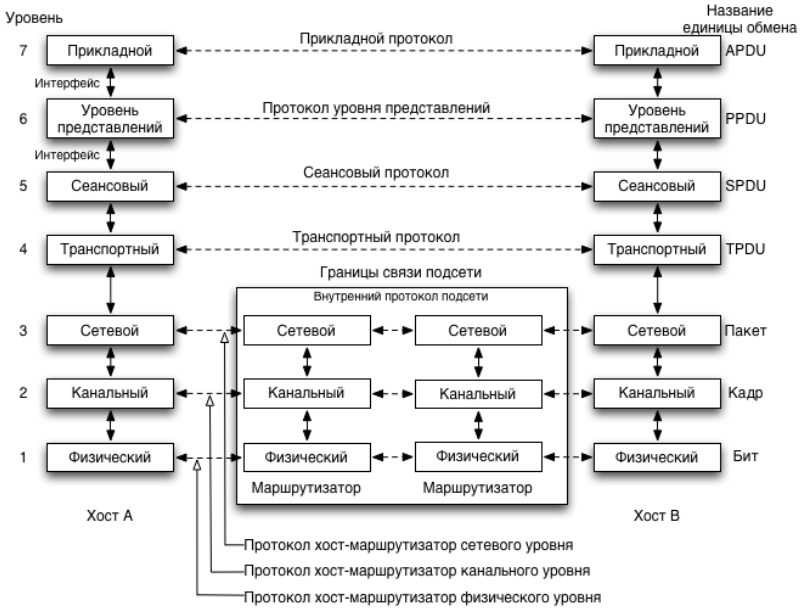


Рис. 13. Взаимодействие двух компьютеров (хостов) в модели ISO/OSI

7. На *прикладном* уровне с помощью специальных приложений пользователь создает документ.

6. На *уровне представления* операционная система его компьютера фиксирует, где находятся созданные данные (в оперативной памяти, в файле на жестком диске и т.п.), и обеспечивает взаимодействие со следующим уровнем.

5. На *сеансовом уровне* компьютер пользователя взаимодействует с локальной или глобальной сетью. Протоколы этого уровня проверяют права пользователя на «выход в эфир» и передают документ к протоколам транспортного уровня.

4. На *транспортном уровне* документ преобразуется в ту форму, в которой положено передавать данные в используемой сети. Например, он может нарезаться на небольшие пакеты стандартного размера.

3. *Сетевой уровень* определяет маршрут движения данных в сети. Например, каждый пакет получает адрес, по которому он должен быть доставлен независимо от прочих пакетов.

2. *Канальный уровень* создает логический канал между двумя устройствами поверх сигналов физического уровня. Для адресации на канальном уровне используются аппаратные MAC-адреса, которые адресуют непосредственно сетевую карту, как устройство.

1. Реальная передача данных происходит на *физическом уровне*. Здесь нет ни документов, ни пакетов, ни даже байтов – только биты, т.е., элементарные единицы представления данных. Восстановление документа из них произойдет при переходе с нижнего на верхний уровень на компьютере клиента.

10.2. Сетевые протоколы

По-настоящему годом рождения Интернета принято считать 1983 г. В этом году произошли революционные изменения в программном обеспечении компьютерной связи. Днем рождения Интернета в современном понимании этого слова стала дата внедрения стека протоколов TCP/IP в сеть ARPANET. На следующий год была основана межуниверситетская сеть NSFNet, которая объединяла локальные сети университетов. Таким образом, Интернет получил не только стек протоколов, который используется по нынешний день, но и свое название – сеть сетей или Интернет.

Стек протоколов TCP/IP получил свое название по названиям двух базовых протоколов, лежащих в его основе. Протокол TCP – протокол *транспортного уровня*. Он управляет тем, как происходит передача информации, и обеспечивает ее надежную доставку получателям. Протокол IP – *адресный*. Он принадлежит *сетевому уровню* и определяет, куда происходит передача. Помимо них в стеке есть еще протокол UDP, который обеспечивает негарантированную передачу сообщений или датаграмм (блоков информации) между узлами сети.

Согласно протоколу TCP, отправляемые данные «нарезаются» на небольшие сегменты, после чего каждый сегмент маркируется таким образом, чтобы в нем были данные, необходимые для правильной сборки документа на компьютере получателя. Два компьютера, связанные между собой одним физическим соединением, могут поддерживать одновременно несколько TCP-соединений. Так, например, два промежуточных сетевых сервера могут одновременно по одной линии связи передавать друг другу в обе стороны множество TCP-пакетов от многочисленных клиентов.

Суть адресного протокола – IP (Internet Protocol) – состоит в том, что у каждого участника Всемирной сети должен быть свой уникальный адрес (IP-адрес). В созданной в 1983 г. версии этого протокола (IPv4) адрес выражается четырьмя байтами, например: 195.38.46.11. Поскольку один байт содержит до 256 различных значений, то теоретически с помощью четырех байтов можно выразить более четырех миллиардов уникальных IP-адресов. В те годы этого казалось достаточно, но к 2015 г. большая часть провайдеров исчерпала доступное количество IP-адресов. В настоящее время вводится в эксплуатацию шестая версия протокола – IPv6, которая позволяет адресовать значительно большее количество узлов, чем IPv4. Длина адреса в протоколе IPv6 составляет 128 бит вместо 32 в протоколе IPv4, что позволяет выделить на каждого жителя Земли порядка 300 млн адресов.

Большинство других протоколов сейчас работают поверх названных выше протоколов стека TCP/IP. Среди них можно особо отметить протоколы DNS, HTTP, SMTP, POP3, которые рассмотрены далее.

10.3. Основные службы Интернета

Когда говорят о работе в Интернете или об использовании Интернета, то на самом деле речь идет не об Интернете в целом, а только об одной или нескольких из его многочисленных служб.

Разные службы имеют разные протоколы. Они называются *прикладными протоколами*. Их соблюдение обеспечивается и поддерживается работой специальных программ. Таким образом, чтобы воспользоваться какой-то из служб Интернета, необходимо установить на компьютере программу, способную работать по протоколу данной службы. Такие программы называют *клиентскими* или просто *клиентами*.

Так, например, чтобы воспользоваться электронной почтой, необходимо соблюсти протоколы отправки и получения сообщений. Для этого надо иметь программу (*почтовый клиент*) и установить связь с *почтовым сервером*.

Терминальный режим. Исторически одной из ранних является служба удаленного управления компьютером *Telnet*. Подключившись к удаленному компьютеру по протоколу этой службы, можно управлять его работой. Часто протоколы Telnet применяют для дистанционного управления техническими объектами, например телескопами, видеокамерами, промышленными роботами.

Сервер, предоставляющий Telnet-услуги, обычно предлагает свое клиентское приложение. Его надо получить по сети, установить на своем компьютере, подключиться к серверу и работать с удаленным оборудованием.

Электронная почта (E-Mail). Эта служба также является одной из наиболее ранних. Ее обеспечением в Интернете занимаются специальные *почтовые серверы*. Здесь и далее под *сервером* может пониматься программное обеспечение. Таким образом, один узловой компьютер Интернета может выполнять функции нескольких серверов и обеспечивать работу различных служб.

Почтовые серверы получают сообщения от клиентов и пересылают их по цепочке к почтовым серверам адресатов, где эти сообщения накапливаются. При установлении соединения между адресатом и его почтовым сервером происходит автоматическая передача поступивших сообщений на компьютер адресата.

Адрес электронной почты состоит из двух частей, разделенных символом @. Первая часть адреса указывает конкретного пользователя. Вторая часть адреса – это домен почтового сервера.

Существуют серверы, где можно бесплатно открыть почтовый ящик, например, gmail.com, mail.yandex.ru и др.

Почтовая служба основана на двух прикладных протоколах: SMTP и POP3. По первому происходит отправка корреспонденции с компьютера на сервер, а по второму – прием поступивших сообщений. Существует большое разнообразие клиентских почтовых программ. К ним относится, например, специализированная почтовая программа – Mozilla Thunderbird (рис. 14).

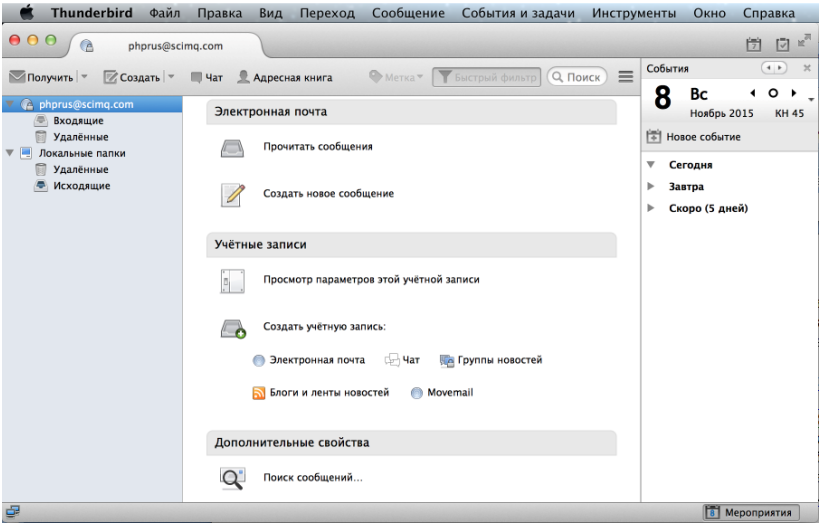


Рис. 14. Главное окно программы Mozilla Thunderbird

Так как сообщения электронной почты поступают и отправляются через почтовый сервер, программе – почтовому клиенту – требуется указать информацию об используемом сервере. Эта информация хранится в виде *учетной записи*.

В программе Mozilla Thunderbird учетную запись создают командой **Инструменты → Параметры учетной записи**. В открывшемся диалоговом окне надо щелкнуть на кнопке <Действия для учетной записи> и выбрать в открывшемся меню пункт <Добавить учетную запись почты>. Последующая информация

вводится под управлением мастера и включает имя, указываемое как имя отправителя, адрес электронной почты и имя используемого сервера.

Для создания сообщения электронной почты следует щелкнуть на кнопке <Создать> на панели инструментов (см. рис. 14). При этом открывается окно **Создание сообщения**, рабочая область которого разбивается на две основные части. В верхней части располагаются поля для ввода служебной информации – адреса получателя, темы сообщения. В нижней части окна – собственно текст сообщения. В ходе создания и редактирования сообщения наличие связи с почтовым сервером не требуется.

Получение и доставка почты осуществляются по щелчку на кнопке <Отправить> в окне создания сообщения или по щелчку на кнопке <Получить> в основном окне программы Mozilla Thunderbird.

Сообщения электронной почты размещаются в системе «внутренних» папок программы Mozilla Thunderbird. Поступившие сообщения заносятся в папку *Входящие*.

Для хранения адресов электронной почты используется специальная программа *Адресная книга*.

Списки рассылки (Mail List). Достаточно большой поток почтовой информации в свой адрес можно обеспечить, подписавшись на *списки рассылки*. Это специальные тематические серверы, собирающие информацию по определенным темам и переправляющие ее подписчикам в виде сообщений электронной почты.

Служба телеконференций. Служба телеконференций похожа на циркулярную рассылку электронной почты, в ходе которой одно сообщение отправляется не одному корреспонденту, а большой группе (такие группы называются *телеконференциями* или *группами новостей*).

Вся система телеконференций разбита на тематические группы (в мире их более 50 000). Основной прием использования групп новостей состоит в том, чтобы задать вопрос, обраща-

ясь ко всему миру, и получить ответ или совет от тех, кто с этим вопросом уже разобрался.

Служба World Wide Web (WWW). Это самая популярная служба современного Интернета. World Wide Web – это единое информационное пространство, состоящее из сотен миллионов взаимосвязанных электронных документов, хранящихся на web-серверах. Отдельные документы, составляющие *пространство web*, называют *web-страницами*. Группы тематически объединенных web-страниц называются *web-сайт* или просто *сайт*.

От обычных текстовых документов web-страницы отличаются тем, что они оформлены без привязки к конкретному носителю. Оформление выполняется непосредственно во время их воспроизведения на компьютере клиента и происходит в соответствии с настройками программы, выполняющей просмотр. Программы для просмотра web-страниц называются *браузерами*. Наиболее важной чертой web-страниц являются *гипертекстовые ссылки*. В этом случае при щелчке левой кнопкой мыши на тексте или рисунке, являющемся гиперссылкой, отправляется запрос на доставку нового документа.

Со стороны Интернета работу службы World Wide web обеспечивают серверные и программные средства – *web-серверы*. Со стороны пользователя работа обеспечивается клиентскими программами – *web-браузерами* (например, Mozilla Firefox или Microsoft Internet Explorer). Для запуска этой программы можно использовать Главное меню (**Пуск → Все программы → Mozilla Firefox**). В открывшемся окне программы в поле **Адрес** необходимо ввести URL – адрес ресурса (если он известен) или воспользоваться одной из поисковых систем – www.yandex.ru, www.rambler.ru, www.google.ru или др.

Адрес любого файла во всемирном масштабе определяется *унифицированным указателем ресурса – URL*. Адрес URL состоит из трех частей:

1. Указание службы, которая осуществляет доступ к данному ресурсу (обычно обозначается именем прикладного прото-

кола, соответствующего данной службе). Так, например, для службы WWW прикладным является протокол *HTTP* (*Hyper Text Transfer Protocol – протокол передачи гипертекста*). После имени протокола ставится двоеточие и два знака «/»:

http://…

2. Указание *доменного имени* компьютера (сервера), на котором хранится данный ресурс:

http://**www.google.com**…

3. Указание полного пути доступа к файлу на данном компьютере. В качестве разделителя используется символ «/»:

http://www.google.com**/intl/ru/options**

При записи URL-адреса важно точно соблюдать регистр символов, так как в Интернете строчные и прописные символы считаются разными.

Служба имен доменов (DNS). В Интернете есть две разные формы записи адреса одного и того же сетевого компьютера: числовая (64.233.162.106) и доменная (www.google.com). Человеку неудобно работать с числовым представлением IP-адреса, зато доменное имя запоминается легко, особенно если учесть, что, как правило, это имя имеет содержание. С другой стороны, автоматическая работа серверов сети организована с использованием четырехзначного числового адреса. Поэтому необходим перевод доменных имен в связанные с ними IP-адреса. Этим и занимаются серверы службы имен доменов DNS.

Служба передачи файлов (FTP). Необходимость в передаче файлов возникает, например, при приеме файлов программ, при пересылке крупных документов (например, книг), а также при передаче архивных файлов, в которых запакованы большие объемы информации.

Со стороны клиента для работы с серверами FTP может быть установлено специальное программное обеспечение, хотя в большинстве случаев браузеры WWW обладают встроенными возможностями для работы и по протоколу FTP.

ICQ. Эта служба предназначена для поиска сетевого IP-адреса человека, подключенного в данный момент к Интернету. Необходимость в подобной услуге связана с тем, что большинство пользователей не имеют постоянного IP-адреса. Для пользования этой службой надо зарегистрироваться на ее центральном сервере (http://www.icq.com) и получить персональный идентификационный номер *UIN* (*Universal Internet Number*). Данный номер нужно сообщить партнерам по контактам.

При каждом подключении к Интернету программа ICQ, установленная на компьютере, определяет текущий IP-адрес и сообщает его центральной службе, которая, в свою очередь, оповещает наших партнеров по контактам.

11. МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ

11.1. Вопросы компьютерной безопасности

Вредоносная программа – любое программное обеспечение, предназначенное для получения несанкционированного доступа к вычислительным ресурсам ЭВМ или к информации, хранимой на ЭВМ с целью несанкционированного использования ресурсов ЭВМ или причинения вреда (нанесения ущерба) владельцу информации, и/или владельцу ЭВМ, и/или владельцу сети ЭВМ, путем копирования, искажения, удаления или подмены информации.

Вредоносное программное обеспечение может быть классифицировано по различным критериям.

По вредоносной нагрузке

- Помехи в работе зараженного компьютера: начиная от открытия-закрытия дисковода оптических дисков и заканчивая уничтожением данных и поломкой аппаратного обеспечения. Блокировка антивирусных сайтов, антивирусного ПО и административных функций ОС с целью усложнить лечение. Саботирование промышленных процессов, управляемых компьютером (червь Stuxnet).

- Загрузка из сети Интернет и инсталляция другого вредоносного ПО.
- Кража, мошенничество, вымогательство и шпионаж за пользователем. Для кражи может применяться сканирование жесткого диска, регистрация нажатий клавиш и перенаправление пользователя на поддельные сайты, в точности повторяющие исходные ресурсы.
    - Похищение данных, представляющих ценность или тайну.
    - Кража аккаунтов различных служб (электронной почты и других).
    - Кража аккаунтов платежных систем.
    - Блокировка компьютера, шифрование файлов пользователя с целью шантажа и вымогательства денежных средств.
- Прочая незаконная деятельность:
    - Получение несанкционированного доступа к ресурсам самого компьютера или третьим ресурсам, доступным через него, в том числе прямое управление компьютером.
    - Зараженный компьютер может быть использован для проведения DDoS-атак (распределенная атака, приводящая к нарушению работы интернет-сервисов).
    - Сбор адресов электронной почты и распространение спама.
- Файлы, не являющиеся истинно вредоносными, но в большинстве случаев нежелательные:
    - Adware – программное обеспечение, показывающее рекламу.
    - Spyware – программное обеспечение, посылающее через интернет не санкционированную пользователем информацию.
    - Программы удаленного администрирования могут применяться как для того, чтобы дистанционно решать проблемы с компьютером, так и для противоправных целей.

○ Руткит – программное обеспечение, позволяющее скрывать другое вредоносное ПО от обнаружения.

По методу размножения

- Эксплойт – теоретически безобидный набор данных (например, графический файл или сетевой пакет), некорректно воспринимаемый программой, работающей с такими данными. Здесь вред наносит не сам файл, а неадекватное поведение ПО с ошибкой.
- Логическая бомба в программе срабатывает при определенном условии и неотделима от полезной программы-носителя.
- Троянская программа не имеет собственного механизма размножения.
- Компьютерный вирус размножается в пределах компьютера и через сменные диски. Размножение через сеть возможно, если пользователь сам выложит зараженный файл в сеть. Вирусы, в свою очередь, делятся по типу заражаемых файлов (файловые, загрузочные, макро-, автозапускающиеся); по способу прикрепления к файлам (паразитирующие, «спутники» и перезаписывающие) и т.д.
- Сетевой червь способен самостоятельно размножаться по сети. Делятся на почтовые, размножающиеся с помощью эксплойтов и т.д.

Вредоносное ПО может образовывать цепочки: например, с помощью эксплойта на компьютере жертвы развертывается загрузчик, устанавливающий из Интернета червя.

Средства антивирусной защиты. Основным средством защиты информации является резервное копирование наиболее ценных данных. Резервные копии должны храниться отдельно от компьютера на внешних носителях.

Вспомогательным средством защиты информации являются антивирусные программы, которые следует регулярно обновлять, так как антивирусная программа ищет вирус путем сравнения кода программ с кодами известных ей вирусов, хранящимися в базе данных. Если база данных устарела, а вирус является новым, сканирующая программа его не обнаружит.

11.2. Защита информации в Интернете. Понятие о шифровании данных

Принцип действия защиты информации в Интернете основан на том, чтобы исключить возможность подбора адекватного метода для преобразования данных в информацию. Одним из приемов такой защиты является *шифрование* данных [1].

Обычный подход состоит в том, что к документу применяется некий *метод шифрования* (*ключ*), после чего документ становится недоступен для чтения обычными средствами. Его может прочитать только тот, кто знает ключ, т.е. может применить *адекватный метод чтения*. Если в процессе обмена информацией для шифрования и чтения пользуются одним и тем же ключом, то такой криптографический процесс является *симметричным*.

Основной недостаток симметричного процесса заключается в том, что, прежде чем начать обмен информацией, надо выполнить передачу ключа, а для этого опять нужна защищенная связь, т.е. проблема повторяется.

Поэтому в настоящее время в Интернете используют *несимметричные* криптографические системы, основанные на использовании не одного, а двух ключей. Компания для работы с клиентами создает два ключа: один – открытый (публичный) ключ, а другой – закрытый (личный) ключ. На самом деле это как бы две «половинки» одного целого ключа, связанные друг с другом. Ключи устроены так, что сообщение, зашифрованное одной половинкой, можно расшифровать только другой половинкой. Таким образом, торговая компания широко распространяет публичный ключ и надежно сохраняет закрытый ключ. Оба эти ключа представляют собой некую кодовую последовательность.

Защиту информации принято считать достаточной, если затраты на ее преодоление превышают ожидаемую ценность самой информации. В этом состоит *принцип достаточности защиты*, которым руководствуются при использовании несимметричных средств шифрования данных.

ВОПРОСЫ ПО ДИСЦИПЛИНЕ «ИНФОРМАТИКА» (ГЛАВЫ 1–11)

1. Информатика: предмет и задачи.
2. Понятие информации. Свойства информации.
3. Носители данных. Операции с данными.
4. Основные структуры данных.
5. Кодирование данных двоичным кодом.
6. Поколения ЭВМ. Классификация компьютеров.
7. Состав вычислительной системы.
8. Базовая аппаратная конфигурация компьютера.
9. Внутренние устройства системного блока.
10. Системы, расположенные на материнской плате.
11. Классификация программного обеспечения компьютера.
12. Классификация операционных систем. Функции операционных систем персональных компьютеров.
13. Текстовый процессор и методы работы с ним.
14. Электронные таблицы и методы работы с ними.
15. Алгоритм и его свойства. Графический способ представления алгоритма.
16. Основные структуры алгоритмов.
17. Алгоритмы линейной, разветвляющейся и циклической структуры.
18. Циклические алгоритмы с неизвестным числом повторений. Вложенные циклы.
19. Параллельные алгоритмы.
20. Классификация языков программирования. Алфавит, синтаксис и семантика языка программирования.
21. Трансляция, интерпретация и компиляция программ.
22. Программирование на языке Pascal: операторы языка программирования, идентификаторы, типы данных, стандартные функции. Структура программы на языке Pascal.
23. Программирование на языке Pascal: операторы ввода/вывода, оператор присваивания, условный оператор, операторы цикла.

24. Реализация алгоритмов линейной, разветвляющейся и циклической структуры на языке Pascal.

25. Массивы. Ввод/вывод и обработка элементов одномерного и двумерного массивов.

26. Процедуры и функции (подпрограммы).

27. Математические, графические пакеты прикладных программ.

28. Базы данных и методы работы с ними.

29. Основные понятия компьютерных сетей. Уровни модели ISO/OSI.

30. Сетевые протоколы.

31. Основные службы Интернета.

32. Вопросы компьютерной безопасности. Вредоносные программы и средства антивирусной защиты.

33. Защита информации в Интернете. Шифрование данных.

12. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ. СТРУКТУРНОЕ И ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

12.1. Жизненный цикл программного обеспечения

Программный продукт (software product) – совокупность компьютерных программ, процедур и, возможно, связанных с ними документации и данных.

Классификация программного обеспечения ПО:

1) по способу исполнения программы делят:

– на интерпретируемые;

– компилируемые.

2) по степени переносимости программы делят:

– на платформозависимые;

– кроссплатформенные;

3) по способу распространения и использования программы делят:

– на несвободные (закрытые) (proprietary software) – программное обеспечение, являющееся частной собственностью авторов или правообладателей и не удовлетворяющее критериям свободного ПО. Правообладатель проприетарного ПО сохраняет за собой монополию на его использование, копирование и модификацию, полностью или в существенных моментах;

– открытые (open-source software) – программное обеспечение с открытым исходным кодом. Исходный код таких программ доступен для просмотра, изучения и изменения, что позволяет пользователю принять участие в доработке самой открытой программы, использовать код для создания новых программ и исправления в них ошибок;

– свободные (free software) – программное обеспечение, пользователи которого имеют права («свободы») на его неограниченную установку, запуск, а также свободное использование, изучение, распространение и изменение (совершенствование) и распространение копий и результатов изменения.

Жизненный цикл программного обеспечения – период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации. Это процесс построения и развития ПО.

Структура жизненного цикла ПО согласно стандарту ISO/IEC 12207 базируется на трех группах процессов:

– основные процессы (приобретение, поставка, разработка, эксплуатация, сопровождение);

– вспомогательные процессы, обеспечивающие выполнение основных процессов (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем);

– организационные процессы (управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого жизненного цикла, обучение).

Разработка проекта включает в себя все работы по созданию ПО и его компонентов в соответствии с заданными требованиями. Этапы жизненного цикла ПО:

– формирование требований;

– проектирование;

– реализация (программирование);

– тестирование;

– внедрение;

– эксплуатация и сопровождение.

Важное место в цикле разработки ПО после этапов программирования и тестирования занимает *отладка* – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Для того чтобы понять, где возникла ошибка, приходится:

– узнавать текущие значения переменных;

– выяснять, по какому пути выполнялась программа.

Существуют две взаимодополняющие технологии отладки:

1) использование отладчиков – программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функци-

ей, с остановками на некоторых строках исходного кода или при достижении определённого условия;

2) вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода – на экран, принтер или в файл. Вывод отладочных сведений в файл называется журналированием.

Эксплуатация содержит работы по внедрению компонентов ПО в эксплуатацию, в том числе конфигурирование базы данных и рабочих мест пользователей.

Сопровождение (поддержка) программного обеспечения – процесс улучшения, оптимизации и устранения дефектов ПО после передачи в эксплуатацию. В ходе сопровождения в программу вносятся изменения, чтобы исправить обнаруженные в процессе использования дефекты и недоработки, а также для добавления новой функциональности с целью повысить удобство использования и применимость ПО.

Переносимость программ. Переносимость программы (портирование) – адаптация некоторой программы или её части для того, чтобы она работала в другой среде, отличающейся от той среды, под которую она была изначально написана с максимальным сохранением её пользовательских свойств.

Необходимость в выполнении портирования возникает обычно из-за различий в системе команд процессора, различий между способами взаимодействия операционной системы и программ (API – Application Program Interface), принципиальных различий в архитектуре вычислительных систем либо по причине некоторых несовместимостей или даже полного отсутствия используемого языка программирования в целевом окружении.

По мере развития операционных систем, языков и техники программирования становится всё проще портировать программы между различными платформами.

Понятие интеллектуальной собственности. Интеллектуальная собственность – закреплённое законом временное исключительное право, а также личные неимущественные права авторов на результат интеллектуальной деятельности.

В России с 1 января 2008 г. вступила в силу 4-я часть Гражданского кодекса (в соответствии с Федеральным законом от 18.12.2006 № 231-ФЗ), раздел VII «Права на результаты интеллектуальной деятельности и средства индивидуализации», который определяет интеллектуальную собственность как список результатов интеллектуальной деятельности и средств индивидуализации, которым предоставляется правовая охрана. Согласно этому документу интеллектуальной собственностью являются в том числе и программы для электронных вычислительных машин (программы для ЭВМ).

Законодательство, которое определяет права на интеллектуальную собственность, устанавливает монополию правообладателей на определённые формы использования результатов интеллектуальной, творческой деятельности, которые, таким образом, могут использоваться другими лицами лишь с разрешения первых.

12.2. Структурное программирование

Структурное программирование – методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 1970-х гг. нидерландским ученым Э. Дейкстрой, разработана и дополнена швейцарским ученым Н. Виртом.

В соответствии с данной методологией любая программа строится без использования оператора `goto` из трёх базовых управляющих структур: последовательность, ветвление, цикл; кроме того, используются подпрограммы. При этом разработка программы ведётся пошагово, методом «сверху вниз».

Принципы структурного программирования:

1. Следует отказаться от использования оператора безусловного перехода `goto`.

2. Любая программа строится из трёх базовых управляющих конструкций: последовательность, ветвление, цикл.

3. В программе базовые управляющие конструкции могут быть вложены друг в друга произвольным образом. Никаких

других средств управления последовательностью выполнения операций не предусматривается.

4. Повторяющиеся фрагменты программы можно оформить в виде подпрограмм (процедур и функций). Таким же образом (в виде подпрограмм) можно оформить логически целостные фрагменты программы, даже если они не повторяются.

5. Каждую логически законченную группу инструкций следует оформить как блок (block). Блоки являются основой структурного программирования.

Блок – это логически сгруппированная часть исходного кода, например набор инструкций, записанных подряд в исходном коде программы. Границы блока строго определены. Например, в языке Pascal в `if`-инструкции блок ограничен кодом `begin...end`.

6. Все перечисленные конструкции должны иметь один вход и один выход.

7. Разработка программы ведётся пошагово, методом «сверху вниз».

Суть метода «сверху вниз» заключается в следующем. Сначала пишется текст основной программы, в который вместо каждого связного логического фрагмента текста вставляется вызов подпрограммы, которая будет выполнять этот фрагмент. Вместо настоящих, работающих подпрограмм в программу вставляются фиктивные части – заглушки, которые, говоря упрощенно, ничего не делают.

После того как программист убедится, что подпрограммы вызываются в правильной последовательности (т.е. общая структура программы верна), подпрограммы-заглушки последовательно заменяются на реально работающие. При этом разработка каждой подпрограммы ведётся тем же методом, что и разработка основной программы. Разработка заканчивается тогда, когда не останется ни одной заглушки.

Рассмотрим применение принципов структурного программирования на примере задач математического моделирования.

Основы математического моделирования

Практически во всех науках о природе, об обществе построение и использование моделей является мощным орудием познания. Реальные объекты и процессы бывают столь многогранны и сложны, что лучшим способом их изучения часто является построение модели, отображающей лишь какую-то грань реальности и потому более простой, чем эта реальность, и исследование вначале этой модели. Многовековой опыт развития науки доказал на практике плодотворность такого подхода.

Таким образом, *модель* представляет собой отражение имеющихся знаний о процессе или объекте моделирования в соответствующей форме. Модель строится на основе обобщения известных данных (например, эксперимента) и отражает лишь те свойства, которые подлежат исследованию. Являясь результатом анализа и обобщений знаний об объекте, модель сама становится эффективным средством глубокого исследования свойств и характеристик объекта, его отношения к другим объектам, его поведения в различных условиях, в том числе и таких, которые в действительности воспроизвести или наблюдать невозможно.

Математическая модель выражает существенные черты объекта или процесса языком уравнений и других математических средств. Собственно говоря, сама математика обязана своим существованием тому, что она пытается отразить, т.е. промоделировать на своем специфическом языке закономерности окружающего мира.

Математическое моделирование представляет собой метод исследования объектов и процессов реального мира с помощью их приближенных описаний на языке математики – математических моделей.

Этапы и цели математического моделирования

Общая схема процесса математического моделирования представлена на рис. 15.

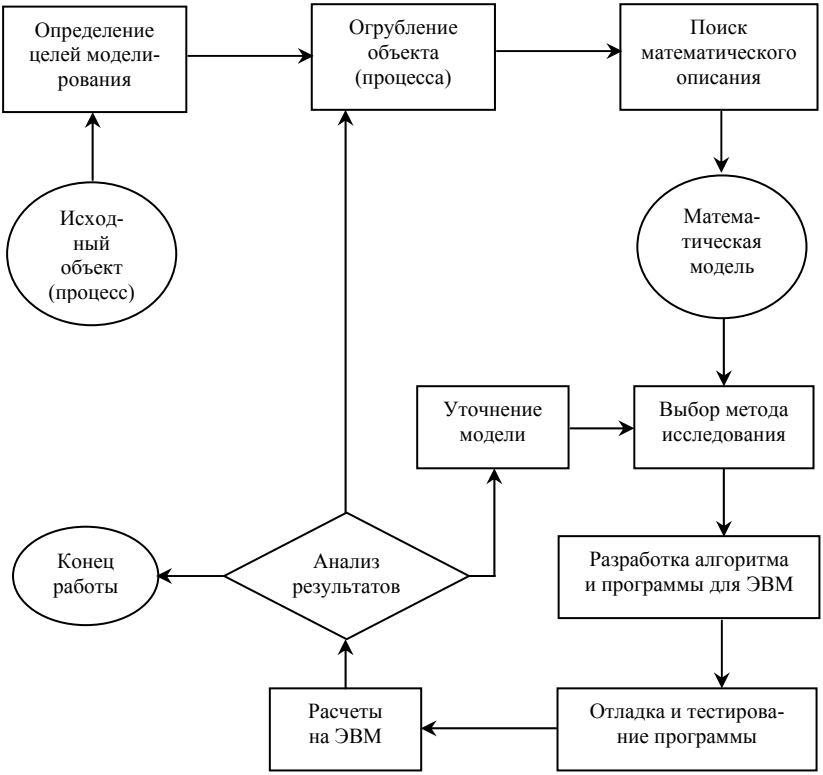


Рис. 15. Общая схема процесса математического моделирования

Первый этап – определение целей моделирования [10]. Основные из них:

1) модель нужна для того, чтобы понять, как устроен конкретный объект, какова его структура, основные свойства, законы развития и взаимодействия с окружающим миром (*понимание*);

2) модель нужна для того, чтобы научиться управлять объектом (или процессом) и определить наилучшие способы управления при заданных целях и критериях (*управление*);

3) модель нужна для того, чтобы прогнозировать прямые и косвенные последствия реализации заданных способов и форм воздействия на объект (*прогнозирование*).

Составим список величин, от которых зависит поведение объекта или ход процесса, а также тех величин, которые желательно получить в результате моделирования. Обозначим первые (входные) величины через x_1, x_2,…, x_n; вторые (выходные) через y_1, y_2,…, y_k. Символически поведение объекта или процесса можно представить в виде

$$y_j = F_j(x_1, x_2, \ldots, x_n), \qquad j = 1, 2, \ldots, k,$$

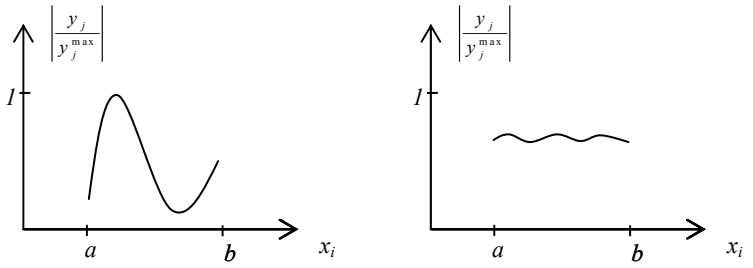
где F_j – те действия, которые следует произвести над входными параметрами, чтобы получить результаты.

Важнейшим этапом моделирования является разделение входных параметров по степени важности влияния их изменений на выходные. Чаще всего невозможно (да и не нужно) учитывать все факторы, которые могут повлиять на значения интересующих нас величин y_j. От того, насколько правильно выделены важнейшие факторы, зависит успех моделирования, быстрота и эффективность достижения цели. Отбрасывание (по крайней мере при первом подходе) менее значимых факторов огрубляет объект моделирования и способствует пониманию его главных свойств и закономерностей. При этом модель должна быть адекватна исходному объекту или процессу в отношении целей моделирования.

На рис. 16 проиллюстрированы две крайние ситуации: а) некоторый параметр x_i очень сильно влияет на результирующую величину y_j; б) почти не влияет на нее. Ясно, что если все представляющие интерес величины y_j реагируют на x_i так, как изображено на рис. 16, *б*, то x_i является параметром, который при первом подходе может быть из модели исключен; если же хотя бы одна из величин y_j реагирует на изменение x_i так, как изображено на рис. 16, *а*, то x_i нельзя исключать из числа важнейших параметров.

Следующий этап – поиск математического описания. На этом этапе необходимо перейти от абстрактной формулировки модели к формулировке, имеющей конкретное математическое

наполнение. В этот момент модель предстает перед нами в виде уравнения, системы уравнений, системы неравенств, дифференциального уравнения или системы таких уравнений и т.д.



а *б*

Рис. 16. Варианты степени влияния величины x_i на результирующую величину y_j

Когда математическая модель сформулирована, выбираем метод ее исследования. От верного выбора метода часто зависит успех всего процесса.

Затем разрабатывается алгоритм и составляется программа для ЭВМ (или используются специализированные пакеты программ решения математических задач).

Следующий этап – решение простейшей тестовой задачи (желательно с заранее известным ответом) с целью устранения грубых ошибок. Это лишь начало процедуры тестирования.

Далее следует собственно численный эксперимент, и выясняется, соответствует ли модель реальному объекту (процессу). Модель адекватна реальному процессу, если некоторые характеристики процесса, полученные на ЭВМ, совпадают с экспериментальными данными с заданной степенью точности. В случае несоответствия модели реальному процессу необходимо возвратиться к одному из предыдущих этапов.

Один из важнейших этапов математического моделирования – это выбор метода исследования, а также разработка для него алгоритма решения задачи.

Все методы решения математических задач можно разделить на 2 группы:

1) точные (аналитические) методы решения задач;

2) численные методы решения задач.

Аналитические решения (т.е. представленные формулами, выражающими результаты исследования через исходные данные) обычно удобнее и информативнее численных. Однако возможности аналитических методов решения сложных математических задач очень ограничены и, как правило, они гораздо сложнее численных. *Численные методы* позволяют свести получение численного решения к последовательному выполнению большого числа простых арифметических операций над численными значениями входных данных. При этом результаты вычислений носят приближенный характер. Важно только добиться того, чтобы ошибки укладывались в рамки требуемой точности.

Численные методы решения нелинейных уравнений

Решение прикладных задач (в расчетах систем автоматического управления и регулирования, собственных колебаний машин и конструкций и др.) часто сводится к нахождению корней уравнений вида $f(x)=0$, где функция $f(x)$ определена и непрерывна на отрезке $[a,b]$. Если $f(x)$ – многочлен, то уравнение называется *алгебраическим*, если в $f(x)$ входят тригонометрические, логарифмические, показательные функции – то *трансцендентным*.

Аналитического решения многих алгебраических и трансцендентных уравнений найти не удается, поэтому используются приближенные итерационные (численные) методы.

Решение уравнения численным методом сводится к отысканию приближенных значений корней, т.е. к нахождению численных их величин, которые с заданной точностью обращают $f(x)$ в нуль. При этом приходится решать две задачи:

1) отделение корней, т.е. отыскание достаточно малых областей $[a,b]$, в каждой из которых содержится только один корень уравнения;

2) уточнение корней, т.е. вычисление корней с заданной точностью на найденном отрезке $[a,b]$.

Рассмотрим два метода решения второй задачи: *метод простых итераций* (метод последовательных приближений) и *метод бисекции* (метод деления отрезка пополам).

Метод простых итераций. Основным преимуществом этого метода является однотипность выполнения операций на каждом шаге, что в значительной степени облегчает составление программ.

Задача заключается в нахождении корня уравнения вида $f(x)=0$ на отрезке $[a,b]$ с заданной точностью ε.

Уравнение $f(x)=0$ заменяется равносильным ему уравнением вида $x=\varphi(x)$, что всегда можно сделать многими способами.

Выберем на отрезке $[a,b]$ начальное приближение, например, $x_0=\dfrac{a+b}{2}$, и, подставив его в правую часть преобразованного уравнения, вычислим следующее приближение к корню:

$$x_1=\varphi(x_0)\,.$$

Аналогично

$$x_2=\varphi(x_1)\,,$$

$$\cdots\cdots\cdots$$

$$\cdots\cdots\cdots$$

$$x_n=\varphi(x_{n-1})\,,$$

т.е. каждое новое значение корня вычисляется через предыдущее по формуле

$$x_i=\varphi(x_{i-1})\,. \qquad (1)$$

Процесс последовательного вычисления значений x_i по формуле (1) называется итерационным процессом (процессом последовательных приближений). Окончание итерационного процесса определяет условие достижения заданной точности ε:

$$\left|x_i-x_{i-1}\right|\le\varepsilon\,.$$

Установлено, что предел последовательности x_1, x_2, …, x_i, если он существует, является корнем уравнения вида $f(x)=0$, т.е. $\lim\limits_{i\to\infty} x_i = x$.

Условием сходимости итерационного процесса, т.е. условием существования предела, является соблюдение неравенства

$$\left|\varphi'(x)\right| < 1 .$$

Сходимость будет тем более быстрой, чем меньше величина $\left|\varphi'(x)\right|$, это необходимо учитывать при выборе функции $\varphi(x)$.

Блок-схема алгоритма нахождения корня уравнения с точностью ε методом итераций приведена на рис. 17, где x_0 и x_1 – предыдущее и последующее приближения к корню, соответственно, n – счетчик числа итераций.

Метод бисекции. Решается уравнение вида $f(x)=0$. Считаем, что отделение корней произведено и на отрезке $[a,b]$ расположен один корень, который необходимо уточнить с точностью ε. Функция $f(x)$ непрерывна на отрезке $[a,b]$ и имеет разные знаки на его концах, т.е. $f(a)\cdot f(b) < 0$. Если непрерывная функция на отрезке меняет знак, то она на этом отрезке имеет по крайней мере одно нулевое значение (соответствующее значение x при этом и есть корень уравнения).

Для нахождения корня отрезок $[a,b]$ делится пополам (рис. 18), т.е. выбирается начальное приближение корня $x = \dfrac{a+b}{2}$.

Вычисляем значение функции $f(x)$ в точке x, и если оно равно нулю или $\left|f(x)\right| < \varepsilon$, то это значение x является корнем уравнения.

Если $f(x) \neq 0$, то выбирается тот из отрезков $[a,x]$ или $[x,b]$, на концах которого функция $f(x)$ имеет противоположные знаки. Для этого проверяем: если условие $f(a)\cdot f(x) < 0$ выполняется, то выбираем отрезок $[a,x]$, а если не выполняется, то отрезок

[x,b]. Для того чтобы отрезок снова можно было записывать как [a,b], необходимо сделать переприсвоение переменных: при выполнении условия $f(a)\cdot f(x)<0$ это будет $b=x$, а при невыполнении – $a=x$.

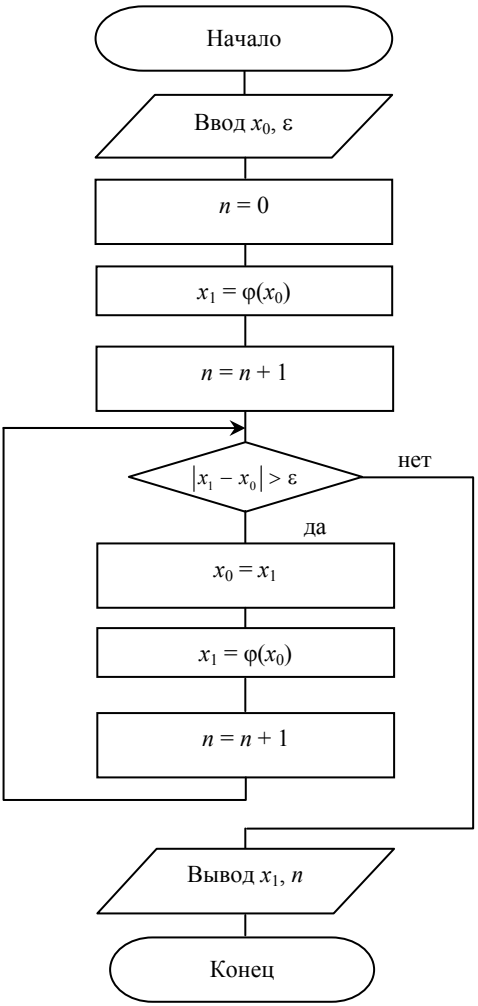


Рис. 17. Блок-схема алгоритма нахождения корня уравнения с точностью ε методом итераций

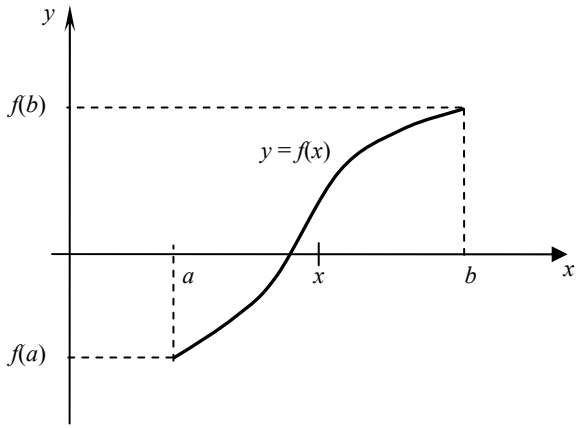


Рис. 18. Деление отрезка пополам
для нахождения корня уравнения

Полученный отрезок снова делится пополам, и повторяется то же рассмотрение.

Процесс деления отрезков пополам продолжается до тех пор, пока не будет найдено значение корня с заданной точностью, т.е. пока не выполнится условие $|f(x)| < \varepsilon$ или пока длина отрезка после очередного деления пополам не станет равной либо меньше заданной точности, т.е. $|b - a| \le \varepsilon$, и тогда корнем уравнения будет значение середины этого отрезка.

Блок-схема алгоритма нахождения корня уравнения с точностью ε методом бисекции приведена на рис. 19.

Решение систем линейных алгебраических уравнений. Метод Гаусса

Метод Гаусса (*метод последовательного исключения неизвестных*) основан на приведении матрицы коэффициентов к треугольному виду, из которой затем определяются значения неизвестных.

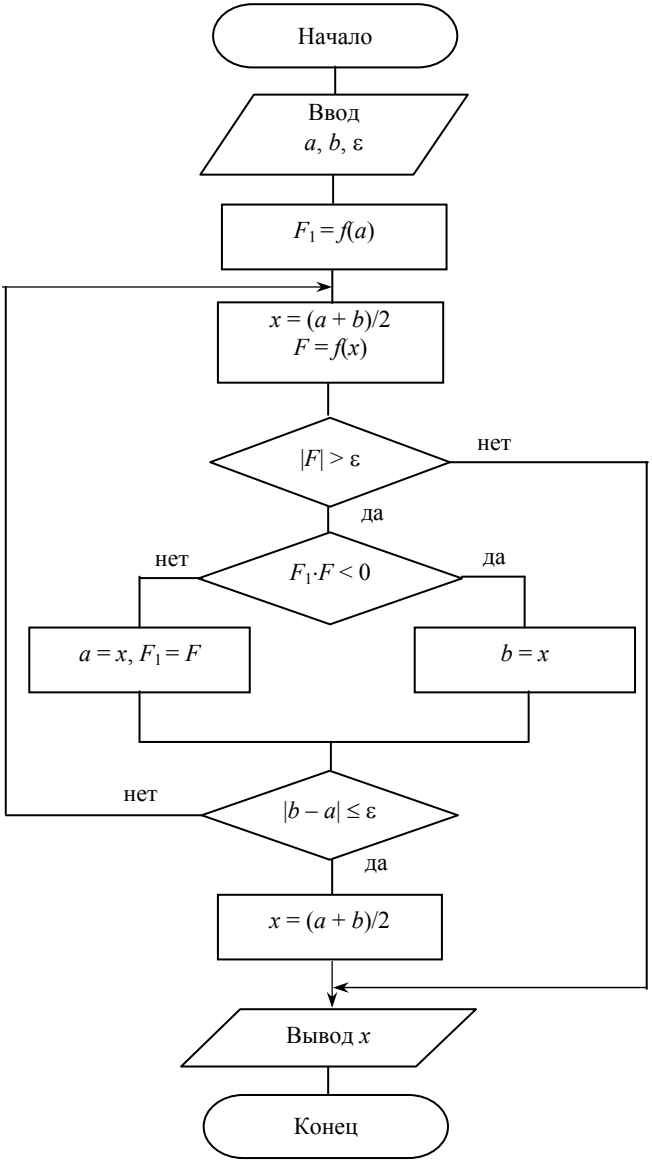


Рис. 19. Блок-схема алгоритма нахождения корня уравнения с точностью ε методом бисекции

Решается система линейных алгебраических уравнений вида

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + ... + a_{1n} \cdot x_n = b_1, \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + ... + a_{2n} \cdot x_n = b_2, \\ ..... \\ a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + ... + a_{nn} \cdot x_n = b_n \end{cases} \qquad (2)$$

или в матричной форме:

$$A \cdot x = b \,,$$

где $A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix}$, $x = \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{pmatrix}$, $b = \begin{pmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{pmatrix}$,

A – матрица коэффициентов системы; x – столбец неизвестных; b – столбец свободных членов.

Система уравнений (2) путем последовательного исключения неизвестных приводится к системе уравнений вида

$$\begin{cases} x_1 + \tilde{a}_{12} \cdot x_2 + ... + \tilde{a}_{1n} \cdot x_n = \tilde{b}_1, \\ x_2 + ... + \tilde{a}_{1n} \cdot x_n = \tilde{b}_2^{(1)}, \\ ..... \\ x_n = \tilde{b}_n^{(n-1)}. \end{cases} \qquad (3)$$

Коэффициенты системы (3) представляют собой треугольную матрицу с единичной главной диагональю. При этом формулы пересчета коэффициентов системы на каждом k-м шаге имеют вид

$$\tilde{a}_{kj}^{(k-1)} = \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad j = k, ..., n \,,$$

$$\tilde{b}_k^{(k-1)} = \frac{b_k^{(k-1)}}{a_{kk}^{(k-1)}},$$

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \cdot \tilde{a}_{kj}^{(k-1)}, \quad i = k+1,...,n; \quad j = k,...,n,$$

$$b_i^{(k)} = b_i^{(k-1)} - a_{ik}^{(k-1)} \cdot \tilde{b}_k^{(k-1)},$$

где верхний индекс (k – 1) соответствует предыдущему шагу прямого хода.

Приведение системы (2) к виду (3) называется *прямым ходом метода Гаусса*. При этом процесс исключения k-го неизвестного называется *k-м шагом прямого хода*.

Определение значений неизвестных из системы (3) называется *обратным ходом метода Гаусса* и осуществляется по формуле

$$x_i = \tilde{b}_i^{(i-1)} - \sum_{j=i+1}^{n} \tilde{a}_{ij}^{(i-1)} \cdot x_j, \quad i = n, n-1,...,1. \qquad (4)$$

Метод Гаусса с выбором главного элемента заключается в том, что при прямом ходе производится выбор наибольшего по модулю (главного) элемента в строке или в столбце матрицы коэффициентов и соответственно выполняется перестановка столбцов или строк. Это исключает деление на ноль, если матрица коэффициентов содержит нулевые элементы, и повышает точность.

При использовании *метода Гаусса с выбором главного элемента в столбце* на каждом k-м шаге сначала ищем элемент, равный $\max\limits_{k \le j \le n} \left| a_{jk}^{(k-1)} \right|$. Пусть это будет элемент $a_{mk}^{(k-1)}$. Меняем местами k-е и m-е уравнения и производим k-й шаг прямого хода метода Гаусса.

Блок-схема алгоритма решения системы линейных алгебраических уравнений методом Гаусса с выбором главного (максимального) элемента в столбце приведена на рис. 20–25.

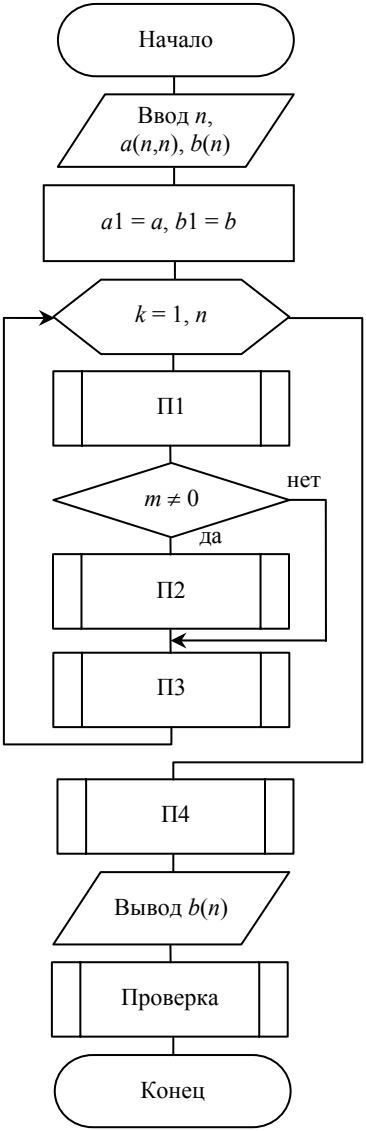


Рис. 20. Блок-схема основной программы

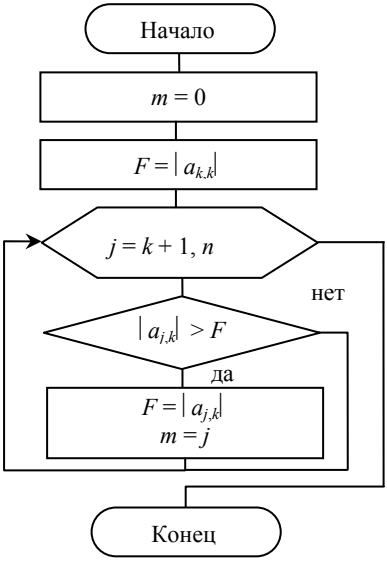


Рис. 21. Блок-схема подпрограммы (П1) поиска главного элемента в k-м столбце

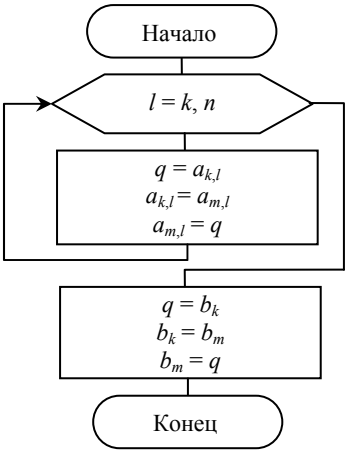


Рис. 22. Блок-схема подпрограммы (П2) перестановки k-го и m-го уравнений при $m \neq 0$

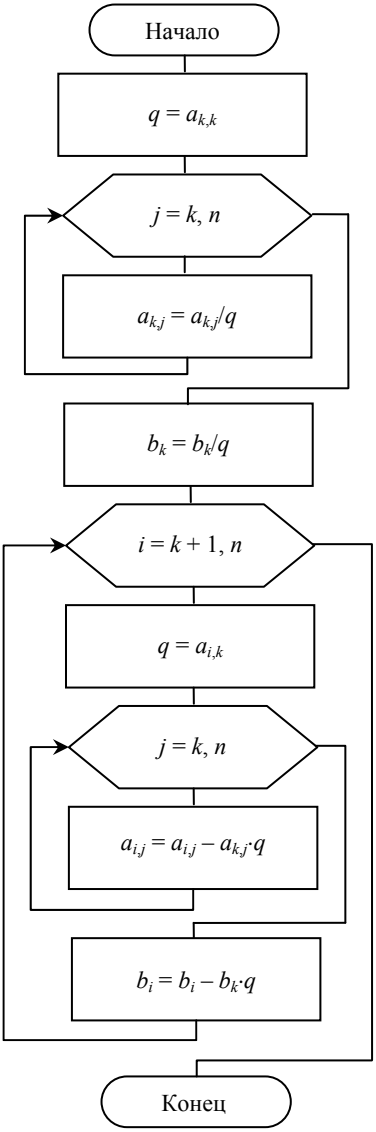


Рис. 23. Блок-схема подпрограммы (П3) прямого хода метода Гаусса (приведение системы к треугольному виду)

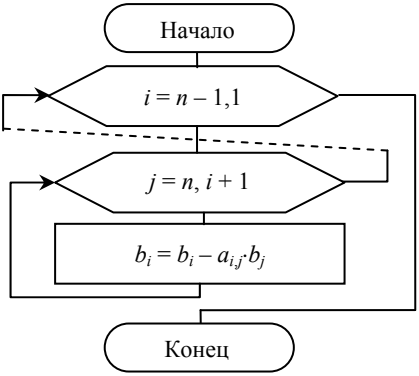


Рис. 24. Блок-схема подпрограммы (П4) обратного хода метода Гаусса

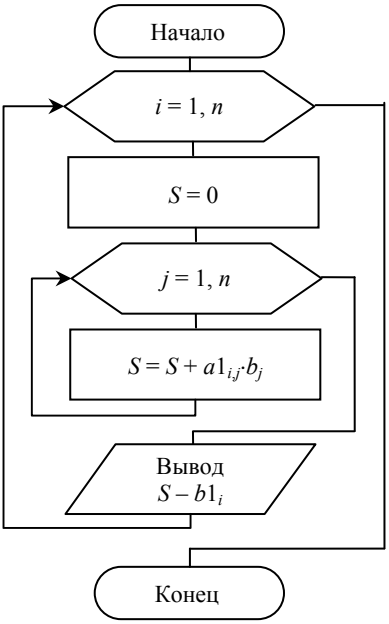


Рис. 25. Блок-схема подпрограммы (Проверка) проверки правильности решения

Решение задачи интерполяции

Интерполяция или *интерполирование* – способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений.

Одним из применений интерполирования является обработка результатов эксперимента с целью определения аналитического вида функции и вычисления ее промежуточных значений.

Задачу интерполяции можно считать обратной задаче табулирования функции. При табулировании по аналитическому выражению функции находится таблица ее значений.

При интерполяции – по таблице значений (например, полученных при проведении эксперимента) строится аналитическое выражение функции $y = f(x)$ для всех значений x на отрезке $[a,b]$, если известны ее значения y_i для некоторых точек x_i этого отрезка, т.е. $y_i = f(x_i), i = 1,2,...,n$, при этом кривая полученной функции должна проходить точно через имеющиеся точки данных. Точки x_i называются узлами интерполяции, а функция $y = f(x)$ – интерполирующей функцией.

Через заданный набор точек можно провести бесконечное число кривых, поэтому задача интерполяции не имеет единственного решения, при этом каждая кривая будет описываться своим аналитическим выражением. Полученную интерполяционную формулу в дальнейшем можно использовать для приближённого вычисления значений функции при значениях аргумента, отличных от таблично заданных, а также исследовать методами математического анализа (например, интегрировать и дифференцировать), рис. 26.

На практике чаще всего применяют *интерполяцию многочленами*.

При *линейной интерполяции* считается, что на каждом отрезке $[x_i,x_{i+1}]$ функция $y = f(x)$ является линейной, представленной алгебраическим двучленом,

$$f(x) = a_0 + a_1 \cdot x\,. \qquad (5)$$

В случае если заданы значения в нескольких точках, то получается кусочно-линейная функция.

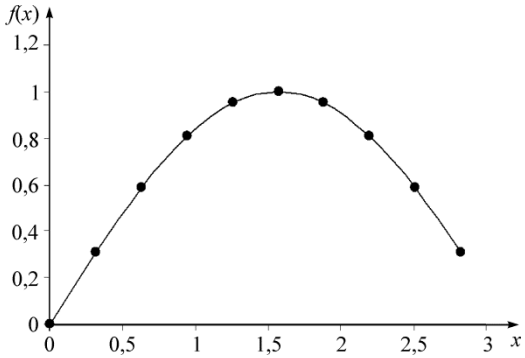


Рис. 26. Построение интерполирующей функции по дискретному набору исходных табличных данных: ● – исходные табличные данные; — интерполирующая функция $f(x)$

Зависимость (5) полностью определена и может быть использована для вычисления y при любых $x \in [a,b]$, если известны a_0 и a_1. Подставив в (5) координаты точек (x_i,y_i) и (x_{i+1},y_{i+1}), получим систему уравнений для нахождения a_0 и a_1

$$\begin{cases} a_0 + a_1 \cdot x_i = y_i, \\ a_0 + a_1 \cdot x_{i+1} = y_{i+1}. \end{cases}$$

Интерполяция алгебраическим многочленом для набора точек (x_i,y_i), $i = 1,2,\ldots,n+1$ отрезка $[a,b]$ – построение многочлена $P_n(x)$ степени не выше n, который записывается в виде

$$P_n(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 \cdot x + a_0 = y. \qquad (6)$$

Значения a_0, a_1, …, a_n, при которых многочлен (6) проходит через все заданные точки, могут быть получены из решения системы уравнений, каждое из которых записывается для узла интерполяции:

$$\begin{cases} a_n \cdot x_1^n + a_{n-1} \cdot x_1^{n-1} + \ldots + a_1 \cdot x_1 + a_0 = y_1, \\ a_n \cdot x_2^n + a_{n-1} \cdot x_2^{n-1} + \ldots + a_1 \cdot x_2 + a_0 = y_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_n \cdot x_{n+1}^n + a_{n-1} \cdot x_{n+1}^{n-1} + \ldots + a_1 \cdot x_{n+1} + a_0 = y_{n+1}. \end{cases} \qquad (7)$$

Решение системы линейных уравнений (7) относительно неизвестных a_0, a_1, …, a_n может быть выполнено методом Гаусса.

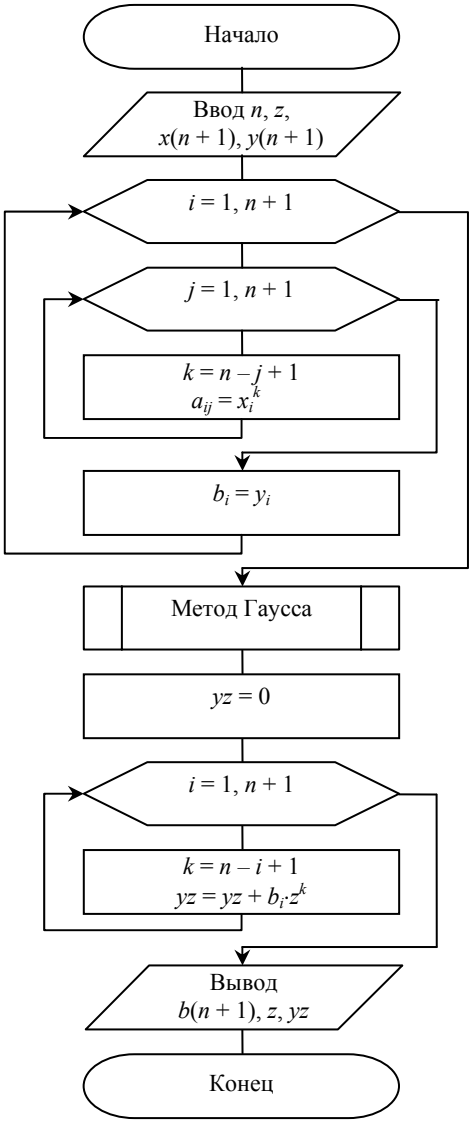


Рис. 27. Блок-схема алгоритма решения задачи интерполяции (интерполяция алгебраическим многочленом)

Блок-схема алгоритма решения задачи интерполяции алгебраическим многочленом приведена на рис. 27, где a_{ij} – матрица коэффициентов системы уравнений; b_i – вектор свободных членов, векторы x и y – исходные табличные данные; z – значение аргумента x, для которого вычисляется промежуточное значение функции yz после нахождения коэффициентов многочлена. В результате выполнения приведенного на рис. 27 алгоритма получаем $a_n = b_1$, $a_{n-1} = b_2$, …, $a_0 = b_{n+1}$.

Аппроксимация табличных зависимостей

Под *аппроксимацией* (приближением) понимается замена одних объектов (зависимостей) другими, близкими к исходным, но более простыми.

Аппроксимация применяется при обработке результатов измерений для получения простых приближенных формул, определяющих зависимость между табличными данными.

В задачах интерполяции требовалось, чтобы искомая зависимость $y = f(x)$ совпадала со значениями таблично заданной функции $y_i = f(x_i), i = 1,2,...,n$ в узлах интерполяции.

На практике часто бывает, что аналитическое выражение такой зависимости получить невозможно или очень сложно. В этом случае решается задача аппроксимации, т.е. задача нахождения такой достаточно простой функции, значения которой на отрезке $[a,b]$, содержащем узлы интерполяции x_i, $i = 1,2,...,n$, возможно, мало отличаются от значений таблично заданной функции (рис. 28).

Для решения задачи аппроксимации необходимо выбрать вид аппроксимирующей функции исходя из анализа графического представления $y_i = f(x_i), i = 1,2,...,n$, а затем по исходным данным определить значения ее параметров.

Аппроксимирующая функция может быть представлена гиперболической, экспоненциальной, логарифмической или степенной зависимостью, а также многочленом $P_m(x)$ степени m вида

$$P_m(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + ... + a_m \cdot x^m = y. \qquad (8)$$

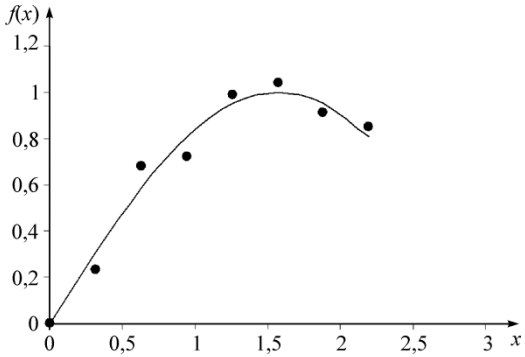


Рис. 28. Построение аппроксимирующей функции по дискретному набору исходных табличных данных: ● – исходные табличные данные; — аппроксимирующая функция $f(x)$

Для определения параметров a_0, a_1, …, a_m зависимости (8) по исходным данным x_i, y_i, $i = 1,2,\ldots,n$ широкое распространение получил *метод наименьших квадратов*. При этом степень многочлена m обычно принимается меньшей числа узлов интерполяции, т.е. $m < n$.

При аппроксимации данных методом наименьших квадратов определяют такие значения a_0, a_1, …, a_m зависимости $P_m(x)$, при которых достигается минимум суммы квадратов отклонений этой функции от табличных значений y_i, т.е.

$$S = \sum_{i=1}^{n} (P_m(x_i) - y_i)^2 \to \min$$

или

$$S = \sum_{i=1}^{n} \left(a_0 + a_1 \cdot x_i + a_2 \cdot x_i^2 + \ldots + a_m \cdot x_i^m - y_i \right)^2 \to \min. \qquad (9)$$

Минимум функции будет найден, если приравнять к нулю частные производные от S по a_0, a_1, …, a_m, при этом получается система $m+1$ уравнений

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 2\sum\limits_{i=1}^{n}(a_0 + a_1 \cdot x_i + a_2 \cdot x_i^2 + \ldots + a_m \cdot x_i^m - y_i) \cdot 1 = 0, \\ \dfrac{\partial S}{\partial a_1} = 2\sum\limits_{i=1}^{n}(a_0 + a_1 \cdot x_i + a_2 \cdot x_i^2 + \ldots + a_m \cdot x_i^m - y_i) \cdot x_i = 0, \\ \dfrac{\partial S}{\partial a_2} = 2\sum\limits_{i=1}^{n}(a_0 + a_1 \cdot x_i + a_2 \cdot x_i^2 + \ldots + a_m \cdot x_i^m - y_i) \cdot x_i^2 = 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \dfrac{\partial S}{\partial a_m} = 2\sum\limits_{i=1}^{n}(a_0 + a_1 \cdot x_i + a_2 \cdot x_i^2 + \ldots + a_m \cdot x_i^m - y_i) \cdot x_i^m = 0. \end{cases} \qquad (10)$$

Сократив каждое уравнение системы (10) на 2, раскрыв скобки и сгруппировав суммы, получаем следующую систему (для краткости записи вместо $\sum\limits_{i=1}^{n}$ используем $\sum$):

$$\begin{cases} a_0 \cdot n + a_1 \sum x_i + a_2 \sum x_i^2 + \ldots + a_m \sum x_i^m = \sum y_i, \\ a_0 \sum x_i + a_1 \sum x_i^2 + a_2 \sum x_i^3 + \ldots + a_m \sum x_i^{m+1} = \sum y_i \cdot x_i, \\ a_0 \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4 + \ldots + a_m \sum x_i^{m+2} = \sum y_i \cdot x_i^2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_0 \sum x_i^m + a_1 \sum x_i^{m+1} + a_2 \sum x_i^{m+2} + \ldots + a_m \sum x_i^{2 \cdot m} = \sum y_i \cdot x_i^m. \end{cases} \qquad (11)$$

Система уравнений (11) относительно неизвестных a_0, a_1, …, a_m является линейной, поэтому может быть решена, например, методом Гаусса. Полученные в результате значения a_0, a_1, …, a_m подставляют в (8), что позволяет полностью определить аппроксимирующую функцию.

Блок-схема алгоритма аппроксимации табличных зависимостей многочленом степени m для n точек методом наименьших квадратов приведена на рис. 29. В связи с тем, что матрица коэффициентов системы уравнений a_{ij} содержит попарно повторяющиеся элементы, для ее формирования используется вспомогательный

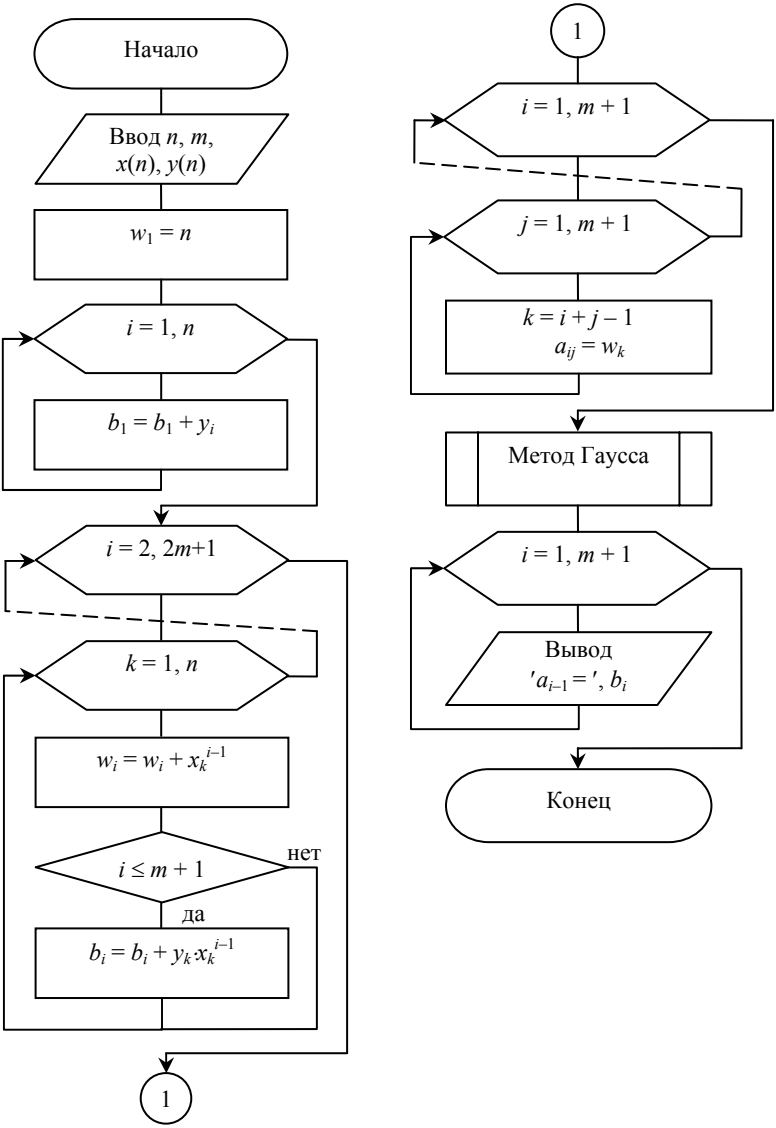


Рис. 29. Блок-схема алгоритма аппроксимации табличных зависимостей многочленом степени m для n точек методом наименьших квадратов

массив w – вектор с количеством элементов, равным $2m+1$. Кроме этого, в блок-схеме использованы следующие обозначения: векторы x и y – исходные табличные данные; n – количество точек таблицы; m – степень многочлена (аппроксимирующей функции); b_i – вектор свободных членов. В результате выполнения приведенного на рис. 29 алгоритма получаем: $a_0 = b_1$, $a_1 = b_2$, …, $a_m = b_{m+1}$.

Численные методы решения дифференциальных уравнений

Моделирование самых разнообразных процессов, разработка и исследование новых систем автоматического управления различными объектами и многие другие проблемы связаны с необходимостью численного решения дифференциальных уравнений и их систем.

Уравнение, связывающее искомую функцию одной или нескольких переменных, эти переменные и производные различных порядков данной функции, называется *дифференциальным уравнением*. Например, дифференциальными уравнениями будут $y' = 6x$, $y'' = 10 - 3x$.

В общем случае дифференциальное уравнение можно записать в виде

$$G(x, y, y', \ldots, y^n) = 0,$$

где G – некоторая функция от $n + 2$ переменных, при этом порядок n старшей производной, входящей в запись уравнения, называется порядком дифференциального уравнения.

Решением дифференциального уравнения называется такая функция $y = y(x)$, которая при подстановке ее в это уравнение обращает его в тождество.

Задача о нахождении решения некоторого дифференциального уравнения называется задачей интегрирования данного дифференциального уравнения. График решения дифференциального уравнения называется интегральной кривой. Без дополнительных предположений решение дифференциального урав-

нения принципиально неоднозначно, так как дифференциальное уравнение задает семейство интегральных кривых на плоскости. Для выделения однозначно определенной кривой (решения) необходимо указать точку плоскости, через которую проходит интегральная кривая, и направление, в котором она проходит через эту точку. Дополнительные условия такого рода называют начальными, поскольку часто дифференциальные уравнения используются для описания динамических процессов – процессов, проходящих во времени.

Математическое описание многих явлений часто сводится к системе дифференциальных уравнений вида

$$\begin{cases} \dfrac{dy_1}{dx} = f_1\left(x, y_1, y_2, ..., y_n\right), \\ \dfrac{dy_2}{dx} = f_2\left(x, y_1, y_2, ..., y_n\right), \\ ..... \\ \dfrac{dy_n}{dx} = f_n\left(x, y_1, y_2, ..., y_n\right). \end{cases} \tag{12}$$

Задача Коши для системы дифференциальных уравнений (12) заключается в отыскании функций y_1, y_2, …, y_n, удовлетворяющих этой системе и начальным условиям

$$\begin{aligned} &y_1(x_0) = y_{1_0}, \\ &y_2(x_0) = y_{2_0}, \\ &..... \\ &y_n(x_0) = y_{n_0}, \end{aligned}$$

где x – независимая переменная; $y_1(x)$, $y_2(x)$, …, $y_n(x)$ – неизвестные (искомые) функции.

Существует возможность перехода от одной формы записи к другой. Так, от записи в виде одного дифференциального уравнения n-го порядка можно перейти к записи в виде системы n дифференциальных уравнений 1-го порядка и наоборот.

Дифференциальное уравнение *n*-го порядка

$$y^{(n)} = f(x, y, y', ..., y^{(n-1)}) \tag{13}$$

приводится к виду (12) с помощью замены (следующих преобразований):

$$y = y_0, \quad y' = y_1, \quad y'' = y_2, \quad ..., \quad y^{(n-1)} = y_{n-1},$$

в результате получаем систему *n* дифференциальных уравнений 1-го порядка:

$$\begin{cases} \dfrac{dy_0}{dx} = y_1, \\ \dfrac{dy_1}{dx} = y_2, \\ ..... \\ \dfrac{dy_{n-2}}{dx} = y_{n-1}, \\ \dfrac{dy_{n-1}}{dx} = f(x, y_0, y_1, y_2, ..., y_{n-1}). \end{cases} \tag{14}$$

Следовательно, решение дифференциального уравнения *n*-го порядка сводится к решению системы *n* дифференциальных уравнений 1-го порядка.

Например, дифференциальное уравнение 2-го порядка $y'' - 2y' + y = 5xe^x$ с помощью замены $y = y_0, \quad y' = y_1$ приводится к системе дифференциальных уравнений 1-го порядка

$$\begin{cases} \dfrac{dy_0}{dx} = y_1, \\ \dfrac{dy_1}{dx} = 5xe^x - y_0 + 2y_1. \end{cases}$$

Методика численного решения дифференциального уравнения 1-го порядка на отрезке $[x_0, x_k]$

$$\frac{dy}{dx} = y'(x) = f(x, y) \tag{15}$$

с начальными условиями x_0, $y(x_0) = y_0$ базируется на разложении искомой функции в ряд Тейлора в h-окрестности точки x_0:

$$y_1 = y(x_0 + h) = y(x_0) + h \cdot y'(x_0) + \frac{h^2}{2!} y''(x_0) + \ldots \qquad (16)$$

Численные методы решения дифференциальных уравнений позволяют получить искомую функцию в виде таблицы ее приближенных значений $y_1(x_1)$, $y_2(x_2)$, …, $y_k(x_k)$.

Метод Эйлера

При отбрасывании всех членов ряда (16), содержащих производные 2-го и высших порядков, с учетом (15) получаем

$$y_1 = y(x_0 + h) = y_0 + h \cdot f(x_0, y_0),$$

где $f(x_0, y_0)$ – правая часть дифференциального уравнения (15).

Пользуясь значением y_1, получим значение искомой функции в следующей точке $x_2 = x_1 + h$:

$$y_2 = y(x_1 + h) = y_1 + h \cdot f(x_1, y_1).$$

Таким образом, если y_i – приближенное значение искомой функции в точке x_i, тогда y_{i+1} в следующей точке $x_{i+1} = x_i + h$ на каждом шаге интегрирования по методу Эйлера вычисляется по формуле

$$y_{i+1} = y_i + h \cdot f(x_i, y_i). \qquad (17)$$

Алгоритм решения дифференциального уравнения 1-го порядка методом Эйлера представляет собой циклический процесс вычислений искомой функции y по формуле (17) при изменении аргумента x от x_0 до x_k с шагом h.

Блок-схема алгоритма численного решения дифференциального уравнения 1-го порядка вида (15) методом Эйлера приведена на рис. 30.

Погрешность метода Эйлера ε определяется отбрасываемой частью разложения функции в ряд Тейлора (16), ε = h^2. Следова-

тельно, точность решения дифференциальных уравнений методом Эйлера тем выше, чем меньше шаг интегрирования h.

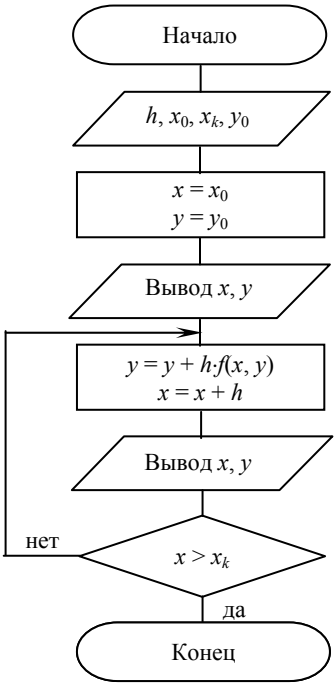


Рис. 30. Блок-схема алгоритма решения дифференциального уравнения 1-го порядка методом Эйлера

Дифференциальное уравнение n-го порядка ($n > 1$) при решении численным методом Эйлера, как и другими методами, приводится к системе дифференциальных уравнений 1-го порядка. При этом формула (17) используется для каждого из полученных уравнений. Совместная система уравнений на каждом шаге интегрирования решается одновременно.

12.3. Объектно-ориентированное программирование

Дальнейшим развитием концепции структурного программирования является концепция объектно-ориентированного программирования. Суть концепции заключается в том, что про-

грамма строится как набор неких сущностей или объектов, которые имеют свое внутреннее состояние и поведение. Объекты взаимодействуют путем передачи сообщений друг другу. Таким образом, концепция объектно-ориентированного программирования объединяет в одну сущность и данные, и процедуры, и функции для работы с этими данными.

Естественным образом выстраивается иерархия объектов: программа в целом – это объект, для выполнения своих функций она обращается к входящим в нее объектам, которые, в свою очередь, выполняют запрошенное путем обращения к другим объектам программы. Чтобы избежать бесконечной рекурсии в обращениях, на каком-то этапе объект трансформирует обращенное к нему сообщение в сообщения к стандартным системным объектам, предоставляемым языком и средой программирования.

Устойчивость и управляемость системы обеспечивается за счет четкого разделения ответственности объектов (за каждое действие отвечает определенный объект), однозначного определения интерфейсов межобъектного взаимодействия и полной изолированности внутренней структуры объекта от внешней среды (инкапсуляции).

В объектно-ориентированном программировании существуют следующие базовые понятия.

Инкапсуляция. Инкапсуляция – это свойство системы, позволяющее объединить данные и методы, работающие с ними, в один класс. Таким образом, от тех, кто использует разработанный класс, скрываются особенности внутренней реализации алгоритмов.

Наследование. Наследование – свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствованной функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс – потомком, наследником, дочерним или производным классом.

Полиморфизм. Полиморфизм – свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Интерфейс. Интерфейс – это определение набора методов, которые должны быть в классе, который реализует данный интерфейс.

Класс. Класс – это описание на языке программирования структуры данных и методов по работе с ними. Класс может реализовывать какой-либо интерфейс или сам определять свой интерфейс. Фактически класс описывает устройство еще не существующей сущности и является своего рода ее чертежом. Для использования класса программист создает объекты класса, которые являются материальными сущностями, расположенными в памяти работающей программы. У каждого объекта имеется своя версия данных, но все объекты используют общие методы того класса, от которого они созданы. Обычно классы разрабатывают таким образом, чтобы их объекты соответствовали объектам предметной области.

Объект. Объект – это сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса.

13. СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Внедрение персональных компьютеров в информационную сферу и применение телекоммуникационных средств связи определили современный этап развития информационных технологий (ИТ).

Основные черты современных ИТ:

– структурированность стандартов цифрового обмена данными;

– широкое использование компьютерного сохранения и предоставление информации в необходимом виде;

– передача информации посредством цифровых технологий на практически безграничные расстояния.

Современные ИТ, наиболее часто используемые в системах различного типа и назначения:

– математическое и компьютерное моделирование;

– базы данных и базы знаний;

– экспертные и интеллектуальные системы;

– средства, технологии планирования и управления с помощью электронных таблиц;

– электронная почта и телекоммуникационные средства;

– интегрированные пакеты прикладных программ и среды;

– средства, методы и технологии компьютерной графики и анимации;

– средства, методы и технологии мультимедиа;

– гипертекстовые технологии и WWW-технологии;

– CASE -технологии и др.

ВОПРОСЫ ПО ДИСЦИПЛИНЕ «ИНФОРМАТИКА» (ГЛАВЫ 12–13)

1. Жизненный цикл программного обеспечения.
2. Переносимость программ. Понятие интеллектуальной собственности.
3. Структурное программирование. Принципы структурного программирования.
4. Основы математического моделирования. Этапы и цели математического моделирования.
5. Численные методы решения нелинейных уравнений. Метод простых итераций и метод дихотомии.
6. Решение систем линейных алгебраических уравнений. Метод Гаусса.
7. Решение задачи интерполяции.
8. Аппроксимация табличных зависимостей.
9. Численные методы решения дифференциальных уравнений. Метод Эйлера.
10. Объектно-ориентированное программирование.
11. Современные информационные технологии.

СПИСОК ЛИТЕРАТУРЫ

1. Информатика. Базовый курс: учеб. пособие для втузов / под ред. С.В. Симонович. – 3-е изд. – СПб.: Питер, 2012. – 637 с.

2. Текстовый процессор: метод. указания к выполнению лаб. работ по дисциплине «Информатика» [Электронный ресурс] / сост. И.Н. Щапова. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2015. – 20 с. – URL: http://pstu.ru/title1/faculties/gnf/?infores=1.

3. Электронные таблицы: метод. указания к выполнению лаб. работ по дисциплине «Информатика» [Электронный ресурс] / сост. И.Н. Щапова. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2015. – 26 с. – URL: http://pstu.ru/title1/faculties/gnf/?infores=1.

4. Аляев Ю.А., Козлов О.А. Алгоритмизация и языки программирования Pascal, C++, Visual Basic: учеб.-справ. пособие. – М.: Финансы и статистика, 2007. – 319 с.

5. Семакин И.Г., Шестаков А.П. Основы программирования: учебник. – 7-е изд. – М.: Академия, 2008. – 431 с.

6. Справка PascalABC.NET. – URL: http://pascalabc.net/downloads/pabcnethelp/index.htm (дата обращения 10.11.2015).

7. Программные средства реализации алгоритмов. Алгоритмизация и программирование задач по обработке массивов: метод. указания к выполнению лаб. работ по дисциплине «Информатика» [Электронный ресурс] / сост. И.Н. Щапова. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2015. – 35 с. – URL: http://pstu.ru/title1/faculties/gnf/?infores=1.

8. Макаров Е. Инженерные расчеты в Mathcad 14. – СПб.: Питер, 2007 – 591 с.

9. Система управления базами данных Microsoft Access: метод. указания к выполнению лаб. работ / сост. И.Н. Щапова. – Пермь: Изд-во Перм. нац. исслед. политехн. ун-та, 2016. – 21 с.

10. Могилев А.В., Пак Н.И., Хеннер Е.К. Информатика: учеб. пособие для вузов / под ред. Е.К. Хеннера. – 8-е изд., стер. – М.: Академия, 2012. – 841 с.

ПРИЛОЖЕНИЕ 1

КОНТРОЛЬНАЯ РАБОТА ДЛЯ СТУДЕНТОВ ЗАОЧНОГО ОТДЕЛЕНИЯ

Цель работы:

– ознакомление с основными принципами работы в программах Microsoft Word, Microsoft Excel, Microsoft Access и Microsoft PowerPoint пакета Microsoft Office (задания 1, 2, 3);

– овладение практическими навыками разработки и программирования алгоритмов решения задач (задания 4, 5).

1. Выполнить задания контрольной работы по дисциплине «Информатика» по варианту, номер которого совпадает с последней цифрой номера зачетной книжки. Так, например, если номер оканчивается цифрой 5, то нужно выполнять в каждом из заданий контрольной работы задание под номером 5.

2. Подготовить презентацию (например, с помощью программы Microsoft Office PowerPoint) по материалу контрольной работы (6–10 слайдов).

Общие методические указания

При выполнении и оформлении контрольной работы необходимо соблюдать следующие указания:

1) Контрольную работу следует оформлять в соответствии с требованиями (см. прил. 2).

2) Титульный лист контрольной работы оформляется в соответствии с образцом (см. прил. 3).

3) Список используемой литературы должен быть оформлен в соответствии с ГОСТ Р 7.05 – 2008 «Библиографическая ссылка. Общие требования и правила составления» (см. прил. 4).

4) В работу должны быть включены все задания строго по варианту. Работы, содержащие не все задания, а также задания не своего варианта, не зачитываются.

Задание 1
Текстовый процессор Microsoft Word

1. Опишите средства рецензирования текста в текстовом процессоре Microsoft Word.

2. Перечислите базовые приемы работы с текстами в текстовом процессоре Microsoft Word.

3. Напишите об использовании средства «Автозамена» при вводе текста в текстовом процессоре Microsoft Word. Ввод специальных символов.

4. Опишите последовательность действий по созданию оглавления средствами текстового процессора Microsoft Word.

5. Опишите возможные настройки и дополнительные параметры, выбираемые в окне «Параметры Word» текстового процессора Microsoft Word.

6. Опишите возможности Microsoft Word для работы с графическими объектами.

7. Опишите последовательность действий по созданию многоуровневых списков и колонтитулов в текстовом процессоре Microsoft Word.

8. Опишите возможности Microsoft Word по вводу и редактированию формул.

9. Опишите последовательность действий по созданию и редактированию таблиц в текстовом процессоре Microsoft Word.

10. Опишите приемы форматирования текста в текстовом процессоре Microsoft Word.

Задание 2
Электронные таблицы Microsoft Excel

1. Опишите назначение и возможности программы Microsoft Excel.

2. Перечислите и поясните основные типы данных, которые могут быть записаны в ячейки электронной таблицы Microsoft Excel. Приведите примеры имен ячеек. Какие действия могут быть выполнены применительно к ячейке и к диапазону ячеек?

3. Опишите возможности программы Microsoft Excel по созданию и редактированию формул. Приведите примеры формул электронной таблицы Microsoft Excel. Опишите последовательность действий, необходимых для отображения в ячейке не результата вычислений по формуле, а самой формулы.

4. Применение абсолютных и относительных ссылок в программе Microsoft Excel. Опишите процесс копирования формул и получаемый при этом результат в зависимости от использования абсолютных и относительных ссылок в формулах.

5. Приведите примеры функций электронной таблицы Microsoft Excel. Опишите назначение Мастера функций. Как обозначается диапазон ячеек и несколько диапазонов ячеек при указании их в качестве аргументов функций?

6. Опишите возможности программы Microsoft Excel по созданию и редактированию диаграмм.

7. Опишите возможные настройки и дополнительные параметры, выбираемые в окне «Параметры Excel» электронной таблицы Microsoft Excel.

8. Опишите последовательность действий для решения уравнений средствами программы Microsoft Excel.

9. Опишите последовательность действий для решения задачи оптимизации средствами программы Microsoft Excel.

10. Опишите возможности программы Microsoft Excel по вводу и форматированию текста в ячейках таблицы.

Задание 3
Система управления базами данных Microsoft Access

1. Опишите назначение и типы Запросов в СУБД Microsoft Access.

2. Опишите возможности СУБД Microsoft Access по формированию структуры Таблицы в режиме конструктора.

3. Опишите возможности СУБД Microsoft Access по созданию и использованию Форм.

4. Опишите основные объекты СУБД Microsoft Access.

5. Опишите назначение и возможности системы управления базами данных Microsoft Access.

6. Опишите основные этапы разработки базы данных в программе Microsoft Access.

7. Опишите возможности СУБД Microsoft Access по формированию новой Таблицы на основе шаблона таблиц.

8. Опишите возможности СУБД Microsoft Access по созданию Отчетов.

9. Опишите типы данных, используемые в СУБД Microsoft Access.

10. Опишите возможности СУБД Microsoft Access по созданию межтабличных связей (схемы данных).

Методические указания к заданиям 4, 5

1. Каждое задание должно включать:

– содержание задания;

– блок-схему алгоритма решения задачи;

– текст программы на языке программирования Pascal ABC.NET;

– результаты решения задачи на компьютере.

В программу должны быть включены комментарии, которые поясняют работу и назначение отдельных частей программы, характеризуют используемые переменные.

Задание 4
Программирование алгоритмов разветвляющейся и циклической структуры

Реализация приема программирования – табулирования функции от одного аргумента (вычисление значений функции при изменении значения аргумента в заданном диапазоне с шагом Δx).

Задание: разработать алгоритм табулирования функции. Вычислить значение функции при изменении аргумента в указанном диапазоне и с заданным шагом. Организовать вывод значений аргумента и вычисленных значений функции в виде таблицы:

ТАБЛИЦА	ФУНКЦИИ $Y(X)$
X	Y
.....	.....
.....	.....
.....	.....

Варианты заданий:

1. $$y = \begin{cases} a \cdot x^2 \cdot \ln(x) & \text{при } 1 \le x \le 2, \\ 1 & \text{при } x < 1, \\ e^{ax} \cdot \cos(b \cdot x) & \text{при } x > 2. \end{cases}$$

$a = -0{,}5$; $b = 2$

$x \in [0{,}15;3]$; $\Delta x = 0{,}15$

2. $$y = \begin{cases} a \cdot x^2 + b \cdot x + c & \text{при } x < 1{,}2, \\ \dfrac{a}{x} + \sqrt{x^2 + 1} & \text{при } x = 1{,}2, \\ \dfrac{a + b \cdot x}{\sqrt{x^2 + 1}} & \text{при } x > 1{,}2. \end{cases}$$

$a = -2{,}8$; $b = -0{,}3$; $c = 4$

$x \in [1;2]$; $\Delta x = 0{,}05$

3. $$y = \begin{cases} \pi \cdot x^2 - \dfrac{7}{x^2} & \text{при } x < 1{,}3, \\ a \cdot x^3 + 7\sqrt{x} & \text{при } x = 1{,}3, \\ \lg(x + 7\sqrt{x}) & \text{при } x > 1{,}3. \end{cases}$$

$a = 1{,}5$

$x \in [0{,}8;2]$; $\Delta x = 0{,}1$

4. $$y=\begin{cases}1,5\cos^2(x) & \text{при } x<1,\\ 1,8a\cdot x & \text{при } x=1,\\ (x-2)^2+6 & \text{при } 1<x<2,\\ 3\operatorname{tg}(x) & \text{при } x\ge 2.\end{cases}$$

$a = 2,3$

$x \in [0,2;\ 2,8];\quad \Delta x = 0,2$

5. $$y=\begin{cases}\pi\cdot x-\dfrac{7}{x} & \text{при } x<1,4,\\ a\cdot x+7\sqrt{x} & \text{при } x=1,4,\\ \ln(x+7\sqrt{|x+a|}) & \text{при } x>1,4.\end{cases}$$

$a = 1,65$

$x \in [0,7;\ 2];\quad \Delta x = 0,1$

6. $$y=\begin{cases}x\sqrt[3]{x-a} & \text{при } x>a,\\ x\cdot\sin(a\cdot x) & \text{при } x=a,\\ e^{-ax}\cdot\cos(a\cdot x) & \text{при } x<a.\end{cases}$$

$a = 2,5$

$x \in [1;5];\quad \Delta x = 0,5$

7. $$y=\begin{cases}b\cdot x-\lg(b\cdot x) & \text{при } b\cdot x<1,\\ 1 & \text{при } b\cdot x=1,\\ b\cdot x+\lg(b\cdot x) & \text{при } b\cdot x>1.\end{cases}$$

$b = 1,5$

$x \in [0,1;1];\quad \Delta x = 0,1$

8. $$y=\begin{cases}\sqrt{a\cdot x^2+b\cdot\sin(x)+1} & \text{при } x<0{,}1,\\ a\cdot x+b & \text{при } x=0{,}1,\\ \sqrt{a\cdot x^2+b\cdot\cos(x)+1} & \text{при } x>0{,}1.\end{cases}$$

$a = 1{,}5$; $b = 0{,}4$

$x \in [-1;1]$; $\Delta x = 0{,}2$

9. $$y=\begin{cases}a\cdot\lg(x)+\sqrt[3]{|x|} & \text{при } x>1,\\ 2a\cdot\cos(x)+3x^2 & \text{при } x\le 1.\end{cases}$$

$a = 0{,}9$

$x \in [0{,}8;2]$; $\Delta x = 0{,}1$

10. $$y=\begin{cases}\dfrac{a+b}{e^x+\cos(x)} & \text{при } x<2{,}8,\\ \dfrac{a+b}{x+1} & \text{при } 2{,}8\le x<6,\\ e^x+\sin(x) & \text{при } x\ge 6.\end{cases}$$

$a = 2{,}6$; $b = -0{,}39$

$x \in [0; 7]$; $\Delta x = 0{,}5$

Пример выполнения задания 4

Таблица функции состоит из заголовка и строк, содержащих значение аргумента в некоторой точке интервала и соответствующее значение функции. Поэтому, записывая программу, сначала программируем вывод заголовка. Затем в цикле вычисляем значение функции и выводим текущую строку таблицы (см. рис. 6).

Список переменных:

a, *b* – коэффициенты функции;

*x*0, *xк* – начальное и конечное значения интервала изменения аргумента;

dx – шаг изменения аргумента;

y – значение функции в точке *x*.

Текст программы, реализующей указанный алгоритм (для задания варианта 1):

```
program z1;
var  a, b, x, y, x0, xk, dx : real;
begin
 write ('Введите a, b, x0, xk, dx');
 readln (a, b, x0, xk, dx);
 writeln (' ТАБЛИЦА ФУНКЦИИ Y(X) ');
 writeln ('X', '         ', 'Y(X) ');
 x:=x0;
 repeat
  if (x>=1) AND (x<=2) then y:=a* *x*x*ln(x);
  if  x<1  then  y:=1;
  if x>2 then y:=exp(a*x)* *cos(b*x);
     writeln (x:8:3, '   ', y:8:3);
     x:=x+dx;
  until  x>xk;
end.
```

С помощью процедуры `writeln (x:8:3,'  ', y:8:3);` осуществляется форматный вывод на экран значений *x* и *y*. Под каждое из этих значений выделяется по восемь позиций, три из них под дробную часть.

Задание 5
Программирование вычисления суммы и произведения

Пример алгоритма вычисления суммы и произведения (рис. 7).

Вычислить:

1. $\sum_{i=1}^{7} \frac{1}{i(i+1)(i+2)}$

   $\prod_{k=1}^{7} \frac{1}{k(k+1)}$

2. $\sum_{i=0}^{10} \frac{1}{4^i + 5^{i+2}}$

   $\prod_{i=1}^{12} \frac{i^2}{i^2 + 2i + 3}$

3. $\sum_{i=1}^{8} \frac{x + \cos(ix)}{2^i}; x = 0,6$

   $\prod_{i=2}^{6} \frac{i+1}{i+2}$

4. $\sum_{k=1}^{6} \frac{1}{(2k+1)k}$

   $\prod_{k=1}^{5} \left(\frac{k}{k+1} - \cos^k |x| \right); x = 0,3$

5. $\sum_{i=1}^{10} \frac{i+2}{i(i+1)}$

   $\prod_{k=1}^{5} \frac{k}{2^k + 3^{k+1}}$

6. $\sum_{n=1}^{6} \frac{x^{2n+1}}{4n^2 + 1}; x = 0,15$

   $\prod_{k=1}^{4} \left(\frac{1}{(2k+1)k} + 1 \right)$

7. $\sum_{k=1}^{5} \frac{k^2}{k^2 + 3k + 5}$

   $\prod_{i=1}^{4} \frac{5^i}{3^i + 2^{i+2}}$

8. $\sum_{n=1}^{4} \frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}; x = 3$

   $\prod_{k=1}^{8} \frac{k+3}{k+5}$

9. $\sum_{k=1}^{6} \frac{k^k \cdot x^{2k}}{x^{k+1}}; x = 2$

   $\prod_{i=1}^{10} \left(2 + \frac{1}{i^2} \right)$

10. $\sum_{k=1}^{6} \frac{1}{k(k+1)}$

    $\prod_{i=2}^{8} \left(1 - \frac{1}{i^2} \right)^2$

ПРИЛОЖЕНИЕ 2

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

1. Файл должен быть подготовлен в текстовом процессоре Microsoft Word в формате docx.

2. Шрифт: Times New Roman. Размер шрифта: 14.

3. Поля: левое – 3 см; правое – 1,5 см; верхнее и нижнее – по 2 см.

4. Интервал: полуторный. Абзацный отступ первой строки: 1,25 см. Дополнительный интервал между абзацами не делается.

5. Ориентация: книжная. Выравнивание текста – по ширине.

6. Нумерация страниц начинается с Титульного листа (на титульном листе номер не ставится) и проставляется внизу страницы по центру.

7. Каждое задание контрольной работы начинается с новой страницы. Тема задания – прописными буквами (точка в конце заголовка не ставится), размер шрифта – 16. Выравнивание – по центру.

8. Все рисунки в контрольной работе должны содержать подробное описание.

9. Общий объём контрольной работы должен составлять 10–15 страниц текста.

10. Оглавление, в котором приводятся все заголовки работы с указанием номеров страниц, размещается после титульного листа. Оглавление должно быть создано с помощью соответствующей команды Microsoft Word.

11. Ссылки на используемую литературу указываются в квадратных скобках (например, [1]) – номер соответствует порядковому номеру источника из списка используемой литературы. Источники в списке используемой литературы нумеруются в порядке обращения к ним в тексте контрольной работы.

ПРИЛОЖЕНИЕ 3

ОБРАЗЕЦ ОФОРМЛЕНИЯ ТИТУЛЬНОГО ЛИСТА

Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Пермский национальный исследовательский
политехнический университет»
Кафедра горной электромеханики

Контрольная работа
по дисциплине «Информатика»
Вариант № 1

Выполнил: студент гр. ЭАГП-15-1с
Иванов С.А.

Проверила: доцент каф. ГЭМ
Щапова И.Н.

Пермь 2016

ГОСТ Р 7.05-2008

Единый формат оформления библиографических списков в соответствии с ГОСТ Р 7.05-2008 «Библиографическая ссылка. Общие требования и правила составления»

(примеры оформления ссылок и пристатейных списков литературы)

Статьи из журналов и сборников

одного автора:

Чаронов В.Я. Экономичные электроприводы для станков-качалок малодебитных скважин // Нефтяное хозяйство. – 1996. – № 12. – С. 46–48.

двух или трех авторов:

Ашрафьян М.О., Нижник А.Е. Оценка возможности вращения и расхаживания обсадных колонн при цементировании скважин // Бурение и нефть. – 2011. – №9. – С. 19–23.

Ильиных М.В., Сарин Л.И., Ширковец А.И. Опыт эксплуатации высоковольтных резисторов типа Р3 в сетях средних классов напряжения // Научные проблемы транспорта Сибири и Дальнего Востока. – 2008. – № 1. – С. 65–68.

четырех и более авторов:

Исследование отклонений свай при погружении / Е.В. Светинский, Ю.М. Приходько, Х.А. Джантимиров, С.А. Кузнецов, М.-Р. Леках // Основания, фундаменты и механика грунтов. – 2007. – №3. – С. 9–10.

Авторефераты

Глухов В.А. Исследование, разработка и построение системы электронной доставки документов в библиотеке: автореф. дис. … канд. техн. наук. – Новосибирск, 2000. – 18 с.

Диссертации

Фенухин В.И. Этнополитические конфликты в современной России: дис. … канд. полит. наук. – М., 2002. – С. 54–55.

Патенты

Пат. 200013051 Российская Федерация. Оптико-электронный аппарат / А.Н. Еськов, Б.Э. Бонштедт, С.Н. Корешев, Г.И. Лебедева, А.Г. Серегин. № 2122745/98; заявл. 18.12.00; опубл. 20.08.02, Бюл. № 23. – С. 3.

Материалы конференций

Марьинских Д.М. Разработка ландшафтного плана как необходимое условие устойчивого развития города (на примере Тюмени) // Экология ландшафта и планирование землепользования: материалы II Всерос. научн.-практ. конф. (Иркутск, 11–12 сент. 2000 г.). – Иркутск, 2000. – 200 с.

Интернет-документы

Официальный сайт компании Schlumberger [Электронный ресурс]. – URL: http://www.slb.com/~/media/Files/resources/oilfield_review/russia97/aut97/seamless.pdf (дата обращения: 27.04.2015).

Технический каталог электродвигателей ВЭМЗ (2010 г.) [Электронный ресурс]. – URL: http://www.vemp.ru/prod/motors.html (дата обращения 26.03.2014).

Учебное издание

Щапова Ирина Николаевна,
Щапов Владислав Алексеевич

ИНФОРМАТИКА

Учебное пособие

Редактор и корректор *И.А. Мангасарова*

Подписано в печать 19.02.2016. Формат 60×90/16.
Усл. печ. л. 9,75. Тираж 100 экз. Номер заказа 13/2016.

Издательство
Пермского национального исследовательского
политехнического университета.
Адрес: 614990, г. Пермь, Комсомольский пр., 29, к. 113.
Тел. (342) 219-80-33.