

## Лекция 1

### Введение

**Целью** изучения учебной дисциплины «**Основы алгоритмизации и программирования**» является изучение основных структур данных, используемых в информатике, и освоение навыков алгоритмизации задач на базе этих структур.

Для достижения поставленной цели нам необходимо решить следующие **задачи**:

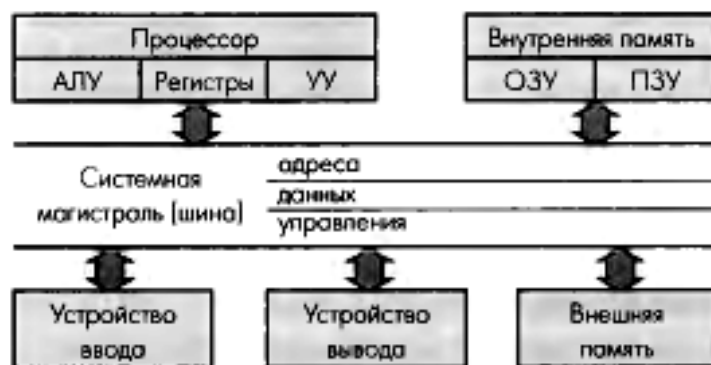
- научиться самостоятельно проектировать простейшие программы на языке высокого уровня с использованием основных структур данных;
- овладеть базовыми понятиями алгоритмизации;
- получить представление о программном обеспечении, используемом для обработки структур данных;
- приобрести системные знания в области программного обеспечения персональных компьютеров в объеме, достаточном для эффективной эксплуатации визуальной среды программирования при решении базовых функциональных задач предметной области.

### Архитектура ПЭВМ. Организация памяти.

**Аппаратное обеспечение персонального компьютера** – система взаимосвязанных технических устройств, выполняющих ввод, хранение, обработку и вывод информации.

Любой компьютер состоит из следующих устройств:

- 1) процессор;
- 2) память (внутренняя и внешняя);
- 3) устройства ввода информации;
- 4) устройства вывода информации.



**Системная шина** – подсистема, служащая для передачи данных между функциональными блоками компьютера.

Шина состоит из трех частей:

- шина адреса, на которой устанавливается адрес требуемой ячейки памяти или устройства, с которым будет происходить обмен информацией;
- шина данных, по которой передается информация;
- шина управления, регулирующая процесс передачи данных

Подключение всех внешних устройств к шине обеспечивается через соответствующие адаптеры или контроллеры.

**Магистрально-модульный принцип** (принцип открытой архитектуры) построения современных компьютеров заключается в том, что:

- все устройства взаимодействуют между собой через системную магистраль передачи данных (системную шину);
- каждое устройство конструктивно оформляется в виде отдельного блока (модуля), который легко подключается к общей схеме через один или несколько разъемов.

Модульный принцип позволяет комплектовать нужную конфигурацию компьютера и производить при необходимости модернизацию компьютера.

**Процессор** — устройство, обеспечивающее преобразование информации и управление другими устройствами компьютера. Основная работа процессора заключается в двух действиях — считывании из программы, находящейся в ОЗУ, очередной команды и выполнении действий, указанных в этой команде.

#### **Компоненты процессора:**

- АЛУ (арифметико-логическое устройство) — выполняет логические и арифметические операции над данными;
- УУ (устройство управления) — управляет работой процессора с помощью электрических сигналов;
- регистры — используются для временного хранения данных и результатов операций над ними

#### **Основные характеристики процессора:**

- разрядность — число двоичных разрядов, одновременно обрабатываемых при выполнении одной команды. Большинство современных процессоров — 64-разрядные;

• тактовая частота — это количество тактов (операций) процессора в секунду. Чем выше тактовая частота процессора, тем выше его производительность.

Тактовая частота:

- определяется временем между активными переходами сигнала с одного значения на другое;
- измеряется в герцах, определяющих число активных переходов в секунду.

**Внутренняя память** — это запоминающее устройство, напрямую связанное с процессором и предназначенное для хранения выполняемых программ и данных,

непосредственно участвующих в вычислениях. Внутренняя память, в свою очередь, делится на постоянную (ПЗУ) и оперативную (ОЗУ) память.

- **Постоянная память (ПЗУ — постоянное запоминающее устройство).** В постоянной памяти хранится информация, необходимая для первоначальной загрузки компьютера в момент включения питания.

- **Оперативная память (ОЗУ — оперативное запоминающее устройство).** Из оперативной памяти процессор берет программы и исходные данные для обработки, в нее он записывает полученные результаты. Оперативная память работает очень быстро, содержащиеся в ней данные сохраняются до тех пор, пока компьютер включен; при выключении компьютера содержимое оперативной памяти стирается.

**Внешняя память.** Для хранения программ и данных в ПК используются накопители различных типов. Накопители разделяются на накопители на магнитных лентах (стримеры), дисковые накопители, флэш-память.

К дисковым накопителям относятся:

- накопители на гибких дисках;
- накопители на жестких дисках;
- накопители на магнитооптических дисках;
- накопители на оптических дисках:
  - CD-R — компакт-диск для однократной записи информации;
  - CD-RW — компакт-диск для многократной записи информации;
  - CD-ROM — компакт-диск с записанными на них данными, доступными только для чтения (read-only memory — память «только для чтения»),
  - DVD-R — записываемый DVD. Данные на DVD-R не могут быть изменены;
  - DVD-RW — перезаписываемый DVD-диск. Может быть перезаписан не более 1000 раз;
  - DVD-RAM — перезаписываемый DVD-диск. Обладает более высокой надежностью, может быть перезаписан 100 000 раз.
  - BD (Blu-Ray Disc) — формат оптического носителя, используемый для записи с повышенной плотностью хранения цифровых данных, включая видео высокой чёткости.

флэш-память (Flash-memory) — новый тип памяти. Флэш-память представляет собой микросхему перепрограммируемого постоянного запоминающего устройства (ППЗУ) с неограниченным числом циклов перезаписи.

**Устройства ввода** позволяют вводить информацию в компьютер. Основными устройствами ввода являются клавиатура, манипулятор мышь, сенсорный экран.

**Устройства вывода** позволяют передать информацию от компьютера в доступной человеческому восприятию форме. Наиболее распространенными устройствами вывода являются монитор и принтер.

### **Дополнительные устройства:**

- устройства ввода (трекбол, джойстик, световое перо, сканер, цифровой фотоаппарат, диджитайзер (графический планшет), микрофон, тачпад, видеокамера, вебкамера);
- устройства вывода (плоттер (графопостроитель), колонки, наушники, проектор);
- устройства хранения (USB флеш-накопитель, стример (ленточный накопитель — запоминающее устройство на принципе магнитной записи на ленточном носителе, с последовательным доступом к данным, по принципу действия аналогичен бытовому магнитофону.), zip-накопитель (сменный магнитный диск большой емкости), магнитооптический накопитель и др.);
- устройство обмена (модем).

### **Представление цифровой и символьной информации в ЭВМ. Единицы информации.**

В ЭВМ применяется двоичная система счисления, т.е. все числа в компьютере представляются с помощью нулей и единиц, поэтому компьютер может обрабатывать только информацию, представленную в цифровой форме.

Для преобразования числовой, текстовой, графической, звуковой информации в цифровую необходимо применить кодирование.

**Кодирование** – это преобразование данных одного типа через данные другого типа. В ЭВМ применяется система двоичного кодирования, основанная на представлении данных последовательностью двух знаков: 1 и 0, которые называются двоичными цифрами (binary digit – сокращенно bit).

**Целые числа** кодируются двоичным кодом довольно просто (путем деления числа на два).

Для кодирования нечисловой информации используется следующий алгоритм: все возможные значения кодируемой информации нумеруются и эти номера кодируются с помощью двоичного кода.

Например, для представления **текстовой информации** используется таблица нумерации символов или таблица кодировки символов, в которой каждому символу соответствует целое число (порядковый номер). Восемь двоичных разрядов могут закодировать 256 различных символов.

Существующий стандарт ASCII (8-разрядная система кодирования) содержит две таблицы кодирования – базовую и расширенную. Первая таблица содержит 128 основных символов, в ней размещены коды символов английского алфавита, а во второй таблице кодирования содержатся 128 расширенных символов (символы национального алфавита).

Для кодирования **графических данных** применяется, например, такой метод кодирования как растр. Координаты точек и их свойства описываются с помощью целых чисел, которые кодируются с помощью двоичного кода.

Режим представления цветной графики в системе RGB с использованием 24 разрядов (по 8 разрядов для каждого из трех основных цветов) называется полноцветным. Для полноцветного режима в системе CMYK необходимо иметь 32 разряда (четыре цвета по 8 разрядов).

Компьютер, имеющий звуковую плату, микрофон и акустическую систему, позволяет кодировать (оцифровывать), сохранять и воспроизводить **звуковую информацию**. Звук — это волна с изменяющейся амплитудой и частотой в диапазоне от 20 Гц до 20 кГц. Чем больше амплитуда, тем громче звук, чем больше частота, тем выше тон.

Микрофон превращает звуковую волну в электрический сигнал, а звуковая плата кодирует его, превращая в последовательность нулей и единиц. Точность преобразования определяется разрешающей способностью преобразователя (8 бит — 256 уровней, 16 бит — 65 536 уровней, 24 бит — 16 777 216 уровней) и числом преобразований (выборки) за 1 с — частотой дискретизации.

Итак, минимальное количество информации, для кодирования которой достаточно одного двоичного разряда, называют **битом**. Он принимает только одно из двух значений — 0 или 1. Для удобства введена более крупная единица измерения информации — байт.

**Байт** — единица измерения количества информации, состоящая из восьми последовательных и взаимосвязанных битов.

$$1 \text{ байт} = 2^3 \text{ бит} = 8 \text{ бит}$$

Для обозначения большего объема информации используются другие единицы измерения:

$$1 \text{ Кбайт (килобайт)} = 1\,024 \text{ байт};$$

$$1 \text{ Мбайт (мегабайт)} = 1\,048\,576 \text{ байт};$$

$$1 \text{ Гбайт (гигабайт)} = 1\,073\,741\,824 \text{ байт};$$

$$1 \text{ Тбайт (терабайт)} = 1\,099\,511\,627\,776 \text{ байт}.$$

Значения данных единиц измерения информации для удобства кодирования связаны со степенью числа 2. В этих единицах измеряются количество (объем) оперативной или внешней памяти компьютера, размеры файлов.

Приставки *кило*, *мега*, *гига*, *тера*, которые используются для обозначения единиц измерения информации, являются общепринятыми. Однако в международной системе единиц (СИ) этим приставкам соответствуют другие числовые значения ( $10^3$ ,  $10^6$ ,  $10^9$ ,  $10^{12}$ ). Для единиц измерения информации в СИ применяются приставки *киби*, *меби*, *гиби*, *теби* и т.д.

Двоичная приставка МЭК	Множитель двоичных единиц измерения	Обозначение МЭК		Десятичная приставка СИ	Множитель десятичных единиц измерения
		биты	байты		
киби-	$2^{10} = 1024$	Кибит	КиБ	кило-	$10^3$
меби-	$2^{20} = 1\,048\,576$	Мибит	МиБ	мега-	$10^6$
гиби-	$2^{30} = 1\,073\,741\,824$	Гибит	ГиБ	гига-	$10^9$
теби-	$2^{40} = 1\,099\,511\,627\,776$	Тибит	ТиБ	тера-	$10^{12}$
пеби-	$2^{50} = 1\,125\,899\,906\,842\,624$	Пибит	ПиБ	пета-	$10^{15}$
эксби-	$2^{60} = 1\,152\,921\,504\,606\,847\,000$	Эибит	ЭиБ	экса-	$10^{18}$
зеби-	$2^{70} = 1\,180\,591\,620\,717\,411\,303\,424$	Зибит	ЗиБ	зетта-	$10^{21}$
йоби-	$2^{80} = 1\,208\,925\,819\,614\,629\,174\,706\,176$	Йибит	ЙиБ	йотта- (иотта-)	$10^{24}$

### Этапы решения вычислительных задач на ЭВМ.

На ЭВМ могут решаться задачи различного характера, например: научно-инженерные; разработки системного программного обеспечения; обучения; управления производственными процессами и т.д. В процессе подготовки и решения на ЭВМ научно -инженерных задач можно выделить следующие *этапы*:

- постановка задачи;
- математическое описание задачи;
- выбор и обоснование метода решения;
- алгоритмизация вычислительного процесса;
- составление программы;
- отладка программы;
- решение задачи на ЭВМ и анализ результатов.

В задачах другого класса некоторые этапы могут отсутствовать, например, в задачах разработки системного программного обеспечения отсутствует математическое описание. Перечисленные этапы связаны друг с другом. Например, анализ результатов может показать необходимость внесения изменений в программу; алгоритм или даже в постановку задачи.

**Постановка задачи.** На данном этапе формулируется цель решения задачи и подробно описывается ее содержание. Анализируются характер и сущность всех величин, используемых в задаче, и определяются условия, при которых она решается. Корректность постановки задачи является важным моментом, так как от нее в значительной степени зависят другие этапы.

**Математическое описание задачи.** Настоящий этап характеризуется математической формализацией задачи, при которой существующие соотношения

между величинами, определяющими результат, выражаются посредством математических формул.

**Выбор и обоснование метода решения.** Модель решения задачи с учетом ее особенностей должна быть доведена до решения при помощи конкретных методов решения. Само по себе математическое описание задачи в большинстве случаев трудно перевести на язык машины. Выбор и использование метода решения задачи позволяет привести решение задачи к конкретным машинным операциям. Одну и ту же задачу можно решить различными методами, при этом в рамках каждого метода можно составить различные алгоритмы.

**Алгоритмизация вычислительного процесса.** На данном этапе составляется алгоритм решения задачи согласно действиям, задаваемым выбранным методом решения. Процесс обработки данных разбивается на отдельные относительно самостоятельные блоки, и устанавливается последовательность выполнения блоков. Разрабатывается блок-схема алгоритма.

**Составление программы.** При составлении программы алгоритм решения задачи переводится на конкретный язык программирования. Для программирования обычно используются языки высокого уровня, поэтому составленная программа требует перевода ее на машинный язык ЭВМ. После такого перевода выполняется уже соответствующая машинная программа.

**Отладка программы.** Отладка заключается в поиске и устранении синтаксических и логических ошибок в программе.

В ходе синтаксического контроля программы транслятором выявляются конструкции и сочетания символов, недопустимые с точки зрения правил их построения или написания, принятых в данном языке. Сообщения об ошибках ЭВМ выдает программисту, при этом вид и форма выдачи подобных сообщений зависят от вида языка и версии используемого транслятора.

После устранения синтаксических ошибок проверяется логика работы программы в процессе ее выполнения с конкретными исходными данными. Для этого используются специальные методы, например, в программе выбираются контрольные точки, для которых вручную рассчитываются промежуточные результаты. Эти результаты сверяются со значениями, получаемыми ЭВМ в данных точках при выполнении отлаживаемой программы. Кроме того, для поиска ошибок могут быть использованы отладчики, выполняющие специальные действия на этапе отладки, например, удаление, замена или вставка отдельных операторов или целых фрагментов программы, вывод или изменение значений заданных переменных.

**Решение задачи на ЭВМ и анализ результатов.** После отладки программы ее можно использовать для решения прикладной задачи. При этом обычно выполняется многократное решение задачи на ЭВМ для различных наборов

исходных данных. Получаемые результаты интерпретируются и анализируются специалистом или пользователем, поставившим задачу.

### **Классификация и способы записи алгоритмов**

**Алгоритм** — понятная и конечная последовательность точных действий (команд), формальное выполнение которых позволяет получить решение поставленной задачи. Алгоритмы помогают нам каждый день решать разнообразные задачи.


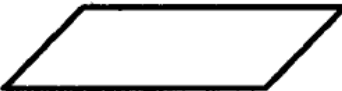
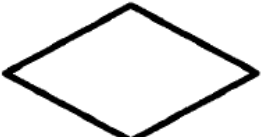

Алгоритмы можно записывать различными способами: словесно, в виде блок-схемы и с помощью программы.

**Словесное описание** алгоритма представляет собой описание структуры алгоритма на естественном языке с тщательным отбором слов и фраз, не допускающих лишних слов, двусмысленностей и повторений. Дополняется язык обычными математическими обозначениями и некоторыми специальными соглашениями. Алгоритм описывается в виде последовательности шагов. На каждом шаге определяется состав выполняемых действий и направление дальнейших вычислений. При этом, если на текущем шаге не указывается какой шаг должен выполняться следующим, то осуществляется переход к следующему шагу.

Наибольшее распространение благодаря своей наглядности получил графический (блок-схемный) способ записи алгоритмов.

**Блок-схемой** называется графическое изображение логической структуры алгоритма, в котором каждый этап процесса обработки информации представляется в виде геометрических символов (блоков), имеющих определенную конфигурацию в зависимости от характера выполняемых операций.

Основные блоки, которые используются при записи алгоритма в виде блок-схемы:

Название фигуры	Изображение	Обозначаемое действие (шаг) алгоритма
Овал		Начало или конец
Параллелограмм		Ввод или вывод
Ромб		Условие для принятия решения о выполнении действия
Прямоугольник		Выполняемое действие



Алгоритм решения задачи можно записать в виде программы на некотором языке программирования. Т.о. **программа** – это алгоритм, записанный на некотором языке программирования.

При всем многообразии алгоритмов решения задач в них можно выделить три основных вида вычислительных процессов:

- линейный;
- ветвящийся;
- циклический.

**Линейным** называется такой вычислительный процесс, при котором все этапы решения задачи выполняются в естественном порядке следования записи этих этапов.

**Ветвящимся** называется такой вычислительный процесс, в котором выбор направления обработки информации зависит от исходных или промежуточных данных (от результатов проверки выполнения какого-либо логического условия).

Циклом называется многократно повторяемый участок вычислений. Вычислительный процесс, содержащий один или несколько циклов, называется **циклическим**.

## **Системы счисления. Правила перевода чисел из одной системы счисления в другую.**

Способ записи чисел с помощью письменных знаков называют **системой счисления**.

### **Позиционные и непозиционные системы счисления.**

Системы счисления делятся на два класса позиционные и непозиционные.

**Непозиционными** называют такие системы счисления, в которых каждый знак (цифра) в записи любого числа имеет одно и то же значение и не зависит от своего расположения в числе.

**Простейшая**, но абсолютно неудобная система счисления, основана на единственной цифре – единице (палочке). Позволяет записывать только натуральные числа. Чтобы представить число в этой системе счисления нужно записать столько палочек, каково само число.

**Римская** система счисления. С помощью семи цифр – I=1, V=5, X=10, L=50, C=100, D=500, M=1000 – можно весьма успешно и довольно выразительно представлять натуральные числа в диапазоне до нескольких тысяч.

**Позиционными** называют такие системы счисления, в которых значение каждого знака (цифры) в записи любого числа зависит от расположения (позиции) этого знака в числе. Количество цифр, используемых для записи чисел в

позиционной системе счисления, называется ее **основанием**. Основанием систем счисления может быть любое число, больше единицы. К числу таких систем относится современная десятичная система счисления (с основанием  $n=10$ ). В ней для обозначения первых десяти чисел служат цифры 0,1,...,9. Комбинируя эти цифры, можно записывать любые числа.

Например, цифры числа 323 в десятичной системе счисления являются коэффициентами его записи в виде суммы степеней числа 10:

$$323 = 3 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 = 3 \cdot 100 + 2 \cdot 10 + 3 \cdot 1.$$

Из этого примера видно, что цифра 3 в зависимости от своей позиции в этом числе означает и 3 сотни, и 3 единицы, а цифра 2 означает два десятка.

### **Двоичная (бинарная) система счисления.**

Для кодирования информации в компьютере вместо привычной десятичной системы счисления используется двоичная система счисления. **Двоичная система счисления** — это позиционная система счисления с основанием 2. Для записи чисел в ней используются только две цифры: 0 и 1.

Для обозначения системы счисления, в которой представляется число, используют нижний индекс, указывающий основание системы. Например,  $11011_2$  — число в двоичной системе счисления.

Цифры в двоичном числе являются коэффициентами его представления в виде суммы степеней с основанием 2, например:

$$1011_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0.$$

В десятичной системе счисления это число будет выглядеть так:

$$1011_2 = 8 + 0 + 2 + 1 = 11_{10}.$$

Максимальное двоичное число, которое может разместиться в одном байте, равно  $11111111_2$ . Представим его в десятичной системе счисления:

$$\begin{aligned} 11111111_2 &= 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = \\ &= 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255_{10}. \end{aligned}$$

Специалисты-профессионалы, работающие с компьютером, часто используют восьмеричную и шестнадцатеричную системы счисления. Это позволяет сократить запись двоичного числа.

**Перевод чисел из одной системы счисления в другую** составляет важную часть машинной арифметики. Рассмотрим основные правила перевода.

1. Для перевода двоичного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 2, и вычислить по правилам десятичной арифметики:

$$X_2 = A_n \cdot 2^{n-1} + A_{n-1} \cdot 2^{n-2} + A_{n-2} \cdot 2^{n-3} + \dots + A_2 \cdot 2^1 + A_1 \cdot 2^0$$

**Пример.** Число  $11101000_2$  перевести в десятичную систему счисления.

$$11101000_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 232_{10}$$

2. Для перевода восьмеричного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 8, и вычислить по правилам десятичной арифметики:

$$X_8 = A_n \cdot 8^{n-1} + A_{n-1} \cdot 8^{n-2} + A_{n-2} \cdot 8^{n-3} + \dots + A_2 \cdot 8^1 + A_1 \cdot 8^0$$

**Пример.** Число  $75013_8$  перевести в десятичную систему счисления.

$$75013_8 = 7 \cdot 8^4 + 5 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 + 3 \cdot 8^0 = 31243_{10}$$

3. Для перевода шестнадцатеричного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 16, и вычислить по правилам десятичной арифметики:

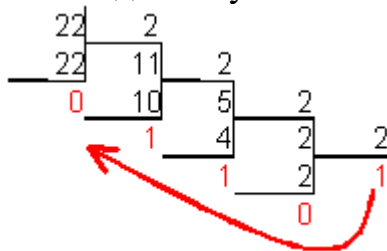
$$X_{16} = A_n \cdot 16^{n-1} + A_{n-1} \cdot 16^{n-2} + A_{n-2} \cdot 16^{n-3} + \dots + A_2 \cdot 16^1 + A_1 \cdot 16^0$$

**Пример.** Число  $FDA1_{16}$  перевести в десятичную систему счисления.

$$FDA1_{16} = 15 \cdot 16^3 + 13 \cdot 16^2 + 10 \cdot 16^1 + 1 \cdot 16^0 = 64929_{10}$$

4. Для перевода десятичного числа в двоичную систему его необходимо последовательно делить на 2 до тех пор, пока не останется остаток, меньший или равный 1. Число в двоичной системе записывается как последовательность последнего результата деления и остатков от деления в обратном порядке.

**Пример.** Число  $22_{10}$  перевести в двоичную систему счисления.



$$22_{10} = 10110_2$$

5. Для перевода десятичного числа в восьмеричную систему его необходимо последовательно делить на 8 до тех пор, пока не останется остаток, меньший или равный 7. Число в восьмеричной системе записывается как последовательность цифр последнего результата деления и остатков от деления в обратном порядке.

**Пример.** Число  $571_{10}$  перевести в восьмеричную систему счисления.

$$\begin{array}{r}
 571 \overline{) 8} \\
 \underline{-56} \phantom{0} \\
 11 \phantom{0} \overline{) 8} \\
 \underline{-8} \phantom{0} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 71 \overline{) 8} \\
 \underline{-64} \phantom{0} \\
 7 \phantom{0} \overline{) 8} \\
 \underline{-8} \phantom{0} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 8 \overline{) 8} \\
 \underline{-8} \phantom{0} \\
 0
 \end{array}$$

3      7      0      1

$$571_{10} = 1073_8$$

6. Для перевода десятичного числа в шестнадцатеричную систему его необходимо последовательно делить на 16 до тех пор, пока не останется остаток, меньший или равный 15. Число в шестнадцатеричной системе записывается как последовательность цифр последнего результата деления и остатков от деления в обратном порядке.

**Пример.** Число  $7467_{10}$  перевести в шестнадцатеричную систему счисления.

$$\begin{array}{r}
 7467 \overline{) 16} \\
 \underline{7456} \phantom{00} \\
 11 \phantom{00} \overline{) 16} \\
 \underline{16} \phantom{00} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 468 \overline{) 16} \\
 \underline{464} \phantom{00} \\
 4 \phantom{00} \overline{) 16} \\
 \underline{16} \phantom{00} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 29 \overline{) 16} \\
 \underline{16} \phantom{00} \\
 13 \phantom{00} \overline{) 16} \\
 \underline{16} \phantom{00} \\
 1
 \end{array}$$

11      4      2      13      1

$$7467_{10} = 1D2B_{16}$$

7. Чтобы перевести число из двоичной системы в восьмеричную, его нужно разбить на триады (тройки цифр), начиная с младшего разряда, в случае необходимости дополнив старшую триаду нулями, и каждую триаду заменить соответствующей восьмеричной цифрой (табл. 3).

**Пример.** Число  $1001011_2$  перевести в восьмеричную систему счисления.

$$001 \ 001 \ 011_2 = 113_8$$

8. Чтобы перевести число из двоичной системы в шестнадцатеричную, его нужно разбить на тетрады (четверки цифр), начиная с младшего разряда, в случае необходимости дополнив старшую тетраду нулями, и каждую тетраду заменить соответствующей шестнадцатеричной цифрой (табл. 3).

**Пример.** Число  $1011100011_2$  перевести в шестнадцатеричную систему счисления.

$$0010 \ 1110 \ 0011_2 = 2E3_{16}$$

9. Для перевода восьмеричного числа в двоичное необходимо каждую цифру заменить эквивалентной ей двоичной триадой.

**Пример.** Число  $531_8$  перевести в двоичную систему счисления.

$$531_8 = 101011001_2$$

10. Для перевода шестнадцатеричного числа в двоичное необходимо каждую цифру заменить эквивалентной ей двоичной тетрадой.

**Пример.** Число  $EE8_{16}$  перевести в двоичную систему счисления.

$$EE8_{16} = 111011101000_2$$

11. При переходе из восьмеричной системы счисления в шестнадцатеричную и обратно, необходим промежуточный перевод чисел в двоичную систему.

**Пример 1.** Число  $FEA_{16}$  перевести в восьмеричную систему счисления.

$$FEA_{16} = 111111101010_2$$

$$111\ 111\ 101\ 010_2 = 7752_8$$

**Пример 2.** Число  $6635_8$  перевести в шестнадцатеричную систему счисления.

$$6635_8 = 110110011101_2$$

$$1101\ 1001\ 1101_2 = D9D_{16}$$

## Логические основы ЭВМ. Основные понятия и операции формальной логики.

В основе всех действий, произведенных компьютером лежат логические выводы, основанные на математической логике. Любая интегральная схема строится с использованием триггеров и логических элементов. Логика – это совокупность правил, которым подчиняется процесс мышления.

Наука, изучающая формы рассуждений, называется **формальной логикой**.

**Математическая логика** использует математические методы для исследования способов построения рассуждений, доказательств, выводов.

Одним из разделов современной математической логики является **логика высказываний**.

На правилах математической логики построены процессы «рассуждений» компьютера. Изучение логики высказываний поможет понять, как можно научить компьютер «думать».

### Понятие высказывания

**Высказывание** — повествовательное предложение (утверждение), о котором в настоящее время можно сказать, истинно оно или ложно.

**Пример** Следующие предложения являются высказываниями:

1. Атом водорода самый легкий (истинно).
2. Клетка — часть атома (ложно).

Как правило, высказывания обозначают заглавными латинскими буквами. Если высказывание  $A$  истинно, пишут  $A = 1$ , если ложно —  $A = 0$

Часто используют такие обозначения:  $A = \text{true}$  (истина) и  $A = \text{false}$  (ложь).

### Логическая операция НЕ

С высказываниями можно производить различные операции, подобно тому как в математике — с числами (сложение, умножение, вычитание и др.).

Логическая операция **НЕ** (отрицание) меняет значение высказывания на противоположное: истинно на ложно, а ложно на истинно.

Логическое отрицание получается из высказывания путем добавления частицы «не» к сказуемому или с использованием оборота «неверно, что...»

**Пример.** Построим отрицание высказываний.

Высказывания:

1. У цветковых растений развивается плод.
2. Фреска — это живопись водяными красками по свежей штукатурке.

**Отрицание** высказываний:

1. У цветковых растений **не** развивается плод.
2. **Неверно, что** фреска — это живопись водяными красками по свежей штукатурке.

Иногда при построении отрицаний некоторые слова заменяют их антонимами, если это возможно.

Если высказывание содержит слова «все», «всякий», «любой», то отрицание такого высказывания строится с использованием слов «некоторые», «хотя бы один». И наоборот, для высказываний со словами «некоторые», «хотя бы один» отрицание будет содержать слова «все», «всякий», «любой»

**Пример.**

Высказывание: Все студенты занимаются спортом.

**Отрицание** высказывания: **Некоторые** студенты **не** занимаются спортом.

Любую операцию над числами в математике обозначают каким-либо знаком: «+», «-», «·», «:». Для логических операций тоже определены свои обозначения. Если операцию отрицания применяют к высказыванию  $A$ , то это можно записать так: **НЕ**  $A$ . Можно встретить и другие обозначения для логической операции отрицания: **Not**  $A$ ,  $\neg A$ ,  $\bar{A}$ ,  $\sim A$ .

Если нас интересует истинность высказывания **НЕ**  $A$ , то ее (вне зависимости от содержания) можно определить по таблице истинности:

$A$	<b>НЕ</b> $A$
1	0
0	1

Из таблицы истинности следует, что отрицанием истинного высказывания будет ложное, а отрицанием ложного — истинное.

**Пример.**

Число  $-7$  является положительным (ложное высказывание). Число  $-7$  не является положительным (истинное высказывание).  $A = 0$ , **НЕ**  $A = 1$ .

**Высказывание** и его отрицание никогда не могут быть истинными или ложными одновременно.

**Логика высказываний** позволяет строить **составные высказывания**. Они создаются из нескольких простых высказываний путем соединения их друг с другом с помощью логических операций **НЕ**, **И**, **ИЛИ** и др.

## Логическая операция И

Составное высказывание  $A \text{ И } B$ , образованное в результате объединения двух простых высказываний  $A$  и  $B$  логической операцией **И**, истинно тогда и только тогда, когда  $A$  и  $B$  одновременно истинны.

**Пример.** Проанализируем высказывание «Число 456 трехзначное и четное».

Данное высказывание является составным, поскольку оно содержит два простых высказывания:

«Число 456 трехзначное» (высказывание  $A$ ) и «Число 456 четное» (высказывание  $B$ ). Высказывания  $A$  и  $B$  соединены вместе логической операцией **И**, в результате получено составное высказывание  $A \text{ И } B$ . Высказывание  $A$  истинно, высказывание  $B$  истинно. Поэтому высказывание  $A \text{ И } B$  истинно:  $(A \text{ И } B) = 1$ .

Операцию **И** называют **логическим умножением**. Равенства  $1 \cdot 1 = 1$ ,  $1 \cdot 0 = 0$ ,  $0 \cdot 1 = 0$ ,  $0 \cdot 0 = 0$ , верные для обычного умножения, верны и для логического умножения.

Представим таблицу истинности для логической операции **И**:

$A$	$B$	$A \text{ И } B$
1	1	1
0	1	0
1	0	0
0	0	0

Для записи логической операции **И** используют следующие обозначения:  $A \text{ И } B$ ,  $A \text{ AND } B$ ,  $A \cdot B$ ,  $A * B$ ,  $A \wedge B$ ,  $A \& B$ .

## Логическая операция ИЛИ

Составное высказывание  $A \text{ ИЛИ } B$ , образованное в результате объединения двух простых высказываний  $A$  и  $B$  логической операцией **ИЛИ**, ложно тогда и только тогда, когда  $A$  и  $B$  одновременно ложны.

**Пример.** Проанализируем высказывание «Школьники изучают философию или высшую математику».

Данное составное высказывание образовано из двух простых высказываний: «Школьники изучают философию» (высказывание  $A$ ), «Школьники изучают высшую математику» (высказывание  $B$ ), которые связаны логической операцией **ИЛИ**. В результате получилось составное высказывание  $A \text{ ИЛИ } B$ . Высказывание  $A$  ложно, высказывание  $B$  ложно. Поэтому высказывание  $A \text{ ИЛИ } B$  ложно:  $(A \text{ ИЛИ } B) = 0$ .

Другими словами, составное высказывание  $A \text{ ИЛИ } B$  будет истинным, если истинно хотя бы одно из двух составляющих его простых высказываний.

**Пример.** Высказывание  $A$ : «Франциск Скорина — белорусский первопечатник». **Истинно**,  $A = 1$ .

Высказывание  $B$ : «Стефан Баторий — турецкий султан». **Ложно**,  $B = 0$ .

**Высказывание** «Франциск Скорина — белорусский первопечатник, **ИЛИ** Стефан Баторий — турецкий султан» будет **истинным**,  $(A \text{ ИЛИ } B) = 1$ .

Таблица истинности для логической операции **ИЛИ** имеет следующий вид:

$A$	$B$	$A \text{ ИЛИ } B$
1	1	1
0	1	1
1	0	1
0	0	0

Операцию **ИЛИ** называют **логическим сложением**. Равенства  $1 + 0 = 1$ ,  $0 + 1 = 1$ ,  $0 + 0 = 0$ , верные для обычного сложения, верны и для логического сложения.

Для записи логической операции **ИЛИ** можно использовать следующие выражения:  $A \text{ ИЛИ } B$ ,  $A \text{ OR } B$ ,  $A + B$ ,  $A \vee B$ ,  $A \mid B$ .

Если в логическом выражении присутствует несколько логических операций, то важно определить порядок их выполнения. Наивысшим приоритетом обладает операция **НЕ**. Логическая операция **И**, т.е. логическое умножение, выполняется раньше операции **ИЛИ** — логического сложения.

Для изменения порядка выполнения логических операций используют скобки: в этом случае сначала выполняются операции в скобках, а затем — все остальные.

Логические операции **И** и **ИЛИ** подчиняются переместительному закону:

$$A \text{ И } B = B \text{ И } A;$$

$$A \text{ ИЛИ } B = B \text{ ИЛИ } A.$$

### Способы представления чисел в ПЭВМ

В ЭВМ применяются две формы представления чисел:

- естественная форма, или форма с фиксированной запятой (точкой)
- нормальная форма, или форма с плавающей запятой (точкой).

**Фиксированная запятая (точка).** В форме представления с *фиксированной запятой (точкой)* числа изображаются в виде последовательности цифр с постоянным для всех чисел положением запятой, отделяющей целую часть от дробной.

Например, пусть числа представлены в десятичной системе счисления и имеют пять разрядов в целой части числа (до запятой) и пять в дробной части (после запятой). Числа, записанные в такую разрядную сетку, имеют вид:

+00721.35500

+00000.00328



-10301.20260

Эта форма наиболее проста, естественна, но имеет небольшой диапазон представления чисел и поэтому чаще всего неприемлема при вычислениях.

В компьютерах естественная форма представления используется как вспомогательная и только для целых чисел.

В памяти ЭВМ числа с фиксированной точкой хранятся в трех форматах:

а) полуслово — это обычно 16 бит, или 2 байта;

б) слово — 32 бита, или 4 байта;

в) двойное слово — 64 бита, или 8 байтов. Отрицательные числа с фиксированной точкой записываются в разрядную сетку в дополнительных кодах, которые образуются прибавлением единицы к младшему разряду обратного кода. Обратный код получается заменой единиц на нули, а нулей на единицы в прямом двоичном коде.

### Плавающая запятая (точка).

В форме представления с плавающей запятой (точкой) число изображается в виде двух групп цифр:

- мантисса;
- порядок.

При этом абсолютная величина мантиссы должна быть меньше 1, а порядок должен быть целым числом. В общем виде число в форме с плавающей запятой может быть представлено так:

$$N = \pm M \cdot P^{\pm r},$$

где  $M$  — мантисса числа ( $|M| < 1$ );  $r$  — порядок числа (целое число);  $P$  — основание системы счисления.

Например, приведенные ранее числа в нормальной форме запишутся следующим образом:

$$+0,721355 \cdot 10^3;$$

$$+0,328 \cdot 10^{-3};$$

$$-0,103012026 \cdot 10^5.$$

Нормальная форма представления обеспечивает большой диапазон отображения чисел и является основной в современных компьютерах.

Следует заметить, что все числа с плавающей запятой хранятся в машине в так называемом *нормализованном* виде.

**Нормализованная запись** отличного от нуля действительного числа — это запись вида  $N = \pm m \cdot P^{\pm r}$ , где  $r$  — целое число (положительное, отрицательное или ноль), а  $m$  — правильная  $P$ -ичная дробь, у которой первая цифра после запятой не

$$\frac{1}{P} \leq m < 1$$

равна нулю, то есть . При этом  $m$  называется **мантиссой** числа,  $r$  — **порядком** числа. У нормализованных двоичных чисел, следовательно,

$$0,5 < |m| < 1.$$

Нормализованные, т. е. приведенные к правильной дроби, числа:

$$10,35_{10} = 0,1035_{10} \cdot 10^{+2};$$

$$0,00007245_8 = 0,7245_8 \cdot 8^{-4};$$
$$F5C,9B_{16} = 0,F5C9B_{16} \cdot 16^{+3};$$

В памяти ЭВМ числа с плавающей точкой хранятся в двух форматах:

- слово — 32 бита, или 4 байта;
- двойное слово — 64 бита, или 8 байт.

Разрядная сетка для чисел с плавающей точкой имеет следующую структуру:

- нулевой разряд — это знак числа (0 — «минус», 1 — «плюс»);
- с 1 по 7 разряд записывается порядок в прямом двоичном коде, пустые разряды заполняются нулями. В первом разряде указывается знак порядка (1 — «плюс» или 0 — «минус»);
- с 8 по 31 (63) указывается мантисса, слева направо без нуля целых в прямом двоичном коде и для отрицательных чисел и пустые разряды заполняются нулями.