МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВПО МГТУ «СТАНКИН»)

Факультет **информационных технологий и систем управления**

Кафедра **информационных технологий и вычислительных систем**

Курс лекций по дисциплине «Информатика».

12.03.01 «Приборостроение»

Очная форма обучения

1 курс, 1 и 2 семестры

Разработчик: доцент кафедры ИТиВС
Носовицкий В.Б.

Раздел 1. Информатика как область интеграции знаний. Лекции 1-4.

Тема 1. Предмет и задачи информатики. Понятие информации

Понятие об информатике и информации.

Информация - это любые сведения об объектах или событиях реального мира, которые подвергаются хранению, передаче или преобразованию. Информация, которая пригодна к внесению в память вычислительной системы, называется данными.

Информатика - это наука о методах и средствах обработки информации и решении разнообразных задач при помощи компьютера.

До 80-х годов XX века информатика часто смешивалась с кибернетикой. В настоящее время существует четкое различие между этими науками. Кибернетика разрабатывает общие принципы создания систем управления и систем для автоматизации умственного труда. Основные технические средства для решения задач у кибернетики и информатики одни и те же — электронные вычислительные машины.

Количество введенной в ЭВМ информации измеряется ее «длиной», которая выражена в двоичных знаках или битах (binary - двоичный). В одном бите информации может содержаться 0 или 1. Бит является минимальной единицей информации, но он не имеет адреса. Минимальной адресуемой единицей информации является байт - последовательность из 8 стоящих рядом битов.

Объем памяти также измеряется в килобайтах (1 Кбайт = 2^{10} байт = 1024 байта), мегабайтах (1 Мбайт = 1024 Кбайт = 2^{20} байт = 1048576 байт), гигабайтах (1 Гбайт = 1024 Мбайт = 2^{30} байт = 1073741824 байт).

Эти единицы измерения количества информации используются для характеристики емкости запоминающих устройств, то есть количества информации, которое может храниться в памяти ЭВМ одновременно.

Архитектура ЭВМ. Концепция фон Неймана.

Под аппаратным обеспечением ЭВМ понимается комплекс технических средств, предназначенных для обработки информации.

В 1945 г. американский ученый Джон фон Нейман опубликовал «Предварительный доклад о машине EDVAC», в котором описывалась сама машина и ее логические свойства. Описанная Нейманом архитектура компьютера получила название архитектуры фон Неймана, и она была положена в основу всех последующих моделей

компьютеров, а сформулированные им логические свойства стали называться принципами фон Неймана.

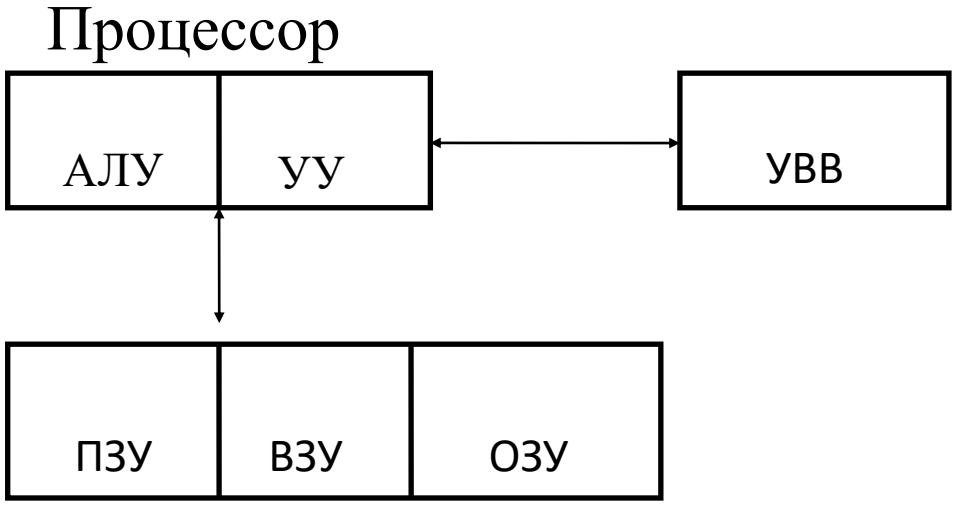
Принципы фон Неймана:

1. Любая ЭВМ должна содержать обязательный набор блоков для выполнения:

- ввода информации извне;
- ее хранения;
- ее преобразования;
- ее вывода в доступном для человека виде.

2. Машина должна работать с двоичными числами.
3. Машина должна быть электронным, а не механическим устройством.
4. Машина должна хранить в своей памяти не только данные, но и программу, отвечающую за обработку данных.

Архитектура фон Неймана изображена на рисунке:



Запоминающие устройства

Все действия над данными выполняются в арифметико-логическом устройстве (АЛУ) процессора. Устройство управления (УУ) процессора отвечает за выполнение программы и синхронизацию работы всех устройств ЭВМ.

Из всех запоминающих устройств существует одно, которое хранит информацию, необходимую процессору для работы в данный момент времени. Это устройство называется оперативной памятью или оперативным запоминающим устройством (RAM - random access memory). ОЗУ является энергозависимым устройством, то есть вся находящаяся там информация в момент выключения компьютера пропадает.

Как правило, процессор использует информацию, которая постоянно хранится в других запоминающих устройствах – внешних, так что при очистке ОЗУ исчезает только копия информации, предварительно скопированная туда из внешних запоминающих устройств. Но в тех случаях, когда информация попала в ОЗУ с устройств ввода, она может пропасть, если до выключения компьютера не сохранить ее специальным образом в каком-нибудь из ВЗУ. Одно такое устройство – жесткий диск, – физически находится внутри системного блока ЭВМ. Большая часть самой важной для нормальной работы ЭВМ программы – операционной системы, – также постоянно находится во внешнем запоминающем устройстве. Но часть операционной системы – базовая система ввода-вывода (BIOS)– находится в еще одном запоминающем устройстве – постоянном (ROM – read only memory).

В постоянном запоминающем устройстве хранится программа тестирования устройств ЭВМ и программа начальной загрузки машины. Информация закладывается в ПЗУ при изготовлении компьютера, и, как правило, не может быть изменена.

Устройства ввода-вывода часто называют периферийными устройствами ЭВМ, так как они (монитор, клавиатура, принтер и т.д.) могут находиться на довольно существенном расстоянии от самого компьютера (процессора). Более того, существуют даже специальные терминальные сети, в которые входит один мощный компьютер и большое число периферийных устройств (терминалов).

Под устройствами ввода-вывода принято понимать приспособления, способные приводить информацию, которая ранее не находилась в компьютере, к компьютерному виду, или производить обратную операцию. В этом смысле дисковод, модем (модулятор – демодулятор) и т.д. к устройствам ввода-вывода не относятся.

К устройствам ввода принято относить клавиатуру, мышь, световое перо, трекбол, джойстик, планшет, микрофон. Кроме того, существуют специальные экраны, при касании которых пальцем в компьютер вводится соответствующая информация. К устройствам вывода относятся монитор, принтер, графопостроитель, динамики.

Тема 2. Меры информации

В современной информатике различают синтаксическую, семантическую и прагматическую меры информации.

Синтаксическая мера количества информации оперирует обезличенной информацией, не выражающей смыслового отношения к объекту. Для измерения информации на синтаксическом уровне вводятся два параметра: объём информации и количество информации.

Факт получения информации всегда связан с уменьшением разнообразия или неопределенности (энтропии) системы. Исходя из этого, количество информации в сообщении определяется как мера уменьшения неопределённости состояния данной системы после получения сообщения. При этом подходе под информацией понимается количественная величина исчезнувшей в ходе какого-либо процесса неопределенности.

$$I = H_{apr} - H_{aps},$$ где

H_{apr} – априорная энтропия системы или процесса;

H_{aps} – апостериорная энтропия.

Коэффициент информативности сообщения определяется отношением количества информации к объёму данных.

Для измерения смыслового содержания информации, то есть ее количества на семантическом уровне, наиболее часто используется тезаурусный подход, когда содержащийся в принятом сообщении смысл оценивается путем соотнесения с тезаурусом получателя, его способностью понимать и усваивать поступившее сообщение.

Тезаурус – совокупность сведений, которыми располагает пользователь или система.

Количество семантической информации I_c, извлекаемой получателем из поступающих сообщений и включаемой им в дальнейшем в свой тезаурус, зависит от степени подготовленности (полноты) тезауруса S_p для восприятия такой информации.

Максимальное количество семантической информации получатель извлекает при согласовании ее смыслового содержания со своим тезаурусом ($S_p = S_{p\ opt}$). В этом случае поступающая информация понятная получателю и несет ему ранее неизвестные (отсутствующие в его тезаурусе) сведения, включаемые в дальнейшем в тезаурус.

Количество семантической информации в сообщении, то есть количество новых знаний, получаемых потребителем, является величиной относительной. Одно и то же сообщение может иметь смысловое содержание для компетентного пользователя и быть бессмысленным (семантический шум) для некомпетентного пользователя.

Относительной мерой количества семантической информации служит коэффициент содержательности, который определяется как отношение количества семантической информации к ее объему.

Прагматическая мера определяет полезность (ценность) информации для достижения пользователем поставленной цели. Эта мера также величина относительная, обусловленная особенностями применения этой информации в той или иной системе.

А.А.Харкевич предложил принять за меру ценности информации количество информации, необходимое для достижения поставленной

цели, то есть рассчитывать приращение вероятности достижения цели.

$$I = \log_2 \frac{p_1}{p_o}, \text{ где}$$

P_0 – вероятность достижения цели до получения информации;
P_1 – вероятность достижения цели после получения информации.

Тема 3. Качество информации

Под качеством информации понимается такая совокупность свойств, которая обуславливает ее способность удовлетворять определенные потребности людей. Основными потребительскими показателями качества информации являются репрезентативность, содержательность, достаточность, доступность, актуальность, своевременность, точность, достоверность и устойчивость.

Репрезентативность информации связана с правильностью ее отбора и формирования в целях адекватного отражения свойств объекта. Важнейшее значение здесь имеют правильность концепции, на базе которой сформулировано исходное понятие, и обоснованность отбора существенных признаков и связей отражаемого явления. Нарушение репрезентативности информации приводит нередко к существенным ее погрешностям.

Содержательность информации отражает семантическую емкость информации. С увеличением содержательности информации растет семантическая пропускная способность системы, так как для получения одних и тех же сведений требуется преобразовать меньший объём данных.

Достаточность (полнота) информации означает, что ее состав (набор показателей) минимален, но достаточен для принятия правильного решения. Как неполная, то есть недостаточная для принятия правильного решения, так и избыточная информация снижает эффективность принимаемых пользователем решений.

Доступность информации восприятию пользователя обеспечивается выполнением соответствующих процедур ее получения и преобразования. Информация должна быть преобразована к удобной для восприятия пользователем форме, а достигается это, в частности, путем согласования ее семантической формы с тезаурусом пользователя.

Актуальность информации определяется степенью сохранения ценности информации для управления на момент использования и зависит от динамики изменения ее характеристик, а также от интервала времени, прошедшего с момента возникновения данной информации.

Своевременность информации означает ее поступление не позже заранее назначенного момента времени, согласованного с временем решения поставленной задачи.

Точность информации определяется степенью близости получаемой информации к реальному состоянию объекта, процесса, явления и т.п.

Достоверность информации определяется ее свойством отражать реально существующие объекты с необходимой точностью. Достоверность информации измеряется вероятностью того, что отображаемое информацией значение параметра отличается от истинного значения этого параметра в пределах необходимой точности.

Устойчивость информации отражает ее способность реагировать на изменения исходных данных без нарушения необходимой точности. Устойчивость информации наравне с ее репрезентативностью обусловлена выбранной методикой ее отбора и формирования.

Тема 4. Кодирование при передаче и хранении информации.

Отображение одного набора знаков в другой производится путем кодирования. Отображаемый набор знаков называется исходным алфавитом, а набор знаков, который используется для отображения – кодовым алфавитом.

Код символа – это совокупность символов кодового алфавита, применяемых для кодирования одного символа или одной комбинации символов исходного алфавита. Взаимосвязь символов или комбинаций символов исходного алфавита с их кодовыми комбинациями составляет таблицу кодов или таблицу соответствия.

Декодирование – это процедура получения исходных символов по кодам символов. Для декодирования необходимо соблюдение условия однозначности кода – одному исходному знаку должен соответствовать только один код и наоборот.

В зависимости от поставленной цели кодирования различают кодирование по образцу, криптографическое кодирование, оптимальное кодирование и помехозащитное кодирование.

При кодировании по образцу каждому знаку дискретного сигнала соответствует знак или набор знаков того алфавита, в котором выполняется кодирование. К системам кодирования по образцу относятся коды ASCII (American Standard Code for Information Interchange) и Unicode.

Криптографическое кодирование (шифрование) используется при необходимости защитить информацию от несанкционированного доступа. Различают симметричное кодирование с закрытым ключом и асимметричное кодирование с открытым ключом. При симметричном

кодировании с закрытым ключом для кодирования и декодирования данных применяется один и тот же ключ, который должен быть доставлен стороне, осуществляющей декодирование по безопасным каналам. При шифровании с ассиметричным ключом сторона, осуществляющая декодирование, публикует открытый ключ, который применяется для кодирования сообщений, а расшифровка осуществляется при помощи другого ключа, который известен только принимающей стороне. В настоящее время для ассиметричного кодирования чаще всего используется криптосистема RSA (Ривест, Шамир, Адлеман – 1977 год).

Оптимальное кодирование служит для устранения избыточности данных путем снижения среднего числа символов кодового алфавита, предназначенных для представления одного исходного символа. Такое кодирование обычно используется в архиваторах. Оптимальное кодирование использует алгоритмы статистического кодирования и словарного кодирования. Статистическое кодирование базируется на предварительном вычислении частоты повторения одних и тех же кодов в сообщении и составлении таблицы кодирования, в которой символам с большей вероятностью ставятся в соответствие более короткие коды (коды Шеннона, коды Хаффмана – 1950 годы). Словарное кодирование основано на нахождении повторяющихся последовательностей символов в сообщении (слов) и замене каждого повторяющегося слова ссылкой на его первое вхождение. В этом случае вместе с закодированным сообщением необходимо хранить словарь (коды LZ77, LZ78, RLE).

Помехозащитное кодирование служит для передачи данных по каналам связи и учитывает возможность возникновения помех и связанного с этим искажения или утраты части данных. Избыточность при таком кодировании увеличивается, тем самым обеспечивается возможность определения факта потери или искажения информации. Возможно кодирование с фиксацией наличия на принимающей стороне самого факта ошибки (в случае обнаружения ошибки сообщение передается повторно) и кодирование с коррекцией ошибок (объём передаваемых данных увеличивается почти в 3 раза).

Тема 5. Кодирование при классификации информации

Система кодирования применяется для замены названия объекта условным обозначением (кодом) в целях обеспечения удобной и более эффективной обработки информации.

Процедура присвоения объекту кодового обозначения называется кодированием. Существует две группы методов, используемых в

системе кодирования – классификационное кодирование и регистрационное кодирование.

Классификационное кодирование применяется после проведения классификации объектов. При последовательном кодировании применяется иерархическая классификационная структура. В результате получается кодовая комбинация, каждый разряд которой содержит информацию о специфике выделенной группы на каждом уровне иерархической структуры. Параллельное кодирование используется для фасетной системы классификации. Все фасеты кодируются независимо друг от друга, для значений каждого фасета выделяется определенное количество разрядов кода.

Регистрационное кодирование используется для однозначной идентификации объектов и не требует предварительной классификации объектов. Может применяться порядковая или серийно-порядковая система. Порядковая система кодирования предполагает последовательную нумерацию объектов числами натурального ряда. Серийно-порядковая система кодирования предусматривает предварительное выделение групп объектов, которые составляют серию, затем в каждой серии производится порядковая нумерация объектов. Каждая серия тоже получает порядковую нумерацию.

Тема 6. Основы моделирования. Классификация моделей. Этапы компьютерного моделирования.

Под моделированием понимается процесс построения, изучения и применения моделей. Модель – это описание или объект-заменитель объекта-оригинала, обеспечивающмй изучение выбранных свойств оригинала в условиях, когда использование оригинала по тем или иным причинам невозможно.

Необходимость моделирования определяется тем, что многие объекты и процессы непосредственно исследовать либо практически невозможно, либо это исследование требует много врмеени и средств. Моделирование заключается в имитации изучаемого явления. Точность имитации определяется путем сравнения пролученного при воспроизведении результата с его прототипом, то есть объектом исследования, и оценки степени их сходства.

Применительно к естественным и техническим наукам различают следующие виды моделирования:

- концептульное моделирование – совокупность уже известных фактов или представлений относительно исследуемого объекта или системы истолковывается с помощью некоторых специальных знаков;

- физическое моделирование – модель и моделируемый объект представляют собой реальные объекты или процессы, причем между процессы в объекте-оригинале и модели имеют некоторое сходство;
- структурно-функциональное моделирование – моделями являются схемы, графики, табрицы, чертежи и т.п;
- математическое моделирование – моделирование, включая построение модели, осуществляется на основе различных математических методов;
- имитационное (программное) моделирование – модель исследуемого объекта представляет собой алгоритм функционирования системы, реализованный в виде программного комплеса.

По способу представления принято разделять материальные и информационные модели. При этом информационные модели могут быть неформализованными и формализованными, которые бывают компьютерными и некомпьютерными.

При компьютерном моделировании выделяют 4 этапа – постановка задачи, разработка модели, компьютерный эксперимент и анализ результатов моделирования.

На этапе постановки задачи прежде всего задача формулируется в форме словесных описаний. Формируется возможно более полное описание объекта, выделяются его элементы, устанавливаются связи между ними.

Раздел 2. Техническая база информатики. Лекции 5-9.

Темы 9-10. Представление данных в компьютере. Арифметические основы информатики.

Арифметические действия в ЭВМ выполняются в двоичной системе счисления. Эта система относится к позиционным системам счисления. Система является позиционной, если удельный вес цифры в числе зависит от позиции этой цифры. Примером непозиционной системы счисления является римская.

В связи с тем, что двоичная система неудобна для человека, компьютер выдает информацию в десятичной системе. В ряде системных программ помимо двоичной используются также восьмеричная и шестнадцатеричная системы счисления.

2	*8*	*10*	*16*
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
10000	20	16	10

Чтобы перевести число из десятичной системы счисления в любую необходимо осуществлять последовательные деления с остатком на основание системы счисления, в которую осуществляется перевод, до тех пор, пока не получится число, меньшее основания системы. Тогда ответом будет последовательность остатков, записанная в обратном порядке, начиная с последнего частного.

Перевод из двоичной системы в восьмеричную или шестнадцатеричную системы или обратно осуществляется не путем деления, а путем замены.

Чтобы перевести число из любой системы счисления в десятичную необходимо пронумеровать разряды числа справа налево начиная с нуля. Результатом будет являться сумма произведений цифры, стоящей в данном разряде, на основание системы в степени, равной номеру разряда.

Чтобы произвести сложение в любой позиционной системе счисления необходимо производить поразрядное сложение так, как это делается в десятичной системе. Если результат в столбце получился меньше основания системы, в которой производятся действия, то он остается без изменения. В противном случае от этого результата отнимается основание системы, а в соседний слева разряд добавляется единица.

Вычитание в машине заменяется сложением с числом, обратным по знаку вычитаемому. Таким образом, приходится иметь дело с отрицательными числами. Эти числа хранятся в памяти ЭВМ в дополнительном коде.

При хранении чисел со знаком под знак выделяется самый левый разряд числа (знаковый разряд). Для положительных чисел там хранится нуль, а для отрицательных - 1.

Чтобы представить число в дополнительном коде необходимо:

- записать соответствующее положительное число в прямом коде;
- заменить все единицы на нули, а нули на единицы (обратный код);
- по правилу сложения прибавить к обратному коду единицу.

Тема 11. Логические основы информатики.

В 1854 английский математик Джордж Буль опубликовал работу «Исследование законов мышления, базирующихся на математической логике и теории вероятностей». Эта работа дала рождение алгебре логики, или булевой алгебре. Буль первым показал, что существует аналогия между алгебраическими и логическими действиями, так как и те, и другие предполагают лишь два варианта ответов — истина или ложь, нуль или единица. Он придумал систему обозначений и правил, пользуясь которыми можно было закодировать любые высказывания, а затем манипулировать ими как обычными числами. Булева алгебра располагала тремя основными операциями — И, ИЛИ, НЕ, которые позволяли производить сложение, вычитание, умножение, деление и сравнение символов и чисел. Впоследствии идеи Буля нашли свое воплощение в вычислительной технике.

Особенностью функций, которые используются в булевой алгебре, является то, что все их аргументы и их значения принадлежат множеству «0,1». Булева функция может быть задана таблицей или формулой. Строка из области определения булевой функции называется набором. Задать булеву функцию - это значит сопоставить каждому набору из области определения значение 0 или 1. Построенная таблица называется таблицей истинности булевой функции.

Всякая булева функция от n переменных имеет 2^n наборов, а всего существует 2^{2^n} функций от n переменных. Часть булевых функций от одной и от двух переменных носит название элементарных. К ним, в частности, относятся функции отрицания, конъюнкции, дизъюнкции, эквивалентности, импликации и сложения по модулю два.

Функция конъюнкции (логическое «и», логическое умножение, $\wedge$) принимает значение 1 только тогда, когда оба ее аргумента истинны.

Функция дизъюнкции (логическое «или», логическое сложение, $\vee$) принимает значение 0 только тогда, когда оба ее аргумента ложны.

Функция импликации ($\Rightarrow$) принимает значение 0 только тогда, когда из верной посылки следует неверный результат.

Функция эквивалентности ($\Leftrightarrow$) принимает значение 1 только тогда, когда ее аргументы принимают одинаковые значения.

Функция сложения по модулю два ($\oplus$) является обратной к функции эквивалентности.

Две функции называются равными, если их таблицы истинности совпадают.

Базисом называется такое подмножество множества всех булевых функций, что через функции, входящие в это подмножество, можно выразить все остальные булевы функции. Одним из наиболее важных базисов является базис «И, ИЛИ, НЕ». И именно этот базис реализован в вычислительной технике. Одним из способов приведения остальных булевых функций к этому базису является построение совершенной дизъюнктивной нормальной формы, представляющей собой дизъюнкцию конъюнктов (конъюнкт в СДНФ - это логическое произведение всех переменных из области определения, причем каждая переменная может входить в это произведение либо с отрицанием, либо без него), по таблице истинности. Построенная СДНФ, по возможности, должна быть минимизирована.

Для построения СДНФ по таблице истинности необходимо:

– выделить наборы, на которых функция принимает значение 1;

– записать эти наборы в виде конъюнктов;

– поставить между конъюнктами знаки дизъюнкции.

Для хранения и преобразования одного бита информации используются два вида электронных схем, которые называются цифровыми автоматами. К первому виду относятся цифровые автоматы с памятью, используемые, в частности, для приема, хранения и выдачи другим логическим элементам одного бита информации. Одной из основных разновидностей цифровых автоматов с памятью является триггер.

Для выполнения двоичных арифметических и логических операций служат цифровые автоматы другого вида - комбинационные схемы или логические элементы без памяти. Несколько таких элементов объединяют в микросхемы. В частности, для выполнения сложения из логических элементов образуют электронные схемы, называемые сумматорами и полусумматорами.

Тема 12. История развития вычислительной техники. Поколения компьютеров.

С древнейших времен человек конструирует себе в помощь различные приспособления для облегчения вычислений. Еще в V веке до нашей эры греки и египтяне использовали абак - доску, разделенную на полосы, где передвигались камешки или кости.

В 40-х годах XVII века один из крупнейших ученых в истории человечества - математик, физик, философ и богослов Блез Паскаль изобрел и изготовил механическое устройство, позволяющее складывать числа. Механическое устройство, позволяющее не только складывать числа, но и умножать их, было изобретено другим великим математиком и философом - Готфридом Вильгельмом Лейбницем в конце XVII века.

Настоящим переворотом в счетных машинах явилось создание в 1874 г. арифмометра В. Т. Однера. Введенное изобретателем колесо с переменным числом зубьев позволило механически переносить десяток в вышестоящий разряд.

Наряду с устройствами, предназначенными для вычислений, развивались и механизмы для автоматической работы по заданной программе (музыкальные автоматы, шарманки, часы с боем и т.п.). В шарманку, например, помещали диски с по-разному расположенными штырьками - в зависимости от расположения штырьков звучала та или иная мелодия. В ткацком станке Жаккарда (начало XIX века) узор ткани задавался с помощью дырочек в тонких картонных картах (перфокартах). Для смены узора достаточно было по-другому пробить дырочки в перфокарте.

В 1822 английский математик Чарльз Бэббидж описал машину, способную рассчитывать и печатать большие математические таблицы, и сконструировал машину для табулирования, состоявшую из валиков и шестеренок, вращаемых с помощью рычага. Машина могла производить некоторые математические вычисления с точностью до восьмого знака после запятой. Это был прообраз его разностной машины, к постройке которой он приступил в 1823 году, получив правительственную субсидию для продолжения работ. Разностная машина должна была производить вычисления с точностью до 20 знака после запятой. Постройка машины отняла у Бэббиджа 10 лет, ее конструкция становилась все более сложной, громоздкой и дорогой. Она так и не была закончена, финансирование проекта было прекращено.

Тем временем Бэббиджем овладела идея создания нового прибора — аналитической машины. Главное ее отличие от разностной машины заключалось в том, что она была программируемой и могла выполнять любые заданные ей вычисления. По существу, аналитическая машина стала прообразом современных компьютеров, так как включала их основные элементы: память, ячейки которой содержали бы числа, и арифметическое устройство, состоящее из рычагов и шестеренок. Бэббидж предусмотрел возможность вводить в машину инструкции при помощи перфокарт. Однако и эта машина не была закончена, поскольку низкий уровень технологий того времени стал главным препятствием на пути ее создания.

Разностная машина в несколько видоизмененном виде была построена в 1854 году шведским изобретателем Шойцем. В 1991 британскими учеными по спецификации Бэббиджа была построена вторая разностная машина, способная производить вычисления с точностью до 31 знака после запятой.

С машиной Бэббиджа связано появление профессии программиста. Первым программистом принято считать Аду Лавлейс, которая создавала программы для работы на этой машине. Несмотря на то, что никто не имел возможности проверить правильность этих программ, впоследствии в честь этого события был назван язык программирования Ада.

В 1888 американский инженер Герман Холлерит сконструировал электромеханическую машину, которая могла считывать и сортировать статистические записи, закодированные на перфокартах. Эта машина, названная табулятором, состояла из реле, счетчиков, сортировочного ящика. Данные на каждого человека наносились на перфокарты, почти не отличающиеся от современных, в виде пробивок. При прохождении перфокарты через машину, данные, отмеченные дырочками, снимались путем прощупывания системой игл. Если напротив иглы оказывалось

отверстие, то игла, пройдя сквозь него, касалась металлической поверхности, расположенной под картой. Возникавший таким образом контакт замыкал электрическую цепь, благодаря чему к результатам расчетов автоматически добавлялась единица, после чего перфокарта попадала в определенное отделение сортировочного ящика.

В 1890 изобретение Холлерита было впервые использовано для 11-й американской переписи населения. Успех вычислительных машин с перфокартами был феноменален. То, чем десятилетием ранее 500 сотрудников занимались в течение семи лет, Холлерит сумел выполнить с 43 помощниками на 43 вычислительных машинах за 4 недели.

Это изобретение имело успех не только в США, но и в Европе, где стало широко применяться для статистических исследований. Несколько таких машин закупила Россия. Холлерит был удостоен нескольких премий и получил звание профессора Колумбийского университета. В 1896 он организовал в Нью-Йорке компанию по производству машин для табуляции (Tabulating Machine Company), которая впоследствии выросла в International Business Machines Corporation — IBM.

В конце 30-х годов XX века американские ученые Дж. Атанасов и К. Берри построили фактически первую электронную счетную машину. Аппарат содержал около 300 вакуумных трубок, с помощью которых производились вычисления, использовал двоичный код, мог осуществлять логические операции. Для ввода и вывода данных применялись перфокарты. Аппарат Атанасова мог достигать точности вычислений, в тысячу раз превышающей точность дифференциального анализатора Х. Буша, считавшегося в ту пору самым передовым вычисляющим прибором.

Машина Атанасова оказала огромное влияние на развитие компьютерных технологий. Это был первый компьютер, в котором для операций с двоичными числами были применены электронные устройства (вакуумные трубки). Некоторые идеи Атанасова до сих пор остаются актуальными, например, использование конденсаторов в запоминающих устройствах с произвольной выборкой, в том числе в оперативной памяти, регенерация конденсаторов, разделение памяти и процесса вычислений.

Тем не менее, машина Атанасова и Берри не была универсальной в полном смысле этого слова. В результате работ американского инженера Джона Мокли была предложена конструкция универсального цифрового компьютера, который мог оперировать закодированными данными. С использованием разработок Дж. Атанасова, к 1946 было завершено создание модели «Эниак» (ENIAC, Electronic Numerical Integrator and Computer), огромной машины,

которая состояла из более 18 тысяч электронных ламп. Вес машины составлял 30 тонн, она требовала для размещения 170 $м^2$. Машина оперировала двоичными числами и могла производить 5 тыс. операций сложения или 300 операций умножения в секунду. Впервые эта машина была применена при баллистических военных исследованиях на Абердинском испытательном полигоне в 1947.

Изобретение не принесло Атанасову никаких дивидендов. Патент на изобретение получили создатели «Эниака», которым Атанасов демонстрировал свою машину. Его же вклад в изобретение был признан лишь в результате судебного разбирательства. Было доказано, что практически все основные узлы «Эниака» заимствованы из машины Атанасова и Берри, и той информации, которую Атанасов передал Джону Мокли в начале 1940-х. В 1973 году патент на «Эниак» был признан недействительным по решению Федерального суда.

Таким образом, аппаратной базой для первых электронных вычислительных машин явились вакуумные лампы. Такие машины принято называть машинами первого поколения. В нашей стране представителями первого поколения ЭВМ стали машины МЭСМ и БЭСМ, созданные соответственно в 1951 и 1953 годах под руководством академика С. А. Лебедева. Впоследствии были созданы полупроводниковые варианты этих машин (второе поколение ЭВМ). Представитель этого поколения - машина БЭСМ-6 долгое время считалась одной из лучших в мире. Быстродействие этих машин достигало 1 млн. операций в секунду.

А первым в мире компьютером, созданным на основе транзисторов, стал TX-0, разработанный в 1953 году в Массачусетском технологическом университете. На этой аппаратной базе были созданы и первые в мире мини-компьютеры фирмы Digital Equipment Corporation PDP-1 (1961 год).

Аппаратной базой ЭВМ третьего поколения стали большие интегральные схемы. В 1968 году американская фирма Burroughs выпустила на рынок первые ЭВМ на БИС B2500 и B3500. Ну а наиболее известными машинами этого поколения стали IBM 360/370, разработанные в конце 60-х годов XX века. Отечественными аналогами этих компьютеров стали машины единой серии ЕС, которые выпускались в рамках социалистической интеграции странами СЭВ.

Настоящая революция в аппаратном обеспечении ЭВМ произошла с появлением СБИС - сверх больших интегральных схем. Именно они стали аппаратной базой для ЭВМ четвертого поколения и основой для создания персональных ЭВМ. Первый же четырехразрядный микропроцессор Intel 4004 был создан в 1970 году. Осознав перспективность своей разработки, Intel выкупила права на изобретение у компании-заказчика и занялась повышением

производительности микропроцессора. Уже в 1973 году выпускается восьмиразрядный процессор Intel 8008, а в 1975 году создается первый персональный компьютер Altair-8800.

В 1976 был представлен персональный компьютер Apple 1, а 1977 году Apple II с цветным монитором, звуком и графикой. Фирма IBM вышла на рынок персональной техники только в 1981 году, представив IBM PC с 16 килобайтами оперативной памяти и процессором Intel 8088. Существенным преимуществом разработок IBM стала открытая архитектура, что впоследствии сыграло против фирмы. Специально для этого компьютера была написана операционная система MS DOS. В 1983 году выходит новая модель IBM PC XT со встроенным жестким диском, памятью 128 килобайт и новой версией DOS, а в 1984 году фирма Apple выпускает свой Macintosh с памятью 128 килобайт на базе процессора Motorola 68000. В этом же году выходит IBM PC AT на базе 16-разрядного процессора Intel 80286.

Тема 13. Классификация компьютеров и вычислительных систем.

По принципу действия:

- аналоговые (вычислительные устройства непрерывного действия, обрабатывающие аналоговые данные без преобразования их в цифровую форму);
- цифровые;
- гибридные.

По вычислительной мощности и габаритам:

- большие ЭВМ (построены с использованием многоядерной, многопроцессорной или многокомпьютерной технологии). Их назначение – мейнфреймы, крупные интернет-серверы, серверы данных в больших распределенных системах;
- средние ЭВМ (серверы баз данных в информационных системах среднего масштаба, управляющие компьютеры в системах автоматизированного производства);
- малые ЭВМ;
- микроЭВМ.

По способу применения:

- суперкомпьютеры (научные вычислительные центры большой мощности, системы противоракетной обороны, управление распределенными космическими спутниками);
- серверы (файловые, интернет-служб, приложений, серверы баз данных, мэйнфреймы);

- персональные (рабочие станции, бытовые, игровые приставки, персональные информационные менеджеры, наладонные и коммуникаторы);
- специализированные (бортовые, управляющие, микроконтроллеры, встроенные, специализированные рабочие станции).

Мэйнфрейм – это мощный компьютер в сетевой инфраструктуре, в которой на стороне пользователя находится терминал.

Производительность мейнфреймов, как правило, вычисляется в миллионах операций в секунду (MIPS),

В контексте общей вычислительной мощности мейнфреймы проигрывают суперкомпьютерам.

Любой современный компьютер создан на основе СБИС и в качестве исполнителя программы и вычислителя имеет один или несколько микропроцессоров.

Самые производительные на сегодняшний день процессоры обеспечивают вычислительную мощность примерно в 4000 раз меньше самого производительного суперкомпьютера. Механизм достижения такой скорости обработки информации – параллельные вычисления. Каждая параллельная задача может выполняться одним микропроцессором.

Все современные суперкомпьютеры по классификации Флинна относятся к категории MIMD (множество потоков команд с множеством потоков данных – Multiple Instruction, Multiple Data). Кроме этой технологии выделяют технологии MISD, SIMD (обработка массивов) и SISD.

Процессоры, входящие в состав суперкомпьютера, могут взаимодействовать с памятью разными способами.

SMP (Symmetric MultiProcessing) – симметричное мультиплицирование. Все микропроцессоры, входящие в состав суперкомпьютера, подключены к одному адресному пространству, к одной памяти при помощи специальной шины памяти. Эта архитектура дает наибольший выигрыш по производительности, но не может содержать большого количества процессоров.

MPP (Massive Parallel Processing) – обработка с массовым параллелизмом. В этом случае общей памяти нет. Каждый процессор или модуль с несколькими процессорами является владельцем своего банка памяти, а между процессорами устанавливаются соединения, которые могут выполняться не только при помощи обычных сетевых устройств, но и посредством специальных вспомогательных компьютеров, предназначенных для высокоскоростной передачи данных (транспьютеров).

NUMA (Nonuniform Memory Access) – неоднородный доступ к памяти. При такой архитектуре несколько процессоров объединяются

между собой в SMP-узел, а SMP-узлы, в свою очередь, образуют MPP-архитектуру.

Кроме того, суперкомпьютеры принято объединять в кластеры – узлы, объединенные при помощи сети таким образом, что для конечного пользователя они предстают одним устройством. Физически кластеры могут быть организованы в одной локальной сети (гомогенная организация) или объединяться через разного рода сетевые соединения, включая Интернет (так называемая гетерогенная структура).

Производительность суперкомпьютеров измеряется в операциях с плавающей запятой (точкой) в секунду (FLOPS – Float Point Operation Per Second). В настоящее время преодолен рубеж в триллион FLOPS.

Сервер – это компьютер или система компьютеров, связанная через сетевые соединения с другими компьютерами и обрабатывающая их запросы.

Современные специализированные серверы изначально конструируются с учетом особенностей их работы. Такими особенностями являются:

- размещение (специальные помещения с вентиляцией, резервной системой питания, сигнализацией и т.д.);
- резервирование (возможность функционирования при выходе из строя части комплектующих, наличие резервных компьютеров, кластерная архитектура);
- защита от вирусов (средства антивирусной защиты могут встраиваться непосредственно в серверный процессор);
- резервное копирование и хранение данных (в системах непрерывного цикла резервное копирование может производиться во время работы);
- удаленное управление (используется отдельный канал связи с сервером);
- масштабирование (сервер должен иметь возможность увеличивать свою мощность путем наращивания вычислительных блоков).

Серверы реализуются в трёх исполнениях – напольное, стоечное, блейд-серверное. В блэйд-сервере отсутствуют или вынесены наружу некоторые типичные компоненты, традиционно присутствующие в компьютере. Функции питания, охлаждения, сетевого подключения, подключения жёстких дисков, межсерверных соединений и управления могут быть возложены на внешние агрегаты.

Персональные компьютеры в соответствии с PC 99 System Design Guide подразделяются на:

- Consumer PC – массовый персональный компьютер домашнего применения, должен обеспечивать возможность выхода в интернет, но не возможность работы в локальной сети;
- Office PC – массовый персональный компьютер офисного применения, должен обеспечивать работу в локальной сети, а также возможность удаленного управления и обслуживания;
- Mobile PC – переносной персональный компьютер;
- Workstation PC – мощный компьютер с возможностью работы в локальной сети и в Интернете;
- Entertainment PC – развлекательный компьютер с повышенным качеством воспроизведения мультимедийного содержимого.

По габаритам и весу персональные компьютеры можно разделить на классы:

- стационарный компьютер (по типу корпуса делятся на Tower, minitower, Desktop);
- переносной компьютер;
- ноутбук;
- субноутбук (NetBook);
- карманный персональный компьютер («наладонник») – как правило выпускаются без жестких дисков;
- коммуникатор – объединение КПК, сети и телефонной связи;
- планшетный компьютер;
- электронная книга.

Тема 14. Устройство персонального компьютера. Материнская плата и центральный процессор.

Реальная конфигурация компьютера не совпадает с концепцией фон Неймана - в минимальном наборе эта конфигурация включает системный блок, монитор и клавиатуру.

Системный блок содержит блок питания, материнскую (системную) плату, в которую вставлены другие платы - контроллеры, устройства внешней памяти - накопители на жестких и гибких магнитных дисках. Устройства ввода-вывода подключаются к портам ввода-вывода, которые могут находиться как на материнской плате, так и на контроллерах.

На материнской плате монтируются:

– центральный процессор с охлаждающим вентилятором;
– постоянное запоминающее устройство, куда зашита BIOS;
– оперативное запоминающее устройство (RAM);
– кэш-память второго уровня;
– энергонезависимая память на основе CMOS;

– системная шина;
– генератор тактовой частоты;
– разъемы расширения системы (слоты);
– порты.

За работу материнской платы отвечает набор микросхем (chipset). Обычно чипсет содержит в себе контроллеры прямого доступа к памяти, контроллеры прерываний, таймеры, систему управления памятью и шиной. До последнего времени чипсеты состояли из двух микросхем, которые носили название северного и южного мостов (North Bridge и South Bridge). Сейчас более распространено построение чипсетов по так называемой интеловской хабовой архитектуре. Тут северный мост переименован в хаб управления графикой, а южный – в хаб управления вводом-выводом. Естественно, эти технологии отличаются не только названиями микросхем, а и способом их соединения.

Существуют четыре основных типоразмера (форм-фактора) материнских плат: AT, ATX, NLX, LPX. Форм-фактор – это стандарт, определяющий размеры материнской платы, места ее крепления к корпусу; расположение на ней интерфейсов шин, портов ввода/вывода, процессорного гнезда и слотов для оперативной памяти, а так же тип разъема для подключения блока питания. Имеет значение также тип корпуса, в котором будет размещаться материнская плата.

В состав центрального процессора, как правило, входят:
– блок интерфейса с магистралью - отвечает за связь процессора с другими блоками;
– блок предварительной выборки команд - для подготовки очереди следующих команд;
– блок декодирования команд - для преобразования команд в специальный микрокод;
– исполнительный блок - выполняет команды, используя регистры;
– блок управления памятью - для сегментации оперативной памяти.

Архитектура набора команд процессора выступает в качестве посредника между аппаратурой и программным обеспечением и представляет собой ту часть аппаратной системы, которую может использовать программист-разработчик.

Существуют два типа архитектуры системы команд – RISC (Reduced Instruction Set Computer) и CISC (Complex Instruction Set Computer). В каждом из этих типов реализована конвейерная обработка, то есть каждая машинная команда разделяется на ступени, а на разных ступенях одного конвейера в определенный момент выполняется несколько команд.

Генератор тактовой частоты генерирует импульсы, которые, по сути, служат метками времени для всех устройств персонального компьютера. Благодаря этим меткам устройства разделяют время

своей работы на короткие фрагменты, которые синхронизированы по тактовым импульсам. Частота тактовых импульсов определяет быстродействие ПК - чем она выше, тем чаще может обрабатываться или передаваться единица информации.

Современные модели процессоров имеют тактовые частоты свыше 500 МГЦ, причем внутренняя частота процессора больше, чем частота, с которой работают другие части машины.

К сервисным устройствам относятся:

- контроллер периферийного устройства;
- шинный формирователь/контроллер;
- контроллер прерываний;
- программируемый внутренний таймер;
- контроллер прямого доступа в память;
- контроллер и банк буферного запоминающего устройства (кэш-памяти).

Контроллеры предназначены для управления подключенным к ним устройствам и обеспечения связи с центральной платой.

Системная шина - это основная интерфейсная система компьютера, обеспечивающая сопряжение и связь всех его устройств между собой. В настоящее время наиболее распространенной является шина PCI (Peripheral Component Interconnect bus).

Гнезда для установки процессора, имеющиеся на материнской плате, бывают двух типов – слоты (процессор расположен под прямым углом к материнской плате) и сокеты (процессор после установки находится параллельно материнской плате).

Слоты предназначены для подключения электронных устройств непосредственно к шине компьютера. Они расширяют возможности системы в целом, а потому и носят название разъемов расширения. Платы, которые вставляются в эти разъемы, называют дочерними, а устройства, выполненные на дочерних платах, носят название адаптеров. К таким устройствам, в частности, относится сетевой адаптер.

Порты предназначены для ввода информации в последовательном или параллельном режиме. При последовательном способе ввода биты поступают по очереди по одному и тому же кабелю (COM-порты). В параллельный порт биты поступают каждый по своему каналу (LPT-порты). Такая передача данных, безусловно, быстрее последовательной, но параллельные кабели стоят дороже, а синхронизация информации при длинных кабелях является довольно сложной технической задачей. Как правило, к параллельным портам подключаются печатающие устройства, а к последовательному – манипулятор «мышь».

Тема 15. Запоминающие устройства.

ПЗУ (ROM) - энергонезависимая память, хранящая, в частности, BIOS - базовую систему ввода-вывода. На сегодняшний день большинство микросхем памяти для BIOS построены на технологии Flash ROM или EEPROM с электрическим стиранием и перезаписью, в результате чего эту память можно неоднократно стирать и перезаписывать более свежую версию BIOS средствами самой материнской платы.

ОЗУ (RAM) - энергозависимая память, используемая для хранения данных и программ, обрабатываемых в данный момент времени. На компьютерах, работавших под управлением MS-DOS, оперативная память разделялась на базовую память, используемую DOS (первые 640 Кбайт), верхнюю или скрытую память (от 640 Кбайт до 1 Мбайт), дополнительную память (ее использовала, в частности, Windows 3.1) и расширенную память. В настоящее время это деление зачастую является только логическим. Физически же принято делить память на динамическую и статическую, причем динамическую память обычно применяют в качестве ОЗУ общего назначения, а статическую – в качестве высокоскоростного процессорного кэша (буфера между процессором и собственно оперативной памятью). Отличие динамической памяти от статической состоит в том, что статическая память не нуждается в регенерации, но имеет существенно более высокую стоимость. Регенерация, которая должна происходить через промежуток времени порядка нескольких миллисекунд, существенно замедляет работу устройства.

Кэш-память второго уровня – это высокоскоростная память, в которую копируются данные из оперативной памяти с целью оптимизации работы процессора с большими объемами данных.

Энергонезависимая память на основе CMOS (Complementary Metal-Oxide-Semicondactor) содержит основные параметры настройки материнской платы, в частности тип и число накопителей на жестких дисках, информацию о дисководах, пароль на загрузку компьютера и т.д.

Устройства для долговременного хранения больших объемов информации называются внешними запоминающими устройствами или накопителями. Конструктивно накопители могут выполняться как внешними, так и внутренними.

Ранее основным носителем информации в ЭВМ являлась магнитная лента. Информация размешалась на катушечной магнитной ленте или на специальных кассетах (стримерах). В настоящее же время в подавляющем большинстве случаев используется память на различных типах дисков или же флэш-память.

При запоминании информации на магнитных носителях происходит изменение намагниченности носителя на участке под магнитной

головкой. Каждый из дисков разбит на концентрические дорожки (треки), которые в свою очередь делятся на сектора. Кластер – это несколько секторов, рассматриваемых как единое целое с точки зрения операций записи/считывания. Для хранения информации о местоположении файлов на диске используется таблица размещения файлов (file allocation table). Эта таблица представляет собой массив элементов, каждый из которых соответствует одному кластеру. При создании нового файла под него выделяется необходимое количество кластеров, номер первого кластера заносится в папку с файлом, соответствующий этому кластеру элемент таблицы содержит адрес следующего кластера файла и т.д.

Чем больше кластеров, тем меньше их размер, а значит и потери на слэк (части кластера, которые не заняты информацией). В системе FAT32 максимальное число кластеров – 4294967296. Кроме системы FAT32 активно используется система NTFS (New Technology File System).

Компакт-диски (CD-ROM) позволяют не только хранить данные и программы, но и воспроизводить звуки (музыку). Закодированная информация наносится на алюминиевый диск лазерным лучом, который создает на его поверхности микроскопические впадины, разделяемые плоскими участками. Поэтому компакт-диски часто называют лазерными. Считывание также происходит при помощи лазерного луча, но меньшей мощности. Скорость доступа к данным на CD-ROM значительно выше, чем на гибких дисках, а их средняя емкость 650-700 МБайт. Для повышения надежности компакт-дисков в данные добавляется служебная информация, в частности контрольные коды Рида-Соломона, служащие для исправления ошибок.

В домашних условиях можно самостоятельно записывать с помощью специального привода компакт-диски CD-R (Compact Disc Recordable). Такие диски дороже обычных, так как в качестве светоотражательного слоя вместо алюминия используется золото. Перезаписывать эти диски нельзя. Для перезаписи используются CD-RW (Compact Disk Rewritable). У таких дисков пленка, составляющая активный слой, выдерживает большое количество смен фазовых состояний. В настоящее время большое распространение имеют цифровые видеодиски – DVD (Digital Versatile Disc). По сравнению с CD-ROM DVD имеет более высокую емкость, а также более продвинутые способы коррекции от ошибок и новый формат секторов.

Флеш-память — разновидность полупроводниковой технологии электрически перепрограммируемой памяти. Эта память в современном виде создана в 1984 году корпорацией Toshiba. Принцип работы полупроводниковой технологии флеш-памяти основан на изменении и регистрации электрического заряда в изолированной

области (кармане) полупроводниковой структуры. С электрической стороны флэш-память выдерживает примерно миллион циклов перезаписи.

Тема 16. Периферийные устройства.

Печатающие устройства.

Принтеры, используемые совместно с персональными компьютерами можно разделить по следующим признакам: на ударные и безударные; на знакопечатающие и знакосинтезирующие; на посимвольные, построчные и постраничные.

Наиболее распространенными в настоящее время являются матричные, струйные, лазерные и термические принтеры.

Матричные принтеры являются ударными, посимвольными и знакосинтезирующими. Каждый символ, выводимый на печать представляется матрицей точек (например, 7x9 точек). Печатающая головка состоит из одного или двух рядов специальных иголок, каждая из которых может независимо от других выдвигаться и ударять по красящей ленте. Печатающая головка движется горизонтально. Фактически каждый символ печатается за несколько ударов, а не за один.

В лазерном печатающем устройстве луч лазера под управлением ЭВМ вырисовывает текст на электрически заряженном вращающемся барабане или же на специальной ленте. Частички красителя прилипают к заряженным участкам, затем все изображение переносится на бумагу, которая подогревается, в результате чего красящие частички спекаются вместе. Этот принтер является построчным, безударным и знакопечатающим.

Термический принтер относится к безударным посимвольным знакосинтезирующим печатающим устройствам. Принцип действия аналогичен тому, который используется в матричных принтерах, только проволочки разогреты и в тем местах, где они практически касаются бумаги, эта бумага меняет цвет. В термических принтерах должна использоваться специальная бумага.

К безударным посимвольным знакосинтезирующим печатающим устройствам относятся также и струйные принтеры, причем они могут основаны на использовании двух разных технологий. В пусковых капельных системах капельки чернил под давлением выбрызгиваются из специальных форсунок, которые открываются по специальным электрическим импульсам. Головка передвигается горизонтально, формируя изображение символ за символом, причем принцип действия также напоминает матричный принтер. В системах с непрерывным потоком чернила выдуваются из отверстия непрерывно, причем поток чернил аналогичен потоку электронов из

электронно-лучевой трубки. В те моменты, когда вырисовывать символ не нужно, чернильная струя не долетает до бумаги, а направляется в резервуар, из которого повторно попадает на разбрызгивание.

В настоящее время, когда любое печатающее устройство способно выводить как тексты, так и различные графические изображения, деление на принтеры и графопостроители является довольно условным. Тем не менее, принято считать, что к графопостроителям (плоттерам) относятся устройства, использующие формат бумаги А2, А1 или А0.

По способу перемещения носителя (бумаги или другого материала) графопостроители подразделяют на планшетные (бумага неподвижна, пишущий узел движется в двух направлениях), рулонные (пишущий узел и бумага движутся по взаимно перпендикулярным направлениям, причем бумага может продвигаться в обе стороны) и растровые (изображение выводится по строкам).

Изображение может наноситься электростатическим способом, а может прорисовываться печатающим узлом, который может быть одноперьевым, а может быть многоперьевым и допускать вывод цветных изображений.

Устройства ввода.

К основным устройства ввода относятся манипулятор «мышь», клавиатура, джойстик, световое перо, графический планшет, сканер.

Мышь преобразует движения руки человека в управляющие сигналы для ЭВМ. На сегодняшний день существую два основных типа мыши – оптомеханическая и чисто оптическая. В оптомеханических манипуляторах движение шарика отслеживается при помощи двух валиков с прорезями и двух оптических пар светодиод-фотодиод. В результате на оптопаре образуются импульсы, которые затем при помощи счетчиков преобразуются в числовые величины, обозначающие величину относительного перемещения мыши по горизонтальной и вертикальной осям.

Оптические манипуляторы основаны на явлении отражения светового луча от поверхности специального коврика (У него должен быть небольшой коэффициент отражения). Излучатель света мыши посылает сигнал, который, отражаясь от коврика, под определенным углом попадает в фотоприемник, расположенный на манипуляторе. По величине этого угла и его изменениям при перемещении мыши вычисляется относительное перемещение по каждой из осей.

Как разновидность мыши можно рассматривать трекбол, представляющий собой перевернутую шариком вверх мышь. Здесь вручную перемещается сам шарик.

Основным устройством ввода информации в компьютер и управления им является клавиатура (keyboard). В настоящее время используются в основном мембранные (пленочные) клавиатуры.

В таких клавиатурах при нажатии на клавишу замыкаются две мембраны. Обычно мембраны для всех клавиш напаиваются на одну пленку и разделяются промежуточной пленкой с отверстиями. Получившаяся трехслойная конструкция помещается под клавишами: при нажатии мембраны замыкаются, при отжатии – размыкаются, а клавиша возвращается назад при помощи резинового купола.

Важным параметром для клавиатур является их эргономичность, именно на нее делают упор производители. На общую эргономичность клавиатуры влияет расположение клавиш, размер и положение наиболее важных из них, общая компоновка основных клавиатурных блоков, наличие или отсутствие регулировки высоты и наклона клавиатуры, а также подставки для рук.

Джойстик наиболее часто используется для компьютерных игр. Он состоит из подставки с некоторым количеством кнопок и подвижного рычага, который, собственно, и осуществляет перемещение.

Световое перо на вид напоминает обычную шариковую ручку и предназначено для формирования изображений путем простого рисования их на экране монитора. Сейчас эти устройства практически вытеснены современными плазменными панелями TouchScreen, совмещающими в себе функции монитора и светового пера: указывание здесь производится путем простого касания нужной точки экрана пальцем.

Современные модели планшетов оснащены специальным пером. Поверхность планшета обладает чувствительностью, основанной на принципе пьезоэлектрического эффекта. При нажатии пером на точку, расположенную в пределах рабочей поверхности планшета, на пластине пьезоэлектрика возникает разность потенциалов. Координаты этой точки отслеживаются и передаются в компьютер вместе с такими параметрами, как сила нажима, наклон, вращение пера и т.д. Затем с помощью специальной программы отображение точки либо отрезка выводится на экран монитора.

Сканеры предназначены для ввода в компьютер изображений. Их основным классифицирующим признаком является тип активного элемента.

По типу активных элементов различают сканеры на основе приборов с зарядовой связью и сканеры на основе фотоэлектронных умножителей. Приборы с зарядовой связью основаны на явлении увеличения проводимости полупроводникового p-n перехода под действием света. Эти приборы состоят из большого количества датчиков, преобразующих световую энергию в аналогичную по

интенсивности электрическую. Фотоэлектронные умножители по свойствам похожи на электронные лампы. Сканируемый оригинал освещается мощной галогенной лампой, отраженный от него свет попадает на катод ФЭУ, выбивая из него электроны и вызывая слабый электрический ток. Затем этот ток после усиления снимается с анода, числовые значения напряжения квантуются, преобразуются в цифровой вид и выдаются как результаты сканирования.

Физическое разрешение сканера – это то, которое определяется исключительно его механическими способностями: шагом датчиков в ПЗЛ-линейке, дискретностью, с которой двигатель способен сдвигать сканирующую каретку и т.д. Достаточным для сканирования непрозрачных оригиналов в масштабе 1:1 является разрешение порядка 300-350 dpi.

Глубина цвета- это количество бит квантования на один пиксель. Этот параметр определяет число промежуточных градаций между самым светлым и самым темным тоном, воспроизводимым сканером. При этом битность (от 8 до 16 бит на пиксель) относится к каждому каналу (то есть к каждому цвету в модели RedGreenBlue).

Тема 17. Основные тенденции развития компьютеров и вычислительных систем.

Согласно закону Гордона Мура примерно каждый год число транзисторов в микросхеме удваивается. Так, в 1971 году на микропроцессоре Intel 4004 содержалось 2300 транзисторов, а в 2003 году на микропроцессоре Intel Itanium 2 processor – 410 миллионов транзисторов. При этом увеличение количества транзисторов на одном микропроцессоре при сохранении размеров самой платы должно вести к уменьшению размера транзисторов. «Технологический тупик» по современным прогнозам ожидается к 2017 году. Поиск новых путей в микроэлектронике сводится к созданию «нетранзисторных» компьютеров – устройств, основанных на химических, биологических или световых процессах. Кроме геометрических размеров существует проблема теплоотдачи.

Основные тенденции:

- повышение количества ядер (на одном кремниевом кристалле создается не один процессор, а несколько; при этом эти несколько процессоров могут совместно использовать общие ресурсы или, напротив, могут быть независимыми друг от друга и на каждом из них может быть запущена своя операционная система);

- увеличение разрядности;
- автонастройка (снабжение встроенными механизмами регулирования производительности, контроля температурного режима, энергопотребления и производительности в зависимости от загруженности процессора);

многопоточность ядра (ядра получают возможность образовывать внутри себя несколько независимых потоков команд).

На сегодняшний день основными тенденциями развития компьютерных технологий кажутся:

- усиление роли интернета, в том числе реализация концепции виртуального офиса;
- постепенное усиление роли дистанционного образования вплоть до вытеснения отдельных форм традиционного учебного процесса;
- миниатюризация элементов компьютера (проецируемые мышь, клавиатура и экран);
- интеллектуализация компьютеров (нейронные сети, ситемы искусственного интеллекта);
- выход в пограничные области знаний.

Раздел 3. Программное обеспечение информатики. Лекции 10-11.

Темы 18-23. Программное обеспечение информатики.

По мере развития компьютеров программное обеспечение постоянно усложняется по своей структуре и составу программных модулей. В настоящее время затраты на разработку и приобретение программных продуктов в несколько раз превышают стоимость технических средств.

Программное обеспечение современных компьютеров и ВС строится по иерархическому модульному принципу. Это обеспечивает возможность адаптации компьютеров и ВС к конкретным условиям применения, открытость системы для расширения состава предоставляемых услуг, способность систем к совершенствованию, наращиванию мощности и т.д.

Программные модули ПО, относящиеся к различным подсистемам, представляют для пользователя своеобразную иерархию программных компонентов, используемую им при решении своих задач.

Нижний уровень образуют программы ОС, которые играют роль посредника между техническими средствами системы и пользователем. Однако прямое использование команд ОС требует от пользователя определенных знаний и специальной компьютерной подготовки, сосредоточенности, точности и внимания. Этот вид работ отличается трудоемкостью и чреват появлением ошибок в работе оператора. Поэтому на практике пользователи, как правило, работают не напрямую с ОС, а через командные системы - пакеты программ, дополняющие возможности ОС или графический интерфейс пользователя.

Программное обеспечение (ПО) - это совокупность программ, используемых при работе на компьютере и обеспечивающих функционирование, диагностику и тестирование аппаратных средств, а так же разработку, отладку и выполнение задач пользователя. Программное обеспечение служит интерфейсом между аппаратными ресурсами ПК и пользователями и позволяет решать задачи любой предметной области.

Программное обеспечение ЭВМ разделяют на:
- общее, или системное;
- специальное, или прикладное;
- инструментальные программные средства (средства разработки программ).

Общее или системное ПО объединяет программные компоненты, обеспечивающие многоцелевое применение ЭВМ и мало зависящие от

специфики вычислительных работ пользователей. Сюда входят программы, организующие вычислительный процесс в различных режимах работы машин, программы контроля работоспособности ЭВМ, диагностики и локализации неисправностей, программы контроля заданий пользователей, их проверки, отладки и т.д.

Системное ПО обычно поставляется потребителям комплектно с ЭВМ. Часть этого ПО может быть реализована в составе самого компьютера. Например, в ПЭВМ часть программ ОС и часть контролирующих тестов записана в ПЗУ этих машин.

Прикладное или специальное ПО (СПО) содержит пакеты прикладных программ (ППП) пользователей, обеспечивающие специфическое применение ЭВМ и вычислительной системы.

Прикладное ПО ПЭВМ комплектуется в зависимости от места и роли автоматизированного рабочего места (АРМ) работника, использующего в своей деятельности компьютер. В ПО ПЭВМ обычно включают небольшое число пакетов программ (табличный процессор, текстовый редактор, систему управления базами данных и др.). В последнее время наметилась тенденция к комплексированию и слиянию их в интегрированные программные продукты. Например, пакет MS Office фирмы Microsoft объединяет все перечисленные продукты.

Общее или системное ПО включает в свой состав операционную систему (ОС), комплекс программ технического обслуживания (КПТО), пакеты программ, дополняющие возможности ОС (ППос), и систему документации (СД).

Операционная система служит для управления вычислительным процессом путем обеспечения его необходимыми ресурсами.

Операционная система представляет собой комплекс программ, предназначенных для управления ресурсами ПК и составляет ядро "универсальной вычислительной машины - компьютера". ОС управляет всеми процессами внутри компьютера; управляет обменом между компьютером и подключенными к нему периферийными устройствами, такими, как принтер, дисплей, дисководы и винчестер и т.д.; обеспечивает возможность общения между прикладными программами и модулями аппаратуры; служит в качестве посредника между компьютером и пользователем.

Драйверы, являющиеся частью ОС, - это специальные программы управления вводом/выводом, позволяющие ОС работать с теми или иными внешними устройствами, обучая ее новому протоколу обмена данными и т.д. Драйверы бывают стандартными и загрузочными. Стандартные драйверы управляют работой стандартных устройств (монитор, клавиатура, диски, принтеры), записываются в ПЗУ ПК и образуют в совокупности "базовую систему ввода/вывода" - BIOS. Загружаемые драйверы (нестандартные) используются для управления

дополнительными внешними устройствами ПК ("мышь, компакт-диск и т.д.); для управления стандартными устройствами, используемыми в режиме, отличном от штатного (русский или казахский шрифт, ввод с клавиатуры и отображение на экран); для управления верхней, высокой и расширяемой памятью; для формированием виртуальных дисков и работой с ними и т.д.

Утилиты - это вспомогательные программы, чаще всего используемые для организации резервирования; для предотвращения заражения ПК вирусом и ликвидации последствий заражения; для архивации информации; для приспособления других программ к работе с нестандартными языками, текстами, пользователями; для диагностики конфигурации и работоспособности ПК; для убыстрения доступа к информации на дисках (организация кэш-буфера); для оптимизации размещения данных на диске; для динамического сжатия дисков (увеличения объема диска); для защиты хранящихся на компьютере данных.

Программы-оболочки - это программы, обеспечивающие более удобный и наглядный способ общения с ПК, чем ОС. Программы-оболочки не заменяют ОС, а дополняют ее. Например, Norton Commander, Norton Navigator.

Операционные оболочки - это программы, которые, как и программы-оболочки, являются надстройкой над ОС, обеспечивают удобство и наглядность общения с ПК, кроме того, расширяют возможности ОС в плане логического уровня интерфейса с пользователем: графический интерфейс, мультипрограммирование, создает интегрированную среду для работы с различными программными средами и информацией различных форматов. Например, WINDOWS 3.10, 3.11 для MS-DOS, Xwindows для Unix.

Средства тестирования и диагностики ЭВМ - это программы, составляющие средства технического обслуживания ЭВМ и предназначены для проверки работоспособности, наладки и технической эксплуатации и делятся на средства диагностики, программно-логического контроля, тестовые, программно-аппаратного контроля.

Важной частью ПО является система документации (СД), хотя она и не является программным продуктом. СД предназначается для изучения программных средств, она определяет порядок их использования, устанавливает требования и правила разработки новых программных компонентов и особенности их включения в состав ПО.

Модули комплекса программ технического обслуживания (КПТО) предназначены для проверки работоспособности вычислительного комплекса. Программы КПТО непосредственного участия в вычислениях не принимают, они только обеспечивают их. Перед

началом вычислений их задачей является проверка работоспособности аппаратуры и параметров сопряжения перечисленных уровней ПО.

Прикладной программой называется программный продукт, предназначенный для решения конкретной задачи пользователя. Обычно прикладные программы объединяются в пакеты, что является необходимым атрибутом автоматизации труда каждого специалиста-прикладника. Комплексный характер автоматизации производственных процессов предопределяет многофункциональную обработку данных и объединение отдельных практических задач. Специализация пакета определяется характером решаемых задач (пакеты для разработки экономических документов, рекламных роликов, планирования и др.) или необходимостью управления специальной техникой (управление сложными технологическими процессами, управление бортовыми системами кораблей, самолетов и т.п.). Такие специальные пакеты программ могут использовать отдельные подразделения, службы, отделы учреждений, предприятий, фирм для разработки различных планов, проектов, документов, исследований. В некоторых случаях прикладное ПО может иметь очень сложную структуру, включающую библиотеки, каталоги, программы-диспетчеры и другие обслуживающие компоненты. Программы СПО разрабатываются с учетом интересов определенной группы пользователей, иногда даже по их заказам и при их непосредственном участии.

ППП подразделяются на несколько классов. ППП общего назначения - это ППП, ориентированные на широкий круг пользователей в различных проблемных областях, позволяющие автоматизировать наиболее часто используемые функции и работы. К пакетам такого типа относятся всевозможные процессоры, текстовые, деловой графики, электронные таблицы, системы управления базами данных (СУБД) и т.д.

Проблемно-ориентированные ППП - это ППП, имеющие достаточно узкое применение, использующие особые методы представления и обработки информации, учитывающие специфику поддерживаемых задач пользователя. Например, CorelDraw, Pbrush, MathCAD, OptiNet, StatGraf, PageMaker и т.д.

Интегрированные ППП — это ППП, объединяющие в себе функции сразу несколько выше перечисленных ППП, как правило, общего назначения. Простейшим типом таких ППП является совокупность функционально-ориентированных, объединенных единым информационным интерфейсом, например, MS Office, MS Woks, Lotus 1-2-3.

ППП расширяющие функции ОС - это пакеты, определяющие достаточно широкий спектр ПС. Они обеспечивают сопряжения ЭВМ с

унифицированными приборными интерфейсами, научными приборами и установками, обеспечивают подключение к ЭВМ дополнительных унифицированных ВУ, обеспечивают поддержку работы в локальных сетях и обеспечивают обмен текстовыми файлами часто используемых форматов, подготовленных на ЭВМ различного типа, обеспечивают расширение функций ВУ ЭВМ (монитора, клавиатуры, мыши, НМД и т.д.).

Инструментальное программное обеспечение обеспечивает создание новых программ, оригинальных пользовательских систем в любой проблемной области, включая системные программы. Современные инструментальные программные средства представляют собой системы программирования со всеми необходимыми функциями.

Системы программирования представляют собой интегрированные инструментальные средства, обеспечивающие все основные функции по разработке программ: создание и редактирование исходных модулей, компиляцию или интерпретацию, создание загрузочных модулей их выполнение, отладку, тестирование, сохранение и документирование и т.д. В состав ОС обычно входят СП какого-либо языка программирования. Другие системы программирования устанавливаются отдельно как обычные прикладные программы.

В основе любой СП находится транслятор с языка программирования. Трансляторы бывают двух видов: интерпретаторы и компиляторы.

Компилятор - это комплекс программ, который транслирует весь текст исходного модуля в машинный код (объектный модуль), который только после обработки редактором связей и загрузчиком (загрузочный модуль) может быть выполнена.

Интерпретатор - это программы, которые выполняют исходный модуль программы в режиме "оператор за оператором", превращая, по ходу работы, каждый оператор ЯВУ в машинные коды.

Раздел 4. Алгоритмы и способы их записи. Лекции 12-13.

Темы 24-28. Алгоритмы и способы их записи.

Понятие алгоритма используется давно. Сам термин «алгоритм» произошел при переводе на европейские языки имени арабского математика IXв. Аль-Хорезми, которым были описаны правила (алгоритмы) выполнения основных арифметических действий в десятичной системе счисления.

В зависимости от характера занятий в своей повседневной жизни люди встречаются с различными практическими задачами: приготовление супа, проезд в общественном транспорте, решение квадратного уравнения, поиск слова в словаре и т.д. При решении любой подобной задачи человек обращается к продуманным заранее со всеми возможными вариантами предписаниям (инструкциям) о том, какие действия и в какой последовательности должны быть выполнены. В подавляющем большинстве случаев успех любой деятельности человека зависит от степени продуманности действий и их последовательности, возможных вариантов. Именно с целью успешного решения какого-либо определенного класса задач люди вырабатывают системы таких предписаний для использования разными людьми.

Алгоритмом называют систему точных и понятных предписаний (команд, директив) о содержании и последовательности выполнения конечного числа действий, необходимых для решения любой задачи данного типа.

Согласно этому определению рецепты изготовления какого-то определенного лекарства или печенья являются алгоритмами. И правило безопасного перехода пешеходом проезжей части улицы, содержащее указание человеку о его действиях, - тоже алгоритм.

По разновидности алгоритмы можно разделить на три крупных вида:

- вычислительные;
- информационные;
- управляющие.

Первые, как правило, работают с простыми видами данных (числа, векторы, матрицы), но зато процесс вычисления может быть длинным и сложным. Информационные алгоритмы, напротив, реализуют сравнительно небольшие процедуры обработки (например, поиск элементов, удовлетворяющих определенному признаку), но для больших объемов информации. Наконец, управляющие алгоритмы непрерывно анализируют информацию, поступающую от тех или иных источников, и выдают результирующие сигналы, управляющие работой тех или иных устройств. Для этого вида алгоритмов существенную роль играет их быстродействие, т.к. управляющие сигналы всегда должны появляться в нужный момент времени.

Таким образом, каждый алгоритм – это правила, описывающие процесс преобразования исходных данных в необходимый результат.

Для того, чтобы произвольное описание последовательности действий было алгоритмом, оно должно обладать следующими свойствами:

1. Дискретность. Процесс решения задачи должен быть разбит на последовательности отдельных шагов, каждый из которых называется командой. Примером команд могут служить пункты инструкции, нажатие на одну из кнопок пульта управления, рисование графического примитива (линии, дуги и т.п.), оператор языка программирования. Наиболее существенным здесь является тот факт, что алгоритм есть последовательность четко выделенных пунктов – такие «прерывные» объекты в науке принято называть дискретными.

2. Понятность. Каждая команда алгоритма должна быть понятна тому, кто исполняет алгоритм; в противном случае эта команда и, следовательно, весь алгоритм в целом не могут быть выполнены. Данное требование можно сформулировать более просто и конкретно. Составим полный список команд, который умеет делать исполнитель алгоритма, и назовем его системой команд исполнителя (СКИ). Тогда понятными будут являться только те команды, которые попадают в этот список.

3. Определенность (или детерминированность). Команды, образующие алгоритм (или, можно сказать, входящие в СКИ), должны быть предельно четкими и однозначными. Их результат не может зависеть от какой-либо дополнительной информации извне алгоритма. Сколько бы раз вы не запускали программу, для одних и тех же исходных данных всегда будет получаться один и тот же результат. При наличии ошибок в алгоритме последнее сформулированное свойство может иногда нарушаться. Например, если не было предусмотрено присвоение переменной начального значения, то результат в некоторых случаях может зависеть от случайного состояния той или иной ячейки памяти компьютера. Но это, скорее, не опровергает, а
подтверждает правило: алгоритм должен быть определенным, в противном случае это не алгоритм.

4. Результативность. Результат выполнения алгоритма должен быть обязательно получен, т.е. правильный алгоритм не может обрываться безрезультатно из-за какого-либо непреодолимого препятствия в ходе выполнения. Кроме того, любой алгоритм должен завершиться за конечное число шагов. Большинство алгоритмов данным требованиям удовлетворяют, но при наличии ошибок возможны нарушения результативности.

5. Корректность. Любой алгоритм создан для решения той или иной задачи, поэтому нам необходима уверенность, что это решение будет правильным для любых допустимых исходных данных. Указанное свойство алгоритма принято называть его корректностью. В связи с обсуждаемым свойством большое значение имеет тщательное тестирование алгоритма перед его использованием. При этом важно не столько количество проверенных сочетаний входных данных, сколько количество их типов. Например, можно сделать сколько угодно проверок для положительных значений аргумента алгоритма,

но это никак не будет гарантировать корректную его работу в случае отрицательной величины аргумента.

6. Массовость. Алгоритм имеет смысл разрабатывать только в том случае, когда он будет применяться многократно для различных наборов исходных данных. Но массовость алгоритма в отдельных случаях может нарушаться: к числу подобных исключений можно отнести алгоритмы пользования некоторыми простыми автоматами (для них входными данными служит единственный тип монет).

На практике наиболее распространены следующие формы представления алгоритмов:

- словесная (запись на естественном языке);
- графическая (изображения из графических символов);
- псевдокоды (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);
- программная (тексты на языках программирования).

Словесный способ записи алгоритмов представляет собой описание последовательных этапов обработки данных. Алгоритм задается в произвольном изложении на естественном языке. Для более сжатого и наглядного описания алгоритма используется графическое описание. Графическое описание алгоритма в форме геометрических фигур с направленными связями, показывающими переходы от одной фигуры, где каждая фигура это определенного предписание (команда) называется блок-схемой.

Раздел 5. Основы языка программирования С/С++. Лекции 14-18.

Тема 29. Понятие о языке программирования и среде программирования. Виды реализаций языков программирования.

Средой (или системой) программирования называют комплекс программ, предназначенный для создания, отладки и выполнения программ, создаваемых программистом на том или ином языке.

В состав среды программирования входят транслятор с языка программирования, компоновщик, отладчик, редактор, система помощи, библиотека стандартных подпрограмм и т.д.

Язык программирования – это набор правил, определяющих систему записей, составляющих программу, синтаксис и семантику используемых грамматических конструкций.

Синтаксис языка программирования – совокупность правил записи, которым должна удовлетворять любая программа.

Семантика языка программирования – набор правил, определяющих смысловое содержание элементов языка программирования.

Реализация языка – это программа, которая преобразует текст программы на каком-либо языке программирования в последовательность машинных команд.

Имеется два вида средств реализации (трансляторов) – компиляторы и интерпретаторы.

Компилятор осуществляет перевод в несколько этапов, причем для перехода к следующему этапу требуется, чтобы предыдущий этап был полностью и успешно завершен. При переводе программы с языка С компилятор работает в три прохода – препроцессорная обработка, синтаксический анализ, перевод в машинный код. После компиляции производится компоновка, во время которой устанавливаются смысловые связи между отдельными частями программы и соединяются модули, откомпилированные отдельно. Результатом компиляции и компоновки является программа в машинных кодах, которая затем выполняется без участия среды программирования.

Интерпретатор работает с программой построчно. Каждая строка программы проверяется на наличие ошибок, если их нет, то сразу переводится и выполняется.

Тема 30. Характеристики и свойства языков программирования.

Основными характеристиками , позволяющими сравнивать языки программирования и выбирать наилучшие для решения той или иной задачи, являются мощность, уровень, надежность, удобочитаемость, полнота, гибкость, простота, мобильность и эффективность.

Мощность языка характеризуется количеством и разнообразием задач, алгоритмы решения которых можно записать, используя этот язык. Любую задачу, запрограммированную на каком-либо языке, можно запрограммировать и на машинном языке.

Уровень языка характеризуется сложностью решения задач с помощью этого языка. Чем проще для человека записать решение задачи, тем выше уровень языка.

Надежность языка обеспечивает при написании программ минимум ошибок, которые не обнаруживаются при компиляции.

Удобочитаемость языка – это свойство, обеспечивающее легкость восприятия программ человеком.

Полнота языка обеспечивает описание на языке решений как можно большего числа задач определенной предметной области.

Гибкость языка обеспечивает легкость выражения на языке необходимых для решения задач действий, предоставляет программисту достаточно возможностей для выражения всех операций в программе.

Простота языка обеспечивает легкость понимания семантики языковых конструкций и запоминания их синтаксиса.

Мобильность языка обеспечивает независимость его от аппаратных средств, позволяет переносить программное обеспечение с одного типа компьютера на другой с относительной легкостью.

Эффективность языка обеспечивает высокоэффективную работу программ, написанных на этом языке.

Темы 31-32. Программа на языке программирования C/C++. Понятие о структурном программировании. Стандартные типы данных.

C – это универсальный язык программирования, сочетающий в себе как высокоуровневые, так и низкоуровневые черты. В частности, низкоуровневой является возможность в некоторых ситуациях принудительно назначать место в памяти для хранения данных. К высокоуровневым особенностям, например, относится отсутствие механизмов ввода-вывода, которые реализованы с помощью явно вызываемых функций.

Этот язык был создан в 1972 году Деннисом Ритчи (Bell Laboratories). Основными его предшественниками являются:

Algol – 1960 год – разработан специально созданным международным комитетом – большое внимание было уделено модульной структуре программы;

CPL (Combined Programming Language) – 1963 год – разработка Лондонского и Кембриджского университетов;

BCPL – 1967 год – разработан Мартином Ричардсом – получен выделением из CPL его основных свойств;

B – 1970 год – разработан Кеном Томпсоном из Bell Laboratories – еще большее упрощение – пригоден для узкого круга системных задач.

Впоследствии Деннис Ритчи дополнил систему типов данных языка B без потери его простоты.

Программа на языке C состоит из произвольного количества функций, причем одна из них должна называться main(). Выполнение программы начинается с первой инструкции этой функции.

Тело функции представляет из себя перечень действий, которые необходимо выполнить. Этот список действий заключается в фигурные скобки. Перед телом функции помещается заголовок.

Как правило, функция main() среди своих действий содержит вызовы других функций, которые могут создаваться самим программистом или содержаться в библиотеках, входящих в комплект поставки.

```
# include <stdio.h>
main()
{       printf("Здравствуй, мир!\n");
}
```

В начале программы помещается конструкция, позволяющая подключить информацию о стандартной библиотеке ввода-вывода. Знак # показывает, что данная строка должна быть выполнена до начала процесса перевода программы в машинный код.

Если необходимо в теле одной функции вызвать другую, то к ней надо обратиться по имени, указав в скобках после имени список параметров, которые надо передать в вызываемую функцию для работы. Если функции не надо ничего передавать, то в строке вызова круглые скобки остаются, но в них ничего не будет.

Комбинация "\n" внутри строки означает на необходимость перейти на новую строку при печати. Существует несколько таких последовательностей, начинающихся с наклонной черты, которые служат для печати трудно представимых символов.

В языке C существует четыре базовых типа данных:

- символьный (char);
- целый (int);
- действительный (float);
- двойной точности(double).

Имеется также несколько квалификаторов, которые можно использовать для расширения числа типов данных. Это квалификаторы long (длинный), short (короткий), unsigned (беззнаковый) и signed (со знаком). Знаковым или беззнаковым может объявляться любой целый тип (в том числе и тип char), а long и short применяются к типу int. Кроме этого существует тп long double.

Стандарт языка С не устанавливает размер памяти, который должен быть выделен на тот или иной тип. При этом некоторые соотношения между типами в стандарте описаны. Например,

short int <= int <= long int.

Реально выделенный размер можно узнать при помощи конструкции sizeof(тип) или sizeof(переменная).

Кроме переменных в программах на С используются константы. Целую константу от действительной можно отличить по наличию десятичной точки.

Символьная константа – это целое число, записанное в виде символа, заключенного в одиночные кавычки. Значением символьной константы является код символа, зависящий от используемой системы кодировки. Символьные константы могут участвовать в операциях по тем же правилам, что и целые переменные или константы.

Некоторые трудно представимые символы записываются как комбинация нескольких других символов. Такие комбинации называются Esc-последовательностями. К ним, например, относятся:

\n – переход на новую строку \a – звонок

\t – горизонтальная табуляция \f – перевод страницы

\\ - собственно наклонная черта.

Строковая константа – это произвольное количество символов, заключенных в двойные кавычки, которые не являются составной частью строки. Такая строка фактически воспринимается как массив символов, а в конце этого массива присутствует специальный невидимый символ завершения строки \0.

Символьная константа и строка, содержащая один символ, воспринимаются компьютером совершенно по разному. Так, “a” воспринимается как массив из двух символов, а ‘a’ – как число, равное коду буквы a.

Константы перечисления – это список целых констант. Первое имя в списке имеет значение 0, следующее – 1 и т.д. Если значение какой-нибудь константы явно определено, то следующие константы продолжаю прогрессию с шагом 1.

enum aaa {A,B=8,C}.

Все переменные должны быть описаны до их использования, при своем описании переменная может быть инициализирована (то есть ей может быть присвоено начальное значение). В языке С описание переменных разрешается в начале функции или начале блока.

К любой переменной может быть применен квалификатор const для запрета в дальнейшем изменения значения этой переменной. Константы принято инициализировать при описании в обязательном порядке.

Темы 33-36. Управляющие структуры языка C/C++.

Структуры, реализующие последовательность.

Для того чтобы любое выражение языка воспринималось как действие, в конце его необходимо поставить точку с запятой.

```
a=5;
```

Если необходимо, чтобы с точки зрения синтаксиса несколько действий воспринимались как одно, то перед первым из этих действий необходимо открыть фигурную скобку, а после последнего – закрыть. Такая конструкция называется составной инструкцией или составным оператором.

Структуры, реализующие развилку.

Основной конструкцией для реализации развилки является оператор if-else.

```
if (условие)
                действие;
else
                действие;
```

Если по какой-то ветви необходимо выполнить более чем одно действие, то нужны фигурные скобки.

```
if(a>5)
          {b=3;
           t=4;}
else
           a=29;
k=11;
```

Очень часто не поставленные или неверно поставленные фигурные скобки вызывают не только синтаксические ошибки, обнаруживаемые при компиляции, но и смысловые ошибки, из-за которых программа работает неверно.

else – часть может отсутствовать. В этом случае, если условие ложно, то управление передается на следующий оператор программы.

```
if (a>5)                        if(a<=5)
     ;                               t=t+1;
else                                 r=7;
     t=t+1;
r=7;
```

В случае, когда нескольким вложенным друг в друга или идущим друг за другом операторам if соответствует один оператор else, этот else считается относящимся к ближайшем if.

```
if(t > t1)                      if(t > t1)     {
          if (n<0)                      if (n<0)
               n=n+m;                        n=n+m; }
```

```
          else                  else
               n=n-m;                n=n-m;
```

При необходимости многоступенчатого принятия решений удобно использовать конструкцию else – if. Условия проверяются по порядку; как только встречается истинное условие, выполняется соответствующая ему инструкция и последовательность проверок завершается. Если ни одно из условий не оказалось истинным, то срабатывает последняя else часть.

```
if (r >= 45)
          printf("Выплачивается повышенная стипендия\n");
else if (r >= 35)
               printf("Выплачивается стипендия\n");
          else if(r >= 25)
                    printf("Нет стипендии\n");
               else
                    printf("Пора гнать\n");
```

Инструкция switch используется для выбора одного из многих путей и называется переключателем. Она проверяет, совпадает ли значение некоторого выражения с одной из целых констант, и выполняет соответствующую этому занчению ветвь, а также последующие ветви, соответствующие константам.

```
switch (выражение)  {
          case константа : инструкции
          case константа : инструкции
          . . . . . .
          default :        инструкции   }
```

Если ни одна из констант не совпала со значением выражения, то выполняются только инструкции, стоящие после ключевого слова default.

Так как в языке С символьный тип является подмножеством целого типа, то в конструкции switch разрешается использование символьных переменных и констант.

При необходимости немедленного выхода из переключателя после выполнения соответствующей ветви следует использовать инструкцию break. После ее выполнения управление передается на инструкцию, следующую за обязательной закрывающей фигурной скобкой оператора switch.

```
i=4;
switch (i+2)   {
          case 4 : i=i+4;
          case 6 : i=i+6; break;
          case 8 : i=i+8;
                    }
```

Структуры, реализующие повторение.

В языке С существует три основных конструкции для реализации циклов. Для представления циклов с постусловием используется конструкция do – while, а для циклов с предусловием – конструкции while и for.

```
while (условие)                    i=2;
                действие;          while (i<=5)
действие;                          { . . . .
                                             i=i+1;}
                                   t=4;
```

```
do                              i=3;
          действие;                  do{
while (условие);                  . . .
                                      i=i+1;
                                      . . .}
                                   while (i<=10);
```

При организации циклов с предусловием и с постусловием условие необходимо формулировать так, чтобы выполнение цикла продолжалось по ветви “да”.

При написании циклов с предусловием более компактным способом записи является использование цикла for, поскольку в его заголовке помимо условия окончания цикла содержатся, если это необходимо, и задание начального значения счетчика, и предписание по изменению значения счетчика.

```
for (выражение 1; выражение 2; выражение 3)
                              действие;
действие;

for (i=2; i<=5; i=i+1)
                {. . .
                 . . .
                }
t=4;
```

Первое выражение из заголовка цикла for воспринимается как действие и выполняется только 1 раз. Второе выражение воспринимается как условие. До тех пор, пока это условие истинно, выполняется тело цикла. После каждого выполнения тела выполняется третье выражение, которое воспринимается как действие. Затем управление вновь передается на проверку условия.

Любое из трех выражений, находящихся в заголовке цикла for, может быть пропущено, а может присутствовать, но не являться действием или условием. Обязательными с точки зрения синтаксиса

являются только обе точки с запятой. Тем не менее, компилятор воспринимает второе выражение как условие. Если оно на самом деле условием не является, то вычисляется его числовое значение. Условие считается ложным, если это значение равно 0, и истинным во всех остальных случаях. Это правило распространяется на любые места в программах на С, где ожидается условие.

В заголовке цикла for часто используется оператор “,”. Выражения, разделенные запятой выполняются слева направо.

```
for (i=0, j=3; i<j;)
                { i=i+1;
                  j=j+0.5;
                }
```

Структуры безусловного перехода.

В языке С существуют три оператора реализующие безусловный переход – break, continue, goto. Несмотря на термин “безусловный”, управление обычно передается на эти операторы при выполнении некоторого условия.

Инструкция break используется в ситуации, когда необходимо прервать выполнение переключателя switch или цикла, организованного при помощи операторов цикла. После выполнения инструкции break управление будет передано на оператор, стоящий после цикла или после переключателя. При этом выход будет осуществлен только из одного цикла или переключателя.

Использование оператора break вне циклов или переключателей приводит к синтаксической ошибке.

```
for(..;..;..)
          { . . .;
            . . .;
            for(..;..;..)
                  {. . .;
                   if(..)
                         break;
                   . . .; }
        . . .;
          }
. . .;
```

Оператор continue вынуждает ближайший цикл, в котором находится этот оператор, начать работу со следующего шага. В циклах while и do while переход осуществляется на проверку условия, а в цикле for – на выполнение действия, стоящего в заголовке за второй точкой с запятой.

При использовании инструкции continue в циклах while и do while существует опасность зацикливания в случае, когда изменение счетчика цикла находится ниже conyinue.

```
i=5;
while (i<=10)
          {. . .;
      if(i==7)
               continue;
           . . .;
      . . .;}
```

Инструкция goto используется для передачи управления в помеченную строку программы. Имена меткам даются по тем же правилам, что и имена переменных, но метки не нужно описывать. Метка ставится в начале строки, на которую осуществляется переход, а после метки ставится двоеточие. Переход при помощи оператора goto принято использовать для передачи управления вниз по программе.

```
for (..;..;..)
     {. . .;
      for (..;..;..)
          {. . .;
           if(условие)
               goto t;
           . . .;
          }
       . . .;
     }
t:. . .;
```

Тема 37. Основные операции языка программирования C/C++.

Арифметические операторы.

К бинарным арифметическим операторам относятся +,-,*,/, и %.

Оператор % вычисляет остаток от деления своего левого операнда на правый. Он применим только к целым типам.

```
int a=4,b=7,c,d;
c=b%a;
d=b/a;
```

Деление целых чисел сопровождается отбрасыванием дробной части. Бинарные операторы + и – имеют одинаковый приоритет, который ниже приоритета операторов *,/,%. Еще выше приоритет унарных операторов + и -. Арифметические операции одного приоритетного уровня выполняются слева направо.

Операторы отношения и логические операторы.

К операторам отношения относятся <,<=,>,>=, которые имеют одинаковый приоритет. Ровно на одну ступень ниже приоритет операторов сравнения на равенство == и !=.

Все операторы отношения имеют приоритет ниже арифметических операторов.

К бинарным логическим операторам относятся && и ||. Особенностью языка С является то, что выполнение цепочки выражений, соединенных логическими && или || прекращается, как только становится известной истинность или ложность результата.

```
while (i<n && a[i]<a[i+1])
```

Приоритет оператора && выше, чем оператора ||, но ниже, чем приоритет операторов отношения и проверки на равенство.

Если же в сложном выражении встречается присваивание, то скобки могут понадобиться, так как у присваивания один из самых низких приоритетов.

```
while (i<a+3 && (c=t[i])!=a)
```

Численным результатом вычисления выражения отношения или логического выражения является 1 в случае, если оно оказалось истинным, и 0 в случае, если оно ложно.

В С также существует унарный оператор отрицания !, который преобразует истину в ложь, а ложь в истину.

Например, if(!a) аналогично if (a==0).

Инкрементные и декрементные операторы.

Инкрементный оператор ++ добавляет 1 к своему операнду, а декрементный оператор -- вычитает 1 от своего операнда.

Операторы ++ и – можно использовать в двух формах: префиксной, помещая их перед переменной (++i), и постфиксной, помещая их после переменной (i++). Значение переменной i будет увеличено на единицу при любой из этих форм записи, но в префиксной форме это изменение происходит до использования значения переменной, а в постфиксной форме уже после использования. Таким образом, разница в этих формах видна при использовании инкрементных и декрементных операторов в составе выражений, а не в качестве отдельных операторов.

```
i=3;                i=3;
a=i++;              a=++i;
d=++i;              d=i++;
```

Инкрементные и декрементные операторы можно применять только к переменным. Запись (a+b)++ не допускается.

Побитовые операторы.

Использование этой группы операторов дает возможность обращения к отдельным битам двоичного представления целочисленных операндов. В С существует 5 бинарных и один унарный побитовый оператор.

& - побитовое И;

| - побитовое ИЛИ;

^ - побитовое исключающее ИЛИ;

<<, >> - побитовые сдвиги;

~ - побитовое отрицание.

Оператор & выполняет побитовую конъюнкцию своих операндов.

```
a=2;
b=6;
c=a&b;
```

Оператор | выполнят побитовую дизъюнкцию своих операндов. Его можно, в частности, использовать для проверки числа на четность:

```
if((a|1)==a)
```

Оператор ^ устанавливает 1 в том разряде результата, в котором соответствующие разряды операндов имеют различное значение, и 0, если в этих разрядах значения совпадают.

```
a=2;
b=6;
c=a^b;
```

Операторы << и >> выполняют сдвиг в соответствующем направлении своего левого операнда на число позиций, задаваемое правым операндом, который должен быть положительным. При сдвиге беззнаковых величин освобождающиеся биты заполняются нулями, а при сдвиге вправо знаковой величины результат зависит от реализации.

```
a=2;
d=a--;
d=d<<a;
```

Унарный оператор ~ превращает единичные биты в нулевые и наоборот.

Операторы присваивания.

В выражениях типа a=a+t;, в которых стоящая в левой части присваивания переменная, повторяется и справа, может использоваться видоизмененная форма оператора присваивания: a+=t;.

Таким образом, в С имеется одиннадцать операторов присваивания: =,+=,-=,*=,/=,%=,&=,|=,^=,<<=,>>=.

В выражениях с использованием видоизмененной формы оператора присваивания вторым операндом является все, находящееся справа от оператора присваивания:

a*=b+1; равносильно a=a*(b+1);

Условное выражение.

Если в программе возникает необходимость реализовать выбор в пределах выражения, и в пределах этого же выражения воспользоваться результатом этого выбора, то можно воспользоваться тернарным оператором ?:. Его синтаксис:

операнд 1 ? операнд 2 : операнд 3

Первым вычисляется операнд 1, который расценивается как условие. Если результатом вычисления операнда 1 оказалась истина, то вычисляется выражение, являющееся операндом 2, и результат вычисления этого выражения оказывается результатом всего тернарного оператора. В противном случае вычисляется выражение, являющееся операндом 3, и снова его значение является значением всего условного выражения.

```
a=4;
b=3;
c=(a==b?b++:--a);
```

Условное выражение обычно берется в скобки, так как этот оператор имеет достаточно низкий приоритет (хотя он выше приоритета оператора присваивания)

```
for(i=1;i<=n;i++)
     for(j=1;j<=m;j++)
          printf("%d%c",a[i][j],j!=m?' ':'\n');
```

Преобразование типов.

В случае, если в одной бинарной операции участвуют операнды разных типов, еще до выполнения операции тип операнда, занимающего меньше места в памяти, приводится к типу другого операнда. Такое неявное приведение типа является временным, только в пределах проводимой операции. Результат будет иметь "старший" тип.

В операциях присваивания правила другие: значение правой части приводится к типу левой части, который и является типом результата.

Кроме неявного преобразования типов существует и унарный оператор явного преобразования, который называется оператором приведения типа. Его синтаксис:

```
(тип)выражение;
a=(float)sum/n;
```

Оператор приведения на самом деле не меняет тип переменной, а временно генерирует значение нового типа.

Таблица приоритетов.

В таблице приоритетов операторы, перечисленные на одной строке, имеют одинаковый приоритет, строки упорядочены по убыванию приоритетов.

Операторы	Выполняются
() [] -> .	слева направо
! ~ ++ -- + - * & (тип) sizeof	справа налево
* / %	слева направо
+ -	слева направо
<< >>	слева направо
< <= > >=	слева направо
== !=	слева направо
&	слева направо
^	слева направо
|	слева направо
&&	слева направо
||	слева направо
?:	справа налево
= += -= *= /= %= &= ^= |= <<= >>=	справа налево
,	слева направо

Темы 38-41. Массивы и указатели. Обработка текстов и файлов.

Основные свойства массивов.

Массив – это набор данных, элементы которого образуют упорядоченную последовательность, причем все эти элементы должны быть одного типа.

Как и обычные переменные, массивы в С описываются в начале функции или в начале блока, причем при описании массива, как правило, указывается количество ячеек, под которое будет выделяться память.

тип имя[константа 1][константа 2]..[константа n]

Количество размерностей массива стандартом не ограничивается. Если константное выражение в квадратных скобках присутствует, то оно должно быть натуральным числом. Если константное выражение отсутствует, то считается, что описание не завершено. Завершить описание можно двумя способами: либо при другом описании, либо при инициализации массива.

```
int a[]={5,6,7,8}.
```

Память при таком описании выделяется по числу инициализаторов. Счет элементов в массивах начинается с 0. Инициализаторы могут присутствовать и при явно заданном размере массива. Если число инициализаторов меньше размерности массива, то оставшиеся элементы получают значение 0. Если число инициализаторов больше размерности, то выдается сообщение о синтаксической ошибке.

int a[50]={0}; - обнуление массива.

В многомерных массивах пустой может быть только первая размерность.

int a[3][3]={3,4,5,6};-массив заполняется по строкам

int a[3][3]={{3,4},{5},{6}};

В многомерных массивах хранение организовано так, что самый правый индекс в паре или наборе индексов меняется чаще всего.

Хотя память под массив выделяется непрерывным куском, особенности статического распределения памяти приводят к тому, что элементы могут и не храниться связным участком.

Символьные массивы можно инициализировать и без использования фигурных скобок:

char mas[]=”Скоро зачет\n”;

Память будет выделена с учетом символа “\0”. При помощи функции printf символьный массив можно распечатать не используя операторы цикла:

printf(“%s”,mas);

Понятие об указателях.

Указатель – это некоторая переменная, значение которой является адресом другой переменной.

Указатели описываются специальным образом:

тип *имя;

float *t; - t будет хранить адреса переменных типа float.

Адрес сам по себе является целым числом, память под указатель выделяется как под целое, но для нормальной работы с указателем необходимо знать тип переменной, адрес которой хранит указатель.

Для извлечения адресов служит унарная операция &, которая называется операцией получения адреса.

t=&b; - t хранит адрес b или указывает на b.

Унарная операция * позволяет извлечь содержимое ячейки, на которую указывает переменная, стоящая после символа *. Эта операция называется операцией доступа по указателю.

k=*t;

t=&b; k=*t; равносильно k=b;

Связь указателей и массивов.

Любой доступ к элементу массива, осуществляемый с помощью операции индексирования, может быть осуществлён и при помощи указателя.

```
int a[10];
int *x=&a[0];
```

x указывает на нулевой элемент массива a, или, иначе говоря, x содержит адрес элемента a[0]. В этом случае адрес любого i-ого элемента массива можно представить как x+i, а его значение – как *(x+i).

Подобные операции верны для любого типа и размера массивов, так как при прибавлении к указателю целого числа i указатель «сдвигается» не на i байтов, а на i элементов этого типа.

x=&a[0]; равносильно x=a;

Если где-либо на протяжении программы используется идентификатор массива без указания индекса, то эта переменная имеет значение, равное адресу нулевого элемента массива.

Существуют и другие способы записи элементов массива и их адресов. Так a[i] можно записать как *(a+i), а запись &a[i] эквивалентна записи a+i. С другой стороны, если x – указатель, то в выражениях его можно использовать с индексом, то есть запись x[i] эквивалентна записи *(x+i).

Между указателем, которому присвоен адрес нулевого элемента массива, и именем массива существует важное различие. Указатель является переменной, поэтому его значение по ходу программы можно менять (x=a; x++;). Имя массива без указания индекса не может находить в левой части оператора присваивания (записи a++; или a=x; - недопустимы). Это связано с тем, что участок памяти, выделенный под массив, не изменяется на протяжении работы программы.

Над указателями допустимо выполнение следующих операций:

- присвоение указателю в качестве начального значения адреса объекта данных или нуля;
- прибавление к указателю или вычитание из указателя целого числа;
- сравнение двух указателей;
- вычитания одного указателя из другого, если оба указателя указывают на объекты одного типа.

Следующий фрагмент программы можно записать разными способами:

1.
```
int a[5]={3,7,11,8,2}, i, max;
max=a[0];
for(i=1;i<=4;i++)
  if(a[i]>=max)
       max=a[i];
```

```
2. int a[]={3,7,11,8,2}, i, max;
   max=*a;
   for(i=1;i<=4;i++)
     if(*(a+i)>=max)
          max=*(a+i);
3. int a[]={3,7,11,8,2}, i, *t;
   int max=*(t=a);
   for(i=1;i<=4;i++)
     if(*(t+i)>=max)
          max=*(t+i);
4. int a[]={3,7,11,8,2}, *t;
   int max=*a;
   for(t=a;t<=a+4;t++)
     if(*t>=max)
          max=*t;
```

При работе с символьными массивами с использованием указателей можно вообще явным образом не выделять память под массив:

```
char a1,a2;
a1="Кругом одни враги";
a2=a1+17;
while (--a2>=a1)
     printf("%c",*a2);
                        printf("\n");
```

Темы 42-43. Классы памяти. Функции. Способы организации взаимодействия между функциями.

Аргументы и возвращаемые значения функций.

В программе на С может быть любое количество функций, среди них нет иерархии. Но первой будет выполняться функция main(), где бы она ни стояла. Любая функция в процессе своей работы может вызвать любую другую функцию или даже саму себя (рекурсия).

Чтобы вызвать функцию необходимо обратиться к ней по имени в качестве оператора или операнда, указав после имени в круглых скобках список аргументов, которые необходимо передать. Имена переменных в этом списке пишутся через запятую без указания типа. Такие аргументы называются фактическими параметрами.

В результате вызова управление будет передано в вызванную функцию, а затем вернётся к вызывающей функции, когда вызванная функция выполнит все свои инструкции или когда встретится ключевое слово return.

```
main()                          r()
     { r();)                         {putchar('r');
```

```
a()                                    a();
     {putchar('a');                    putchar('r');
      d();                               }
      putchar('a');
     }
d()
     {putchar('d');}
```

Заголовок любой функции имеет вид:

тип возвращаемого значения ИМЯ (список параметров)

Список параметров представляет собой перечень переменных, в которые будут попадать значения, передаваемые из вызывающей функции. Имена в этом списке пишутся через запятую, для каждой переменной указывается тип. Параметры, задаваемые в заголовке функции, называются формальными.

Если функция возвращает значение, то её можно вызывать и в качестве оператора, и в качестве операнда. Но, если вызвать её в качестве оператора, то возвращаемое значение будет пропадать

Если функция ничего не возвращает, то её можно вызвать только в качестве оператора. У таких функций в заголовке в качестве типа возвращаемого значения указывается ключевое слово void.

При передаче аргументов между функциями типы, количество и порядок следования фактических аргументов в вызывающей функции должны соответствовать типам, количеству и порядку следования формальных параметров в вызываемой функции.

```
void main()
     {int a,b,t;
      . . .
      t=power(a,b);
      . . .
     }
int power(int base, int n)
     {int a;
      for(a=1;n>0;--n)
          a=a*base;
      return a; }
```

Никакие действия, которые происходят с формальными параметрами в вызываемой функции, не могут привести к изменению значений фактических параметров. Это связано с тем, что сами переменные не передаются в вызываемую функцию. Компилятор перед передачей управления делает копии фактических параметров, и именно эти копии попадают в вызываемую функцию. Такой способ передачи аргументов называется передачей по значению. При передаче по значению обратно в вызывающую функцию может

вернуться только значение переменной, упомянутой в операторе return. И тип именно этой переменной и является типом возвращаемого значения, который указывается в заголовке функции.

Если из одной функции в другую в качестве параметра передаётся массив, то механизм передачи другой. В функцию передаётся адрес нулевого элемента массива, и она фактически получает доступ ко всем элементам массива. Копирование элементов в этом случае не производится.

```
void main()              void funk(int t[100])
{int a[100];             { . . .}
 . . .
 funk(a); . . .}
```

Такая передача называется передачей по адресу, и таким образом можно передавать не только массивы, но и простые переменные. Передача переменных по адресу используется, в частности, для того, чтобы позволять функции менять значения более чем одной переменной, переданной из другой функции, и неявно возвращать эти изменённые значения в вызвавшую функцию.

```
void main(void)
{ int a,b;
  . . .
  top(&a,&b);
  . . .
}
void top (int *w, int *e)
{ int c;
  c=*w;
  *w= *e;
  *e=c;
}
```

Стандарт языка С разрешает не указывать в заголовке функции тип возвращаемого значения только в том случае, если этот тип – целый. Во всех остальных функциях необходимо явное указание типа возвращаемого значения. Кроме этого, все такие функции нуждаются в дополнительном описании, которое называется прототипом и имеет вид:

тип возвращаемого значения ИМЯ(); или
тип возвращаемого значения ИМЯ(список типов параметров);

Прототип можно не указывать, если заголовок функции оказывается в программном коде раньше её первого вызова. Если всё-таки приходиться писать прототипы, то их часто указывают один раз перед самой первой функцией в файле. Однако для документирования программ и с целью возможности использования

функций в разных проектах иногда советуют дополнительно описывать каждую функцию в каждой вызывающей функции.

С целью полного соответствия программы требованиям стандарта для функций, не имеющих аргументов на входе, принято в круглых скобках после имени в заголовке указывать ключевое слово void.

```
void main(void)
```

Область действия переменных. Классы памяти.

Все переменные, которые описаны внутри каких-либо фигурных скобок, являются локальными, что означает ограничение области действия переменных той функцией или тем блоком, в котором они описаны. При временном выходе из функции такая переменная недоступна, но её значение не теряется. При окончании работы функции переменная прекращает своё существование. Для описания таких локальных переменных может дополнительно использоваться ключевое слово auto, и, соответственно, такие переменные называются автоматическими.

```
void main(void)
     {int x=1;
          {int x=7;
           printf("%d\n",x--);
          }
      printf("%d\n",++x);
     }
```

Вообще, в С существуют четыре класса памяти (класса переменных) – автоматические, регистровые (являются разновидность автоматических), статические и внешние. Глобальную область действия имеют только внешние переменные, все остальные являются локальными.

Регистровые переменные хранятся не в оперативной памяти, а в регистрах процессора, благодаря чему программа, возможно, будет работать быстрее. Таким переменными можно назначать только автоматические переменные или формальные параметры функций. При их описании перед типом указывается ключевое слово register.

Компилятор может не выделить регистр для такой переменной, но, независимо от того, выделялся на самом деле регистр или нет, для таких переменных не определено понятие адреса.

Статические переменные, подобно автоматическим, локальны в той функции или в том блоке, в которых они описаны. Разница заключается в том, что значения статических переменных не теряются, когда выполнение функции (но не всей программы) завершается. Если управление вновь будет передано в данную функцию, то выяснится, что статические переменные имеют именно

те значения, с которыми они остались после предыдущего вызова подпрограммы.

```
void main(void)
{kirpich();
 kirpich();
}
void kirpich(void)
{ static int x;
  x+=3;
  printf("%d\n";x);
}
```

Если переменная описана как статическая, то начальное значение присваивается ей только один раз – перед началом работы программы. При вызовах функции инициализации не происходит. При этом автоматические переменные получают начальные значения по ходу выполнения программы. Если значение статической переменной не присвоено явно, то она полагается равной нулю.

Внешние переменные отличаются от всех ранее рассмотренных классов памяти тем, что имеют глобальную область действия. Они существуют вне отдельных функций и доступны любой функции программы, которая захочет к ним обратиться. Фактически, внешние переменные неявно передаются между функциями, и их значения меняются в процессе использования в любой из этих функций.

По умолчанию внешние переменные получают начальные значения равные нулю.

Внешние переменные в обязательном порядке описываются вне функций, а кроме того должны дополнительно описываться в функции, которая намеревается их использовать, если только они не описаны выше этой функции в том же самом программно файле. Для такого дополнительного описания внешних переменных внутри функций используется ключевое слово extern.

```
int x;
void main(void)
{printf("x=%d\n",x);
 gnus();
 giena();
}
void gnus(void)
{extern int y;
 x=x+7;
 y=y-8;
 printf("x=%d\n",x);
}
int y=11;
```

```
void giena(void)
{x=x-3;
 printf(“x=%d\n”,x);
}
```