**1)Понятие и классификация программного обеспечения**

***Программное обеспечение***(англ. *software*) – это совокупность программ, обеспечивающих функционирование компьютеров и решение с их помощью задач предметных областей.

1.     Системное программное обеспечение (системные программы);

2.     Прикладное программное обеспечение (прикладные программы);

3.    Инструментальное обеспечение (инструментальные системы).

***Системное программное обеспечение*** (СПО) – это программы, управляющие работой компьютера и выполняющие различные вспомогательные функции, например, управление ресурсами компьютера, создание копий информации, проверка работоспособности устройств компьютера, выдача справочной информации о компьютере и др. Они предназначены для всех категорий пользователей, используются для эффективной работы компьютера и пользователя, а также эффективного выполнения прикладных программ.

***Прикладное программное обеспечение*** (ППО) предназначено для решения задач пользователя. В его состав входят ***прикладные программы пользователей***и***пакеты прикладных программ*** (ППП) различного назначения***.***

*Прикладная программа* *пользователя* – это любая программа, способствующая решению какой-либо задачи в пределах данной проблемной области. Прикладные программы могут использоваться либо автономно, либо в составе программных комплексов или пакетов.

*Пакеты прикладных программ* (ППП) – это специальным образом организованные программные комплексы, рассчитанные на общее применение в определенной проблемной области и дополненные соответствующей технической документацией. Различают следующие типы ППП:

К ***инструментальному программному обеспечению*** относят: ***системы программирования* –**для разработки новых программ, например, Паскаль, Бейсик. Обычно они включают: *редактор* *текстов*, обеспечивающий создание и редактирование программ на исходном языке программирования (исходных программ), *транслятор*, а также *библиотеки подпрограмм*;***инструментальные среды***для разработки приложений, например, C++, Delphi, Visual Basic, Java, которые включают средства визуального программирования;

*Транслятор*(англ. *translator* – переводчик) – это программа-переводчик, которая преобразует программу с языка высокого уровня в программу, состоящую из машинных команд.

*Компилятор* (англ. *compiler* – составитель, собиратель) читает всю программу *целиком*, делает ее перевод и создает законченный вариант программы на машинном языке, который затем и выполняется

*Интерпретатор* (англ. *interpreter* – истолкователь, устный переводчик) переводит и выполняет программу *строка за строкой*.

**2) Понятие и классификация системного программного обеспечения**

**Системное программное обеспечение** (System Software) - совокупность программ и программных комплексов для обеспечения работы компьютера и сетей ЭВМ.

СПО управляет ресурсами компьютерной системы и позволяет пользователям программировать в более выразительных языках, чем машинных язык компьютера. Состав СПО мало зависит от характера решаемых задач пользователя.

**Назначение системного программного обеспечения**

Системное программное обеспечение предназначено для:

      создания операционной среды функционирования других программ (другими словами, для организации выполнения программ);

      автоматизации разработки (создания) новых программ;

      обеспечения надежной и эффективной работы самого компьютера и вычислительной сети;

      проведения диагностики и профилактики аппаратуры компьютера и вычислительных сетей;

      выполнения вспомогательных технологических процессов (копирование, архивирование, восстановление файлов программ и баз данных и т.д.).

**Классификация системного программного обеспечения**

В СПО традиционно включают

      системные управляющие и

      системные обрабатывающие программы.

**Управляющие системные программы** организуют корректное функционирование всех устройств системы.

Основные системные функции управляющих программ -

      управление вычислительными процессами и вычислительными комплексами и

      работа с внутренними данными ОС.

В настоящее время системные управляющие программы поставляются фирмами-разработчиками и фирмами-дистрибьюторами в виде инсталляционных пакетов операционных систем и драйверов специальных устройств.

**Обрабатывающие системные программы** выполняются как специальные прикладные задачи, или приложения.

Эти программы поставляются чаще в виде дистрибутивных пакетов, включающих ПО

**3) Жизненный цикл программного обеспечения. Процессы жизненного**

**Цикла**

**Жизненный цикл**– это модель создания и использования программной системы. Он отражает различные состояния программной системы, начиная с момента возникновения необходимости в этой программной системе и принятия решения о ее создании и заканчивая полным изъятием программной системы из эксплуатации.

Жизненный цикл программного обеспечения базируется на трех группах процессов:

1. основные процессы жизненного цикла, то есть приобретение, поставка, разработка, эксплуатация и сопровождение;
2. вспомогательные процессы, обеспечивающие выполнение основных процессов, то есть документирование, верификация, аттестация, оценка качества и другие;
3. организационные процессы, то есть управление проектами, создание инфраструктуры проекта и обучение.

Разработка включает в себя все работы по созданию программного обеспечения в соответствии с заданными требованиями. Сюда включаются оформление проектной и эксплуатационной документации, подготовка материалов, необходимых для проверки работоспособности и качества программных продуктов.

Основные этапы процесса разработки:

1. анализ требований заказчика;
2. проектирование;
3. реализация (программирование).

Процесс эксплуатации включает в себя работы по внедрению программного обеспечения в эксплуатацию, в том числе конфигурирование рабочих мест, обучение персонала, локализация проблем эксплуатации и устранение причин их возникновения, модификация программного обеспечения в рамках установленного регламента и подготовка предложений по модернизации системы.

Каждый процесс характеризуется определенными задачами и методами их решения, а также исходными данными и результатами.

Жизненный цикл программного обеспечения носит, как правило, итерационный характер, то есть реализуются этапы, начиная с самых ранних, которые циклически повторяются в соответствии с изменением требований внешних условий и введением ограничений.

**4)Этапы разработки программного обеспечения**

Типовой проект включает в себя следующие этапы разработки программного обеспечения:

* анализ требований к проекту;
* проектирование;
* реализация;
* тестирование продукта;
* внедрение и поддержка.

Анализ требований к проекту

На этом этапе формулируются цели и задачи проекта, выделяются базовые сущности и взаимосвязи между ними. То есть, создается основа для дальнейшего проектирования системы.

В рамках данного этапа не только фиксируются требования заказчика, но и проводится их формирование – клиентам подбирается оптимальное решение их проблем, определяется необходимая степень автоматизации, выявляются наиболее актуальные для автоматизации бизнес-процессы.

При анализе требований определяются сроки и стоимость разработки ПО, формируется и подписывается ТЗ на разработку программного обеспечения.

Проектирование

На основе предыдущего этапа проводится проектирование системы. Эта методология проектирования соединяет в себе объектную декомпозицию, приемы представления физической, логической, а также динамической и статической моделей системы.

Во время проектирования разрабатываются проектные решения по выбору платформы, где будет функционировать система языка или языков реализации, назначаются требования к пользовательскому интерфейсу, определяется наиболее подходящая СУБД. Разрабатывается функциональная спецификация ПО: выбирается архитектура системы, оговариваются требования к аппаратному обеспечению, определяется набор орг. мероприятий, которые необходимы для внедрения ПО, а также перечень документов, регламентирующих его использование.

Реализация

Данный этап разработки программного обеспечения организован в соответствии с моделями эволюционного типа жизненного цикла ПО. При разработке применяются экспериментирование и анализ, строятся прототипы, как целой системы, так и ее частей. Прототипы дают возможность глубже вникнуть в проблему и принять все необходимые проектные решения еще на ранних этапах проектирования. Такие решения могут затрагивать разные части системы: внутреннюю организацию, пользовательский интерфейс, разграничение доступа и т.д. В результате этапа реализации появляется рабочая версия продукта.

Тестирование продукта

Тестирование тесно связано с такими этапами разработки программного обеспечения как проектирование и реализация. В систему встраиваются специальные механизмы, которые дают возможность производить тестирование системы на соответствие требований к ней, проверку оформления и наличие необходимого пакета документации.

Результатом тестирования является устранение всех недостатков системы и заключение о ее качестве.

Внедрение и поддержка

Внедрения системы обычно предусматривает следующие шаги:

* установка системы,
* обучение пользователей,
* эксплуатация.

К любой разработке прилагается полный пакет документации, который включает в себя описание системы, руководства пользователей и алгоритмы работы.

. **5) Модели разработки ПО (каскадная)**

**Каскадная модель** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *waterfall model*, иногда переводят как **модель «Водопад»**) — [модель](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)&action=edit&redlink=1) процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки. В качестве источника названия часто указывают статью, опубликованную У. У. Ройсом (*W. W. Royce*) в [1970 году](https://ru.wikipedia.org/wiki/1970_%D0%B3%D0%BE%D0%B4); при том, что сам Ройс использовал [итеративную модель разработки](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0).

В исходной каскадной модели следующие фазы шли в таком порядке:

1. Определение требований
2. Проектирование
3. Конструирование (также «реализация» либо «кодирование»)
4. Воплощение
5. Тестирование и отладка (также «***верификация***»)
6. Инсталляция
7. Поддержка

****

**7)Прямой, обратный и дополнительный код**

**Прямой код** числа это представление беззнакового двоичного числа. Если речь идет о машинной арифметике, то как правило на представление числа отводится определенное ограниченное число разрядов. Диапазон чисел, который можно представить числом разрядов n равен 2^n

**Обратный код** числа, или **дополнение до единицы** (**one’s complement**) это инвертирование прямого кода (поэтому его еще называют **инверсный код**). То есть все нули заменяются на единицы, а единицы на нули.

**Дополнительный код** числа, или **дополнение до двойки** (**two’s complement**) это обратный код, к младшему значащему разряду которого прибавлена единица

**8) Понятие и классификация операционных систем**

Операционная система (ОС) – это комплекс программного обеспечения, основная задача которого обеспечивать возможность рационального использования оборудования компьютера наиболее удобным для пользователя образом.

Системное программное обеспечение – комплекс программ, способствующих функционированию и разработке прикладных программ. Таким образом, операционная система является фундаментальным компонентом системного программного обеспечения.

***Классификация ОС***

Существует несколько подходов для классификации операционных систем. Можно отметить следующие критерии классификации:

– реализация многозадачности.

*По числу одновременно выполняемых задач операционные системы могут быть разделены на два класса:*

1 - многозадачные (Unix, OS/2, Windows), полностью реализует мультипрограммный режим;

2 - однозадачные (например, MS-DOS).

– поддержка многопользовательского режима.

*По числу одновременно работающих пользователей ОС можно разделить на:*

1 - однопользовательские (MS-DOS, Windows 3.x);

2 - многопользовательские (Windows NT, Unix), характеризуются наличием у механизмов защиты персональных данных каждого пользователя.

– многопроцессорная обработка.

*По этому критерию ОС делятся на:*

1 - однопроцессорные;

2 - многопроцессорные, характеризуются поддержкой мультипроцессирования и более сложными алгоритмами управления ресурсами (Linux, Solaris, Windows NT и в ряде других). Многопроцессорные системы состоят из двух или более центральных процессоров, осуществляющих параллельное выполнение команд.

*Многопроцессорные ОС делятся на:*

1 - симметричные, в которых на каждом процессоре функционирует одно и то же ядро и задача может быть выполнена на любом процессоре, то есть обработка полностью децентрализована;

2 - асимметричные, в которых процессоры неравноправны, т.е. существует главный процессор (master) и подчиненные (slave), загрузку и характер работы которых определяет главный процессор.

**9)Понятие и классификация драйверов**

|  |
| --- |
| **Классификация драйверов**  **Дра́йвер** — [компьютерное программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), с помощью которого другое программное обеспечение ([операционная система](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)) получает доступ к [аппаратному обеспечению](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) некоторого устройства.  По расположению в стеке драйверов:  Драйверы высшего уровня -- получают запросы от пользовательского приложения и взаимодействуют с нижестоящими драйверами;  Промежуточные драйверы -- получают запросы от вышестоящих драйверов и взаимодействуют с нижестоящими драйверами;  Драйверы низшего уровня -- получают запросы от вышестоящих драйверов, осуществляют конечную обработку пакетов запросов.  **10)Понятие, функции и классификация утилит**  Утилиты - программы, служащие для выполнения вспомогательных операций обработки данных или обслуживания компьютеров и расширяющие стандартные возможности оборудования и операционных систем, выполняющие узкий круг специфических задач. Утилиты предоставляют доступ к возможностям (параметрам и установкам), недоступным без их применения, либо делают процесс изменения некоторых параметров проще, т. е. автоматизируют его. Утилиты используются для:  • мониторинга показателей датчиков и производительности оборудования — мониторинг температур процессора, видеоадаптера; чтение S.M.A.R.T. жёстких дисков;  • управления параметрами оборудования — ограничение максимальной скорости вращения CD-привода; изменение скорости вращения вентиляторов.  • контроля показателей — проверка ссылочной целостности; правильности записи данных.  • расширения возможностей — форматирование и/или переразметка диска с сохранением данных, удаление без возможности восстановления  Классификация утилит по связи с ОС:  - независимые утилиты (не требующие наличия установленной ОС для своей работе)  - системные утилиты (входят в состав ОС или требуют её наличия) .  **11.Проблемы разработки программного обеспечения**  Тенденции развития современных информационных технологий определяют постоянное возрастание сложности ПО, создаваемого в различных областях экономики. Современные крупные проекты ИС характеризуют, как правило, следующие особенности:  • сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующая тщательного моделирования и анализа данных и процессов;  • наличие совокупности тесно взаимодействующих компонентов (подсистем), имеющих локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач, и приложений аналитической обработки (поддержки принятия решений), использующих нерегламентированные запросы к данным);  • отсутствие полных аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;  • необходимость интеграции существующих и вновь разрабатываемых приложений;  • функционирование в неоднородной среде на нескольких аппаратных платформах;  • разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;  • значительная временная протяженность проекта, обусловленная, с одной стороны, ограниченными возможностями коллектива разработчиков и, с другой стороны, масштабами организации-заказчика и различной степенью готовности отдельных ее подразделений к внедрению ИС.  Большинство современных программных систем объективно очень сложны. Эта сложность обуславливается многими причинами, главной из которых является *логическая сложность решаемых*ими*задач.*  Пока вычислительных установок было мало, и их возможности были ограничены, ЭВМ применяли в очень узких областях науки и техники, причем, в первую очередь, там, где решаемые задачи были хорошо детерминированы и требовали значительных вычислений. В наше время, когда созданы мощные компьютерные сети, появилась возможность переложить на них решение сложных ресурсоемких задач, о компьютеризации которых раньше никто, и не думал. Сейчас в процесс компьютеризации вовлекаются совершенно новые предметные области, а для уже освоенных областей усложняются уже сложившиеся постановки задач.  Дополнительными факторами, увеличивающими сложность разработки программных систем, являются :  •сложность формального определения требований к программным системам;  •отсутствие удовлетворительных средств описания поведения дискретных систем с большим числом состояний при недетерминированной последовательности входных воздействий;  •коллективная разработка;  •необходимость увеличения степени повторяемости кодов.  **12.Системы счисления (история создания, классификация)**  **Система счисления** - это совокупность приемов и правил для обозначения и именования чисел.  В древности людям приходилось считать на пальцах. Кроме пальцев считать нужно было много предметов, к счету привлекали больше участников. Один считал единицы, второй - десятки, третий - сотни. Очевидно, такой счет лег в основу системы счисления, принятой почти у всех народов, она называется десятичной системой. Счет с основанием десять применяли и у восточных славян. **В** древнем Вавилоне, где математика была очень высоко развита, существовала весьма сложная шестидесятеричная система счисления. В наше время мы тоже используем эту систему. Например: 1 час=60 минут; 1 минута=60 секунд.  Самой древней из пальцевых систем счисления считается пятеричная. Эта система зародилась, и наибольшее распространение получила в Америке. Ее создание относится к эпохе, когда человек считал по пальцам одной руки. До последнего времени у некоторых племен пятеричная система счисления сохранилась еще в чистом виде.  **Классификация систем счисления.**  Все системы делятся на позиционные и непозиционные.  В **непозиционных**системах каждая цифра имеет свой вес и ее значение не зависит от положения в числе — от позиции. Пример — римская система.  В **позиционных**системах значения цифр зависят от их положения (позиции) в числе.  Так, например, человек привык пользоваться десятичной позиционной системой — числа записываются с помощью 10 цифр. Самая правая цифра обозначает единицы, левее — десятки, ещё левее — сотни и т.д.  **13.Системная шина. Магистрально - модульный принцип**  Магистраль (системная шина) включает в себя три многоразрядные шины (рис.2): шину данных, шину адреса и шину управления.  **Шина данных.** По этой шине данные передаются между устройствами. Например, считанные из оперативной памяти данные могут быть переданы процессору для обработки, а затем полученные данные могут быть отправлены в оперативную память, или в устройство вывода. Данные могут передаваться от устройства к устройству в любом направлении.  **Шина адреса.** Выбор устройств или ячейки памяти, куда пересылаются или откуда считываются данные по шине данных, производит процессор. Каждое устройство или ячейка памяти имеют свой адрес. Адрес передается по адресной шине, причем сигналы по ней передаются в одном направлении от процессора к памяти или устройству.  **Шина управления.** По шине управления передаются управляющие сигналы, которые определяют, какую операцию нужно производить: записи или считывания информации, синхронизации обмена между устройствами и т.д.  **14.Арифметико – логическое устройство (назначение, классификация)**  **Арифме́тико-логи́ческое устро́йство** (АЛУ) — блок [процессора](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80), который под управлением [устройства управления](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) (УУ) служит для выполнения арифметических и логических преобразований (начиная от [элементарных](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F)) над данными, называемыми в этом случае [операндами](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D0%BD%D0%B4). Разрядность операндов обычно называют размером или длиной [машинного слова](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D1%81%D0%BB%D0%BE%D0%B2%D0%BE).  Арифметико-логическое устройство в зависимости от выполнения функций можно разделить на две части:   1. микропрограммное устройство (устройство управления), задающее последовательность микрокоманд (команд); 2. операционное устройство, в котором реализуется заданная последовательность микрокоманд (команд).   Назначение - обработка информации (операции +, -, <<, >>, и т.д.) и логические операции. Кроме того в малых и средних машинах, в которых нету отдельного БУО, связ. с формированием действительных адресов в АЛУ выполняется действия адресной арифметики или действия связанные с преобразованием адресов.  **15.Основные логические функции (назначение, принцип работы)**  Современная вычислительная техника строится на основе цифровых микросхем. При этом сами цифровые микросхемы реализуются на базе простейших логических функций:  Информация (данные, машинные команды и т. д.) в компьютере представлена в двоичной системе счисления, в которой используется две цифры – 0 и 1. Электрический сигнал, проходящий по электронным схемам и соединительным проводникам (шинам) компьютера, может принимать значения 1 (высокий уровень электрического напряжения) и 0 (низкий уровень электрического напряжения) и рассматривается как импульсный сигнал, который математически может быть описан в виде двоичной переменной, принимающей также значения 0 или 1. , широко используются логические функции и логические операции с двоичными переменными, которые называются также логическими переменными.   * ИЛИ – логическое сложение (**дизъюнкция**) – **OR**; * И – логическое умножение (**конъюнкция**) – **AND**; * НЕ – логическое отрицание (**инверсия**) – **NOT**.     **16.Интерфейс. Работа с диалогами**  Пользовательский интерфейс - совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основа взаимодействия - диалоги.  Диалог - регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи: обмен информацией и координация действий. Каждый диалог состоит из отдельных процессов ввода-вывода, которые физически обеспечивают связь пользователя и компьютера.  Обмен информацией осуществляется передачей сообщений и управляющих сигналов.  Диалог должен подчиняться определённым правилам:   * · участники диалога должны понимать язык друг друга; * · порядок высказываний в диалоге строго определён: "вопрос - ответ"; * · очередное высказывание должно учитывать как общий контекст диалога, так и последнюю информацию, полученную от собеседника.  Типы диалога Тип диалога определяет, кто из "собеседников" управляет процессом обмена информацией.  Различают два типа диалога:  управляемые программой;  управляемые пользователем.  Диалог, управляемый программой, предусматривает наличие жесткого, линейного или древовидного, т.е. включающего возможные альтернативные варианты, сценария диалога, заложенного в программное обеспечение. Такой диалог обычно сопровождают большим количеством подсказок, которые уточняют, какую информацию необходимо вводить на каждом шаге.  Диалог, управляемый пользователем, подразумевает, что сценарий диалога зависит от пользователя, который применяет систему для выполнения необходимых ему операций. При этом система обеспечивает возможность реализации различных пользовательских сценариев.  **17.Основные характеристики монитора**  Компьютерный монитор — устройство, предназначенное для визуального отображения графической и текстовой информации.  К основным характеристикам мониторов относятся:  *Контрастность.*  Этот параметр показывает разницу между самым светлым и наиболее темным участком поверхности дисплея. Чем больше его величина, тем монитор считается более качественным.  *Яркость.*  Параметр определяет наибольшую удельную светимость отображающей поверхности, а ее единицей измерения служит 1 нит, равный отношению 1 кд к 1 кв. м.  *Разрешение.*  Это один из важнейших параметров, на который обращают внимание, выбирая компьютерный монитор. Он определяет число всех пикселей, формируемых отображаемую картинку. Чем выше разрешение, тем четче будет изображение, выводимое на экран монитора.  *Частота горизонтальной развертки.*  Этот параметр измеряется в герцах и показывает частоту отображения изображение на экран монитора.  *Частота вертикальной развертки.*  Параметр характеризует наибольшее количество горизонтальных строк, выводимых электронным лучом на экране за ед. времени.  **18.История развития и поколения ЭВМ** Первое поколение Элементная база ЭВМ первого поколения (конец 40-х — середина 50-х) — электронные схемы, построенные с использованием радиоламп. Пример ЭВМ первого поколения — БЭСМ-1(1950г.) — содержала около 7000 радиоламп, выполняла около 8000 арифметических и логических операций в секунду. Задачи, решаемые этой машиной: эксперименты по переводу научно-технических текстов с английского на русский язык, шахматные задачи. БЭСМ-1 занимала большой зал, где требовалась вентиляция, стабильные источники питания, смена ламп. Примеры других ЭВМ первого поколения: ЦВМ Стрела, Минск-1, М-20 и др. Второе поколение Основа элементной базы ЭВМ второго поколения (сер. 50-х — 70-х гг.) — полупроводниковые элементы (транзисторы, диоды). Полупроводниковые приборы позволили резко увеличить скорость выполнения операций, и быстродействие достигло нескольких миллионов операций в секунду. В ЭВМ второго поколения использовались ЗУ с объемом памяти на сотни тысяч машинных слов. ЭВМ второго поколения: БЭСМ-2, БЭСМ-3, БЭСМ-3М, машины серии УРАЛ, МИР, НАИРИ и др. Третье поколение Третье поколение ЭВМ связывают с использованием интегральных схем (ИС) среднего уровня интеграции. На одном кристалле реализуются достаточно сложные логические функции, и из них собираются сложные узлы машин. Примеры ЭВМ третьего поколения: IBM — 360, УРАЛ-12, УРАЛ-13, УРАЛ-14. Самые большие семейства машин третьего поколения — это ЕС ЭВМ (единой серии). Быстродействие этих машин: сотни тысяч операций в секунду, объем памяти — сотни тысяч машинных слов. Четвертое поколение Элементная база — большие интегральные схемы. В этих машинах используются элементы ОЗУ, микропроцессоры. К этому поколению относятся современные машины, мультипроцессорные системы и персональные компьютеры. Пятое поколение ЭВМ пятого поколения будут отличаться организацией искусственного интеллекта.  **19.Классификация**ЭВМ **Электронная вычислительная машина (ЭВМ)** или **компьютер** — это устройство, выполняющее операции ввода данных, их сохранение и обработку по определенной программе, вывод полученных результатов в форме, пригодной для восприятия человеком. Классификация эвм по назначению Универсальные ЭВМ – для решения широкого круга задач.  Проблемно-ориентированные ЭВМ – служат для решения более узкого круга задач связанных, как правило, с управлением технологическими объектами, регистрацией, накоплением и обработкой относительно небольших объемов данных.  Специализированные ЭВМ – используются для решения узкого круга задач или реализации строго определенной группы функций. Классификация эвм по принципу действия Цифровые вычислительные машины (ЦВМ) – вычислительные машины дискретного действия, работают с информацией, представленной в дискретной (цифровой) форме. ЦВМ отличаются высокой точностью вычисления и удобством хранения информации.  Аналоговые вычислительные машины (АВМ) – вычислительные машины непрерывного действия, работают с информацией, представленной в непрерывной (аналоговой) форме, т.е. в виде непрерывного рядя значений какой-либо физической величины. АВМ просты и удобны в эксплуатации, характеризуются высоким быстродействием и относительно высокой тонностью.  Гибридные вычислительные машины (ГВМ) – вычислительные машины комбинированного действия, работают с информацией, представленной в цифровой и аналоговой форме. Они совмещают преимущества ЦВМ и ГВМ.  ***По размерам и функциональным:***   * *сверхмалые (микро ЭВМ*) обязаны своим появлением изобретению микропроцессора, наличие которого первоначально служило определяющим признаком микро ЭВМ, хотя сейчас микропроцессоры используются во всех без исключения классах ЭВМ; * *малые (мини-ЭВМ)* используются чаще всего для управления технологическими процессами; * *большие ЭВМ* чаще всего называют мэйнфреймами (mainframe). Основные направления эффективного применения мэйнфреймов – это решение научно-технических задач, работа в вычислительных системах с пакетной обработкой информации, работа с большими базами данных, управление вычислительными сетями и их ресурсами; * *сверхбольшие (суперЭВМ)*– мощные многопроцессорные вычислительные машины быстродействием десятки миллиардов операций в секунду и объемом оперативной памяти десятки Гбайт.   **20.Классы архитектур вычислительных систем**  **Архитектура ВС***—*совокупность характеристик и параметров, определяющих функционально-логическую и структурную органи­зацию системы. существует четыре основных ар­хитектуры ВС:  • одиночный поток команд - одиночный поток данных (ОКОД),  • одиночный поток команд — множественный поток данных (ОКМД)  • множественный поток команд — одиночный поток данных (МКОД),  • множественный поток команд — множественный поток данных (МКМД),  **Архитектура ОКОД**охватывает все однопроцессорные и одно­машинные варианты систем, т.е. с одним вычислителем. Все ЭВМ классической структуры попадают в этот класс. Здесь параллелизм вычислений обеспечивается путем совмещения выполнения операций отдельными блоками АЛУ, а также параллельной работы устройств ввода-вывода информации и процессора.  **Архитектура ОКМД**предполагает создание структур вектор­ной или матричной обработки. Системы этого типа обычно строят­ся как однородные, т.е. процессорные, элементы, входящие в систему, идентичны, и все они управляются одной и той же последова­тельностью команд. Но каждый процессор обрабатывает свой поток данных. Под эту схему хорошо подходят задачи обработки матриц или векторов (массивов), задачи решения систем линейных и нелинейных, алгебраических и дифференциальных уравнений, зада­чи теории поля и др. В структурах данной архитектуры желательно обеспечивать соединения между процессорами, соответствующие реализуемым математическим зависимостям. Эти свя­зи напоминают матрицу, в которой каждый процессорный элемент связан с соседними.  **Архитектура МКОД**предполагает построение свое­образного процессорного конвейера, в котором результаты обработ­ки передаются от одного процессора к другому по цепочке. Выгоды такого вида обработки понятны. Прототипом таких вычислений мо­жет служить схема любого производственного конвейера. В совре­менных ЭВМ по этому принципу реализована схема совмещения опе­раций, в которой параллельно работают различные функциональные блоки, и каждый из них делает свою часть в общем цикле обработки команды.  **Архитектура МКМД**предполагает, что все процессоры систе­мы работают по своим программам с собственным потоком команд. В простейшем случае они могут быть автономны и независимы. Такая схема использования ВС часто применяется на многих круп­ных вычислительных центрах для увеличения пропускной способно­сти центра. Больший интерес представляет возможность согласован­ной работы ЭВМ (процессоров), когда каждый элемент делает часть общей задачи. Общая теоретическая база такого вида работ прак­тически отсутствует. Но можно привести примеры большой эффективности этой модели вычислений. Подобные системы могут быть многомашинными и многопроцессорными. |