ЛЕКЦИЯ 1. АВТОМАТИЗАЦИЯ. ОПРЕДЕЛЕНИЯ, ТЕРМИНОЛОГИЯ И НАПРАВЛЕНИЯ АПП

Одним из основных направлений деятельности человека является совершенствование процессов производства с целью облегчения тяжёлого физического труда и повышение эффективности процесса в целом – это направление может реализоваться через **автоматизацию производственных процессов**.

Итак, **целью АПП** является:

- повышение производительности;

- повышение качества изделий;

- улучшений условий труда.

Цель порождает вопросы, что и как автоматизировать, целесообразность и необходимость автоматизации и др. задачи.

Как известно **технологический процесс состоит из трёх основных частей:**

- рабочего цикла – основной технологический процесс;

- холостых ходов- вспомогательные операции;

- транспортно-накопительных операций.

**Автоматизация** – направление развития производства, характеризуемое освобождением человека не только от мускульных усилий, для выполнения тех или иных движений, но и от оперативного управления, механизмами, выполняющими эти движения.

Автоматизация может быть **частичной** и **полной**.

**Частичная автоматизация** – автоматизация части операции по управлению производственным процессом при условии, что остальная часть всех операций выполняется автоматически (управление и контроль человеком). Примером может служить – автоматическая линия (АЛ), состоящая из нескольких станков- автоматов и имеющих автоматическую межоперационную транспортную систему. Управление линий осуществляется одной управляющей ЭВМ.

**Полная автоматизация** – характеризуется автоматическим выполнением всех функций для осуществления производственного процесса без непосредственного вмешательства человека в работу оборудования. В обязанности человека входят настройка машины или группа машин, включение и контроль.

Пример: автоматический участок или цех.

**Промышленный РС: блок сопряжения между устройством смены рулонов и печатным станком**

Хороший пример «современной» промышленной системы – устройство смены рулонов для ротационных печатных машин. Последнее высокоэффективное устройство смены рулонов фирмы Koenig& Bauer-Albert AG установило новый стандарт в быстродействии и простоте использования.

**Автоматическая эксплуатационная диагностика**

В установке имеются различные опции: они включают диагностическую систему для оценки эксплуатационных данных, из которых может следовать необходимость в регламентных работах, базу данных для бумажных рулонов, соединение к весовому оборудованию, порт модема для удаленной диагностики.

-**Прочный и надежный**

Все оборудование, необходимое для работы и управления, встраивается в боковую стенку Postomat RC. Для этого метра в устройстве мены рулонов требуются компоненты автоматизации и визуализации с повышенной стойкостью к ударам и вибрации. Это было решающим фактором, который привел к решению использовать IPC 5000 от B&R. Этот новый промышленный контроллер PC выполняет и вывод всей эксплуатационной информации, и управление всеми параметрами установки. Он оборудован двумя платами сети ARCHEN – первая устанавливает соединение с котроллером, а вторая соединяет котроллер с управляющей станцией, которая наблюдает за работой в целом. Модульная конструкция системы позволяет одновременно…

-**Простота использования**

**Гибкие автоматизированные системы**

**Гибкие автоматизированные системы** – предназначены для автоматизации тех. процессов со сменным объектом производства, в том числе для единичного и мелкосерийного производства. Включает в себя основное и вспомогательное оборудование, работающее от одной СУ.

Использование ГАП в Японии и Европе даже в автоматизированном режиме и при недостаточном опыте работы приводит увеличение отдачи станков на 80-200%, к сокращению продолжительности из обслуживания на 60-70%, что в свою очередь уменьшает время производственного процесса и стоимость живого труда примерно на 80%.

Создание ГАП является очень сложным и многоэтажным процессом, поэтому он по возможности автоматизируется, в результате образуется системное окружение ГАП.

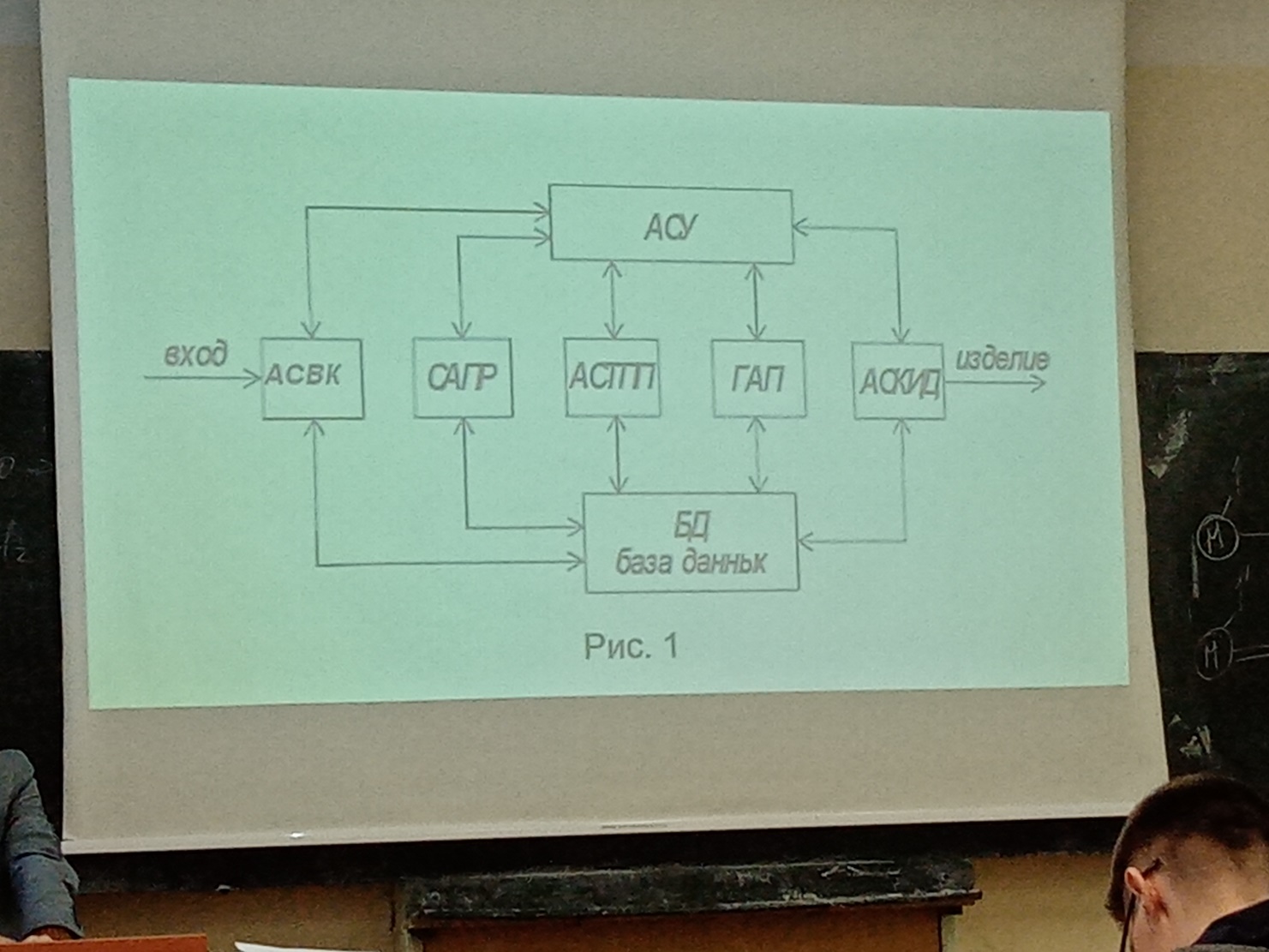


Рис. 1

Создание такой сложной системы (рис. 1) должно протекать поэтапно, тем более что эксплуатация её обуславливает значительные изменения в структуре предприятия. Начальным этапом является разработка подсистемы АСВК (Входной контроль), САПР (NASTRAN, ADAMS), АСТПП (технология), ГАП, АСКИД (Контроль и измерение детали). Основным элементов при создании такой подсистемы – **унификация банка данных (БД)** для информационного обеспечения функционирования подсистемы. Унификация банка данных предполагает прежде всего применение групповой технологии, технологических процессов обработки деталей, объединенных рядом технологических и конструктивных признаков.

В состав ГАП включены подсистемы: обработки, манипулирования, контроля и управления. Проектирование, разработка и внедрение осуществляется в установленном нормами и правилами порядке.

**Система автоматического управления средствами автоматизации (общие понятия)**

**Классификация СУ**

Системы автоматического управления обеспечивают работу оборудования по заранее заданной программе. СУ автоматами могут различаться по **признакам**, а именно:

- по принципу синхронизации;

- по степени централизации управления;

- по методу воздействия;

-по числу управляемых координат;

- по виду программоносителя;

- по наличию или отсутствию обратной связи.

**СУ бывают централизованными и децентрализованными.**

У **централизованной** СУ весь технологический цикл управляется с центрального командного устройства ( контроллера, пульта, ЭВМ) независимо от действия и положения исполнительных органов. Особенности централизованной СУ:

- продолжительность рабочего цикла для каждого исполнительного органа, является как правило величиной постоянной;

- простота схем управления;

- надёжность в работе;

- удобство обслуживания и наладки;

- необходимость иметь дополнительные предохранительные устройства, т.к. команды с центрального командного пульта подаются вне зависимости от действия и положения исполнительных рабочих органов.

Особенности **децентрализованной** системы управления:

- осуществляют управление при помощи датчиков (чаще всего конечных выключателей и путёвых переключателей) включаемых движущимися исполнительными рабочими органами с помощью упоров;

- исполнительные органы связаны между собой так, что каждое последующее движение одного может происходить после окончания движения предыдущий;

- отсутствие сложной блокировки, т.к. команды подаются только после окончания предыдущей операции;

- датчики расположены в рабочей зоне оборудования и нередко выходят из строя из—попадания стружки, пыли, масла и выдают неправильную информацию об объекте.

Наиболее важным и характерным признаком любой СУ программного управления является **способ задания программы обработки или движения**, т.е. от программоносителя:

- СУ копирами;

- СУ с электронными кулачками (наиболее надёжен);

- система циклового программного управления (упоры, коммутаторы, штекерное табло);

- система числового программного управления (перфокарта, перфолента, магнитные ленты, диски, FLASH-память).

В зависимости от решающих технологических задач системы ЧПУ делятся на **четыре вида**:

- позиционные управления положением;

- прямоугольные системы управления рабочими перемещениями;

- непрерывные контурные СУ;

- комбинированные (контурно-позиционные) СУ.

ЛЕКЦИЯ 2. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ АС. СТАНДАРТ ОРС.

ОРС- аббревиатура от OLE for Process Control, или OLE для Управления Процессами. Ключевыми словами для понимания изложения являются технология Microsoft OLE (Object Linking and Embedding – связывание и встраивание объектов) и интеграция..

**COM/DCOM**

COM Component Object Model или Модель Составных Объектов – и ёё сетевое расширение DCOM – Districted COM, или Распределённая COM – это технология, введённая первоначально Microsoft для интеграции между различными офисными приложениями в windows. Интеграция подразумевала использование объектов одного приложения, например, таблицы excel, в другом приложении, например, в редакторе Word. Всё это известно под аббревиатурой OLE (Object Linking and Embedding – Связывание и Встраивание Объектов). Начиная с версии OLE 2.0, в её основу была положена модель COM.

Модель COM оперирует объектами, очень похожими на объекты в объектно-ориентированных языках программирования типа С++. Но сама технология COM не является языком программирования. Она только регламентирует поведение своих объектов. **Нам нужно знать, что объект может быть создан, после чего он предоставляет свою функциональность вызвавшему процессу, а после использования - уничтожен.**

**Интерфейс в COM объединяет группу взаимосвязанных функций, предоставляемых объектов. Главная особенность интерфейсов COM заключается в их «публичности». Интерфейсы используются после того, как они «опубликованы», и после этого их нельзя изменять никогда.**

**Доступ к объектам.** Именно интерфейс, вернее указатель на него, является тем, с чем работает вызывающий процесс (читай программист). Объект может предоставлять несколько интерфейсов. Чтобы получить указатель на любой интерфейс, нужно воспользоваться функций **QueryInterface** Указатель на этот интерфейс передаётся инициирующему процессу при создании объекта.

**Обмен в COM. Объект COM является пассивной стороной. Он лишь предоставляет через интерфейс свои функции.** В этом смысле употребляется термин COM-сервис. **Запрашивающая программа, соответственно называется COM-клиент**. Но это не исключает того, что обе программы одновременно могут являться и COM-серверами, и COM-клиентами. **OPC-сервер может поставлять данные «по подписке», то есть сам инициализировать обмен с OPC-клиентом при их обновлении.**

**Регистрация**. Чтобы создать объект, нужно знать, где он находится. В Windows для этого используется регистрация объектов в системном реестре. При этом каждый «COM-предмет» регистрации имеет уникальный в полном смысле этого слова идентификатор, называемый GUID ( Globally Unique Identifier –глобально уникальный идентификатор). Присваивает идентификаторы своим «COM-детищам» их создатель, используя, например, программу GUIDGEN.EXE.. Регистрация делает создание или унич…..

Отказ ПО не приводит к разрушению или поломке прогрммного элемента. Отказы По не связаны с физическим износом

**Обслуживание объектов.** Вопросы, затрагиваемые здесь, очень важны для понимания всего излагаемого. Объекты Com должны быть достаточно независимыми. Они зачастую, если не сказать в большинстве случаев, находятся вне программы COM-клиента, а могут быть запущены даже на другом компьютере. Это имеет принципиальные последствия. Даже на одном компьютере разные приложения Windows функционируют в своих собственных адресных пространствах. Это означает, что требуется кто-то, кто будет передавать вызову из одного процесса в другой. Даже простое создание или уничтожение объекта в другом адресном пространстве вовсе не тривиальное дело.

**Удалённый объекты.** Без сетевых решений разговора об интеграции в настоящее время можно даже и не начинать. В СОМ по этому поводу существует DCOM – расширение COM, позволяющее «добираться» до объектов на других компьютерах. Существенно то, что с точки зрения программирования, ничего не меняется: **DCOM – это системный сервис, делающий COM прозрачным в локальных сетях. И это четвёртый шаг к интеграции. Но с тем же очевидным недостатком: DCOM должен присутствовать в операционной системе.**

**Предоставление объектов.** Чтобы использовать объект, необходимо знать, как он устроен, вернее, как устроены его интерфейсы. Для этого они должны быть опубликованы. Например, в виде официальной документации. Или в виде стандарта.

**Использование объектов.** Технология COM здесь предусматривает две возможности. **Либо вы программируете на C++и тогда для описания интерфейсов используете в проекте предоставляемые с документацией h- и с-файлы. В этом случае говорят об custom-интерфейсе (не путать с COM-интерфейсом!). Либо вы используете для скриптовых запросов так называемую автоматизацию (OLE Automation).** В этом случае для доступа к функциям объекта используется специальный COM-интерфейс IDispatch, который COM- объект в этом случае обязан поддерживать, предоставляя интерфейс Автоматизации (опять не путать с COM-интерфейсами!). При этом никакие компилируемые файлы не нужны. Но нужна так называемая библиотека типов.

**Реализация объектов.** Программирование COM – занятие не из лёгких. С помощью С-подобного языка MIDL Microsoft Interface Definition Language – язык определения интерфейсов) описываются интерфейсы. С помощью **компилятора** MIDL.EXE они преобразовываются в описанные выше файлы, в том числе и в библиотеку типов. А далее используется библиотека ATL Active Template Library – библиотека активных шаблонов), «умеющая» интегрировать эти файлы и многое другое, связанное с COM.

**OPC в свете COM**

Технология. Как уже отмечалось выше, технология OPC реализована и продолжает реализовываться по типу – разработка стандартов. OPC Foundation определяет направления, по которым ведутся разработки, издаёт по этим направлениям комитеты. Комитеты делают следующие:

* Разрабатывают спецификации COM-интерфейсов и COM-объектов;
* …..

**Спецификации.** В настоящее время имеются следующее OPC-стандарты.

**OPC Common Definitions and Interfaces** – общие для всех OPC-спецификации интерфейсы.

**Data Access Custom Interface Standard** – спецификация COM-интерфейсов для обмена оперативными данными, программирование на C++.

**Data Access Automation Interface Standard** – спецификации COM- интерфейсов для обмена оперативными данными. Программирование на языках типа Visual Basic.

**OPC Batch Custom Interface Specification** –спецификация COM-интерфейса конфигурирования оборудования, программирование на C++.

**OPC Batch Automation Interface Specification** – спецификация COM-интерфейсов для конфигурирования оборудования, программирование на языках типа Visual Basic.

**OPC Alarms and Events Interface Specification** – спецификация COM-интерфейсов для обслуживания событий (event) и нештатных ситуаций (alarm), программирование на C++.

**Historical Data Access Custom Interface Standard** – спецификация COM-интерфейсов для работы с хранилищами данными, программирование на C++.

**OPC Security Custom Interface** -спецификация COM-интерфейсов для обработки прав доступа к данным, программирование на C++.

**OPC-сервер (потребители «снизу»).** Кто же использует OPC? Первая категория – производители оборудования автоматизации, или OEM (Original Equipment Mставщик комплексного оборудования). Предполагается, что тот….

Что должен сделать производитель, если он задался целью обеспечить свой продукт стандартным интерфейсом? Он должен получить нужную спецификацию и прилагаемые программные компоненты. Затем он должен изучить COM-интерфейсы тех COM-объектов…..

**OPC Data Access.** Стандарт DA предназначен для поставки оперативных данных от оборудования и/или к оборудованию. Для стандарта DA реализованы спецификации как Custom-интерфейса, так интерфейса Автоматизации. С точки зрения функциональных интерфейсов, последний ничем не отличается от Custom, кроме того, что не позволяет одновременно работать с несколькими OPC-серверами и добавлен упоминавшийся выше COM-интерфейс IDispatch, обязательный в OLE Automation. Это позволило OPC Foundation издать «обёртку» (wrapper) в виде dll, преобразующую один интерфейс в другой.

**Данные. *Основной единицей данных в OPC является переменная (Item). Переменная может быть любого типа, допустимого в OLE: различные целые и вещественные типы, логический тип, строковый, дата, валюта, вариантный тип и так далее. Кроме того, переменная может быть массивом.***

ЛЕКЦИЯ 3

Одной из основных частей АСУ ТП является ПО, представляющее собой совокупность взаимосвязанных и автономных программ, описаний, инструкций программиста и пользователя, тестов и т.п.

**Надёжность ПО** – есть свойство программного обеспечения своевременно выполнять в заранее указанных условиях эксплуатации вперёд установленных функций.

В самом общем случае основную функцию ПО АСУТП можно рассматривать как своевременное получение некоторого результата или решения y при переработке входной информации x из множества X.

Под x понимается контрольная информация, сигналы о состоянии технологического оборудования, команды управленческого персонала и вышестоящих АСУ и т.п.

Результат y зависит как от случайного x принадлежит X, так и от свойства ПО, носящих во многом стохастический характер. Поэтому установление каких-либо диапазонов изменения y и тем более границ допустимых или разумных результатов Y оказывается в этом случае невозможным. Вследствие этого становится затруднительной строгая качественная оценка принадлежности данного y множеству «разумных» результатов Y.

**Надёжность ПО** устанавливается по результатам работы ПО, т.е. при динамической проверке всех программ на множестве входной информации. Некорректное ПО заведомо ненадёжно, одна и корректное ПО может быть ненадёжным. Рассмотренное определение надёжности ПО базируется на понятии отказа программы, под которым понимается событие, заключающееся в появлении «неразумного» результата y принадлежит Y при x принадлежит X и нормальной работе УВМ и ТСА.

Отказы ПО:

1. Неслучайные отказы обусловлены действием компьютерных вирусов.
2. Случайные отказы наблюдаются в случайные моменты времени работы УВМ или процессора. По своим последствиям эти отказы классифицируются на.
   1. Случайные сбои программ
   2. Устойчивые отказы ПО

**Под сбоем ПО** понимают случайные события, которые заключаются в появлении «неразумного» результата y «принадлежит» Y и исчезающее при последующих прогонах (запусках) программ.

**Сбой ПО –** это самоустраняющийся отказ программы, возникающий при некоторых, возможно случайных, состояниях УВМ и информации x «принадлежит» X, наблюдаемый пользователем в случайные моменты времени и исчезающий без вмешательства программиста.

**Устойчивый отказ** – наблюдается в случайный момент процессорного времени в форме «неразумного» результата y принадлежит Y при x принадлежит X в нормальном функционировании УВМ.

**Причиной отказа ПО** служит некоторая систематическая ошибка программы, после устранения которой данный отказ исчезает, т.е. имеет место восстановление ПО.

Различают ошибки **первичного** и **вторичного типа.**

**Ошибки первичного типа** связаны с неточностями в текстах программ и возникают при подготовке носителей и документации ПО, при записях кодов на алгоритмических языках и трансляции программ на машинный язык. А также из-за неточностей алгоритмов и при неверных или некорректных постановках, решаемых на УВМ.

**Ошибки вторичного типа** во многом являются следствием первичных ошибок программ. К ним относят ошибки:

1. **Вычислительные** (неверная индексация и подсчёт временных параметров, расхождение результата ручного и машинного счёта, появление неустойчивых операций и т.п.)
2. **Логические** (пропуск логических условий, неверные краевые условия и др.)
3. **Сопряжение интерфейсов** (межмодульных, программно-технических, информационных)

**Итог:** ошибки первичного и вторичного типов порождаются на этапах разработки спецификаций на ПО: проектирования ПО: реализации программ.

**Отличия отказов ПО** при его эксплуатации от отказов технических элементов

1. Отказ ПО не приводит к разрушению или поломке программного элемента. Отказы ПО не связаны с физическим износом элемента (в частности носителя программ),
2. Отказ ПО не коррелирован с процессорным и, тем более, астрономическим временем (с процессорным временем или числом прогонов ПО программ пользователем),
3. При длительной эксплуатации ПО все его ошибки могут быть устранены и программы становятся абсолютно надёжными. Если обозначить через N(t) число не выявленных ошибок ПО в произвольный момент процессорного времени t, то формально имеет место соотношение lim N(t) = 0, справедливо при условии, что в процессе восстановления программ в них не вносятся новые ошибки.

Опыт создания и эксплуатации ПО реальному времени показывает, что **при устранении одних ошибок вносятся другие.** Поэтому, при длительной

…

Основные показатели надёжности ПО

1. **Функция ненадёжности** или **отказа**. ПО Q(t) = P (того, что отказ ПО появится до момента времени t),
2. **Функция надёжности.** ПО P(t) = P (того, что отказ ПО появится после момента времени t),
3. **Интенсивность отказов** ПО: λ(t) = dQ/dt,
4. **Средняя наработка на отказ** ПО: t = интеграл (t \* f(t) / dt) = интеграл ((P(t) \* d(t))

Программное обеспечение АСУТП состоит из …

Если программы имеют взаимные информационные связи или предназначены для получения одного результата y (вычисление одной функции), то в отношении надёжности такой программный комплекс представляет собой простую систему без избыточности и вероятность его безотказной работы равна произведению вероятностей безотказной работы каждой i-ой программы?

P(t) = П(от 1 до m) pi (t), где

m – общее число программ.

Надёжность такого ПО определяется надёжностью отказов самой «ненадёжной» программы, имеющей наибольшее значение …

Понижению интенсивности λ способствует и переход на другой более высокий ЯП. Другой путь повышения надёжности ПО связано с резервированием и введением в программную систему некоторой избыточности.

Применительно к ПО АСУТП различают **три вида резервирования**:

1. **Временное**
2. **Информационное**
3. **Программное**

Вот краткие объяснения для каждого из упомянутых терминов:

1. Типы данных — это классификация данных, используемых в программировании для указания компилятору или интерпретатору, как программист намерен использовать данные. Основные типы данных:

- Примитивные: например, int (целые числа), float (вещественные числа), char (символы), boolean (логические значения).

- Структурные: массивы, списки, структуры.

- Пользовательские: классы и структуры, определенные разработчиками.

2. Операционные системы (ОС) — это программное обеспечение, которое управляет аппаратными ресурсами компьютера и предоставляет интерфейсы для взаимодействия пользователя с системой. Примеры ОС:

- Windows

- Linux

- macOS

- Android

3. Языки высокого уровня — это языки программирования, которые предоставляют абстракции от машинного кода и удобны для написания программ. Они ближе к человеческому языку и автоматически управляют многими аспектами работы с памятью. Примеры:

- Python

- JavaScript

4. Язык низкого уровня — это язык программирования, который близок к машинному коду и обеспечивает высокую степень контроля над оборудованием. Примеры:

- Ассемблер — язык, напрямую связанный с архитектурой процессора.

- Машинный код — двоичные команды, которые процессор может выполнять напрямую.

- Java

- C++

- Fortran

5. Шлюз (Gateway) — это устройство или программа, которая соединяет две разные сети и управляет их взаимодействием. Шлюз может преобразовывать форматы данных, протоколы и может работать на разных уровнях модели OSI. Например, шлюз может соединять локальную сеть с интернетом.

6. Сетевой протокол — это набор правил, определяющих, как данные передаются по сети. Примеры сетевых протоколов:

- TCP/IP — основной набор протоколов для интернета.

- HTTP — протокол передачи гипертекста, используемый в веб-браузерах для взаимодействия с веб-серверами.

- FTP — протокол передачи файлов.

- DNS — протокол преобразования доменных имен в IP-адреса.