- РК 1. Итерационные циклы - 2 часа – 5-6 недели.

- РК 2. Матрицы и подпрограммы - 2 часа – 10-11 недели.

- РК 3. Файлы и дин. память - 2 часа – 15-16 недели.

***Паскаль*** – универсальный язык программирования высокого уровня. Поддерживает структурный и объектный подходы. Первоначально предназначен для обучения студентов, затем, в совокупности со средой программирования Turbo Pascal, стал профессиональным.

***Автор языка***: Николаус Вирт, Цюрих, Швейцария.

***Год создания языка****:* 1971 г.

В основе языка хорошо продуманные, логически стройные концепции. Язык имеет простой, но хорошо защищенный синтаксис и сравнительно ясную семантику, что упрощает обучение азам программирования.

**Синтаксис** – правила, определяющие допустимые конструкции языка. «Защищенный» синтаксис предполагает, что предложения языка строятся по правилам, которые позволяют автоматически выявлять большой процент ошибок в программах.

**Семантика** – правила, определяющие смысл синтаксически корректных предложений. Ясная или «интуитивно-понятная» семантика – семантика, позволяющая без большого труда определять смысл программы или «читать» ее.

***Среда программирования***– собранная в единую программную систему совокупность программных средств, предназначенный для разработки программных продуктов. Обычно включает: редактор текстов, компилятор языка программирования, компоновщик, отладчик, библиотеки подпрограмм и/или классов и т.п.

**Этапы создания ПО**

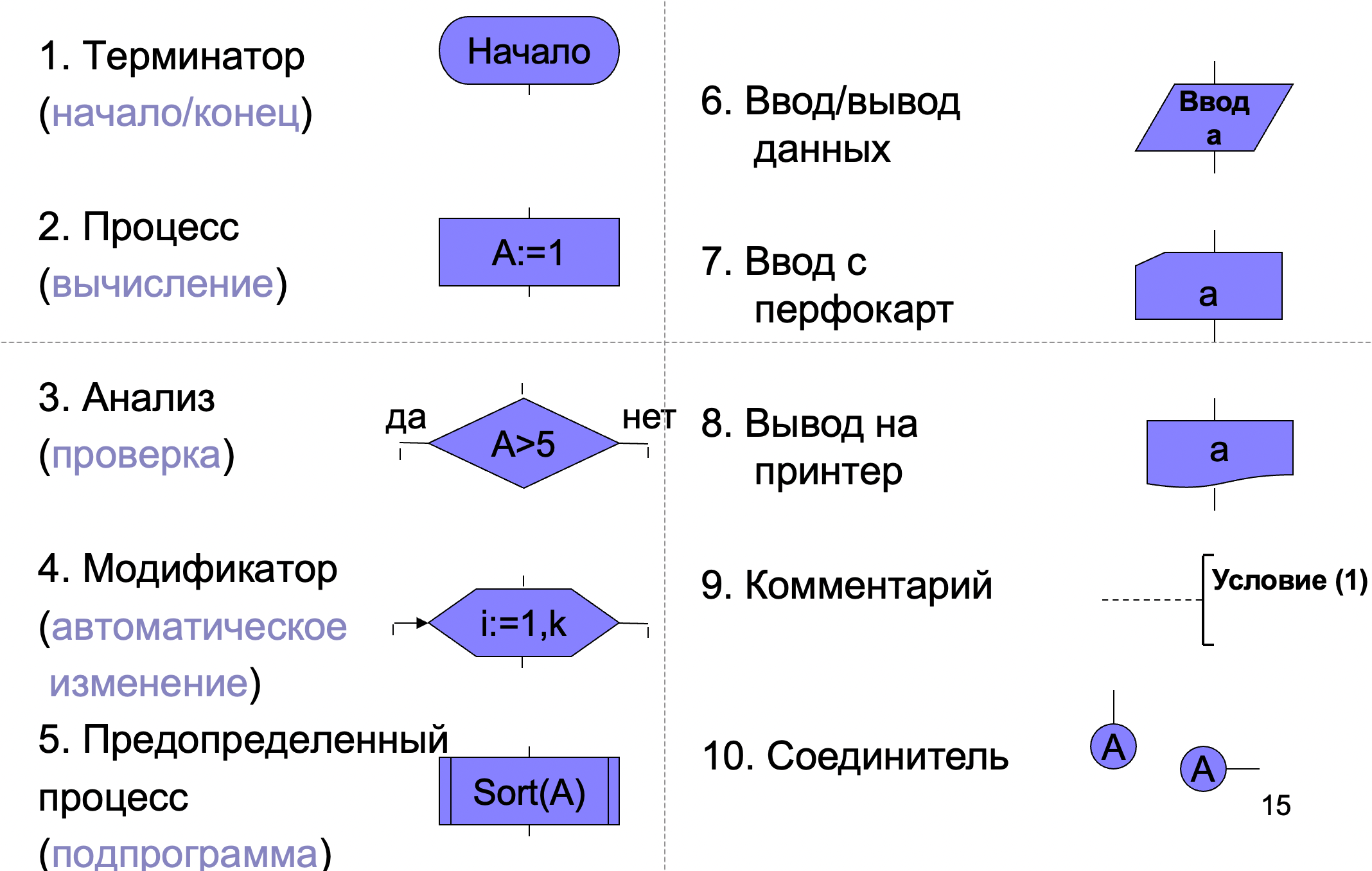
**1. Постановка задачи** – неформальное описание задачи.

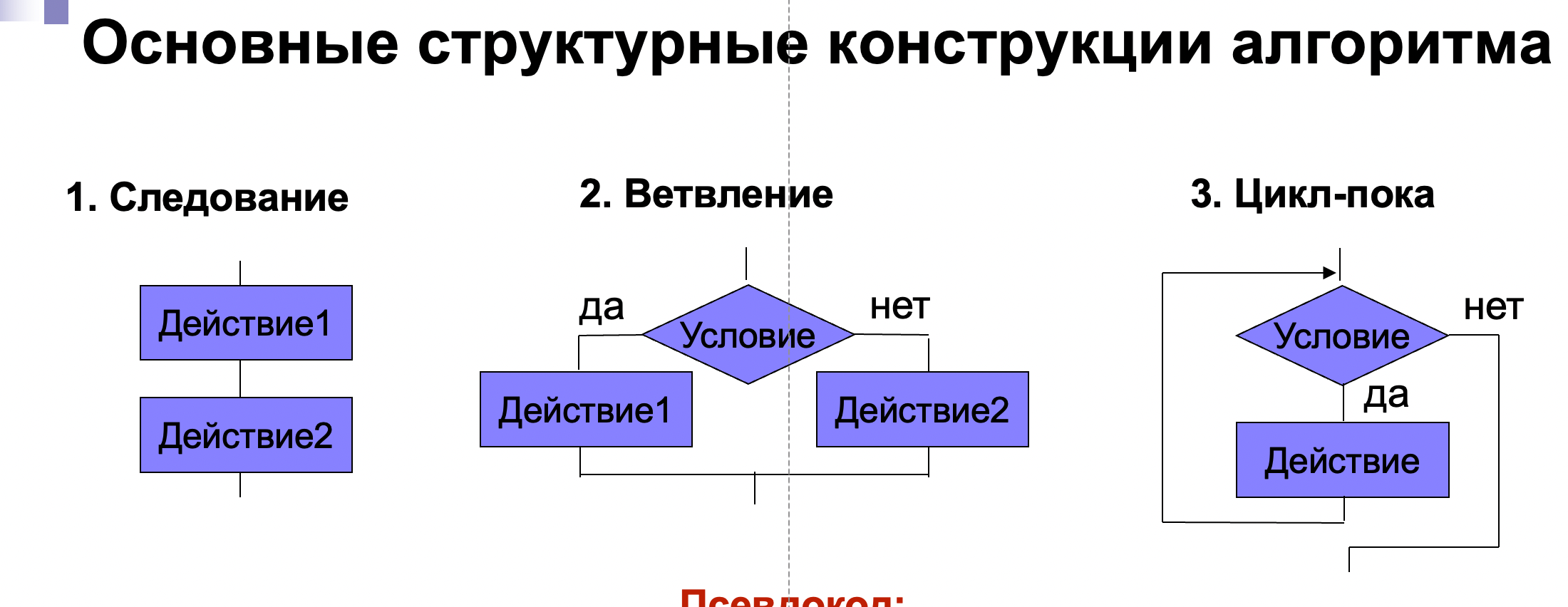
**2. Анализ и уточнение требований** – формальная постановка задачи и выбор метода решения.

**3. Проектирование** – разработка структуры программного продукта, выбор структур данных, выбор метода решения, разработка алгоритмов обработки данных, определение особенностей взаимодействия с программной средой и т.п.

**4. Реализация** – составление программ, их тестирование и отладка.

**5. Модификация** – выпуск новых версий.





**Программа** – последовательность инструкций, адресованных компьютеру, которая точно определяет, как следует решать задачу.

**Константы** – данные, не изменяемые в процессе выполнения программы.

**Литералы** – константы, указанные непосредственно в тексте программы.

**Поименованные константы** – константы, обращение к которым выполняется по имени.

**Переменные** – поименованные данные, которые могут изменяться в процессе выполнения программы.

**Тип** – описатель данных, который определяет:

а) ***диапазон изменения значения*** переменной, задавая размер ее внутреннего представления;

б) ***множество операций***, которые могут выполняться над этой переменной.

**Выражения**

1. **Арифметические** **операции**
2. **Операции отношения**
3. **Логические операции**
4. **Поразрядные операции**

**Правила вычисления выражений**

1. Порядок выполнения операций определяется ***приоритетами и скобками***

2. При выполнении арифметических операций над числами различных типов автоматически осуществляется ***неявное преобразование:***

* целого и вещественного типов – к вещественному,
* с разными интервалами представлений – к типу с большим интервалом.

3.При сравнении вещественных чисел из-за их неточного представления проверку равенства и неравенства следует осуществлять ***с явным указанием допуска*.**

Корректное выполнение оператора предполагает, что результат вычисления и переменная правой части ***одного типа*** или ***совместимы по типу*.**

По правилам ***совместимы*:**

а) все целые типы между собой;

б) все вещественные типы между собой;

в) отрезок базового типа и базовый тип;

г) два отрезка одного и того же базового типа;

д) символ и строка

Если типы результата и переменной не совпадают, но совместимы, то при выполнении присваивания выполняется ***неявное автоматическое преобразование***.

Ввод – операция по передаче данных от источника в память компьютера.

Вывод – операция по передаче данных из компьютера на внешнее устройство.

Конец 1 презентации

Оператор условной передачи управления используется при обработке вариантов вычислений и реализует конструкцию ветвления.

Cчетный цикл – цикл, количество повторений которого известно или можно посчитать. Выход из такого цикла программируется по счетчику.

Итерационный цикл – цикл, количество повторений которого неизвестно или считается неизвестным при построении цикла. Выход из цикла программируется по выполнению или нарушению условия.

Поисковый цикл имеет два выхода – нашли и перебрали все и не нашли.

Конец 2 презентации

***Массив*** – это упорядоченная совокупность *однотипных данных*. Каждому элементу массива соответствует один или несколько *индексов порядкового типа*, определяющих положение элемента в массиве.

Количество *типов* индексов задает ***размерность*** массива.

***Тип индекса*** – порядковый – определяет доступ к элементу.

***Тип элемента*** – любой кроме файла, в том числе массивы, строки и т.п.

**Операции над массивами**

1. Операция присваивания
2. Доступ к элементу массива
3. Ввод/вывод массивов осуществляется поэлементно

Строка – последовательность символов.

**Операции над строками**

1. Присваивание строк
2. Обращение к элементу
3. Конкатенация (сцепление) строк
4. Операции отношения
5. Ввод-вывод строк

***Множество*** – неупорядоченная совокупность неповторяющихся элементов.

**Операции над множествами**

1. Присваивание
2. Объединение, пересечение и дополнение
3. Операции отношения
4. Проверка вхождения элемента во множество

Запись – это структура данных, образованная фиксированным числом разнотипных компонентов, называемых **полями** записи.

Конец 3 презентации

**Процедуры и функции** – самостоятельные фрагменты программы, соответствующим образом оформленные и вызываемые по имени (программные блоки).

***Модуль*** – это автономно компилируемая коллекция программных ресурсов, предназначенных для использования другими модулями и программами.

***Ресурсы*** – переменные, константы, описания типов и подпрограммы.

Все ресурсы, определенные в модуле делят на:

1) *внешние* – предназначенные для использования другими программами и модулями.

2) *внутренние* – предназначенные для использования внутри модуля.

**Открытый массив** – конструкция описания типа массива без указания типа индексов. Используется при объявлении формальных параметров.

***Нетипизированные параметры*** – параметры-переменные, тип которых при объявлении не указан.

Параметры процедурного типа используются для передачи в подпрограмму имен процедур и функций.

**Рекурсия** – организация вычислений, при которой процедура или функция обращаются к самим себе.

Различают *явную* и *косвенную* рекурсии. При явной – в теле подпрограммы существует вызов самой себя, при косвенной – вызов осуществляется в подпрограммах, вызываемых из рассматриваемой.

Каждое обращение к рекурсивной подпрограмме вызывает независимую *активацию* этой подпрограммы.

Совокупность данных, необходимых для *одной* активации рекурсивной подпрограммы, называется *фреймом активации.*

Конец 4 презентации

***Файл*** – поименованная последовательность элементов данных (компонентов файла), хранящихся, как правило, во внешней памяти.

***Файл языка Pascal*** – последовательность однотипных компонентов: файл записей, файл целых чисел, файл строк.

В зависимости от типа компонентов различают три типа файлов: *типизированные*, *текстовые* и *нетипизированные*.

Количество компонентов файла при объявлении файловой переменной не указывается.

Максимальный размер файла определяется свободным пространством на устройстве, например, диске.

Физически операции ввода-вывода с файлами выполняются с использованием буфера.

***Текстовый* *файл*** – файл, компонентами которого являются символьные строки переменной длины, заканчивающиеся специальным маркером – маркером «Конец строки».

***Типизированный* файл** – файл, все компоненты которого одного типа, заданного при объявлении файловой переменной.

***Нетипизированными*** называют файлы, объявленные без указания типа компонентов.

Конец презентации 5

***Указатель*** – тип данных, используемый для хранения ***смещений***.

Различают указатели:

* типизированные – адресующие данные конкретного типа;
* нетипизированные – не связанные с данными определенного типа.

*Список* – способ организации данных, предполагающий использова-ние указателей для определения следующего элемента.

Элемент списка состоит из двух частей: *информационной* и *адресной*.

Информационная часть содержит поля данных.

Адресная – включает от одного до n указателей, содержащих адреса следующих элементов. Количество связей, между соседними элементами списка определяет его связность: односвязные, двусвязные, n-связные.

**Односвязные списки**

Аналогично одномерным массивам реализуют последовательность элементов. Однако в отличие от одномерных массивов позволяют:

* работать **с произвольным количеством элементов**, добавляя и удаляя их по мере необходимости;
* осуществлять вставку и удаление записей, **не перемещая остальных элементов последовательности**;

но

* **не допускают прямого обращения к элементу по индексу;**
* **требуют больше памяти для размещения.**

В математике ***бинарным*** ***деревом*** называют конечное множество вершин, которое либо пусто, либо состоит из корня и не более чем двух непересекающихся бинарных деревьев, называемых левым и правым поддеревьями данного корня.

**Сортированные** бинарные деревья, строятся по правилу: *ключевое поле левого поддерева должно содержать значение меньше, чем в корне, а ключевое поле правого поддерева – значение больше или равное значению в корне*.

Конец 6 презентации

**Объектно-ориентированное программирование** – технология создания сложного программного обеспечения, основанная на представлении программы в виде системы объектов, каждый из которых является экземпляром определенного типа (класса), а классы образуют иерархию с наследованием свойств.

**Компонентный подход** – сконца 90-х годов ХХ века (COM-техноло-гия, Corba, SOAP) – подключение объектов через универсальные интерфейсы – развитие сетевого программирования – появление CASE-технологий.

**Процедурная декомпозиция –** процесс разбиения программы на подпрограммы.

**Структурной** называют декомпозицию,если:

* каждая подпрограмма имеет один вход и один выход;
* подпрограммы нижних уровней не вызывают подпрограмм верхних уровней;
* размер подпрограммы не превышает 40-50 операторов;
* в алгоритме использованы только структурные конструкции.

**Объектная декомпозиция** – процесспредставления предметной области задачи в виде отдельных функциональных элементов (объектов предметной области), обменивающихся в процессе выполнения программы входными воздействиями (сообщениями) .

**Объект предметной области** характеризуется:

* именем;
* состоянием;
* поведением.

**Состояние** – совокупность значений характеристик объекта, существенных с т. з. решаемой задачи.

**Поведение** – совокупность реакций на сообщения.

***Класс*** – это структурный тип данных, который включает описание полей данных, а также процедур и функций, работающих с этими полями данных.

Применительно к классам такие процедуры и функции получили название ***методов****.*

***Объект-переменная*** – переменная типа «класс».

Основные понятия ООП

* ***абстрагирование*** – выделение абстракций в предметной области задачи; под ***абстракцией*** при этом понимается совокупность существенных характеристик некоторого объекта предметной области, которые отличают его от всех других видов объектов;
* ***инкапсуляция*** – размещение в одном программном компоненте (объекте) данных и подпрограмм, которые с этими данными работают;
* ***ограничение доступа*** – сокрытие отдельных элементов реализации абстракции, не затрагивающих существенных характеристик ее как целого;
* ***модульность*** – принцип разработки программной системы, предполагающий реализацию ее в виде отдельных частей;
* ***иерархичность*** – использование иерархий при разработке программных систем; при этом используются как иерархии "целое-часть", так и иерархии "общее-частное";
* ***типизация*** – ограничение, накладываемое на свойства объектов и препятствующее взаимозаменяемости абстракций различных типов (или сильно сужающее возможность такой замены).

**Методы построения классов**

1. ***Наследование* –** *механизм*, позволяющий строить класс на базе более простого посредством добавления полей и определения новых методов. При этом исходный класс, на базе которого выполняется построение, называют *родительским* или *базовым*, а строящейся класс – *потомком* или *производным* классом. Если при наследовании какие-либо методы переопределяются, то такое наследование называется ***полиморфным***.
2. ***Композиция* –** *механизм*, позволяющий включать один или несколько объектов других классов в объекты конструируемого.
3. ***Наполнение*** – *механизм*, позволяющих включать или не включать объекты других классов в объект конструируемого..

C точки зрения синтаксиса ***класс*** – структурный тип данных, в котором помимо полей разрешается описывать *прототипы* (заголовки) процедур и функций, работающих с этими полями данных.

Любой метод неявно получает параметр **Self –** ссылку (адрес) на поля объекта, и обращение к полям происходит через это имя.

Конец презентации 7

Управление техническими средствами осуществляется через **API** (***Application Program Interface***) – набор из нескольких тысяч функций, выполняющих все системно-зависимые действия, такие как выделе-ние памяти, вывод на экран и т.д.

***Приложение*** (в отличие от программы) – набор подпрограмм, вызываемых при наступлении некоторого *события*, которым считается любое изменение в системе, касающееся данного приложения.

***Окно*** – самостоятельно существующий объект, параметры которого хранятся в специальной структуре данных, а поведение определяется обработчиками сообщений, составляющими ***оконную функцию***.

Появившиеся в очереди сообщения выбираются циклом обработки сообщений и передаются *через Windows* соответствующей оконной функции приложения.

Для выполнения действий обработчики сообщений обращаются к функциям API.

**Метка**

Компонент представляет собой окно с текстом и может использоваться для формирования на форме некоторых надписей или подписей.

***TEdit***

Компонент представляет собой окно, обычно выделенное цветом, которое может использоваться, например, для организации ввода информации.

***Кнопка***

Компонент представляет собой окно, в котором размещается название кнопки. Используется для инициирования каких-либо действий.

Конец презентации 8

**Свойство** - это средство Pascal Delphi, позволяющее определять интерфейс доступа к полям и методам класса.

Конец презентации 9