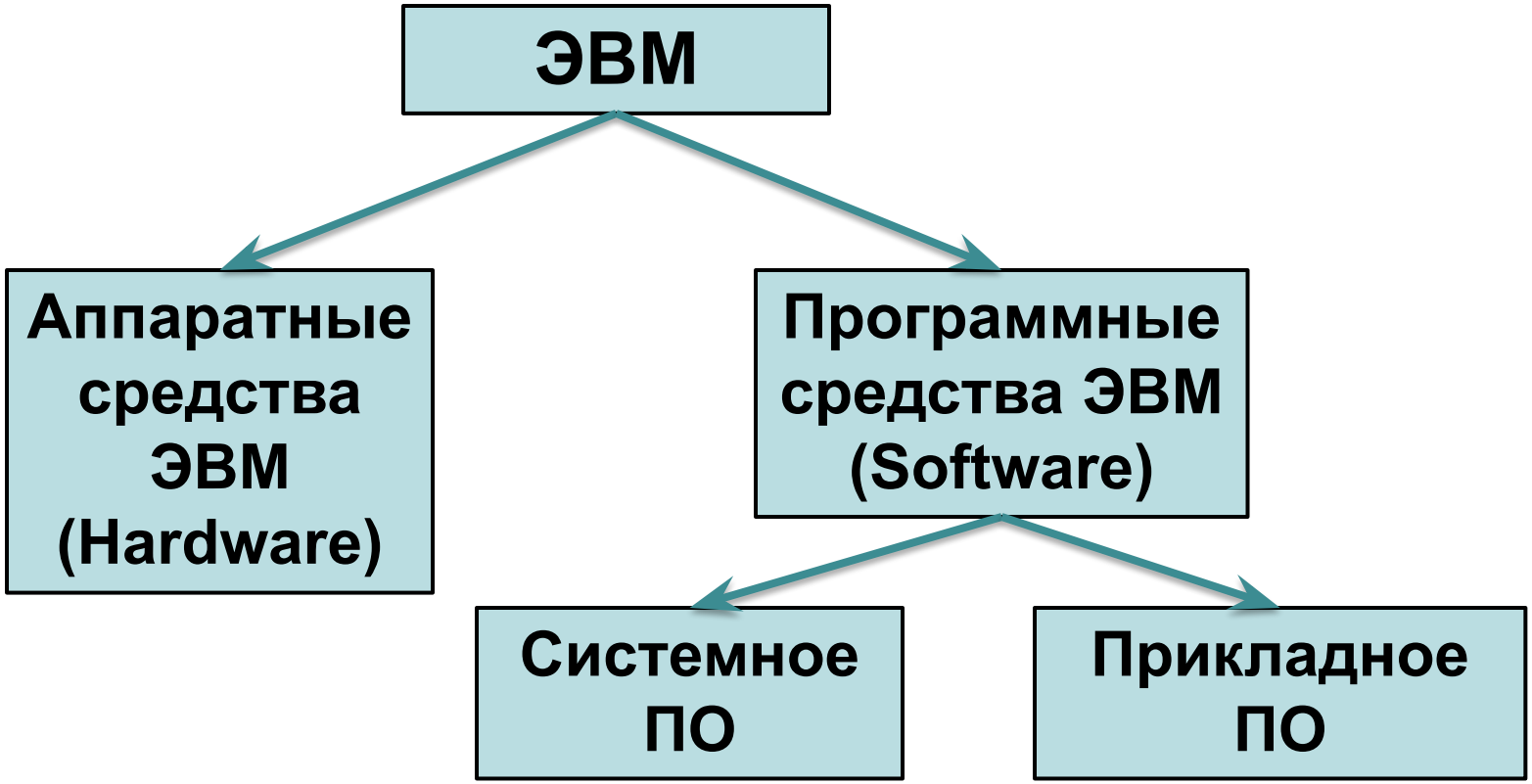
Лекция 13

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ

- **Системное программное обеспечение**
- **Прикладное программное обеспечение**

- **ОБЩИЕ ВОПРОСЫ РАЗРАБОТКИ ПО**
- **КЛАССИФИКАЦИЯ МЕТОДОВ ПРОЕКТИРОВАНИЯ ПРОГРАММНЫХ СРЕДСТВ**

ЭВМ как единство двух начал



Системное ПО предназначено для обеспечения работоспособности ЭВМ и разработки других программных средств

1. Общесистемное
2. Инструментальное
3. Диагностическое

1. Общесистемное ПО

1. Операционные системы (ОС)
2. Операционные оболочки – NC, VC
3. Операционные среды (Win 3.1) – надстройка над DOS
4. Драйверы
5. Утилиты

Операционная система

ОС – программа, которая автоматически загружается в оперативную память и выполняет управление физическими и логическими ресурсами ЭВМ

-- Физические – память, процессор, внешние устройства

-- Логические – программы, файлы, события

(MS DOS, Windows, Windows NT Server, LUNIX)

Операционная среда – надстройка над ОС с развитым пользовательским интерфейсом (Windows 3.1)

Операционная оболочка – это программа, которая позволяет более удобно выполнять команды ОС (Norton Commander, Total Commander).

Драйвер – набор инструкций или программа, расширяющая возможности ОС по управлению ЭВМ (например, раскладка клавиатуры, управление памятью)

Утилита – программа вспомогательного назначения, которая представляет пользователю возможность реализовать набор некоторых действий (обслуживание дисков, шифрование, архивация и пр.). Существуют отдельные утилиты и многофункциональные (Norton Utilities).

2. Инструментальное ПО

1. Системы программирования

(включают компилятор или интерпретатор, библиотеки подпрограмм, интегрированную оболочку для редактирования и отладки программ)
(Pascal, Delphi, C++, C#, Assembler, Microsoft Visual Studio 2005).

2. СУБД – системы управления базами данных, которые обеспечивают операции по созданию больших информационных массивов, сортировки, поиску данных, выводу отчетов (Access, FoxPro, Oracle, MS SQL Server)

3. Case-системы – система, поддерживающие разработку крупных программных средств на протяжении всего жизненного цикла, от моделирования бизнес-процессов до протоколирования всех этапов работы (CASE, Design/IDEF, Designer, BP Win).
   Case-системы являются инструментарием для системных аналитиков и разработчиков программных средств

4. Нетрадиционные средства разработки ПО – инструментальные средства с закрытой непубликуемой технологией (игры, мультимедиа) а также новые специализированные системы разработки игр, мультимедийных приложений, экспертных систем, средств организации WEB- серверов в сети Internet и пр.)

3. Диагностическое ПО

1. **Антивирусное программное обеспечение** – это ПО (например, Dr Web, Антивирус Касперского, Norton Antivirus, MS Antivirus и др.), предназначенное для ликвидации последствий или предотвращения заражения ПК специальными программами (вирусами), выполняющими на ПЭВМ нежелательные для пользователя действия

2. Средства тестирования аппаратных устройств – это программы, позволяющие проверить исправность отдельных узлов ПЭВМ, например, Vtest – визуальное тестирование качества изображения монитора, CoreTest – тестирование винчестера.
3. Диагностическое ПО - для поиска и определения характера неисправности в блоках ЭВМ, например, пакеты Check It, Win Check It.

4. Средства корректировки – это программы, позволяющие произвести настройку режимов работы отдельного узла ЭВМ, например, калибровка цветности монитора)
5. Вспомогательные программные средства – используется при ремонте узлов ЭВМ, например, Screen-Test генерирует тестовые сигналы в процессе ремонта монитора)

ПРИКЛАДНОЕ ПО - предназначено для решения определенной задачи в конкретной предметной области

1. **ПО общего назначения** (текстовые, табличные, графические процессоры, электронные секретари, эл. почта, игры)
2. **Специализированное ПО** (САПР, обучающие системы, математические системы, издательские системы, финансовые системы, системы управления проектами)
3. **Нетрадиционное ПО** (системы мультимедиа, интеллектуальные системы: экспертные системы, системы распознавания, перевод текста)

ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ

(ППП) как средство организации прикладного ПО

1. Проблемно-ориентированные ППП
2. Интегрированные ППП
3. Пакеты ППП для решения научно-технических задач

Классификация ППП

1.Проблемно-ориентированные ППП

Проблемно-ориентированные ППП включают следующие программные продукты:

- Текстовые процессоры
- Настольные издательские системы (НИС)
- Графические редакторы
- Пакеты для работы с векторной графикой
- Электронные таблицы
- Организаторы работ
- Системы управления базами данных (СУБД)
- Пакеты демонстрационной графики
- Пакеты программ мультимедиа
- Системы автоматизации проектирования
- Программы распознавания символов

2. Интегрированные ППП

Традиционные, или полносвязанные, интегрированные комплексы представляют собой многофункциональный автономный пакет, в котором в одно целое соединены функции и возможности различных специализированных пакетов, родственных в смысле технологии обработки данных на отдельном рабочем месте

3. Пакеты ПП для решения научно-технических задач

- Пакет прикладных программ представляет собой набор подпрограмм, объединяемый управляющей программой и предназначенный для решения конкретных задач в какой-либо области знаний
- Обычно все подпрограммы делаются свободными от ввода-вывода и размер массивов указывается условный. Программы ввода-вывода выполняются в виде отдельного модуля
- В зависимости от структуры ППП модули могут быть различных структур:
  - простой
  - оверлейной
  - динамически последовательной
  - динамически параллельной

Общие вопросы разработки программных средств

1. Жизненный цикл ПО
2. Этапы решения научно-технических задач

Жизненный цикл ПО

Жизненный цикл программного обеспечения (ПО) — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации. Этот цикл — процесс построения и развития ПО.

Этапы решения научно-технических задач на ЭВМ

1. **Постановка задачи** (описывается цель решения задачи, проблема, подробное содержание характеристик, условия задачи, входные и выходные данные)
2. **Математическое описание** (все существующие соотношения между величинами выражаются посредством математических формул, формируется математическая модель задачи с определенной точностью и ограничениями и допущениями, математическая модель должна быть реалистичной и реализуемой)

3. **Выбор и обоснование метода решения** (одну и ту же задачу можно решать различными методами: процедурное программирование, объектно-ориентированное программирование, использование известного ПО)
4. **Проектирование** (создается общая структура программы, описывается взаимодействие между компонентами программы, блок-схема)
5. **Кодирование** (все конструкции, записанные на языке проектирования, переводятся на язык программирования высокого уровня)

6. **Тестирование** (всесторонняя проверка программы на *правильность*, *эффективность*, на *вычислительную сложность*- состоит в экспериментальном сравнении двух алгоритмов, решающих одну и ту же задачу)
7. **Составление рабочей документации** (требования ЕСПД: описание применения, руководство пользователя, руководство программисту)
8. **Сопровождение** (этапы эксплуатации программы: обучение пользователей, обновления программы, консультации)

ОСНОВНЫЕ НАПРАВЛЕНИЯ В программировании

- **ПРОЦЕДУРНОЕ программирование**
- **МОДУЛЬНОЕ программирование**
- **ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ программирование**

1. Процедурное программирование

- В процедурном программировании основное внимание уделяется алгоритмам, т.е. некоторой последовательности действий, выполнение которых приводит к определенному результату.
- Языки программирования, которые поддерживают эту модель, называются процедурными, и главное внимание в них уделяется построению подпрограмм (процедур).

Основные вопросы процедурного программирования:

- передача аргументов в процедуры
- получение вычисленных значений из процедур.

Первыми процедурными языками были FORTRAN, ALGOL 60, PASKAL, СИ.

При разработке большой программы коллективом разработчиков необходимо решать очень сложные проблемы (надо договариваться об используемых именах, об организации общих данных и способах доступа к ним).

В процедурном программировании

Алгоритм записывается на выбранном языке программирования с помощью команд описания данных, вычисления значений и управления последовательностью выполнения программы.

Используемые типы данных:

1. Переменные и константы

Реальные данные, с которыми работает программа, - это *числа*, *строки* и *логические величины*. Эти типы называются *базовыми*.

2. Числовые данные

3. Арифметические операции

«+» (сложение)

«-» (вычитание)

«*» (умножение)

«/» (деление)

4. Арифметические выражения

С помощью арифметических операций формируются арифметические выражения, которые состоят из операций и *операндов* (переменных и констант).

5. Логические выражения

При записи логических выражений используются операции сравнения и логические операции. Операции сравнения сличают значения правого и левого операндов. Результатом сравнения является *true*, если оно удачно, и *false* в противном случае.

Описание	Паскаль, Бейсик	Си ++
Равно	=	= =
Не равно	< >	! =
Меньше	<	<
Меньше или равно	< =	<=
Больше	>	>
Больше или равно	> =	> =

Логическая операция	Бейсик	Паскаль	Си ++
И	AND	and	**&&**
ИЛИ	OR	or	**||**
НЕ	NOT	not	**!**

6. Строчные выражения

7. Указатели

Указатели – адреса физической памяти

8. Структуры

Современные языки программирования позволяют применять такие сложные типы данных, составляющиеся из базовых и определенных ранее сложных типов.

В результате удается организовать структуры данных произвольной сложности: списки, деревья и т.п. При этом структура объединяет группу разных данных под одним названием. Например, чтобы организовать обработку данных по студентам, в программе удобно не просто описать десяток различных переменных, а объединить их в структуру «студент», состоящую из полей разного типа «имя», «пол», «группа» и т. д.

9. Массивы

Массив – сложный тип данных, доступ к элементам которого происходит по их положению, по номеру или индексу. Например, можно описать массив, состоящий из тысячи элементов численного типа, и затем обратиться к десятому или сотому элементу по его номеру.

10. Операторы

11. Комментарии

Комментарии – часть исходных текстов, выделяемых с помощью специальных обозначений.

	Бейсик	**Паскаль**	**Си ++**
Однострочные комментарии	REM или ‘	//	//
Многострочные комментарии	нет	{ } или (* *)	/* */

Процедуры и функции

Подпрограммы бывают двух видов:

- Процедуры
- Функции

Отличаются они тем, что процедура просто выполняет группу операторов,

а функция вдобавок вычисляет некоторое значение и передает его обратно в главную программу (*возвращает* значение).

Структура процедуры

	Бейсик	Паскаль	Си ++
Заголовок функции	SUB имя (список_параметров)	procedure имя (список_параметров) ;	Void имя (список_параметров)

Тело	Последовательность операторов	begin
		
		Последовательность операторов
		
		end;
			{
			
			Последователь-ность операторов
			
			};

Завершение	END SUB	нет	нет

Структура функции

	Бейсик	**Паскаль**	**Си ++**
Заголовок функции	FUNCTION имя (список_параметров)	Function имя (список_параметров) : тип_функции;	тип_функции имя (список_параметров)

Тело	Последовательность операторов	begin
		
		Последовательность операторов
		
		end;
			{
			
			Последовательность операторов
			
			};

Завершение	END FUNCTION	нет	нет

2. Модульное программирование

В модульном программирование основные акценты переносятся на построение модулей и их взаимодействию в программе.

Модуль – это множество взаимосвязанных подпрограмм (процедур) вместе с данными, которые эти процедуры обрабатывают.

Основная цель этого направления состоит в *скрытии данных в модулях*, что не позволяет обратиться к ним из-за пределов модуля.

Организация данных, а не алгоритмов – это основная задача модульного программирования

- При создании ПО необходимо определить все модули, которые будут использоваться, и разделить программу на модули так, чтобы ее данные были скрыты в этих модулях.
- Модуль – это самостоятельная часть программы, которая разрабатывается одним программистом, например.
- Поскольку доступ к данным из-за пределов модуля запрещен (скрыт), то соответственно, предотвращено их случайное изменение (ошибки в программе).

- Язык MODULA2 был специально сконструирован для поддержки модульного программирования.
- Языки C++, C# не были специально для этого созданы, однако реализованная в них концепция **классов**, позволила работать с модулями. Эти языки содержат все необходимое для поддержки как процедурного, так и модульного программирования. Эти направления дополняют друг друга, а не исключают.

3. Объектно-ориентированное программирование

Программа представляется в виде набора **объектов**, взаимодействующих между собой посредством сообщений.

Объект = данные + процедуры

Объект – совокупность свойств (структур данных, характерных для этого объекта), методов их обработки (подпрограмм изменения свойств) и событий, на которые данный объект может реагировать и которые приводят, как правило, к изменению свойств объекта.

Существует 4 важнейших механизма объектно-ориентированного программирования:

1. **Наследование**
2. **Полиморфизм**
3. **Инкапсуляция**
4. **Абстракция**

Для описания объектов служат классы.

Класс определяет свойства и методы объекта, принадлежащего этому классу. Соответственно, любой объект можно определить как *экземпляр класса*.

Важнейшая характеристика класса – возможность создания на его основе новых классов с **наследованием** всех его свойств и методов и добавлением собственных. Класс, не имеющий предшественника, называется *базовым*.

Полиморфизм – возможность использования методов с одинаковыми именами для обработки данных разных типов.

Инкапсуляция — свойство языка программирования, позволяющее объединить и защитить данные и код в объект и скрыть реализацию объекта от пользователя (прикладного программиста) .
При этом пользователю предоставляется только спецификация (интерфейс) объекта.
Пользователь может взаимодействовать с объектом только через этот интерфейс.
Реализуется с помощью директив:
public, private, protected.

Абстракция в объектно-ориентированном программировании – это придание объекту характеристик, которые чётко определяют его концептуальные границы, отличая от всех других объектов, при этом особенность выбранных характеристик такова, что при работе с объектами не потребуется вникания в особенности реализации объектов