ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ. ОСНОВНЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Основные понятия, факты

Классификация программного обеспечения ЭВМ. Назначение прикладного и системного программного обеспечения. Структурное программирование. Модульное программирование. Объектно-ориентированное программирование.

Навыки и умения

Разработка программ с использованием технологий структурного, модульного и объектно–ориентированного программирования.

Классификация программного обеспечения

Традиционно программное обеспечение подразделяют на два класса:

1) 1) системное программное обеспечение (СПО) и
2) 2) прикладное (пользовательское) программное обеспечение (ППО)

Выделим еще один класс (скорее группу) программ – (3) специальное программное обеспечение информационных и управляющих систем.

Прикладные программы предназначены для решения функциональных задач, они выполняют обработку информации различных предметных областей.
Это самый многочисленный класс программных продуктов.
В свою очередь он условно подразделяется на две группы.
1 группа – программы для решения отдельных, самостоятельных задач. Эти программы выполняются независимо друг от друга и представляют собой набор разрозненных, не связанных между собой знаний. Конечно, и они могут быть весьма сложными и очень необходимыми задачами.
2 группа - системы программ для решения классов задач из различных специализированных областей науки техники и промышленности. Часто такие системы называют пакетами прикладных программ. Программы, входящие в пакет, выполняются не отдельно друг от друга, а совместно, в различных комбинациях, в зависимости от конкретной решаемой задачи.

К **специальному программному обеспечению информационных и управляющих систем** относятся

- • программы (системы) управления базами данных;
- • программы управления языком интерфейса информационных систем;
- • программы сбора и предварительной обработки информации (в информационно-измерительных системах, например, бортовые системы).

ПО этого класса часто оказывается скрытым в составе драйверов оборудования или поставляется в виде библиотек функционального расширения языков программирования.

Поэтому часто такое ПО относят к системному программному обеспечению.
Система управления базами данных (СУБД) - это сложная программная система накопления и последующего манипулирования данными. Каждая СУБД предоставляет интерфейс с базой данных и может располагать средствами непосредственного доступа к последней ее пользователей.
С ***помощью языка описания данных*** создаются описания элементов и записей данных, а также взаимосвязей между ними. Для выполнения операция с базой данных (например, выборка или обновление) в прикладных программах используется язык манипулирования данными. Фактическая структура физического хранения данных известна только СУБД.

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 машинный язык компьютера. Состав СПО мало зависит от характера решаемых задач пользователя.

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.).

Данный класс программных продуктов тесно связан с типом компьютера и является его неотъемлемой частью.
Программные продукты данного класса в основном ориентированы на квалифицированных пользователей - профессионалов в компьютерной области: системного программиста, администратора сети, прикладного программиста, оператора.

Часто системное ПО компьютера подразделяют на БАЗОВОЕ и СЕРВИСНОЕ программное обеспечение.
БАЗОВОЕ программное обеспечение (base software) - минимальный набор программных средств, обеспечивающих работу компьютера.
К базовому программному обеспечению компьютера относятся
- • операционные системы и драйверы в составе ОС;
- • интерфейсные оболочки для взаимодействия пользователя с ОС (операционные оболочки) и программные среды;
- • системы управления файлами.

Операционная система - совокупность программных средств, обеспечивающая управление аппаратной частью компьютера и прикладными программами, а также их взаимодействием между собой и пользователем.

Операционная система предназначена для управления выполнением пользовательских программ, планирования и управления вычислительными ресурсами ЭВМ.
Операционная система, с одной стороны, выступает как интерфейс между аппаратурой компьютера и пользователем с его задачами, с другой стороны, предназначена для эффективного использования ресурсов вычислительной системы и организации надежных вычислений.

Системы управления файлами предназначены для организации более удобного доступа к данным, организованным как файлы.
Вместо низкоуровневого доступа к данным с указанием конкретных физических адресов система управления файлами позволяет использовать логический доступ с указанием имени файла.
Любая система управления файлами не существует сама по себе - она разработана для работы в конкретной ОС и с конкретной файловой системой. То есть можно было бы систему управления файлами отнести к ОС.
Но в связи с тем, что

1) 1) ряд ОС позволяет работать с несколькими файловыми системами (либо с одной из нескольких, либо сразу с несколькими одновременно); а дополнительную файловую систему можно установить (т.е. они самостоятельны)
2) 2) простейшие ОС могут работать и без файловых систем;

системы управления файлами выделяются в отдельную группу системных программ.
Заметим, что часто в специальной литературе системы управления файлами относят все-таки к операционным системам.

СЕРВИСНОЕ программное обеспечение - программы и программные комплексы, которые расширяют возможности базового программного обеспечения и организуют более удобную среду работы пользователя.
Это набор сервисных, дополнительно устанавливаемых программ, которые можно классифицировать по функциональному признаку следующим образом:

- драйверы специфических и специальных устройств (те, которые не поставляются в составе ОС).
- программы диагностики работоспособности компьютера;
- антивирусные программы, обеспечивающие защиту компьютера, обнаружение и восстановление зараженных файлов;
- программы обслуживания дисков, обеспечивающие проверку качества поверхности магнитного диска, контроль сохранности файловой системы на логическом и физической уровнях, сжатие дисков, создание страховых копий дисков, резервирование данных на внешних носителях и др.;
- программы архивирования данных, которые обеспечивают процесс сжатия информации в файлах с целью уменьшения объема памяти для ее хранения;
- программы обслуживания сети.

Эти программы часто называются **утилитами**. (Заметим, что к антивирусным средствам этот термин обычно не применяется)

Утилиты - программы, служащие для выполнения вспомогательных операций обработки данных или обслуживания компьютеров (диагностики, тестирования аппаратных и программных средств, оптимизации использования дискового пространства, восстановления разрушенной на магнитном диске информации и т.п.).

Отдельно вспомним о такой группе системного ПО как **системы программирования**.
Это набор специализированных программных продуктов, которые являются инструментальны средствами разработчика. Программные продукты данного класса поддерживают все этапы процесса программирования, отладки и тестирования создаваемых программ.
Система программирования включает следующие программные компоненты:

- редактор текста;
- транслятор с соответствующего языка;
- компоновщик (редактор связей);
- отладчик;
- библиотеки подпрограмм.

Таким образом, в системном ПО мы выделили **пять групп системных программ**:

- операционные системы;
- интерфейсные оболочки для взаимодействия пользователя с ОС (операционная оболочка) и программные среды;
- системы управления файлами;
- системы программирования;
- утилиты;
- средства сетевого доступа.

ОСНОВНЫМИ ТЕХНОЛОГИЯМИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ являются

1. 1. Структурное программирование
2. 2. Модульное программирование
3. 3. Объектно-ориентированное программирование

Структурное программирование

Сутью структурного программирования является возможность разбиения программы на составляющие элементы.
Идеи структурного программирования появились в начале 70-годов в компании IBM, в их разработке участвовали известные ученые Э. Дейкстра, Х. Милс, Э. Кнут, С. Хоор.

Распространены две методики (стратегии) разработки программ, относящиеся к структурному программированию: программирование "сверху вниз" и программирование "снизу вверх".

Программирование "сверху вниз", или нисходящее программирование – это методика разработки программ, при которой разработка начинается с определения

целей решения проблемы, после чего идет последовательная детализация, заканчивающаяся детальной программой. Является противоположной методике программирования «снизу вверх».
При нисходящем проектировании задача анализируется с целью определения возможности разбиения ее на ряд подзадач. Затем каждая из полученных подзадач также анализируется для возможного разбиения на подзадачи. Процесс заканчивается, когда подзадачу невозможно или нецелесообразно далее разбивать на подзадачи.
В данном случае программа конструируется иерархически - сверху вниз: от главной программы к подпрограммам самого нижнего уровня, причем на каждом уровне используются только простые последовательности инструкций, циклы и условные разветвления.

Программирование "снизу вверх", или восходящее программирование – это методика разработки программ, начинающаяся с разработки подпрограмм (процедур, функций), в то время когда проработка общей схемы не закончилась. Является противоположной методике программирования «сверху вниз».

Такая методика является менее предпочтительной по сравнению с нисходящим программированием так как часто приводит к нежелательным результатам, переделкам и увеличению времени разработки.

Достоинства структурного программирования:

1) 1) повышается надежность программ (благодаря хорошему структурированию при проектировании, программа легко поддается тестированию и не создает проблем при отладке);
2) 2) повышается эффективность программ (структурирование программы позволяет легко находить и корректировать ошибки, а отдельные подпрограммы можно переделывать (модифицировать) независимо от других);
3) 3) уменьшается время и стоимость программной разработки;
4) 4) улучшается читабельность программ.

Резюме

Технология структурного программирования при разработке серьезных программных комплексов, основана на следующих принципах:

- - программирование должно осуществляться сверху вниз;
- - весь проект должен быть разбит на модули (подпрограммы) с одним входом и одним выходом;
- - подпрограмма должна допускать только три основные структуры – последовательное выполнение, ветвление (if, case) и повторение (for, while, repeat).
- - недопустим оператор передачи управления в любую точку программы (goto);
- - документация должна создаваться одновременно с программированием в виде комментариев к программе.

Структурное программирование эффективно используется для решения различных математических задач, имеющих алгоритмический характер.

Модульное программирование

Модульное программирование - это организация программы как совокупности небольших независимых блоков (модулей), структура и поведение которых подчиняется определенным заранее правилам.
Модулем (в модульном программировании) называется множество взаимосвязанных подпрограмм (процедур) вместе с данными, которые эти подпрограммы обрабатывают.

Модульное программирование предназначено для разработки больших программ.

Разработкой больших программ занимается коллектив программистов. Каждому программисту поручается разработка некоторой самостоятельной части программы. И он в таком случае отвечает за конструирование всех необходимых процедур и данных для этих процедур. Сокрытие данных (запрет доступа к данным из-за пределов модуля) предотвращает их случайное изменение и соответственно нарушение работы программы. Для взаимодействия отдельных частей (модулей) программы коллективу программистов необходимо продумать только интерфейс (взаимодействие) сконструированных модулей в основной программе.

Напомним понятие и структуру модуля в терминах языка Pascal.
Модуль (unit) – программная единица, текст которой компилируется независимо (автономно).
Модуль содержит 4 раздела: заголовок, интерфейсная часть (раздел объявлений), раздел реализации и раздел инициализации.

```
UNIT <имя модуля>;                    {заголовок}
INTERFACE                             {интерфейсная часть}
  Uses <используемые модули>;
  Const <объявления глобальных констант>;
  Type <объявления глобальных типов>;
  Var <описание глобальных переменных>;
  Procedure <заголовки(!) доступных процедур>;
    . . .
  Function <заголовки(!) доступных функций>;
    . . .
IMPLEMENTATION                        {раздел реализации}
  Uses <используемые при реализации модули>;
  Const <объявления скрытых (локальных) констант>;
  Type <объявления скрытых (локальных) типов>;
  Var <описание скрытых (локальных) переменных>;
  Procedure <тела(!) скрытых (локальных) процедур>;
    . . .
  Function <тела(!) скрытых (локальных) функций>;
                                      BEGIN
    <основной блок модуля = раздел инициализации>
END.
```

Замечание. Различие всех описаний, содержащихся в разделах интерфейса и реализации, заключается в сфере их использования. В интерфейсном разделе – внешние описания, в разделе реализации – внутренние.

Объектно-ориентированное программирование (ООП)

Чтобы написать еще более сложную программу, необходим новый подход к программированию - технология объектно-ориентированного программирования.

ООП включает лучшие идеи, воплощённые как в структурном программировании, так и в модульном. «Является еще более структурным программированием, еще более модульным» (Джеф Дантеманн?).
И, кроме того, ООП сочетает старые принципы с мощными новыми концепциями, которые позволяют оптимально организовывать программы.

Объектно-ориентированное программирование позволяет разложить проблему на составные части. Каждая составляющая становится самостоятельным объектом, содержащим свои собственные коды и данные, которые относятся к этому объекту. В этом случае программирование в целом упрощается, и программист получает возможность оперировать гораздо большими по объёму программами.
Таким образом, ООП – «это методология, основанная на представлении программы в виде совокупности объектов, каждый из которых является реализацией собственного класса» (А.Д. Александровский).

Основным понятием ООП является понятие класса.
Класс – множество объектов, связанных общностью структуры и поведения (класс содержит описание структуры и поведение всех объектов, связанных отношением общности). Любой объект является экземпляром класса.
Методом называется процедура или функция, определенная внутри класса.

Базовые принципы ООП

ООП характеризуется тремя базовыми принципами:

1. 1. Инкапсуляция
2. 2. Наследование
3. 3. Полиморфизм

Инкапсуляция

Инкапсуляция представляет собой комбинирование данных (записи, структуры) с процедурами и функциями для получения нового типа данных.
Здесь проводится аналогия с физическими объектами. Конкретные физические свойства определяются данными различных типов. Кроме того, любой физический объект характеризуется и своим поведением во внешнем мире. Поведение объекта задается процедурами и функциями.
Итак, инкапсуляция означает, что методы (коды) и данные одновременно представлены в одной и той же структуре.
Например,
Type

```
Coordinates = class
                x, y : byte;
                procedure Init (Xinit, Yinit: byte);
                function GetX : byte;
                function GetY : byte;
              end;
```

Наследование

Наследование служит для использования однажды определенного класса в построении целой иерархии производных классов, каждый из которых наследует доступ к данным и методам работы с ними всех своих «родителей».
То есть, можно построить иерархию классов, которая выражает родословное дерево классов. Классы организованы в единую древовидную структуру с общим корнем. Свойства и методы определенного класса автоматически доступны любому классу, расположенному ниже в иерархическом дереве.
Таким образом, наследование – это особенность ООП, посредством которой класс может быть определен как расширение уже существующего класса.
Например,

Type

```
Cursor = class (coordinates)
           Hidden : Boolean;
         end;
```

Здесь класс «курсор» наследует все свойства и методы предварительно описанного класса «координаты», т.е. для него определены координаты x, y, а также методы «инициализация», «определить координату x», «определить координату y» (см. выше описание класса «координаты»). Кроме того, класс «курсор» обладает собственным методом - «быть невидимым».
Наследование позволяет различным типам данных совместно использовать один и тот же код, приводя к уменьшению его размера и повышению функциональности.

Полиморфизм

Полиморфизм – это придание действию (методу) одного имени, которое совместно используется объектами всей иерархии класса, причем каждый объект реализует это действие своим собственным, подходящим для него образом.
Другими словами, полиморфизм – это использование одинаковых имен методов на разных уровнях иерархии классов.

Достоинства ООП

С помощью уменьшения взаимозависимости между компонентами программного обеспечения ООП позволяет разрабатывать системы, пригодные для многократного использования. Такие компоненты могут быть созданы и отлажены как независимые программные единицы.

Несколько слов об использовании технологии ООП в среде Delphi на языке Object Pascal.

Для разработки приложений в Delphi используются специальным образом оформленные классы – компоненты.

Компонент обладает набором свойств и методов. Свойства компонента изменяются либо на этапе сборки приложения (под воздействием системы), либо программно, в процессе работы приложения (под воздействием пользователя).
Особым видом свойства компонента является событие. Процедура обработки события – это реакция приложения на изменение свойства компонента под воздействием системы или пользователя (нажатие клавиши, перемещение курсора и т.п.)
В Object Pascal объекты существуют только в динамической памяти (т.е. переменная, являющаяся объектом, по сути является указателем на объект, и содержит адрес объекта).