ТЕМА:

Проектирование объектно – ориентированного приложения. Создание интерфейса пользователя. Программирование приложений. Тестирование, отладка приложения. Создание документации.

Методы проектирования алгоритмов и программ очень разнообразны, их можно классифицировать по различным признакам, важнейшими из которых являются:

- степень автоматизации проектных работ;
- принятая методология процесса разработки.

По степени *автоматизации* проектирования алгоритмов и программ можно выделить:

- Методы традиционного (неавтоматизированного) проектирования;
- методы автоматизированного проектирования (CASE-технология и ее элементы).

Неавтоматизированное проектирование алгоритмов и программ преимущественно используется при разработке небольших по трудоемкости и структурной сложности программных продуктов, не требующих участия большого числа разработчиков.

Автоматизированное проектирование алгоритмов и программ возникло с необходимостью уменьшить затраты на проектные работы, сократить сроки их выполнения, создать типовые "заготовки" алгоритмов и программ, многократно тиражируемых для различных разработок, координации работ большого коллектива разработчиков, стандартизации алгоритмов и программ.

Проектирование алгоритмов и программ может основываться на различных подходах, среди которых наиболее распространены:

- структурное проектирование программных продуктов;
- информационное моделирование предметной области и связанных с ней приложений;
- объектно-ориентированное проектирование программных продуктов.

В основе структурного проектирования лежит последовательная декомпозиция, целенаправленное структурирование на отдельные составляющие.

Типичными методами структурного проектирования являются:

- нисходящее проектирование, кодирование и тестирование программ; Нисходящим - называют проектирование, в котором решение задач высоких иерархических уровней предшествует решению задач более низких иерархических уровней. Система разрабатывается, в условиях, когда ее элементы не определены и, следовательно, сведения об их возможных свойствах носят предположительный характер.
- модульное программирование;
- структурное проектирование (программирование) и др.

В зависимости от объекта структурирования различают:

- ***функционально-ориентированные методы*** - последовательное разложение задачи или целостной проблемы на отдельные, достаточно простые составляющие, обладающие функциональной определенностью;
- методы ***структурирования данных.***

Структурный подход использует:

- диаграммы потоков данных (информационно-технологические схемы) - показывают *процессы* и информационные потоки между ними с учетом "событий", инициирующих процессы обработки;
- интегрированную структуру данных предметной области (инфологическая модель, ER- диаграммы);
- диаграммы декомпозиции - структура и декомпозиция целей, функций управления, приложений;
- структурные схемы - архитектура программного продукта в виде иерархии взаимосвязанных программных модулей с идентификацией связей между ними, детальная логика обработки данных программных модулей (блок-схемы).

Один из основоположников информационной инженерии - Дж. Мартин - выделяет следующие составляющие данного подхода:

- информационный анализ предметных областей *(бизнес - областей);*
- информационное моделирование - построение комплекса взаимосвязанных моделей данных;
- системное проектирование функций обработки данных;
- детальное конструирование процедур обработки данных.

Первоначально строятся информационные модели различных уровней представления:

- информационно-логическая модель, не зависящая от средств программной реализации хранения и обработки данных, отражающая интегрированные структуры данных предметной области;
- даталогические модели, ориентированные на среду хранения и обработки данных.

Даталогические модели имеют логический и физический уровни представления. *Физический уровень* соответствует организации хранения данных в памяти компьютера. *Логический уровень* данных применительно к СУБД реализован в виде:

- концептуальной модели базы данных - интегрированные структуры данных под управлением СУБД;
- внешних моделей данных - подмножество структур данных для реализации приложений.

Объектно-ориентированный подход к проектированию программных продуктов основан на:

- выделении классов объектов;
- установлении характерных *свойств* объектов и *методов* их обработки;
- создании иерархии *классов,* наследовании свойств объектов и методов их обработки.

Объектный подход при разработке алгоритмов и программ предполагает:

- объектно-ориентированный анализ предметной области;
- объектно-ориентированное проектирование.

Объектно-ориентированный анализ - анализ предметной области и выделение объектов, определение свойств и методов обработки объектов, установление их взаимосвязей.

Объектно-ориентированное проектирование соединяет процесс объектной декомпозиции и представления с использованием моделей данных проектируемой системы на логическом и физическом уровнях, в статике и динамике.

ЭТАПЫ СОЗДАНИЯ ПРОГРАММНЫХ ПРОДУКТОВ

При традиционной неавтоматизированной разработке программ независимо от принятого метода проектирования и используемого инструментария выполняют следующие работы.

1. Составление технического задания на программирование

Данная работа соответствует этапу анализа и спецификации программ жизненного цикла программных продуктов.

При составлений технического задания требуется:

- определить платформу разрабатываемой программы - тип операционной системы;
- оценить необходимость сетевого варианта работы программы;
- определить необходимость разработки программы, которую можно переносить на различные платформы;
- обосновать целесообразность работы с базами данных под управлением СУБД.

2. Технический проект

На данном этапе выполняется комплекс наиболее важных работ, а именно:

- с учетом принятого подхода к проектированию программного продукта разрабатывается детальный алгоритм обработки данных или уточняется состав объектов и их свойств, методов обработки, событий, запускающих методы обработки;
- определяется состав общесистемного программного обеспечения, включающий базовые средства;
- разрабатывается внутренняя структура программного продукта, образованная отдельными программными модулями;
- осуществляется выбор инструментальных средств разработки программных модулей.

3. Рабочая документация (рабочий проект)

На данном этапе осуществляется *адаптация базовых средств* программного обеспечения.

Выполняется разработка программных модулей или методов обработки объектов - собственно *программирование* или создание программного кода.

Проводятся автономная и комплексная *отладка* программного продукта, *испытание* работоспособности программных модулей и базовых программных средств.

Для комплексной отладки готовится *контрольный пример,* который позволяет проверить соответствие возможностей программного продукта заданным спецификациям.

Основной результат работ этого этапа - также создание эксплуатационной *документации* на программный продукт:

- ***описание применения*** - дает общую характеристику программного изделия с указанием сферы его применения, требований к базовому программному обеспечению, комплексу технических средств;
- ***руководство пользователя*** - включает детальное описание функциональных возможностей и технологии работы с программным продуктом. Данный вид документации ориентирован на *конечного* пользователя и содержит необходимую информацию для самостоятельного освоения и нормальной работы пользователя (с учетом требуемой квалификации пользователя);
- ***руководство программиста (оператора)*** - указывает особенности установки (инсталляции) программного продукта и его внутренней структуры - состав и назначение модулей, правила эксплуатации и обеспечения

4. Ввод в действие

Готовый программный продукт сначала проходит *опытную эксплуатацию* (пробный рынок продаж), а затем сдается в *промышленную эксплуатацию* (тиражирование и распространение программного продукта).

СТРУКТУРА ПРОГРАММНЫХ ПРОДУКТОВ

В большей степени программные продукты не являются монолитом и имеют *конструкцию* (архитектуру) построения - состав и взаимосвязь программных модулей.

Модуль - это самостоятельная часть программы, имеющая определенное назначение и обеспечивающая заданные функции обработки автономно от других программных модулей.

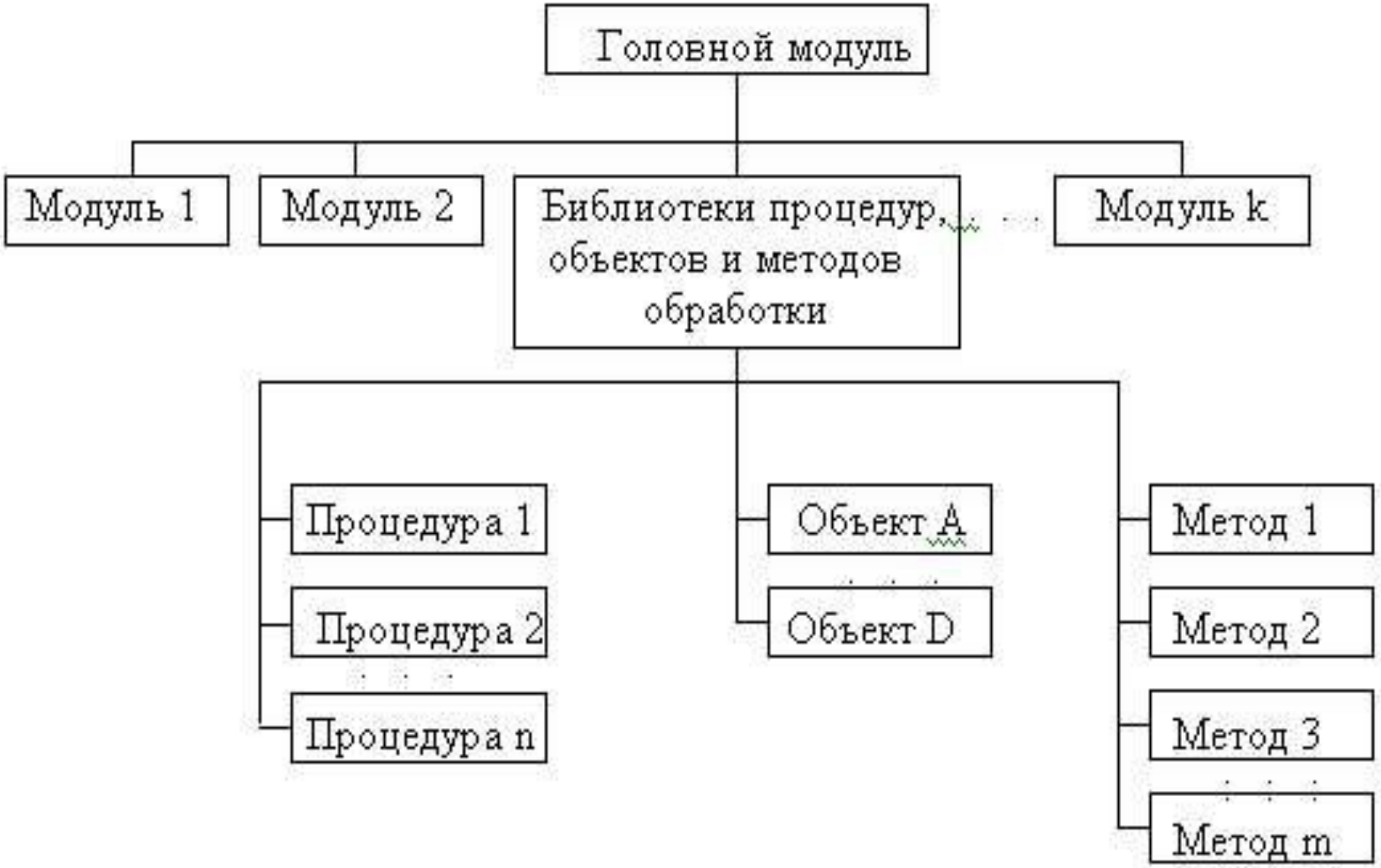
Таким образом, программный продукт обладает *внутренней организацией,* или *внутренней структурой,* образованной взаимосвязанными программными модулями.

Это справедливо для сложных и многофункциональных программных продуктов, которые часто называются *программными системами.*

Таким образом, структуризация программных продуктов преследует основные цели:

- распределить работы по исполнителям, обеспечив приемлемую их загрузку и требуемые сроки разработки программных продуктов;
- построить календарные графики проектных работ и осуществлять их координацию в процессе создания программных изделий;
- контролировать трудозатраты и стоимость проектных работ и др.

СТРУКТУРА ПРОГРАММНОГО ПРОДУКТА



Среди множества модулей различают:

- **головной модуль** - управляет запуском программного продукта (существует в единственном числе);
- **управляющий модуль** - обеспечивает вызов других модулей на обработку;
- **рабочие модули** - выполняют функции обработки;
- **сервисные модули и библиотеки, утилиты** - осуществляют обслуживающие функции.

Структурно-сложные программные продукты разрабатываются как *пакеты программ,* и чаще всего они имеют прикладной характер - *пакеты прикладных программ,* или ППП.

ППП *(application program package)* - это система программ, предназначенных для решения задач определенного класса.

Компоненты ППП объединены общими данными (базой данных), информационно и функционально связаны между собой и обладают свойством *системности,* т.е. объединению программ присуще новое качество, которое отсутствует для отдельного компонента ППП. Структура ППП, как правило, многомодульная.

ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

Диалоговый режим

Системы, поддерживающие диалоговые процессы, классифицируются на:

- **системы с *жестким сценарием диалога*** - стандартизированное представление информации обмена;
- ***дескрипторные системы*** - формат ключевых слов сообщений;
- ***тезаурусные системы*** - семантическая сеть дескрипторов, образующих словарь системы (аналог - гипертекстовые системы);
- **системы с *языком деловой прозы*** - представление сообщений на языке, естественном для профессионального пользования.

Наиболее просты для реализации и распространены диалоговые системы с жестким сценарием диалога, которые предоставлены в виде:

- **меню- диалог** инициируется программой; пользователю предлагается выбор альтернативы функций обработки из фиксированного перечня; предоставляемое меню может быть иерархическим и содержать вложенные подменю следующего уровня;
- **действия запрос-ответ** - фиксирован перечень возможных значений, выбираемых из списка, или ответы типа *Да/Нет;*
- **запрос по формату** - с помощью ключевых слов, фраз или путем заполнения экранной формы с регламентированным по составу и структуре набором реквизитов осуществляется подготовка сообщений.

Диалоговый процесс управляется согласно созданному сценарию, для которого определяются:

- **точки** (момент, условие) начала диалога;
- **инициатор диалога** - человек или программный продукт;
- **параметры и содержание диалога** - сообщения, состав и структура меню, экранные формы и т.п.;
- **реакция программного продукта** на завершение диалога.

Описание сценария диалога выполняют:

- **блок-схема**, в которой предусмотрены блоки выдачи сообщений и обработки полученных ответов;
- **ориентированный граф,** вершины которого - сообщения и выполняемые действия, дуги - связь сообщений; словесное описание;
- **специализированные объектно-ориентированные** языки построения сценариев.

В ряде СУБД и электронных таблиц, текстовых редакторов существуют различные типы *диалоговых окон* содержащих разнообразные объекты управления:

- тексты сообщения;
- поля ввода информации пользователя;
- списки возможных альтернатив для выбора;
- кнопки и т.п.

В среде электронных таблиц и текстовых редакторов имеются возможности настройки главных меню (удаление ненужных, добавление новых режимов и команд), создания системы подсказок с помощью встроенных средств и языков программирования.

Графический интерфейс пользователя

Графический интерфейс пользователя (Graphics User Interface - GUI) - ГИП является обязательным компонентом большинства современных программных продуктов, ориентированных на работу конечного пользователя.

К графическому интерфейсу пользователя предъявляются высокие требования как с чисто инженерной, так и с художественной стороны разработки, при его разработке ориентируются на возможности человека.

К числу типовых объектов управлении графического интерфейса относятся:

- **метка (label)** - постоянный текст, не подлежащий изменению при работе пользователя с экранной формой (например, слова *Фамилия Имя Отчество);*
- **текстовое окно (text box)** - используется для ввода информации произвольного вида, отображения хранимой информации в базе данных (например, для ввода фамилии студента);
- **рамка (frame)** - объединение объектов управления в группу по функциональному или другому принципу (например, для изменения их параметров);
- **командная кнопка (command button)** - обеспечивает передачу управляющего воздействия, например, кнопки «Cance1», «ОК», «Отмена»; выбор режима обработки типа «Ввод», «Удаление», «Редактирование», «Выход» и др.;

- **кнопка-переключатель «option button»** - для альтернативного выбора кнопки из группы однотипных кнопок (например, *семейное положение);*
- **помечаемая кнопка «check button»** - для аддитивного выбора несколько кнопок из группы однотипных кнопок (например, *факультатив для посещения);*
- **окно-список (list box)** - содержит список альтернативных значений для выбора (например, Спортивная секция);
- **комбинированное окно (combo box)** - объединяет возможности окна-списка и текстового окна (например, Предметы по выбору - можно указать новый предмет или выбрать один из предлагаемого списка);

- **линейка горизонтальной прокрутки** - для быстрого перемещения внутри длинного списка или текста по горизонтали;
- **линейка вертикальной прокрутки** - для быстрого перемещения внутри длинного списка или текста по вертикали;
- **окно-список каталогов;**
- **окно-список накопителей;**
- **окно-список файлов и др.**

Стандартный графический интерфейс пользователя должен отвечать ряду требований:

- **поддерживать информационную технологию работы пользователя с программным продуктом** - содержать привычные и понятные пользователю пункты меню, соответствующие функциям обработки, расположенные в естественной последовательности использования;
- **ориентироваться на конечного пользователя**, который общается с программой на *внешнем* уровне взаимодействия;
- **удовлетворять правилу "шести"** - в одну линейку меню включать не более 6 понятий, каждое из которых содержит не более 6 опций;
- **графические объекты** сохраняют свое стандартизованное назначение и по возможности местоположение на экране.