1. Сложности при разработке программного обеспечения

На сегодняшний день очень возросла сложность создаваемых систем. Проявлениями этой сложности являются не только увеличение размеров и функциональности. Не меньше, если не больше, проблем порождается частой сменой потребностей пользователей и ростом требований к качеству систем.

Традиционные способы разрешения проблем- - увеличение коллективов разработчиков, их специализация, Это ведёт к повышению затрат, еще большим трудностям согласования результатов и сборки готовых систем.

Серьезным шагом к победе над сложностью явилось появление объектно-ориентированного программирования (ООП).

Несмотря на свои преимущества ООП полностью не устранило проблем недостаточного взаимопонимания разработчиков и пользователей, неэффективного управления разработкой в условиях изменяющихся требований, неконтролируемости изменений в процессе выполнения работ, субъективности в оценке качества продуктов разработки и т.д.

2. Причины возникновения ООП

На первых компьютерах для написания программ сначала применялись машинные коды, а затем язык ассемблера. Но этот язык не соответствует сегодняшним стандартам. По мере роста сложности программ оказалось, что разработчики не в состоянии помнить всю информацию, нужную для отладки и совершенствования их программ. Какие значения хранятся в регистрах? Есть ли уже переменная с этим именем? Какие переменные надо инициализировать, перед тем как передать управление следующему коду?

Rgm.asm

Частично эти проблемы решили первые языки высокого уровня: Фортран, Кобол, Алгол, С.

Но рост сложности программ продолжался, и появились проекты, в которых ни один программист не мог удержать в голове все детали. Над проектами стали работать команды программистов.

Значительная взаимозависимость частей ПО мешает создавать ПО по типу конструирования материальных объектов.

Например, здание, автомобиль и электроприборы обычно собираются из готовых компонент, которые не надо разрабатывать «с нуля».

Многократное использование ПО – цель, к которой постоянно стремятся, но и которой редко достигают. Из программной системы тяжело извлечь независимые фрагменты. Облегчает эту задачу именно – ООП.

Базовые понятия

Чтобы эффективно использовать ООП, требуется рассматривать задачи иным способом, нежели это принято в процедур□ном программировании.

Известно утверждение, применимое к естественным языкам, что язык, на котором высказывается идея, направляет мышление. Как для компьютерных, так и для естественных языков справедливо: язык направляет мысли, но не предписывает их.

Аналогично, объектно-ориентированная техника не снабжает программиста новой вычислительной мощностью, которая бы позволила решить проблемы, недоступные для других средств. Но объектно-ориентированный подход делает задачу проще и приводит ее к более естественной форме.

ООП часто называется новой парадигмой программирования.

Парадигма программирования – способ концептуализации, который определяет, как проводить вычисления и как работа, выполняемая компьютером, должна быть структурирована и организована.

Процесс разбиения задачи на отдельные, структурно связанные, части, называется декомпозицией.

При процедурной декомпозиции в задаче выделяются алгоритмы и обрабатываемые ими структуры данных, при логической – правила, связывающие отдельные понятия.

При объектно-ориентированной декомпозиции в задаче выделяются классы и способы взаимодействия объектов этих классов друг с другом.

Главный способ борьбы со сложностью ПО – абстрагирование, т.е. способность отделить логический смысл фрагмента программы от проблемы его реализации.

Абстрагирование – это выделение существенных характеристик объекта, которые отличают его от всех других видов объектов. Абстрагирование позволяет отделить самые существенные особенности поведения от несущественных.

Модульность – улучшенный метод создания и управления совокупностями имен и связанными с ними значениями. Суть модуля состоит в разбиении пространства имен на две части: открытая (public) часть является доступной извне модуля, закрытая (private) часть доступна только внутри модуля.

Иерархия – ранжированная (упорядоченная) система абстракций.

Обзор существующих методологий

4.1. Методология процедурно-ориентированного программирования Появление первых электронных вычислительных машин или компьютеров ознаменовало новый этап в развитии техники вычислений. Казалось, достаточно разработать последовательность элементарных действий, каждое из которых преобразовать в понятные компьютеру ин□струкции, и любая вычислительная задача может быть решена.

Эта идея оказалась настолько жизнеспособной, что долгое время доминировала над всем процессом разработки программ. Появились специальные языки программирования, которые позволили преобразовывать отдельные вычислительные операции в соответствующий программный код. К таким языкам относятся Assembler, C, Pascal и другие. Основой данной методологии разработки программ являлась процедурная или алгоритмическая организация структуры программного кода. Это было настолько естественно для решения вычислительных задач, что ни у кого не вызывала сомнений целесообразность такого подхода.

Исходным понятием этой методологии являлось понятие алгоритма, под которым, в общем случае, понимается некоторое предписание выполнить точно определенную последовательность действий, направленных на достижение заданной цели или решение поставленной задачи.

Принято считать, что сам термин алгоритм происходит от имени средневекового математика Аль-Хорезми, который в 825 г. описал правила выполнения арифметических действий в десятичной системе счисления. Вся история математики тесно связана с разработкой тех или иных алгоритмов решения актуальных для своей эпохи задач.

Более того, само понятие алгоритма стало предметом соответствующей теории – теории алгоритмов, которая занимается изучением общих свойств алгоритмов.

Поэтому какое-то время языки программирования назывались алгоритмическими, а первое графическое средство документирования программ получило название блок-схемы алгоритма. Потребности практики не всегда требовали установления вычислимости конкретных функций или разрешимости отдельных задач.

В языках программирования возникло и закрепилось новое понятие процедуры, которое конкретизировало общее понятие алгоритма применительно к решению задач на компьютерах. Так же, как и алгоритм, процедура представляет собой законченную последовательность действий или операций, направленных на решение отдельной задачи. В языках программирования появилась специальная синтаксическая конструкция, которая получила название процедуры.

Со временем разработка больших программ превратилась в серьезную проблему и потребовала их разбиение на более мелкие фрагменты.

Основой для такого разбиения как раз и стала процедурная декомпозиция, при которой отдельные части программы или модули представляли собой совокупность процедур для решения некоторой совокупности задач.

Главная особенность процедурного программирования заключается в том, что программа всегда имеет начало во времени или начальную процедуру и окончание. При этом вся программа может быть представлена визуально в виде направленной последовательности блоков.

Графическое представление программы в виде последовательных процедур

Интенсивное использование условных операторов и оператора безусловного перехода стало предметом острых дискуссий среди специалистов по программированию.

Дело в том, что бесконтрольное применение в программе оператора безусловного перехода goto способно серьезно осложнить понимание кода. Именно с тех пор принято считать хорошим стилем программирование – программирование без goto.

Рассмотренные идеи способствовали становлению системы взглядов на процесс разработки программ и написания программных кодов, которая получила название методологии **структурного программирования**. Основой данной методологии является процедурная декомпозиция программной системы и организация отдельных модулей в виде совокупности выполняемых процедур.

Как вспомогательное средство структуризации программного кода было рекомендовано использование отступов в начале каждой строки, которые должны выделять вложенные циклы и условные операторы. Все это призвано способствовать пониманию или читабельности самой программы.

V.cpp

В этот период основным показателем сложности разработки программ считали ее размер. Вполне серьезно обсуждали такие оценки сложности программ, как количество строк программного кода.

Общая трудоемкость разработки программ оценивалась специальной единицей измерения – «человеко-месяц» или «человеко-год». А профессионализм программиста напрямую связывался с количеством строк программного кода, который он мог написать и отладить в течение, скажем, месяца.

4.2. Методология объектно-ориентированного программирования

2.2. Диаграмма классов (class diagram)

Диаграмма классов – диаграмма, которая описывает структуру приложения (программного проекта, системы). Цель данной диаграммы показать классы, их свойства (атрибуты, методы), связи между классами, интерфейсы и т.д. Приведем пример типичной диаграммы