Вопросы к экзамену по дисциплине «Основы алгоритмизации и программирования» для студентов 2-го курса специальности 09.02.07 «Информационные системы и программирование»

1. Понятие объектно-ориентированного программирования

ООП – технология создания сложного программного обеспечения, которая представляет программный продукт в виде совокупности объектов.

1. Понятия класса и объекта в ООП

Объект – модель реальной сущности в программной системе.

Класс – описание предметной области.

1. Принципы ООП

В ООП существует ряд принципов, лежащих в основе правил создания и использования всех структурных элементов, включая классы, объекты, методы и прочие компоненты:

1) Инкапсуляция

2) Наследование

3) Полиморфизм

1. Принцип инкапсуляции в ООП

Инкапсуляция – объединение в единое целое данных и методов обработки этих данных, позволяющие изолировать объект от внешнего окружения.

Пример:

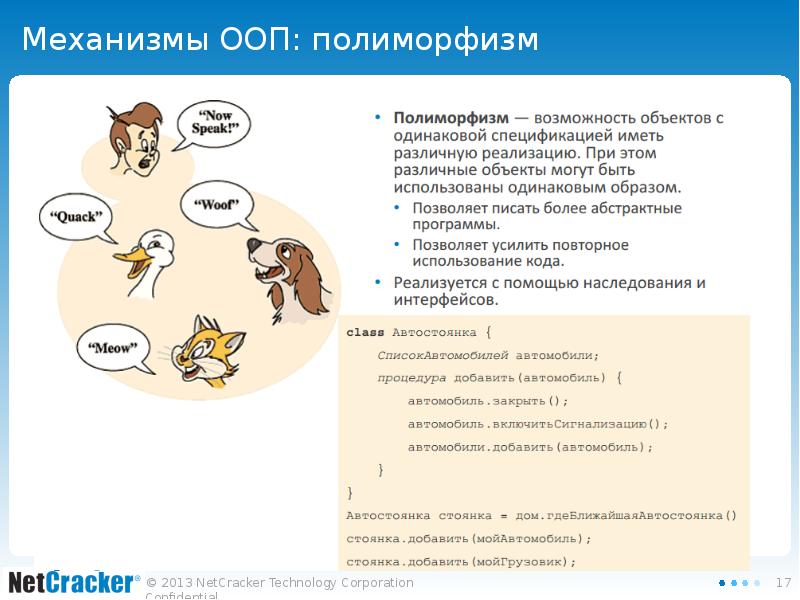
Пользователь может взаимодействовать с объектом только через этот интерфейс. Реализуется с помощью ключевого слова: public.

Пользователь не может использовать закрытые данные и методы. Реализуется с помощью ключевых слов: private, protected, internal.

1. Принцип полиморфизма в ООП

Полиморфизм – возможность использовать одинаковые имена для методов, входящих в различные классы, таким образом имеющих различные алгоритмы.

Пример: вычисление периметра в четырёхугольнике, параллелограмме, квадрате называется одинаково, но реализация разная.



1. Принцип наследования в ООП

Наследование – возможность описать новый класс на базе существующего, при этом свойства и функциональность родительского класса заимствуется новым классом.

Пример:



1. Понятие исключения, виды исключений, способы обработки исключений

Исключение - событие периода выполнения, которое может стать причиной отказа программы.

Виды:

- Аппаратные. Генерируются процессором. К ним относятся, например:

* деление на 0;
* выход за границы массива;
* обращение к невыделенной памяти;
* переполнение разрядной сетки.

- Программные.

Генерируемые операционной системой и прикладными программами – возникают тогда, когда программа их явно инициирует. Когда встречается аномальная ситуация, та часть программы, которая ее обнаружила, может сгенерировать, или ***возбудить***, исключение.

1. Понятие оптимизации программного кода, способы оптимизации

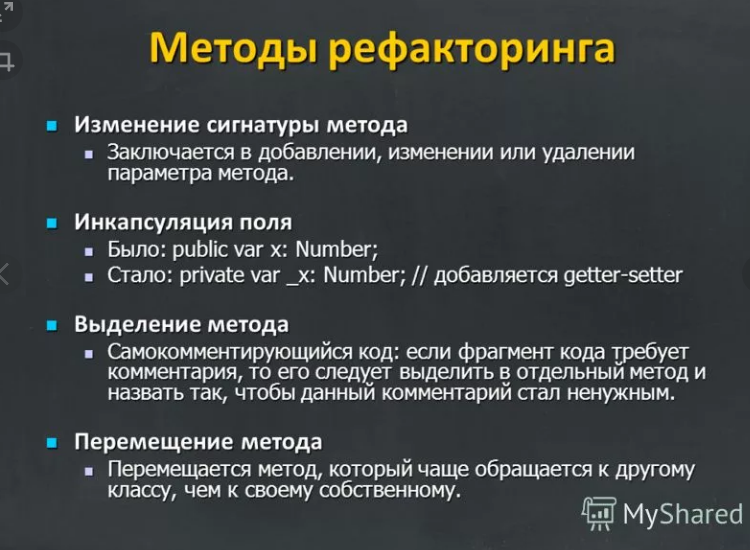
Оптимизация – обработка, связанная с переупорядочиванием и изменением операций в компилируемой программе с целью получения более эффективной результирующей объектной программы.

Способы:

1. Линейных участков программы
2. Логических выражений
3. Циклов
4. Вызов процедур и функций
5. Других конструкций входного языка
6. Понятие рефакторинга программного кода, способы рефакторинга

Рефакторинг – процесс изменения программы, при котором не меняется внешнее поведение кода, но улучшается его внутренняя структура.

Способы:



1. Понятие тестирования программного обеспечения. Виды тестирования

Тестирование ПО **—** процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов.

Виды:

1. Функциональные — это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям.
2. Нефункциональные:

- контролирует работоспособность системы под различными нагрузками;

- адаптация приложения для различных платформ;

- требования к масштабируемости приложения;

- гарантия продолжения работы приложения в случаях непредвиденных ситуаций.

1. Связанные с изменениями – тестирование, связанное с изменениями, предназначено для проверки исправления дефектов и проверки работоспособности системы после внесения изменений, таких так добавление нового функционала или корректировки старого.
2. Понятие отладки программного обеспечения. Способы отладки

Отладка - это поиск (локализация), анализ и устранение ошибок в программном обеспечении, которые были найдены во время тестирования.

Методы:

* ручного тестирования - при обнаружении ошибки необходимо выполнить тестируемую программу вручную, используя тестовый набор, при работе с которым была обнаружена ошибка.
* Индукции - Метод основан на тщательном анализе симптомов ошибки, которые могут проявляться как неверные результаты вычислений или как сообщение об ошибке.
* Дедукции - по методу дедукции вначале формируют множество причин, которые могли бы вызвать данное проявление ошибки. Затем анализируя причины, исключают те, которые противоречат имеющимся данным.
* обратного прослеживания - строится гипотеза о значениях основных переменных, которые могли бы привести к получению имеющегося результата. Исходя из этой гипотезы, делают предложения о значениях переменных в предыдущей точке. Процесс продолжают, пока не обнаружат причину ошибки.

1. Понятие алгоритма. Свойства алгоритмов

Алгоритм – последовательность действий, приводящая к определённому результату за конечное число шагов.

Св-ва:

- определённость – получение однозначного результата при заданных исходных данных.

- результативность – алгоритм должен приводить к результату.

- массовость – алгоритм должен быть применим для решения множества однотипных задач.

- дискретность – алгоритм состоит из отдельных шагов, понятных исполнителю.

1. Способы записи алгоритма

- словесный

- графический (в виде блок-схем)

- на алгоритмическом языке

- на языке программирования

1. Базовые алгоритмические структуры, их описание

- линейный – действия выполняются последовательно, друг за другом

- разветвляющийся – содержит блок условие, в результате выполнения которого возможно 2 варианта действий

- циклический – содержит повторение действий по некоторому условию

1. Понятие интегрированной среды разработки (ИСР). Примеры ИСР. Состав ИСР

Интегрированная среда разработки - система программных средств, используемая программистами для разработки программного обеспечения (ПО).

Состав:

* средства управления проектами
* [система управления версиями](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8)
* инструменты для упрощения разработки [интерфейса пользователя](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F)
* стандартные заготовки, упрощающие разработку стандартных задач

Примеры:

* Microsoft Visual Studio
* NetBean
* PyCharm
* Turbo Pascal

1. Язык программирования C#. Возникновение, основное назначение, особенности

C# — современный объектно-ориентированный и типобезопасный язык программирования.

C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET.

Разработан в 1998-2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core.

Особенности:

* Типы данных имеют фиксированный размер (32-битный int и 64-битный long), что повышает «мобильность» языка и упрощает программирование, так как вы всегда знаете точно, с чем вы имеете дело.
* сигнатура метода (называемая делегатом), позволяющая реализовывать безопасное оповещение о происходящем событии;
* свойство, определяющее функции акцессора для закрытой переменной;
* атрибут, предоставляющий данные о типе функции/переменной/объекта при написании кода;
* внутристрочное комментирование (для XML-документов);
* LINQ – языковой запрос в отношении различных источников данных.

1. Понятие интерфейса. Виды интерфейсов

Интерфейс – это «мост», посредник между человеком, программами и машинами, иными системами. Это инструменты взаимодействия, с помощью которых одна система контактирует и строит диалог с другой.

Виды:

Пользовательский интерфейс - это способ взаимодействия пользователя и программы.

Графический интерфейс – это интерфейс, предоставляемый пользователю, суперпользователю или администратору для простого взаимодействия с системой.  
Текстовый интерфейс  
Голосовой интерфейс   
Жестовый интерфейс   
Интерфейс программ  
Интерфейс программ с машинами  
Аппаратный интерфейс

1. Принципы разработки интерфейса

Принципы разработки интерфейса - это высокоуровневые концепции и представления, которые могут использоваться при проектировании программного обеспечения.



1. Понятие жизненного цикла программного обеспечения. Модели жизненного цикла

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.

**Модель жизненного цикла программного обеспечения** — структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта.

#### **Модель кодирования и устранения ошибок**

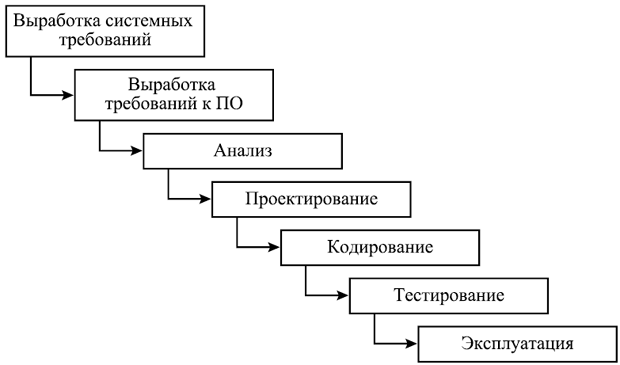
Совершенно простая модель, характерная для студентов ВУЗов. Именно по этой модели большинство студентов разрабатывают, ну скажем лабораторные работы.  
Данная модель имеет следующий алгоритм:

1. Постановка задачи
2. Выполнение
3. Проверка результата
4. При необходимости переход к первому пункту

#### **Модель на основе разработки прототипа**

Данная модель основывается на разработки прототипов и прототипирования продукта.  
**Прототипирование** используется на ранних стадиях жизненного цикла программного обеспечения:

1. Прояснить не ясные требования (прототип UI)
2. Выбрать одно из ряда концептуальных решений (реализация сценариев)
3. Проанализировать осуществимость проекта
4. Этапы разработки приложений



1. Понятие строки в C#. Методы строк

**Строка** — это объект типа String, значением которого является текст.

* Compare: сравнивает две строки с учетом текущей культуры (локали) пользователя
* Contains: определяет, содержится ли подстрока в строке
* Concat: соединяет строки
* CopyTo: копирует часть строки, начиная с определенного индекса в массив
* IndexOf: находит индекс первого вхождения символа или подстроки в строке
* Insert: вставляет в строку подстроку
* Join: соединяет элементы массива строк
* LastIndexOf: находит индекс последнего вхождения символа или подстроки в строке
* Replace: замещает в строке символ или подстроку другим символом или подстрокой
* Split: разделяет одну строку на массив строк
* Substring: извлекает из строки подстроку, начиная с указанной позиции
* ToLower: переводит все символы строки в нижний регистр
* Trim: удаляет начальные и конечные пробелы из строки

1. Понятие массива в C#. Одномерные и многомерные массивы, принципы работы с массивами

**Массив**-это набор однотипных элементов, хранящихся в смежных ячейках памяти.

Одномерный массив — это конечное количество однотипных элементов, объединенных общим именем.

**Двумерный** **массив** **-** **это** **одномерный** **массив,** **элементами** **которого** **являются** **одномерные** **массивы.** Другими словами, это набор однотипных данных, имеющий общее имя, доступ к элементам которого осуществляется по двум индексам.

Например, определим массив целых чисел:

|  |  |
| --- | --- |
| 1 | int[] numbers; |

После определения переменной массива мы можем присвоить ей определенное значение:

|  |  |
| --- | --- |
| 1 | int[] nums = new int[4]; |

Здесь вначале мы объявили массив nums, который будет хранить данные типа int. Далее используя операцию new, мы выделили память для 4 элементов массива: new int[4]. Число 4 еще называется **длиной массива**. При таком определении все элементы получают значение по умолчанию, которое предусмотренно для их типа. Для типа int значение по умолчанию - 0.

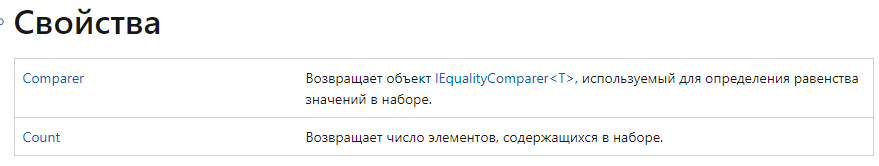
Также мы сразу можем указать значения для этих элементов:

|  |  |
| --- | --- |
|  | int[] nums2 = new int[4] { 1, 2, 3, 5 };    int[] nums3 = new int[] { 1, 2, 3, 5 };    int[] nums4 = new[] { 1, 2, 3, 5 };    int[] nums5 = { 1, 2, 3, 5 };  Методы: |

1. Понятие множества в C#. Класс HashSet – свойства, методы

**Множество** (set) — это структура данных, представляющая собой не организованный набор уникальных элементов одного типа.

HashSet — неупорядоченная структура, то есть элементы этой коллекции могут быть неупорядоченными — например 2, 14, 4, 55, 53 и т.д.



1. Понятие списка в C#. Класс List<T> - свойства, методы
2. Понятие файла в C#. Классы StreamWriter и StreamReader – свойства, методы
3. Принцип модульности при разработке приложений. Добавление класса в проект и его использование
4. Создание и использование библиотеки класса