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

Оглавление

[1. Понятие объектно-ориентированного программирования 2](#_Toc103806185)

[2. Понятия класса и объекта в ООП 2](#_Toc103806186)

[3. Принципы ООП 2](#_Toc103806187)

[4. Принцип инкапсуляции в ООП 2](#_Toc103806188)

[5. Принцип полиморфизма в ООП 2](#_Toc103806189)

[6. Принцип наследования в ООП 2](#_Toc103806190)

[7. Понятие исключения, виды исключений, способы обработки исключений 2](#_Toc103806191)

[8. Понятие оптимизации программного кода, способы оптимизации 2](#_Toc103806192)

[9. Понятие рефакторинга программного кода, способы рефакторинга 3](#_Toc103806193)

[10. Понятие тестирования программного обеспечения. Виды тестирования 3](#_Toc103806194)

[11. Понятие отладки программного обеспечения. Способы отладки 3](#_Toc103806195)

[12. Понятие алгоритма. Свойства алгоритмов 3](#_Toc103806196)

[13. Способы записи алгоритма 3](#_Toc103806197)

[14. Базовые алгоритмические структуры, их описание 3](#_Toc103806198)

[15. Понятие интегрированной среды разработки (ИСР). Примеры ИСР. Состав ИСР 4](#_Toc103806199)

[16. Язык программирования C#. Возникновение, основное назначение, особенности 4](#_Toc103806200)

[17. Понятие интерфейса. Виды интерфейсов 4](#_Toc103806201)

[19. Понятие жизненного цикла программного обеспечения. Модели жизненного цикла 5](#_Toc103806202)

[20. Этапы разработки приложений 5](#_Toc103806203)

[21. Понятие строки в C#. Методы строк 5](#_Toc103806204)

[22. Понятие массива в C#. Одномерные и многомерные массивы, принципы работы с массивами 5](#_Toc103806205)

[23. Понятие множества в C#. Класс HashSet – свойства, методы 6](#_Toc103806206)

[24. Понятие списка в C#. Класс List<T> - свойства, методы 6](#_Toc103806207)

[25. Понятие файла в C#. Классы StreamWriter и StreamReader – свойства, методы 7](#_Toc103806208)

[26. Принцип модульности при разработке приложений. Добавление класса в проект и его использование 7](#_Toc103806209)

[27. Создание и использование библиотеки класса 8](#_Toc103806210)

# Понятие объектно-ориентированного программирования

ООП – это технология создания сложного ПО, которое представляет программный продукт в виде совокупности объектов.

# Понятия класса и объекта в ООП

Класс – представляет собой формальное описание некоторой предметной области.

Объект - это модель реальной сущности в программной системе.

Объекты является представителями класса

# Принципы ООП

1. Инкапсуляция
2. Наследование
3. Полиморфизм

# Принцип инкапсуляции в ООП

Инкапсуляция – объединение в единое целое данных и методов обработки этих данных, позволяющее изолировать объект от внешнего окружения с целью минимизировать внешнее воздействие.

# Принцип полиморфизма в ООП

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

# Принцип наследования в ООП

Наследование - возможность описать новых класс на базе / основе существующего, при этом свойства и функциональность родительского класса взаимствуются новым классом.

# Понятие исключения, виды исключений, способы обработки исключений

Исключения – ошибки, возникающие при выполнение программы, которые трудно предусмотреть или предвидеть, а иногда и вовсе невозможно.

Виды:

Базовым видом исключения является Exception

Способ обработки исключения **try...catch...finally**

# Понятие оптимизации программного кода, способы оптимизации

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

Способы оптимизации:

1. Линейных участков программы
2. Логических выражений
3. Циклов
4. Вызов процедур и функций
5. Других конструкций входного языка

# Понятие рефакторинга программного кода, способы рефакторинга

Рефакторинг – это процесс такого изменения программной системы при котором не меняется внешнее поведение кода, но улучшается его внутренняя структура.

# Понятие тестирования программного обеспечения. Виды тестирования

Тестирование программного обеспечения— проверка соответствия реальных и ожидаемых результатов поведения программы.

Виды:

* Компонентное/модульное тестирование
* Интеграционное тестирование
* Системное тестирование
* Приемочное тестирование

# Понятие отладки программного обеспечения. Способы отладки

Отладка - этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.

Способы:

1. Пошаговая отладка программ с заходом в подпрограммы;
2. Пошаговая отладка программ с выполнением подпрограммы как одного оператора;
3. Выполнение программы до точки останова.

# Понятие алгоритма. Свойства алгоритмов

Алгоритм - система последовательных операций (в соответствии с определёнными правилами) для решения какой-н. задачи.

Свойства:

1. Массовость
2. Дискретность
3. Результативность
4. Детерминированность (определенность)

# Способы записи алгоритма

1. Словесный
2. Графический, в виде блок-схем
3. На алгоритмическом языке
4. На языке программирования

# Базовые алгоритмические структуры, их описание

1. Линейный - представляет собой последовательное выполнение операций
2. Разветвляющийся алгоритм:

* неполную развилку - выполняет последовательность операций только по одной из веток
* полную развилку - выполняет последовательность операций по каждой из двух веток (при выполнении или невыполнении условия)
* ветвление - представляет собой операцию множественного выбора, при которой проверка условия может иметь более двух возможных вариантов

1. циклический алгоритм:

* цикл с предусловием - осуществляет проверку условия перед началом своего выполнения.
* цикл с постусловием - всегда выполняется хотя бы один раз, поскольку проверка условия осуществляется после выполнения операций цикла.
* параметрический цикл - это цикл с заданным числом повторений

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

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

Примеры:

* Microsoft Visual Studio
* Visual Studio Code
* PhpStorm
* IntelliJ IDEA

Состав ИСР:

текстовый редактор;

компилятор и / или интерпретатор;

средства автоматизации сборки; отладчик.

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

C# - объектно-ориентированный язык программирования.

Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года.

В настоящий момент на нём пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов. Позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

# Понятие интерфейса. Виды интерфейсов

Интерфейс — граница между двумя функциональными объектами, требования к которой определяются стандартом; совокупность средств, методов и правил взаимодействия между элементами системы

Виды:

1. Текстовый
2. Тактильный
3. Жестовой
4. Голосовой
5. Материальный
6. Принципы разработки интерфейса
7. контроль пользователем интерфейса
8. уменьшение загрузки памяти пользователя
9. последовательность пользовательского интерфейса

# Понятие жизненного цикла программного обеспечения. Модели жизненного цикла

Жизненный цикл - период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации.

Модели:

1. Каскадная - модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки.
2. Инкрементная - это метод, в котором проект проектируется, реализуется и тестируется инкрементно (то есть каждый раз с небольшими добавлениями) до самого окончания разработки.
3. Эволюционная - система разрабатывается в виде последовательности блоков структур (конструкций).

# Этапы разработки приложений

1. Анализ предметной области и формирование требований(концепции)
2. Проектирование
3. Реализация
4. Тестирование
5. Ввод в эксплуатацию
6. Сопровождение проекта(Эксплуатация)
7. Вывод из эксплуатации

# Понятие строки в C#. Методы строк

Строки – набор символов.

Методы строк:

1. Foreach перебор символов строки
2. For перебор символов по их номерам
3. Split разделяет одну строку на массив строк
4. Contains определяет содержится ли подстрока в строке
5. IndexOf находит индекс первого вхождения символа или подстроки в строке
6. LastIndexOf находит индекс последнего вхождения символа или подстроки в строке
7. Insert вставляет в строку подстроку
8. ToLower переводит все символы строки в нижний регистр
9. ToUpper переводит все символы строки в верхний регистр
10. Trim удаляет начальный и конечный пробелы из строки

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

Массив – упорядоченный набор однотипных данных, имеющих имя.

Массивы характеризуются таким понятием как ранг или количество измерений. массивы, которые имеют одно измерение (то есть их ранг равен 1) - такие массивы можно представлять в виде ряда (строки или столбца) элемента. Но массивы также бывают многомерными. У таких массивов количество измерений (то есть ранг) больше 1.

Массивы которые имеют два измерения (ранг равен 2) называют двухмерными.

Для работы с массивами будь то задание, или вывод массива создаётся цикл

For – перебирает по номерам

Foreach – перебирает по символьно.

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

Множество HashSet<T> – неупорядоченный набор однотипных данных.

Свойства:

* Comparer
* Count

Методы:

* Add
* Union, Except, Intersect
* Clear
* Contains
* Remove и др.

# Понятие списка в C#. Класс List<T> - свойства, методы

List<T> - представляет собой простейший список однотипных объектов.

Свойства

* Ёмкость List<T> (N)
* Длина списка (Count)

Методы:

* **Add(T item)**: добавление нового элемента в список
* **AddRange(IEnumerable<T> collection)**: добавление в список коллекции или массива
* **CopyTo(T[] array)**: копирует список в массив array
* **Contains(T item)**: возвращает true, если элемент item есть в списке
* **Clear()**: удаляет из списка все элементы
* **IndexOf(T item)**: возвращает индекс первого вхождения элемента в списке
* **LastIndexOf(T item)**: возвращает индекс последнего вхождения элемента в списке
* **Insert(int index, T item)**: вставляет элемент item в список по индексу index. Если такого индекса в списке нет, то генерируется исключение
* **InsertRange(int index, collection)**: вставляет коллекцию элементов collection в текущий список начиная с индекса index. Если такого индекса в списке нет, то генерируется исключение
* **Remove(T item)**: удаляет элемент item из списка, и если удаление прошло успешно, то возвращает true. Если в списке несколько одинаковых элементов, то удаляется только первый из них
* **Reverse()**: изменяет порядок элементов
* **Sort()**: сортировка списка

# Понятие файла в C#. Классы StreamWriter и StreamReader – свойства, методы

Файл – это набор данных, который хранится на внешнем запоминающем устройстве (например на жестком диске).

Для записи в текстовый файл используется класс StreamWriter.

**Свойства:**

StreamWriter(string path, bool append, System.Text.Encoding encoding)

**Методы:**

* int Close()
* void Flush()
* Task FlushAsyncvoid Write()

Класс StreamReader позволяет нам легко считывать весь текст или отдельные строки из текстового файла.

**Свойства:**

StreamReader(string path, System.Text.Encoding encoding)

**Методы:**

* void Close()
* int Read()
* string ReadLine()
* string ReadToEnd()

# Принцип модульности при разработке приложений. Добавление класса в проект и его использование

Модульное программирование — это организация программы как совокупности небольших независимых блоков, называемых модулями.

Добавление класса:

1. В обозревателе решений в названии проекта нажимается ПКМ 🡪Добавить 🡪Класс
2. Задаётся имя и создаётся класс.
3. В обозревателе решений должен появиться класс
4. Прописывается решение, которое должен иметь проект класс.

Использование класса (подключение):

В блок using прописываем название проекта и наш класс.

# Создание и использование библиотеки класса

Библиотека класса – это отдельный проект, который компилируется в файле расширения dll и может подключаться в другие проекты.

Файл dll не имеет интерфейса.

Добавление класса:

1. Создаётся проект «Библиотека классов (NET. Framework)»
2. Задаётся имя, расположение файла и создаётся проект.
3. Сразу переименовывается начальное название класса
4. Прописывается решение, которое должен иметь проект и компилируется для сохранения.

Использование библиотеки (подключение):

1. Создаётся новый проект консольного, либо WPF приложения
   1. В обозревателе решений находим узел ссылки, нажимаем ПКМ 🡪 «Добавить ссылку»
   2. В обзоре, если не находим нужного нажимаем внизу «Обзор», находим проект библиотеки, выбираем файл с расширением dll и нажимаем «Ок»
   3. Ссылка на библиотеку должна добавиться и отображаться в обозреватели решения.
2. В блок using вписываем нашу библиотеку.