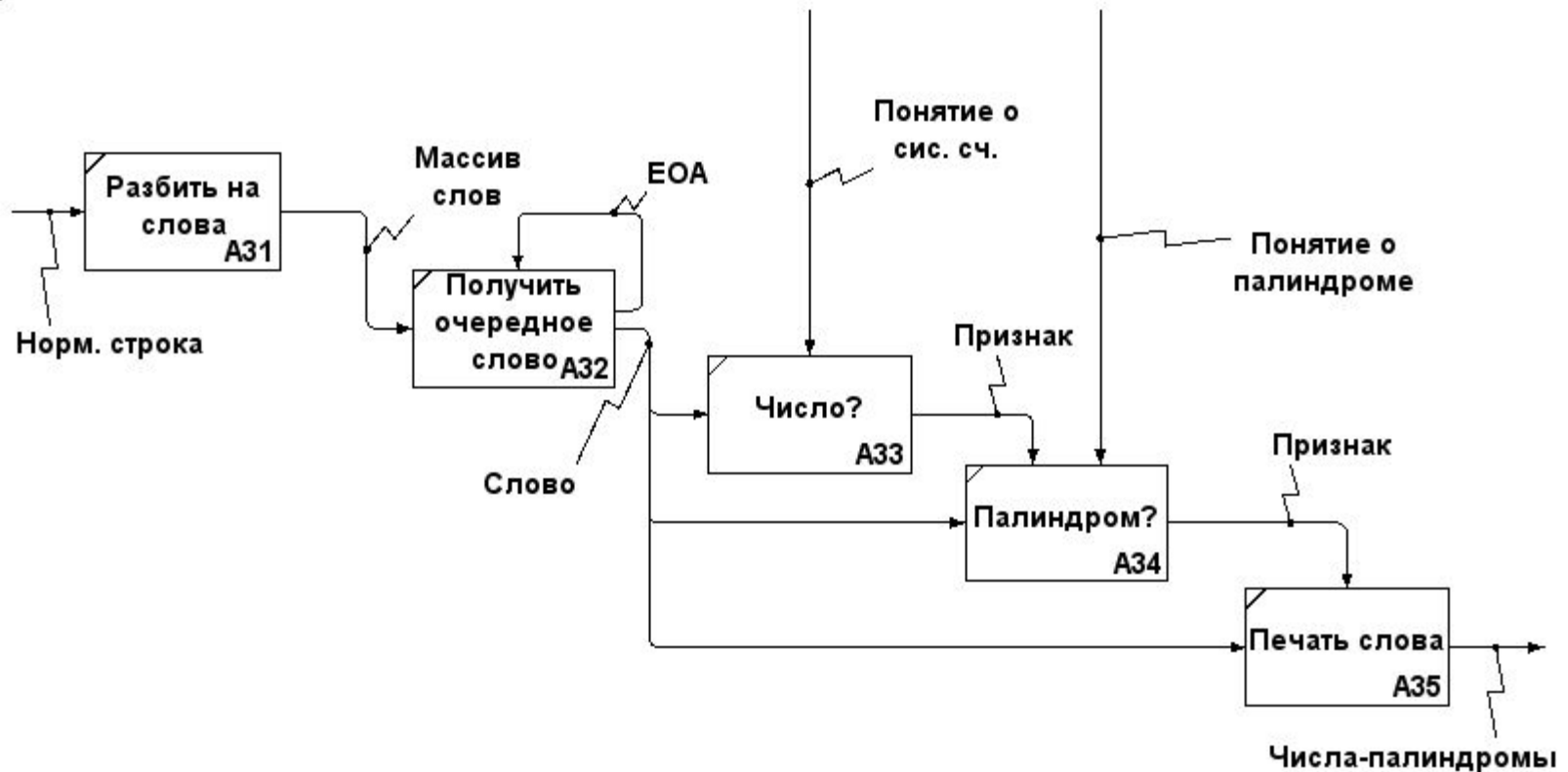


Моменты, на которые
необходимо обратить внимание
в программной реализации
домашнего задания

Найти в файле числа-палиндромы



Найти в файле числа-палиндромы



Спецификация подзадачи А33

Имя	IsNumber
Функция	Определяет является ли слово числом в десятичной системе счисления.(Наличие знака – признак не числа.)
Список параметров	Слово (строка), признак (да/нет)
Входные параметры	Слово
Выходные параметры	Признак
Внешние эффекты	Нет
Допущения	Слово не может быть пустым / Нет

Данные для тестирования

- K1: число
 - 5, 123, 77
- K2: не число
 - K2.1: слова только из букв
 - A, qwerty
 - K2.2: смесь из букв и цифр
 - 12a3, d8, 345x, -57, +2
 - K2.3: пустая строка

Алгоритм подзадачи А33 (в.1)

Вход: слово (не может быть пустым)

Выход: признак

признак = да

пока (признак == да) и (не конец слова) **делать**

если очередная буква слова не цифра **то**

признак = нет

все если

все пока

Алгоритм подзадачи А33 (в.2)

Вход: слово

Выход: признак

если слово пустое **то**

 признак = нет

иначе

 признак = да

пока (признак == да) и (не конец слова) **делать**

если очередная буква слова не цифра **то**

 признак = нет

все если

все пока

все если

Модульное тестирование (1)

Идея модульного тестирования состоит в том, чтобы писать тесты для каждой нетривиальной функции.

Преимущества

- Модульное тестирование облегчает обнаружение и устранение ошибок, позволяет достаточно быстро проверить не привело ли очередное изменение к появлению ошибок в оттестированных местах программы.
- Модульное тестирование можно использовать как документирование кода.

Модульное тестирование (2)

Преимущества (продолжение)

- Модульное тестирование способствует отделению интерфейса от реализации.
- Модульное тестирование поощряет внесение изменений.

Недостатки (мнимые?)

- Написание тестов увеличивает срок разработки.
- В процессе разработки программы требования могут измениться и придется менять тесты.
- «Мои подпрограммы слишком сложно протестировать».

Утверждения (asserts)

Утверждения – это код, используемый во время разработки, с помощью которого программа проверяет правильность своего выполнения.

Общие положения по применению утверждений.

- Используйте обработку ошибок для ожидаемых событий и утверждения для событий, которые происходить не должны.
- Используйте утверждения для документирования и проверки предусловий, постусловий, инвариантов цикла.
- Не помещайте выполняемый код в утверждения.

Pascal: директивы компилятора

- **{SC+/-}**
 - Поддержка процедуры Assert.
- **{SI+/-}**
 - Проверка ввода/вывода.
- **{SR+/-}**
 - Проверка диапазона.
- **{SQ+/-}**
 - Проверка переполнения.
- **{INCLUDE имя_файла}**
 - Включить (вставить) указанный файл.

Pascal: директивы условной КОМПИЛЯЦИИ

```
program ... ;  
...  
{ $define DEBUG }  
...  
begin  
    { $ifdef DEBUG }  
    WriteLn ( 'Отладочная версия' ) ;  
    { $else }  
    WriteLn ( 'Промышленная версия' ) ;  
    { $endif }
```

Функциональное тестирование: автоматизация

Функциональное тестирование - это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям.
(wikipedia)

Идеи для автоматизации функционального тестирования

- перенаправление ввода/вывода;
- командные файлы.

Документирование

Концепция грамотного программирования настаивает на включение в текст программы настолько подробных и продуманных комментариев, чтобы она стала исходным текстом не только для исполняемого кода, но и для сопроводительной документации.

Генератор документации - программа или пакет программ, позволяющая получать документацию, предназначенную для программистов и/или для конечных пользователей системы, по особым образом комментированному исходному коду.

Документирование:дохуген

/// Этот комментарий обрабатается Дохуген

/// Эта строка будет «прилеплена» к предыдущей (и отделена пробелом)

// эта строка будет проигнорирована Дохуген

Для оформления текста внутри комментария используются специальные *параметры*.

Параметром называется определенное ключевое слово, которое служит для уведомления Дохуген выполнить особую обработку следующего (или следующих) слов комментария.

Чтобы отделить ключевое слово от текста комментария, каждое ключевое слово начинается с ESC-символа.

Документирование: параметры

- **\brief**
 - Начало краткого описания.
- **\details**
 - Начало подробного описания.
- **\param ([dir]) parameter_name description**
 - Описания параметра подпрограммы с именем `parameter_name`. Необязательный параметр `dir`, указывает «направление» параметра. Возможные значения `[in]`, `[out]`, `[in,out]`.
- **\return**
 - Описание возвращаемого значения.

Документирование: параметры

- **\field**
 - Описание полей записи.
- **\author**
- **\note**
- **\remark**
- **\bug**
- **\warning**
- **\par**

Документирование: delphidoc.py

1. Установить Python 2.7, пакет Cheetah-2.4.4.
1. Изменить кодировку исходных файлов на UTF8.
1. Написать комментарии.
1. Установить галочку "Generate XML Documentation" в настройках Вашего проекта.
1. Скомпилировать приложение.
1. Запустить delphidoc.py.

Защитное программирование

Идею защитного программирования можно сформулировать следующим образом: «прежде чем делать что-то - проверь, с корректными ли данными ты начинаешь это делать».

- Проверка данных из внешних источников.
- Проверка данных из внутренних источников.
- Выработка правил обработки некорректных входных данных:
 - Возвращение нейтрального значение.
 - Выбор ближайшего допустимого значения.
 - Возвращение кода ошибки.
 - Запись логов в файл.