Программирование, лекция 15

Кафедра ИУ7 МГТУ им. Н. Э. Баумана, 2021 год

Декомпозиция программных задач

Декомпозиция - разделение задачи на множество частных задач, не превосходящих по совокупной сложности исходную.

Один из способов декомпозиции - использование подпрограмм.

Подпрограммы необходимо использовать для:

- уменьшения дублирования кода
- возможности повторного использования кода в других программах
- упрощения отладки

Регулярные выражения

Регулярные выражения - формальный язык поиска и модификации подстрок в тексте, основанные на использовании метасимволов.

Основные специальные символы:

- . - любой символ (кроме перевода строки)
- ^ - начало строки
- $ - конец строки
- * - любое количество вхождений, включая 0
- + - любое количество вхождений > 0
- ? - 0 или 1 вхождений

Специальные символы

- {m} - m вхождений
- {m, n} - от m до n вхождений
- \ - экранирование символа (escape-последовательность)
- [] - набор символов
  - [aeiouy]
  - “-” - диапазон символов [A-F]
  - ^ - инверсия набора [^.!?] (только при включении первым символом)
- | - выбор из двух выражений
- () - группировка подвыражений

Специальные символы

- \d - цифра
- \D - не цифра
- \s - пробельный символ (\t, \n, \r, …)
- \S - непробельный символ
- \w - буквенно-цифровой символ
- \W - не буквенно-цифровой символ

“Ленивый” и “жадный” поиск

“Жадный” (greedy) режим поиска найдёт первый самый длинный фрагмент текста, удовлетворяющий выражению

“Ленивый” (non-greedy) режим поиска найдёт первый самый короткий фрагмент

Например, для текста “abc defg” и выражения “\w+”:

- жадный - abc
- ленивый - a

Для включения “ленивого” режима требуется добавлять “?”: *?, +?, ??, {m,n}?

Разновидности регулярных выражений

- базовые регулярные выражения POSIX
  - требуется экранировать {, }, (,)
  - отсутствуют +, ?, |
- расширенные регулярные выражения POSIX
- регулярные выражения, совместимые с Perl (PCRE, Perl Compatible Regular Expressions)

Модуль re

Методы:

- compile(pattern, flags=0)
- search(pattern, string, flags=0)
- match(pattern, string, flags=0)
- fullmatch(pattern, string, flags=0)
- split(pattern, string, maxsplit=0, flags=0)
- findall(pattern, string, flags=0)
- finditer(pattern, string, flags=0)
- sub(pattern, repl, string, count=0, flags=0)
- subn(pattern, repl, string, count=0, flags=0)
- escape(pattern)
- purge()

Методы и свойства класса Match

expand(template)

group(number)

groups()

groupdict()

start(group)

end(group)

span(group)

pos

endpos

lastindex

pastgroup

re

string