ФП, ИС, ‘15/’16

Първо упражнение, 25 февруари 2016

Калоян Йовчев

- email: k.yovchev@fmi.uni-sofia.bg

Начин на оценяване

- **О = 1/3 ТК + 1/3 ИЗ + 1/3 ИТ**, ако е изпълнено условието (*) О ≥ 3 & ИЗ ≥ 3 & ИТ ≥ 3

или

- ТК: оценка от текущ контрол
- ИЗ: оценка на изпита върху задачи
- ИТ: оценка на изпита върху теория (тест с отворени въпроси)
-
- ТК = 1/4 Д + 1/4 КТ + 1/2 КЗ
- Д: оценка на домашни работи
- КТ: оценка на контролни работи върху теория
- КЗ: оценка на контролни работи върху задачи

- При КЗ ≥ 4,50: възможност за освобождаване от изпита върху задачи (и тогава ИЗ = КЗ)
- При КТ ≥ 4,50: възможност за освобождаване от изпита върху теория (и тогава ИТ = КТ)

Упражнения

- Всеки четвъртък от 25 февруари 2016 до 9 юни 2016 включително
- Няма да има на 3 март и на 5 май 2016
- Общо: 14 занятия
- Език: Haskell
- Компилатор: GHC
- Редактор: SciTE

Примерна задача 1

Да се напише на Haskell функция sumUnique, която по списък от списъци от цели числа намира сумата на тези от числата, които са уникални в рамките на списъка, в който се срещат.

Примери:

- sumUnique [[1,2,3,2],[-4,-4],[5]] → 9 (= 1+3+5)
- sumUnique [[2,2,2],[3,3,3],[4,4,4]] → 0
- sumUnique [[1,2,3],[4,5,6],[7,8,9]] → 45

Примерна задача 1 - Решение

Да се напише на Haskell функция sumUnique, която по списък от списъци от цели числа намира сумата на тези от числата, които са уникални в рамките на списъка, в който се срещат.

Решение:

- sumUnique =

  sum . map (\l -> sum (filter (\x -> notElem x (delete x l)) l))

Примерна задача 2

- Да се дефинира функция maximize, за която оценката на обръщението maximize l, където l е непразен списък от едноместни числови функции, да е едноместна числова функция на аргумент *x*, която дава стойността *f(x)* на тази фунция *f* от списъка l, за която числото *f(x)* е най-голямо по абсолютна стойност.

Пример:

- Ако fn = maximize [(\x -> x*x*x),(\x -> x+1)],
  то fn 0.5 → 1.5, а fn (-2) → -8

Примерна задача 2 – Решение

- Да се дефинира функция maximize, за която оценката на обръщението maximize l, където l е непразен списък от едноместни числови функции, да е едноместна числова функция на аргумент *x*, която дава стойността *f(x)* на тази функция *f* от списъка l, за която числото *f(x)* е най-голямо по абсолютна стойност.

Решение:

- maximize fs = maximum . (\x -> [(f x) | f <- fs])

Ниво на абстракция

?

declarative languages (Haskell, ML, Prolog...)

OO languages (C++, Java, Python...)

procedural languages (C, Fortran, COBOL...)

assembly languages

binary code

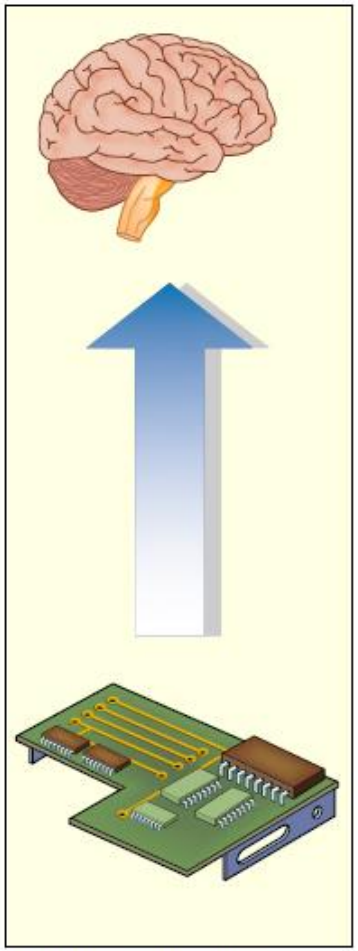

Figure by MIT OpenCourseWare.

Процедурна парадигма

- позната от часовете по УП
- програма = множество от процедури, една начална (входна) и множество извиквания помежду им, евентуално разделени в отделни файлове
- може да има процедури със странични ефекти
- императивен стил на програмиране
- примери за езици, в които е застъпена - C

Обектно-ориентирана парадигма

- позната от часовете по ООП
- основната програмна единица е класът. Класовете моделират някакви обекти от реалния свят или домейна на проблема, който решаваме.
- всеки обект има поведение (множество от методи, които притежава) и състояние (множество от неговите полета)
- може да имаме методи със странични ефекти
- императивен стил на програмиране
- примери за езици, в които е застъпена - Java, C++, Python, Ruby, други

Функционална парадигма

- програмите са минималистични
- декларативен стил на програмиране
- програма = съвкупност от функции, в математическия смисъл на понятието
- всяка функция връща резултат и няма странични ефекти (чисто функционални езици)
- примери за езици, в които е застъпена - диалекти на Lisp - Scheme, Common Lisp, Haskell. Елементи в Javascript, Ruby, Python, други езици (ламбда изрази в Java 8, C#)

Функционална парадигма - специфики

- липса на странични ефекти
- от една страна неоптималност на изчисленията - многократни пресмятания на едни и същи функции, поради невъзможност за запазване на резултата
- голяма полза при разработване на многонишкови приложения - решава основния проблем при разработка на такива приложения - необходимост от синхронизация при достъп до споделено състояние. Имаме липса на състояние (странични ефекти) липса на необходимост от синхронизация
- обикновено, в “чисто” функционалните езици няма оператори за цикъл - симулира се чрез рекурсия
- липсата на странични ефекти и декларативния стил ни карат да разбиваме решаваната задача на подзадачи, за които да пишем отделни функции - добър стил

Задачата „Факториел“

- n! = 1.2.3....n, където n е естествено число

- Решение в процедурен стил:

```
int fact(int n) {
        int res = 1;
        for (int i = 2; i <= n; ++i)
                res *= i;
        return res;
}
```

- Решение във функционален стил:

```
fact n = if (n == 0) then 1 else (n * (fact (n – 1)))
```

Haskell

- Препоръчвам за целта на упражненията да използвате за редактор SciTE
- Конфигуриран за Haskell може да свалите от: http://haskell.fenix1112.net

- Главна функция:

```
main = do
   print "SciTE“
   print (3 * 3)
```

Заб.: Идентацията, както и скобите, са важни!!!

Основен синтаксис на Haskell - 1

- Основни типове:
  - Цели числа: 1, 2, 3, ...
  - Реални числа: 1.2, 1.22, ...
  - Символи (с единични кавички): ‘a’, ‘b’, ...
  - Низове (с двойни кавички): “aaaa”, “bbbb”, ...
  - Булеви стойности: True, False

Основен синтаксис на Haskell - 2

- Основни оператори:
  - аритметични операции: +, -, *, ^ (повдигане в степен естествено число), ** (повдигане в реална степен), div (частно при целочислено деление), mod (остатък при целочислено деление), abs (абсолютна стойност)
  - оператори за сравнение: >, >=, <, <=, ==, /= (различно)
  - булеви оператори: && (конюнкция), || (дизюнкция), not (отрицание)

Основен синтаксис на Haskell - 3

- Дефиниране на константи:

  <име> = <стойност>

Примери:

  - n = 1
  - name = 2

- Дефиниране на функция:

  <име> <арг-1> <арг-2> ... <арг-n> = <тяло>

Примери:

  - f1 a b c = a * b * c
  - f2 a b = a * (b + b)

Заб.: = не е оператор за присвояване!!!

Основен синтаксис на Haskell - 4

- Условни изрази:

  if <условие> then <израз-за-да> else <израз-за-не>

Примери:

- if (1 == 2) then “yes” else “no”
- if (n == 0) then 1 else (n * (fact (n – 1)))

Факториел на Haskell

```
main = do
   print (fact 5)

fact n = if (n == 0) then 1 else (n * (fact (n – 1)))
```

Задачи

- **Задача 1:** Да се напише функция *mymin*, която приема два аргумента и връща по-малкият от тях.
- **Задача 2:** Да се напише функция *mymax*, която приема два аргумента и връща по-големият от тях.
- **Задача 3:** Да се напише функция *myfunc*, която пресмята на средно аритметичното от квадратите на 2 числа.

Рекурсия - видове

- линейно рекурсивна функция: такава, в която има единствено рекурсивно обръщение. Пример: рекурсивно пресмятане на факториел
- дървовидно рекурсивна функция: такава, в която рекурсивните обръщения са повече от едно. Пример: пресмятане на n-тото число на Фибоначи, чрез рекурсия
- опашково рекурсивна функция: такава, в която рекурсивното обръщение е последното нещо в тялото. В повечето случаи се връща стойността на това рекурсивно извикване. Пример: пресмятате на НОД

Рекурсия - задачи

- **Задача 4:** Да се напише myfib, която получава един аргумент n и връща n-тото число на Фибоначи. (Заб.: редицата е 1, 1, 2, 3, 5, ... и е индексирана от 0.)
- **Задача 5:** Да се напише функция *mygcd a b*, която връща НОД(a, b).
- **Задача 6:** Да се напише функция *mymaxdivisor x*, която намира най-големия делител d на цялото число x > 1, за който d < x.

Домашно 1

- Ще бъде качено в Moodle до 23:59 на 26 февруари 2016.
- Срок за предаване: **22:00 на 7 март 2016**.
- Предаване в Moodle, като е важно да се спазят изискванията в условието, защото задачите ще бъдат проверявани автоматично.

Въпроси?

- Моля да използвате темата в новинарския форум в Moodle:
  - Въпроси и дискусии – 1ва и 2ра група
  - http://moodle.openfmi.net/mod/forum/discuss.php?d=10701

До следващия път!