ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტის მეორე საფაკულტეტო კონფერენცია ზუსტ და საბუნებისმეტყველო მეცნიერებებში, 29 იანვარი-3 ბერვალი, 2014

**ნათელა არჩვაძე**, ასოცირებული პროფესორი, თსუ, კომპიუტერულ მეცნიერებათა დეპარტამენტი, პრაქტიკული ინფორმატიკა

# ფუნქციების განსაზღვრა დამგროვებელი პარამეტრებით ფუნქციონალურ ენებში

# ფუნქციების ზოგადი თვისებები ფუნქციონალურ ენებში

- სისუფთავე – არ არის გვერდითი მოვლენები და დეტერმინირებულობა
- ზარმაცი გამოთვლები – ნაწილობრივი გამოთვლების შესაძლებლობა
- კარირებული ფუნქციები

# სისუფთავე – არ არის გვერდითი მოვლენები და დეტერმინირებულობა

- *ფუნქციას შეუძლია მართოს მხოლოდ მისთვის გამოყოფილი მეხსიერება ისე, რომ არ შეუძლია შეცვალოს მეხსიერება მისთვის გამოყოფილი მეხსიერების გარეთ.*
- *დეტერმინირებული ეწოდება ფუნქციას, რომლის გამოსასვლელი მნიშვნელობა დამოკიდებულია მხოლოდ საწყის პარამეტრებზე.*

გამონაკლისები:

- შეტანა–გამოტანის სისტემა გვერდითი მოვლენების გარეშე შეუძლებელია
- შემთხვევითი რიცხვების გენერაცია ხორციელდება არადეტერმინირებული ფუნქციით
- გამოსავალი: გატანა ენის ბირთვის გარეთ მონადების სახით

# ზარმაცი გამოთვლები – ნაწილობრივი გამოთვლების შესაძლებლობა

- *გამოთვლების ზარმაცი სტრატეგია მდგომარეობს მასში, რომ ფუნქცია არ ითვლის მანამ, სანამ მისი შედეგი არ არის აუცილებელი პროგრამის მუშაობისთვის. ეს იძლევა საშუალებას შეიქმნას პოტენციურად უსასრულო მონაცემთა სტრუქტურები (სიები, ხეები და ა.შ), რომლებიც შეზღუდულია მხოლოდ კომპიუტერის ფიზიკური მეხსიერებით.*

*მაგალითად:* ა) firstNumbers n = take n [1..]

*ბ) ლოგიკური ფუნქციების გამოთვლა*

# ფუნქციის ტიპი არის კარირებული:

*A1 –> (A2 –> . . . (An –> B) . . .),*

*A1, A2, ...An – შემავალი პარამეტრების ტიპი, ხოლო B — შედეგის ტიპი*

- ფუნქციის კარირებულობა ნიშნავს, რომ იგი თავის პარამეტრებს იღებს თანმიმდევრულად, რის შედეგადაც მიიღება ახალი ფუნქცია.
- ასე, რომ თუ ფუნქციას გადავაწოდებთ *A1* ტიპის პირველ პარამეტრს, შედეგად მივიღებთ ფუნქციას, რომლის ტიპია:

*A2 –> (A3 –> . . . (An –> B) . . .).*

შედეგი:

- *ნაწილობრივი გამოთვლების გამოყენება*
- *ფუნქცია თვითონ არის გამოთვლების ობიექტი– იყოს არგუმენტი სხვა ფუნქციის. შემოდის ფუნქციონალების ცნება.*

# ფუნქციები დამგროვებელი პარამეტრით (აკუმულატორებით)

- ენები, რომლებიც ფუნქციონალურ პარადიგმას მიეკუთვნებიან, ბუნებრივია ფლობენ მრავალფეროვან საშუალებებს ფუნქციების აღსაწერად.

- გამონაკლისი არც ენა MicroSoft Haskell-ია და მასში განსაზღვრულია ფუნქციების წარმოდგენის რამდენიმე საშუალება

# რეკურსიული ფუნქციები

- ფუნქციის შესრულებისას შეიძლება დადგეს მეხსიერების ხარჯვის სერიოზული პრობლემა.

**Factorial 0 = 1**

**Factorial N = N * Factorial (N - 1)**

- Factorial (3)

  3 * Factorial (2)

  3 * 2 * Factorial (1)

  3 * 2 * 1 * Factorial (0)

  3 * 2 * 1 * 1

  3 * 2 * 1

  3 * 2

  6

# დამატების ოპერაციის გამორიცხვა

სიის შექცევა –Prelude ბიბლიოთეკიდან

reverse :: [a] → [a]

reverse [] = []

Reverse (x : xs) = reverse xs ++ [x]

- ++ *ოპერაც. ხანგრძლივობა წრფივად არის დამოკიდებული პირველი არგუმენტის სიგრძეზე. n სიგრძის სიისთვის* reverse xs *მოთხოვნილი ბიჯების რაოდენობაა 1-დან (n+1)-მდე რიცხვების ჯამი:* $(n+1)(n+2)/2 = (n^2 + 3n + 2)/2$
- *ათი ათასი ელემენტის შემცველი სიის შექცევა მოითხოვს დაახლოებით, რედუქციის 50 მლნ. ბიჯს.*

# დამატებითი პარამეტრი

Factorial_A  N = F N 1

F  0  A = A  --მეორე არგუმენტი–აკუმულატორი

F  N  A = F (N – 1) (N * A)

- აკუმულატორი შეიცავს შედეგს, რომელიც ბრუნდება რეკურსიის დამთავრებისას.
- თვითონ რეკურსიას ამ დროს აქვს „კუდური“ რეკურსიის სახე
- მეხსიერება გამოიყენება მხოლოდ ფუნქციის მნიშვნელობების დასაბრუნებელი მისამართების შენახვისათვის.

*კუდური რეკურსია წარმოადგენს რეკურსიის სპეციალურ სახეს, რომლის დროსაც მხოლოდ ერთხელ ხდება რეკურსიული ფუნქციის გამოძახება და ეს გამოძახებაც სრულდება ყველა გამოთვლის შემდეგ.*

კუდური რეკურსიის რეალიზაცია შეიძლება შესრულდეს იტერაციის პროცესის საშუალებით. პრაქტიკაში ეს ნიშნავს, რომ ფუნქციონალური ენის „კარგ“ ტრანსლიატორს უნდა „შეეძლოს“ აღმოაჩინოს კუდური რეკურსია და მოახდინოს მისი რეალიზება ციკლის სახით. თავის მხრივ,  პარამეტრის დაგროვების მეთოდს ყოველთვის არ მოვყავართ კუდურ რეკურსიამდე, თუმცა იგი ცალსახად გვეხმარება საერთო მეხსიერების მოცულობის შემცირებაში.

```
reverse'              ::      [a] → [a] → [a]
reverse' [] ys        =       ys
reverse' (x : xs) ys  =       reverse' xs (x : ys)
```

```
reverse               ::      [a] → [a]
reverse xs            =       reverse' xs []
```

*მაგალითი:*

```
reverse [1, 2, 3]
=                   { reverse-ის გამოყენება}
reverse' [1, 2, 3] []
=                   { reverse'- ის გამოყენება}
reverse' [2, 3] (1 : [])
=                   { reverse'- ის გამოყენება}
reverse' [3] (2 : (1 : []))
=                   { reverse'- ის გამოყენება}
reverse' [] (3 : (2 : (1 : [])))
=                   { reverse'- ის გამოყენება}
3 : (2 : (1 : []))
```

- 10 000 ელემენტიანი სიის შექცევა ახლა დაიკავებს დაახლოებით 10000 ბიჯს

# სიის ელემენტების საშუალო არითმეტიკულის გამოთვლა

```
sasharit  l = sasharit' l 0 0
sasharit' []  s n =s/n
sasharit' (x:xs)  s n = sasharit' xs (x+s) (n+1)
```

- აკუმულატორი–ორი ცვლადი
- პირველი ცვლადი აგროვებს სიის ელემენტების ჯამს
- მეორე ცვლადი აგროვებს წევრების რაოდენობას

# დამგროვებელი პარამეტრებით განსაზღვრებების აგების პრინციპები

1. შემოდის ახალი ფუნქცია დამატებითი არგუმენტით (აკუმულატორით), რომელშიც გროვდება გამოთვლების შედეგები.

2. აკუმულატორი არგუმენტის საწყისი მნიშვნელობა მოიცემა ტოლობით, რომელიც აკავშირებს ძველ და ახალ ფუნქციებს.

3. საწყისი ფუნქციის ის ტოლობა, რომელიც შეესაბამება რეკურსიიდან გამოსავალს, იცვლება აკუმულატორით დაბრუნების გამოსახულებით.

4. ტოლობა, რომელიც შეესაბამება რეკურსიულ განსაზღვრებას, გამოიხატება როგორც ახალ ფუნქციაზე მიმართვა, რომელშიც აკუმულატორი იღებს იმ მნიშვნელობას, რომელიც ბრუნდება საწყისი ფუნქციით.

# ნებისმიერი ფუნქცია შეიძლება გარდავქმნათ აკუმულატორის გამოთვლის ფორმით?

- ამ შეკითხვის პასუხი არის უარყოფითი
- დამგროვებელი პარამეტრიანი ფუნქციის აგების ხერხი არ არის უნივერსალური და იგი არ არის კუდური რეკურსიის აგების გარანტია
- განსაზღვრების აგება დამგროვებელი პარამეტრით არის შემოქმედებითი საქმე. ამ პროცესში აუცილებელია ზოგიერთი ევრისტიკის გამოყენება.

# განზოგადოებული ფორმები კუდური რეკურსიით განსაზღვრული ფუნქციები სთვის HASKELL–ზე

- f [ ] = g1 [ ]
- f ( x : xs ) = g2 ( g3 x ) ( g4 ( f ( g5 xs ) ) )

- აქ g1, g2, g3, g4, g5 ფუნქციები დამოკიდებულია ამოცანის პირობებზე:
- g1–არის ფუნქცია ცარიელი სიის დასამუშავებლად
- g2–არის ფუნქცია, რომელიც აერთიანებს სიის თავისა და კუდის დამუშავების შედეგებს.
- g3–არის ფუნქცია, რომელიც ამუშავებს სიის თავს.
- g4–არის ფუნქცია, რომელიც ამუშავებს არაცარიელი სიის კუდისთვის რეკურსიულ გამოძახებას.
- g5–არის ფუნქცია, რომელიც წინასწარ ამუშავებს არაცარიელი სიის კუდს რეკურსიული გამოძახებისთვის.

# განზოგადოებული ფორმები დამგროვებელპარამეტრიანი ფუნქციებისთვის HASKELL–ზე

- Fun n =Fun’ n a --გამოძახება, a-ს აქვს კონკრეტ. მნიშვნ
- Fun’ n a= g1 a
- Fun’ ( x : xs ) = g2 ( g3 x ) ( g4 (Fun’ ( g5 xs ) g6 a) )
- g1 - არის ფუნქცია, რომელიც ღებულობს იმ მნიშვნელობას, რომელიც ბრუნდება საწყისი ფუნქციით;
- g2–არის ფუნქცია, რომელიც აერთიანებს სიის თავისა და კუდის დამუშავების შედეგებს.
- g3–არის ფუნქცია, რომელიც ამუშავებს სიის თავს.
- g4–არის ფუნქცია, რომელიც ამუშავებს არაცარიელი სიის კუდისთვის რეკურსიულ გამოძახებას.
- g5–არის ფუნქცია, რომელიც წინასწარ ამუშავებს არაცარიელი სიის კუდს რეკურსიული გამოძახებისთვის.
- g6-არის ფუნქცია, რომელიც აკუმულატორს ამუშავებს

# შემდგომი კვლევა

- გაგრძელდება მრავალჯერადი რეკურსიის კვლევით.

დიდი მადლობა ყურადღებისთვის!