**სასწავლო კურსის სილაბუსი**

|  |  |
| --- | --- |
| **სასწავლო კურსის დასახელება** | **ფუნქციონალური დაპროგრამება**  **Functional Programming** |
| **ავტორი (ავტორები)** | **ნათელა არჩვაძე** - ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტის ასოცირებული პროფესორი |
| **ლექტორი (ლექტორები)** | **ნათელა არჩვაძე** - ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი, კომპიუტერულ მეცნიერებათა დეპარტამენტის ასოცირებული პროფესორი  *საკონტაქტო ინფორმაცია:* ტელ. 599 474 632  *ელ. ფოსტა:* [natela.archvadze@tsu.ge](mailto:natela.archvadze@tsu.ge)  თსუ XI კორპუსი, ოთახი №354  კონსულტაციის საათები: კვირაში 2 სთ (განრიგი იგეგმება სემესტრულად) |
| **სასწავლო კურსის კოდი** | CS 505 |
| **სასწავლო კურსის სტატუსი** | ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი,  საბაკალავრო პროგრამა „კომპიუტერული მეცნიერება“, სავალდებულო არჩევითი |
| **სასწავლო კურსის მიზნები** | შეასწავლოს სტუდენტებს ფუნქციონალური პროგრამირების პარადიგმა, მისცეს მათ ფუნქციონალური პროგრამირების ძირითადი პრინციპების და საშუალებების თანმიმდევრული და საფუძვლიანი ცოდნა.  ფუნქციონალური პროგრამირების მეთოდების ილუსტრირება ხდება ენა Haskell-ზე.  სტუდენტი შეიძენს ფუნქციონალური პროგრამების შექმნისათვის აუცილე­ბელ პრაქტიკულ უნარ-ჩვევებს და მას შეეძლება:   * ფუნქციების წერა ფუნქციონალურ სტილზე; * ფორმალური მსჯელობა ფუნქციონალურ პროგრამებზე; * პოლიმორფიზმისა და მაღალი რიგირ ფუნქციების გამოყენება; * განსხვავება ფუნქციონალურ დაპროგრამებას და ა) იმპერატიულ და ობიექტებზე ორიენტირებულ დაპროგრამებას შორის და ბ) ლოგიკურ დაპროგრამებას შორის. |
| **კრედიტების რაოდენობა და საათების განაწილება სტუდენტის დატვირთვის შესაბამისად (ECTS)** | **5 კრედიტი (125 საათი)**  საკონტაქტო - 45 სთ:   * ლექცია**-** 15 სთ. * ლაბორატორია – 15სთ. * პრაქტიკული მეცადინეობა - 15 სთ.   შუალედური გამოცდა - 2 სთ.  დასკვნითი გამოცდა - 3 სთ.  დამოუკიდებელი მუშაობა - 75 სთ. |
| **დაშვების წინაპირობები** | CS104 ობიექტზე ორიენტირებული დაპროგრამება 1 (C++) |
| **სწავლის შედეგები** | **ცოდნა და გაცნობიერება**  კურსის დასრულების შემდეგ სტუდენტს ექნება საფუძვლიანი ცოდნა შემდეგ სფეროებში:   * სტანდარტული ფუნქციონალური სტილის (პარადიგმის) ალგორითმების იმპლემენტირებაში; * რეკურსიული ალგორითმების შედგენა-რეალიზებაში.   **ცოდნის პრაქტიკაში გამოყენების უნარი**  სტუდენტს ექნება ცოდნის პრაქტიკაში გამოყენების უნარ\_ჩვევები. კერძოდ:   * სხვადასხვა ამოცანის გადასაწყვეტად შეეძლება წარმოადგინოს ალგორითმები და მონაცემთა სტრუქტურები ფუნქციონალური მიდგომის ტერმინებში, ამავე დროს შეძლებს ადექვატურად გამოიყენოს ეს მიდგომა მომხმარებლის კონკრეტული ამოცანის ამოხსნისას; * დამუშავებული ალგორითმის და/ან მონაცემთ სტრუქტურის ფუნქციონალურ ენაზე რეალიზების უნარი - პროგრამის შედგენის, ინტერპრეტატორში დამოუკიდებლად მუშაობის უნარი, შესწორებისა და ტესტირების უნარი.   **დასკვნის უნარი**  კურსის გავლის შემდეგ სტუდენტი შეძლებს:   * დასმული ამოცანის გაანალიზებას და მისთვის საჭირო მონაცემების დადგენას, შესწავლის მეთოდების შერჩევას, განყენებული მონაცემებისა და სიტუაციების ანალიზს სტანდარტული და ზოგიერთი გამორჩეული მეთოდის გამოყენებით, დასაბუთებული დასკვნის ჩამოყალიბებას. * პრობლემის გადასაწყვეტად შეიძინოს ფუნქციონალური პროგრამირების შესაბამისი მეთოდების შერჩევა-დამუშავების უნარი, მაღალი დონის ფუნქციების გამოყენების უნარი; * შეიძინოს კარირებული, რეკურსიული, ლამბდა ფუნქციების აგებისა და გამოყენების უნარი ტიპიური ამოცანებისთვის; * შეიძინოს სიმბოლური ინფორმაციის დამუშავების უნარი, ასევე ინფორმაციის გარდაქმნის ეფექტური მეთოდების გამოყენების უნარი, რომელთა რეალიზება მოითხოვს მრავალდონიან აბსტრაქციასა და განზოგადოებას, რაც უფრო ბუნებრივად გამოიხატება ფუნქციონალური პროგრამირების ენის ტერმინებში.   **კომუნიკაციის უნარი**  კურსის გავლის შემდეგ სტუდენტი შეძლებს შეიძინოს:   * ფუნქციონალური პროექტირების მეთოდოლოგიისა და ტექნოლოგიების გამოყენების უნარი, ასევე გაეცნოს სხვის შემთავაზებულ პროექტს, ასახოს პრობლემა, შესთავაზოს გადაწყვეტა; * ფუნქციონალური პროგრამირების ენაზე ალგორითმის ჩაწერის და გამართვის უნარი. შედეგების წერილობითი სახით გაფორმებისა და ზეპირად გადაცემის უნარი; * გუნდში მუშაობის უნარი; * დროის მენეჯმენტის უნარი.   **სწავლის უნარი**  კურსის გავლის შემდეგ სტუდენტი შეძლებს:   * შეაფასოს სწავლის პროცესისას მიღებული თეორიული ცოდნა და მისი პრაქტიკაში გამოყენების შესაძლებლობა, რათა დაადგინოს შემდგომი სწავლის საჭიროება; * შეაფასოს საკუთარი ცოდნის გაღრმავების აუცილებლობა, რათა განსაზღვროს შემდგომი სწავლის გაგრძელების მიმართულება; * დამოუკიდებლად გაიღრმაოს ცოდნა ფუნქციონალურ პროგრამირებაში სპეციალური ლიტერატურის დახმარებით;   **ღირებულებები:**  კურსის გავლის შემდეგ სტუდენტი შეძლებს:   * ინფორმაციული ტექნოლოგიების სფეროში პროფესიული ეთიკის ნორმების მნიშვნელობის გააზრებას და დაცვას; * თეორიის და პრაქტიკის მნიშვნელობებისა და მათს შორის ჰარმონიული ბალანსის აუცილებლობის გააზრებას და დაცვას, გამოცდილებისა და ცოდნის მუდმივ განახლებას.   დარგობრივი პროექტის განხორციელებისას უზრუნველყოს სხვისი მონაცემების დაცვა. მიაღწიოს პროფესიულ კარერაში წარმატებს, რისი აუცილებელი წინაპირობა იქნება მისი ღირებულებების სისტემა. |
| **სასწავლო კურსის შინაარსი** | **თეორიული კურსი მოიცავს:** (იხ.დანართი 1);  **ლამორატორიული კურსი მოიცავს:** (იხ.დანართი 2); |
| **სწავლების/ სწავლის მეთოდები** | * ვერბალური, ანუ ზეპირსიტყვიერი მეთოდი; * წიგნზე მუშაობის მეთოდი; * ლაბორატორიული მეთოდი და დემონსტრირების მეთოდი; * ახსნა–განმარტებითი მეთოდი; * ქმედებაზე ორიენტირებული სწავლება; * პროექტის შემუშავება და პრეზენტაცია; * ცოცხალი კოდის მეთოდი. |
| **შეფასების ფორმები/**  **კომპონენტები/**  **მეთოდები/**  **კრიტერიუმები** | შეფასების კომპონენტები:   1. *შუალედური გამოცდა* - 25 ქულა 2. *საკონტროლო - პრაქტიკული შეფასება* - 20 ქულა 3. *ლაბორატორიული შეფასება*- 15 ქულა 4. *დასკვნითი გამოცდა* - 40 ქულა   სტუდენტის მიღწევების შეფასება ხორციელდება 100 ქულიანი სისტემით, რომელიც მოიცავს შუალედურ შეფასებას და დასკვნით შეფასებას-გამოცდას.  შეფასების სისტემა უშვებს შემდეგი სახის შეფასებებს:  დადებითი:  (A) ფრიადი – მაქსიმალური შეფასების 91% და მეტი;  (B) ძალიან კარგი – მაქსიმალური შეფასების 81-90%;  (C) კარგი – მაქსიმალური შეფასების 71-80%;  (D) დამაკმაყოფილებელი – მაქსიმალური შეფასების 61-70%;  (E) საკმარისი – მაქსიმალური შეფასების 51-60%;  უარყოფითი:  (FX) ვერ ჩააბარა –0 მაქსიმალური შეფასების 41-50%, რაც ნიშნავს, რომ სტუდენტს ჩასაბარებლად მეტი მუშაობა სჭირდება და ეძლევა დამოუკიდებელი მუშაობით დამატებით გამოცდაზე ერთხელ გასვლის უფლება;  (F) ჩაიჭრა – მაქსიმალური შეფასების 40% და ნაკლები, რაც ნიშნავს, რომ სტუდენტის მიერ ჩატარებული სამუშაო არ არის საკმარისი და მას საგანი ახლიდან აქვს შესასწავლი.  სტუდენტს დამატებით გამოცდაზე გასვლის უფლება აქვს იმავე სემესტრში, თუ მან მიიღო FX (41-50 – ვერ ჩააბარა) შეფასება;  შეფასების მეთოდები:   1. ლაბორატორიული შეფასება: ტარდება ლაბორატორიულ მეცადინეობაზე და გულისხმობს სტუდენტის მიერ კომპიუტერთან დავალების შესრულებას. დავალება ითვალისწინებს 5 ამოცანისთვის პროგრამების შედგენას ფუნქციონალური ენის გამოყენებით, მის გამართვას და პასუხის მიღებას. სტუდენტის მიერ მიღებული შედეგი პროფესორს ჩაბარდება დაარქივებული რამდენიმე ფაილის სახით ან (სხვადასხვა გარემოების გამო) წერილობითი სახით (ნაბეჭდი ან ხელნაწერი). ქვიზის მაქსიმალური 15 ქულა წარმოადგენს 5 ამოცანის (თითოეული მაქსიმუმ 3 ქულა) შეფასებების ქულების ჯამს. ერთი ამოცანის ქულის დაგროვება ხდება შემდეგი მდგენელებში დაგროვილი ქულების შეკრებით:   0-2 ქულა - დასმული ამოცანის რეალიზაცია. ა) 2 ქულა - ამოცანა სრულყოფილად არის რეალიზებული; ბ) 1 ქულა - ამოცანა ამოხსნილია ნაწილობრივ, რეალიზაცია შეიცავს ხარვეზებს; დ) 0 ქულა- რეალიზაცია სრულიად მცდარია.  0-0.5 ქულა - დასმული ამოცანის გაფორმება დაპროგრამების კარგი სტილის დაცვით (ტექსტის ფორმატირება, კომენტარების დადება).  0-0.5 ქულა - გამოყენებული ალგორითმების ან მონაცემთა სტრუქტურების ეფექტურობა.   1. შუალედური გამოცდა: წერითი გამოკითხვა, 5 (საკითხი) X 5(ქულა) = 25 ქულა, აფასებს 1-7 კვირების მასალის ცოდნას. სტუდენტს მიეცემა 1 თეორიული საკითხი და 4 პრაქტიკული დავალება, თითოეული ფასდება 5 ქულით. 2. საკონტროლო-პრაქტიკული შეფასება - 4(საკითხი) X 5 (ქულა) = 20 ქულა. სტუდენტს მიეცემა 4 პრაქტიკული შინაარსის წერილობითი დავალება, თითოეული ფასდება 5 ქულით. 3. დასკვნითი გამოცდა - წერითი გამოკითხვა, 8(საკითხი) X 5 (ქულა) = 40 ქულა. დასკვნითი გამოცდა ტარდება წერითი ფორმით სემესტრის დასრულების შემდეგ. ფასდება ძირითადად საშუალედო შეფასების შემდეგ გავლილი საკითხების ცოდნისა და გაცნობიერების დონე, დამუშავებული ლიტერატურის სისრულე, ლოგიკური აზროვნების, აზრის მკაფიოდ გადმოცემისა და მსჯელობის უნარი. სტუდენტს დასკვნით გამოცდაზე მიეცემა 2 თეორიული საკითხი და 6 პრაქტიკული დავალება, თითოეული 5-ქულიანი.   *თეორიულ დავალებაში* *სტუდენტის შეფასებაა:*   * *5 ქულა, თუ: პასუხი სრულია; საკითხი ზუსტად და ამომწურავად არის გადმოცემული; ტერმინოლოგია დაცულია. სტუდენტი ზედმიწევნით კარგად ფლობს პროგრამით გათვალისწინებულ განვლილ მასალას, ღრმად და საფუძვლიანად აქვს ათვისებული როგორც ძირითადი, ისე დამხმარე ლიტერატურა.* * *4 ქულა, თუ: პასუხი სრულია, მაგარამ შეკვეცილი; ტერმინოლოგიურად გამართულია; საკითხი ამომწურავად არის გადმოცემული; არსებითი შეცდომა არ არის; სტუდენტი კარგად ფლობს პროგრამით გათვალისწინებულ განვლილ მასალას; ათვისებული აქვს ძირითადი ლიტერატურა.* * *3 ქულა, თუ: პასუხი არასრულია; საკითხი დამაკმაყოფილებლად არის გადმოცემული; ტერმინოლოგია ნაკლოვანია; სტუდენტი ფლობს პროგრამით გათვალისწინებულ მასალას, მაგარამ აღნიშნება მცირეოდენი შეცდომები.* * *2 ქულა, თუ: თუ: პასუხი არასრულია; ტერმინოლოგია მცდარია; საკითხის შესაბამისი მასალა გადმოცემულია ნაწილობრივ; სტუდენტს არასაკმარისად აქვს ათვისებული ძირითადი ლიტერატურა; აღინიშნება რამდენიმე არსებითი შეცდომა.* * *1 ქულა, თუ: პასუხი ნაკლოვანია; ტერმინოლოგია არ არის გამოყენებული, ან არ არის შესაბამისი; პასუხი არსებითად მცდარია. გადმოცემულია საკითხის შესაბამისი მასალის მხოლოდ ცალკეული ფრაგმენტები.* * *0 ქულა, თუ: პასუხი არ არის ან არ შეესაბამება დასმულ ამოცანას.*   *პრაქტიკულ და შუალედური გამოცდის დავალებაში* *სტუდენტის შეფასებაა:*   * *5 ქულა, თუ ამოცანა ამოხსნილია სწორად. ჩანს შესაბამისი მეთოდების და/ან დაპროგრამების გარემოს ზედმიწევნით კარგი ცოდნა.* * *4 ქულა, თუ ამოცანა ამოხსნილია სწორად. დაშვებულია უმნიშვნელო შეცდომები.* * *3 ქულა, თუ დავალების მნიშვნელოვანი ნაწილი შესრულებულია, დაშვებულია არსებითი ხასიათის 1-2 შეცდომა.* * *2 ქულა, თუ დავალების დაახლოებით ნახევარია შესრულებული, ზოგიერთ მეთოდს ვერ იყენებს სრულყოფილად.* * *1 ქულა, თუ ქულა - დავალების მცირე ნაწილია შესრულებული, ძირითად მასალას კარგად ვერ ფლობს.* * *0 ქულა, თუ დავალების ვერც ერთ ნაწილს ვერ ასრულებს შესაბამისი მასალის არცოდნის გამო.*   დასკვნით გამოცდაზე გასვლის უფლება ეძლევა სტუდენტს, რომელსაც შუალედური შეფასებების ჯამური ქულის გათვალისწინებით უგროვდება **20** ქულა მაინც.  დასკვნითი გამოცდა ჩაბარებულად ჩაითვლება მხოლოდ იმ შემთხვევაში, თუ სტუდენტი დასკვნით გამოცდაზე მიიღებს მინიმუმ **21** ქულას. |
| **ძირითადი ლიტერატურა** | 1. ნათელა არჩვაძე. ფუნქციონალური დაპროგრამება Haskell-ზე. ISBN 978-9941-13-709-9 (PDF). თბილისის სახელმწიფო უნივერსიტეტის გამომცემლობა. 366 გვ. 2018.   http://press.tsu.ge/data/image\_db\_innova/Haskell-Dak.pdf |
| **დამხმარე ლიტერატურა და სხვა სასწავლო მასალა** | 1. Graham Hutton. Programming in Haskell. 2007. ISBN:9780521871723 <http://ebooks.cambridge.org/ebook.jsf?bid=CBO9780511813672> 2. Душкин Р. В. Функциональное программирование на языке Haskell. — М.: ДМК-Пресс, 2007. — 608 стр. ISBN 5-94074-335-8 3. პრაქტიკული და ლაბორატორიული მეცადინეობების მასალა: http://e-learning.tsu.ge/course 4. Simon Marlow. Parallel and Concurent Programming in Haskell. 2012. 5. [Learn You a Haskell for Great Good!: A Beginner's Guide](http://learnyouahaskell.com/chapters) by Miran Lipovaca <http://learnyouahaskell.com/chapters> 6. Richard Bird. Thinking Functionally With Haskell. Cambridge University Press. 2015 |

**დანართი 1**

**სასწავლო კურსის შინაარსი**

|  |  |  |
| --- | --- | --- |
| **№** | **ლექციის თემა** | **ლიტერატურა**  **(შესაბამისი გვერდების მითითებით)** |
| 1 | **პირველი კვირა**  **პროგრამირების საშუალებების ისტორიული მიმოხილვა.**  აპარატურული და პროგრამული უზრუნველყოფის განვითარების ტენდენციები. პროგრამული კომპონენტების აბსტრაქცია და უნიფიკაცია. იმპერატიული და დეკლარატიული პროგრამირების ენები, ამ ტიპის ენების ნაკლოვანებები. ფუნქციონალური პროგრამირების თავისებურებანი. ტიპური ამოცანები, რომლებიც იხსნება ფუნქციონალური პროგრამირების მეთოდებით. | [1] - ნაწილი 2, თავი 2.1  [2] - გვ. 1-9  [3] - გვ. 27-86 |
| 2 | **მეორე კვირა**  **პროგრამირების ენა Haskell -ის ლექსიკა.**  ფუნქციონალური ტიპის ენის ძირითადი ლექსიკური კონსტრუქციები ენა Haskell -ის მაგალითზე. გამოსახულებები, მათი გამოცხადება და სახელებთან დაკავშირება. Hugs სისტემა. სკრიპტები Haskell -ზე. | [1] – თავი 2.2, 2.5  [2] - გვ. 10-16 |
| 3 | **მესამე კვირა**  **ტიპები და კლასები.**  ძირითადი ცნებები და ტიპები. სიის და კორტეჟის ტიპები. ფუნქციის ტიპები. კარირებული ფუნქციები. პოლიმორფული ტიპები. ძირითადი კლასები. | [1] – თავი 1.3, 2.3  [2] - გვ. 17-30  [3] - გვ. 164-192 |
| 4 | **მეოთხე კვირა**  **ფუნქციების განსაზღვრა.**  პირობითი გამოსახულებები. დამცავი განტოლებები. შაბლონები. | [1] – თავი 1.1,1.3, 2.4  [2] - გვ. 30-32  [3] - გვ. 118-130 |
| 5 | **მეხუთე კვირა**  **λ გამოთვლები**  ფუნქციონალური პროგრამირების თეორიული საფუძვლები. მონაცემთა კოდირება λ აღრიცხვაში. რედუქცია და გამოთვლები. | [1] – თავი 2.4  [2] - გვ. 34-37  [3] - გვ. 273-298 |
| 6 | **მეექვსე კვირა**  **რეკურსიული ფუნქციები.**  რეკურსია სიებზე. რეკურსია რიცხვებზე. რეკურსია მრავალარგუმენტიანი ფუნქციების გამოყენებით. | [1] – თავი 1.4, 2.8  [2] - გვ. 48-59  [3] - გვ. 142-152 |
| 7 | **მეშვიდე კვირა**  რეკომენდაციები რეკურსიუი ფუნქციების შედგენაზე. მრავალჯერადი რეკურსია. ურთიერთრეკურსია. | [1] – თავი 1.4 |
| 8 | **მერვე კვირა**  **სიის კონსტრუქტორები**  გენერატორები. მცველები. | [1] – თავი 1.5  [2] - გვ. 38-46 |
| 9 | **მეცხრე კვირა**  **სიის კონსტრუქტორები**  ფუნქციების აგება სიის კონსტრუქტორების გამოყენებით. სტრიქონის კონსტრუქტორები. | [1] – თავი 1.5 |
| 10 | **მეათე კვირა**  **სიებთან მუშაობის უტილიტები.**  უტილიტები Maybe. სიმბოლოებთან მუშაობის უტილიტები.ბიბლიოთეკა List | [1] – თავი 2.6 |
| 11 | **მეთერთმეტე კვირა**  **სტანდარტული Prelude ფაილი.**  კლასები, კორტეჟები, სიები, ფუნქციები | [1] –  დანართი A |
| 12 | **მეთორმეტე კვირა**  **მაღალი რიგის ფუნქციები.**  სიების დამუშავება. foldr, fordl ფუნქციები. კომპოზიციის ოპერატირი. | [1] – თავი 1.6  [2] - გვ. 61-72 |
| 13 | **მეცამეტე კვირა**  **ტიპებისა და კლასების გამოცხადება.**  რეკურსიული ტიპები | [1] – თავი 10  [2] - გვ. 99-114 |
| 14 | **მეთოთხმეტე კვირა**  **გადადებული გამოთვლები.**  გამოთვლების სტრატეგიები. უსასრულო სტრუქტურები. მოდულური დაპროგრამება. მკაცრი გამოყენება | [1] – თავი12  [2] - გვ. 124-137 |
| 15 | **მეთხუთმეტი კვირა**  **პარალელიზმი Haskell-ში.**  მონადა Eval | [4] – თავი 1, 2 |

|  |  |  |
| --- | --- | --- |
| **N** | **ლაბორატორიული სამუშაოების თემა** | **ლიტერატურა**  **(შესაბამისი გვერდების მითითებით)** |
| 1 | **პირველი კვირა**  **მიზანი:** *Haskell ენის ინტერპრეტატორთან მუშაობის უნარ-ჩვევების გამომუშავება. ძირითად ტიპებზე წარმოდგენის შექმნა.*  Haskell ენის ინტერპრეტატორთან მუშაობის საფუძვლები.  ტიპები. ინტერპრეტატორის ბრძანება :type. მარტივი ტიპები: Integer, Int, Float, Double, Bool, Char.  არითმეტიკული ოპერაციები, მათემატიკური ფუნქციები.  კორტეჟები. რთული ტიპები. წყვილები. წყვილებთან მუშაობის ფუნქციები: fst, snd. კორტეჟი. კორტეჟის ელემენტი-მეორე კორტეჟი.  სია. ცარიელი სია. ოპერაცია :, ასოციურობის თვისება. ფუნქციები: head, tail, length.  სტრიქონები. ტიპი String.რიცხვითი მნიშვნელობების გარდაქმნა სტრიქონად და პირიქით. ფუნქციები: read, show. | [5] თემა 1 |
| 2 | **მეორე კვირა**  **მიზანი:** *Haskell ენის ინტერპრეტატორთან მუშაობის უნარ-ჩვევების გამომუშავება. შესწავლა, თუ როგორ განისაზღვრება მარტივი ფუნქციები.*  ფუნქციები. Haskell-ის ჩადგმული ფუნქციები. ბრძანებები: :load, :edit, :reload. Hugs (სახელი.hs) ფაილის შექმნა.  ფუნქციის განმარტება. ფუნქციის ტიპი.  პირობითი გამოსახულება. ოპერატორი if then else. ლოგიკური ფუნქციები &&, || და not.  მრავალარგუმენტიანი ფუნქცია. ფუნქციის ტიპის განსაზღვრა. | [5] თემა 1 |
| 3 | **მესამე კვირა**  **მიზანი:** *რეკურსიული ფუნქციების განსაზღვრის შესწავლა. ნიმუშთან შედარების მექანიზმების შესახებ წარმოდგენის შექმნა. სიებზე მომუშავე ფუნქციების განსაზღვრის უნარ-ჩვევების გამომუშავება.*  კომენტარები. სტრიქონული (- -) და ბლოკური კომენტარები( { })  რეკურსია. ფაქტორიალის განსაზღვრა.  ამორჩევის ოპერატორი (case) და ჩანაწერის გასწორების წესი.  ფუნქციის ფრაგმენტული განსაზღვრების საშუალება.  ნიმუშთან შედარება.  სიების შედგენა. ზოგიერთი სასარგებლო ფუნქციის შედგენა. | [5] თემა 2 |
| 4 | **მეოთხე კვირა**  let დაკავშირება. დროებითი ცვლადების გამოყენება.  შეტყობინებები შეცდომების შესახებ.  დაცული პირობები. | [5] თემა 2 |
| 5 | **მეხუთე კვირა**  ბაზური ტიპები: კორტეჟები, სიები.  პოლიმორფული ტიპები.  მომხმარებლის ტიპები. გასაღები სიტყვა data. წყვილები. მრავლობითი კონსტრუქტორები. ტიპების კლასები. ექვივალენტობის კლასი. | [5] თემა 3 |
| 6 | **მეექვსე კვირა**  სიის კონსტრუქტორები: გენერატორები,მცველები | [5] თემა 3 |
| 7 | **მეშვიდე კვირა**  ფუნქციების აგება სიის კონსტრუქტორების გამოყენებით. სტრიქონის კონსტრუქტორები. | [5] თემა 3 |
| 8 | **მერვე კვირა**  ოპერატორების განმარტება. ინფიქსური სტილი.  რეკურსიული ტიპები. ხის განმარტება. | [5] თემა 4 |
| 9 | **მეცხრე კვირა**  სია, როგორც რეკურსიული ტიპი.  სინქატსური ხე. | [5] თემა 4 |
| 10 | **მეათე კვირა**  მიზანი: ფუნქციონალების შესწავლა  მაღალი დონის ფუნქციები. ფუნქცია map, filter. foldr და foldl.  მაღალი დონის სხვა ფუნქციები. | [5] თემა 5 |
| 11 | **მეთერთმეტე კვირა**  მიზანი: ფუნქციონალების შესწავლა  ფუნქცია zip. lambda აბსტრაქცია.  სექციები. | [5] თემა 5 |
| 12 | **მეთორმეტე კვირა**  მოდულები. module Tree. მოდულში ექსპორტირება. მოდულის იმპორტირება.  მონაცემთა აბსტრაქტული ტიპები. მაგალითი ლექსიკონის აგების შესახებ. სიტყვის დამატება. სიტყვის მოძებნა.  ტიპების სინომიმები. | [5] თემა 6 |
| 13 | **მეცამეტე კვირა**  შეტანა–გამოტანის ოპერაციები. შეტანა–გამოტანის ბაზური ოპერაციები: getChar, putChar, ready, return, getLine. შეტანა–გამოტანის სტანდარტული ოპერაციები: FilePath, openFile, hGetChar, hGetLine, hPutChar, hPutstr.  მაგალითი: ფაილების კოპირება.  შესასრულებელი ფაილების შექმნა. | [5] თემა 6 |
| 14 | **მეთოთხმეტე კვირა**  გამოთვლების სტრატეგიები: ზარმაცი და მკაცრი გამოთვლები | [5] თემა 6 |
| 15 | **მეთხუთმეტე კვირა**  მონადა Eval-ის განხილვა, ძირითადი ფუნქციები. | [5] თემა 6 |

დანართი ABET - ***სწავლის შედეგების შეფასების მეთოდები***

**ეფექტურობის მაჩვენებლები საგნისთვის „ფუნქციონალური დაპროგრამება“**

***ცხრილი A***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Performance Indicator**  ეფექტურობის მაჩვენებლები | **Beginning**  **დამწყები** | **Developing**  **ვითარდება** | **Competent**  **კომპეტენტური** | **Accomplished**  **პროფესიონალი** |
| **PI 1. კომპიუტინგის კომპლექსური პრინციპების ანალიზი, მათი და სხვა დისციპლინების გამოყენება გადაწყვეტილებების მისაღებად** | | | | |
| **PI 1.1 კომპიუტინგის კომპლექსური პრობლემის ანალიზი გადაწყვეტილების მისაღებად** | ძნელად შეუძლია ფუნქციონალური პარადიგმის შესაბამისი მარტივი ტიპის ამოცანისათვის პროგრამის შექმნა | შეუძლია ფუნქციონალური პარადიგმის შესაბამისი მარტივი ტიპის ამოცანისათვის პროგრამის შექმნა | შეუძლია ფუნქციონალური პარადიგმის შესაბამისი საშუალო სირთულის ამოცანისათვის პროგრამის შექმნა | შეუძლია ფუნქციონალური პარადიგმის შესაბამისი რთული ტიპის ამოცანისათვის პროგრამის შექმნა |
| **PI 1.2**  **კომპუტინგის პრინციპების გამოყენება გადაწყვეტილების მისაღებად კომპიუტინგის კომპლექსური პრობლემის გადასაწყვეტად** |  |  |  |  |
| **PI 1.3**  **რელევანტური დისციპლინების პრინციპების გამოყენება გადაწყვეტილების მისაღებად კომპიუტინგის კომპლექსური პრობლემის გადასაწყვეტად** |  |  |  |  |
| **PI 2 კომპიუტინგზე დაფუძნებული გადაწყვეტილების შემუშავება, განხორციელება და შეფასება, რომელიც აკმაყოფილებს კომპუტინგის მოცემულ მოთხოვნებს პროგრამის დისციპლინის კონტექსტის შესაბამისად.** | | | | |
| **PI 2.1 პროგრამული უზრუნველყოფის შემუშავება, რომელიც აკმაყოფილებს კომპიუტინგის მოცემულ მოთხოვნებს.** | ძნელად შეუძლია შეიმუშაოს და დაწეროს ძირითადი ალგორითმები, სადაც დაუშვებელია მდგომარეობების შეცვლა მინიჭების ოპერატორების გამოყენებით ან მითითების (reference) ტოლობების გათვალისწინებით | შეუძლია შეიმუშაოს და დაწეროს ძირი  თადი ალგორითმები, სადაც დაუშვებელია მდგომარეობების შეცვლა მინიჭების ოპერატორების გამოყენებით ან მითითების (reference) ტოლობების გათვალისწინებით | შეუძლია შეიმუშაოს და დაწეროს საშუალო სირთულის ძირითადი ალგორითმები, სადაც დაუშვებელია მდგომარეობების შეცვლა მინიჭების ოპერატორების გამოყენებით ან მითითების (reference) ტოლობების გათვალისწინებით | შეუძლია შეიმუშაოს და დაწეროს რთული ალგორითმები, სადაც დაუშვებელია მდგომარეობების შეცვლა მინიჭების ოპერატორების გამოყენებით ან მითითების (reference) ტოლობების გათვალისწინებით |
| **PI 2.2 პროგრამული უზრუნველყოფის განხორციელება რომელიც აკმაყოფილებს კომპიუტინგის მოცემულ მოთხოვნებს.** | ძნელად შეუძლია განსაზღვროს და დაწეროს სასარგებლო ფუნქციები, რომლებიც არგუმენტად იღებენ და შედეგადაც აბრუნებენ სხვა ფუნქციებს | შეუძლია განსაზღვროს და დაწეროს სასარგებლო ფუნქციები, რომლებიც არგუმენტად იღებენ და შედეგადაც აბრუნებენ სხვა ფუნქციებს | ძნელად შეუძლია განსაზღვროს და დაწეროს საშუალო სირთულის სასარგებლო ფუნქციები, რომლებიც არგუმენტად იღებენ და შედეგადაც აბრუნებენ სხვა ფუნქციებს | შეუძლია განსაზღვროს და დაწეროს რთული ფუნქციები, რომლებიც არგუმენტად იღებენ და შედეგადაც აბრუნებენ სხვა ფუნქციებს |
| **PI 2.3 კომპიუტინგზე დაფუძნებული გადაწყვეტილების შეფასება, რომელიც აკმაყოფილებს კომპიუტინგის მოცემულ მოთხოვნებს.** |  |  |  |  |
| **PI 3 ეფექტური კომუნიკაცია სხვადასხვა პროფესიულ კონტექსტში** | | | | |
| **PI 3.1**  **ეფექტური მონაწილეობა ჯგუფურ დისკუსიებში** | ძნელად ერთვება ფუნქციონალური პარადიგმის შესაბამის საკითხებზე მიმდინარე ჯგუფურ დისკუსიაში | ერთვება ფუნქციონალური პარადიგმის შესაბამის საკითხებზე მიმდინარე ჯგუფურ დისკუსიაში | არგუმენტირებულად აყალიბებს საკუთარ აზრს ფუნქციონალური პარადიგმის შესაბამის საკითხებზე მიმდინარე ჯგუფურ დისკუსიაში | არგუმენტირებულად აყალიბებს და იცავს საკუთარ აზრს ფუნქციონალური პარადიგმის შესაბამის საკითხებზე მიმდინარე ჯგუფურ დისკუსიაში |
| **PI 3.2**  **ეფექტური პრეზენტაციის მომზადება** |  |  |  |  |
| **PI 3.3**  **პროექტის ეფექტური ანგარიშის მომზადება** | პროექტის ანგარიში დაუსრულებელია | პროექტის ანგარიში დასრულებულია არასრული დეტალიზებით | პროექტის ანგარიში დასრულებულია სრული დეტალიზებით | პროექტის ანგარიში დასრულებულია ეტაპების და შედეგების სრული აღწერით. |
| **PI 4 პროფესიული პასუხისმგებლობის აღიარება და დასაბუთებული გადაწყვეტილებების მიღება იურიდიული და ეთიკური პრინციპების საფუძველზე.** | | | | |
| **PI 4.1 : პროფესიულ პასუხისმგებლობის აღიარება კომპიუტინგის პრაქტიკის სფეროში სამართლებრივი და ეთიკური პრინციპების საფუძველზე.** |  |  |  |  |
| **PI 4.2 დასაბუთებული გადაწყვეტილების მიღება კომპიუტინგის პრაქტიკის სფეროში სამართლებრივი და ეთიკური პრინციპების საფუძველზე.** |  |  |  |  |
| **PI 5. ფუნქციების ეფექტურად შესრულება, როგორც წევრის ან ლიდერის პროგრამის შესაბამისი დისციპლინის ფარგლებში** | | | | |
| **PI 5. 1**  **გუნდში ჩართულია ეფექტურად, როგორც წევრის ან ლიდერი** |  |  |  |  |
| **PI 5.2**  **დასაბუთებული გადაწყვეტილების მიღება კომპიუტინგის პრაქტიკის სფეროში სამართლებრივი და ეთიკური პრინციპების საფუძველზე.** |  |  |  |  |
| **PI 6. კომპიუტერული მეცნიერებების და პროგრამუზი უზრუნველყოფის დეველოპმენთის საფუძვლების გამოყენება კომპიუტინგზე დაფუძნებული გადაწყვეტილებების მისაღებად.** | | | | |
| **PI 6.1 კომპიუტერული მეცნიერებების საფუძვლების გამოყენება გადაწყვეტილებების მისაღებად.** |  |  |  |  |
| **PI 6.2**  **პროგრამული უზრუნველყოფის დეველოპმენთის საფუძვლების გამოყენება გადაწყვეტილებების მისაღებად** |  |  |  |  |

**ABET კრიტერიუმები ( Criterion 3) მიღწევადობის დონე კურსის გავლის შემდეგ**

***ცხრილი B***

|  |  |
| --- | --- |
| **ABET კრიტერიუმები სტუდენტების შედეგების შესაფასებლად** | **მიღწევის დონე**  **(H-მაღალი, M-საშუალო, L-დაბალი)** |
| **1** | **2** |
| 1. კომპუტინგის რთული პრობლემის გაანალიზება და კომპიუტინგის და სხვა შესაბამისი დისციპლინების პრინციპების გამოყენება გადაწყვეტილების განსასაზღვრად. | M |
| 1. კომპიუტინგზე დაფუძნებული გადაწყვეტილების შემუშავება, იმპლემენტაცია და შეფასება კომპიუტინგის მოცემული რიგი მოთხოვნების დასაკმაყოფილებლად პროგრამის დისციპლინის კონტექსტში. | M |
| 1. ეფექტური კომუნიკაცია სხვადასხვა პროფესიულ კონტექსტში |  |
| 1. პროფესიული პასუხისმგებლობის გააზრება და დასაბუთებული დასკვნების გაკეთება კომპიუტინგის პრაქტიკაში იურიდიული და ეთიკური პრინციპების საფუძველზე. | L |
| 1. ეფექტური მუშაობა, როგორც გუნდის წევრის და ლიდერის, რომელიც ეწევა პროგრამის დისციპლინის შესაბამის საქმიანობას. |  |
| 1. კომპიუტერულ მეცნიერებათა თეორიის და პროგრამული უზრუნველყოფის შემუშავების საფუძველბის გამოყენება კომპიუტინგზე დაფუძნებული გადაწყვეტილებების მისაღებად. | L |

**სასწავლო კურსის შედეგების შესაბამისობა ABET კრიტერიუმებთან ( Criterion 3) და მათი შეფასების მეთოდები**

*ცხრილი 2*

|  |  |  |  |
| --- | --- | --- | --- |
| **სწავლების შედეგი** | **რუბრიკის თემა** | **ABET კრიტერიუმები სტუდენტების შედეგების შესაფასებლად Criterion 3** | **შეფასების მეთოდები** |
|  | **1** | **2** | **3** |
| RT1 | დაწეროს ძირითადი ალგორითმები, სადაც დაუშვებელია მდგომარეობების შეცვლა მინიჭების ოპერატორების გამოყენებით ან მითითების (reference) ტოლობების გათვალისწინებით | 1, 6 | 1 - ლაბორატორიული შეფასება  2- შუალედური გამოცდა  3- საკონტროლო |
| RT2 | დაწეროს სასარგებლო ფუნქციები, რომლებსაც არგუმენტად იღებენ და შედეგადაც აბრუნებენ სხვა ფუნქციებს | 2, 6 | 1-ლაბორატორიული შეფასება  2- შუალედური გამოცდა  3- საკონტროლო  4 - დასკვნითი გამოცდა |
| RT3 | ეფექტურად გამოიყენოს ერთი ან მეტი პროგრამული გარემო | 3 | 1-ლაბორატორიული შეფასება  3- საკონტროლო |
| RT4 | არჩევს მსგავსებასა და განსხვავებას პროცედურულ / ფუნქციონალურ მიდგომებს და ობიექტებზე ორიენტირებულ მიდგომას შორის | 1 , 2, 6 | 1-ლაბორატორიული შეფასება  3- საკონტროლო  4 - დასკვნითი გამოცდა |

**შეფასების ცხრილი**

E1- ლაბორატორიული შეფასება - 15 ქულა

E2- შუალედური გამოცდა - 25 ქულა

E3- საკონტროლო - პრაქტიკული შეფასება - 20 ქულა

E4- დასკვნითი გამოცდა - 40 ქულა.

*ცხრილი 3*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| სწავლის შედეგები | E1 | E2 | E3 | E4 |
| RT1 | X | X | X |  |
| RT2 | X | X | X | X |
| RT3 | X |  |  |  |
| RT4 | X |  | X | X |

*ცხრილი 4*

|  |
| --- |
| **Course number and name** |
| CS 101 – Functional Programming |
| **Semester Credit Hours/Contact Hours per week** |
| 5 (ECTS) (3 hours in week - 1 lecture, 2 lab) |
| **Instructor’s name** |
| **Natela Archvadze,** Assoc.Prof. Mob: +995 599 474 632: *e-mail:* [natela.archvadze@tsu.ge](mailto:natela.archvadze@tsu.ge) |
| **Textbook, Supplemental materials** |
| * Natela archvadze. punktsionaluri dapʼrogrameba Haskell- ze. ISBN 978-9941-13-709-9 (PDF). tbilisis sakhelmtsʼipo universitʼetʼis gamomtsemloba. 366 gv. 2018. (ge) * Graham Hutton. Programming in Haskell. 2007. ISBN:9780521871723 <http://ebooks.cambridge.org/ebook.jsf?bid=CBO9780511813672> * Dushkin R. V. Funktsional'noye programmirovaniye na yazyke Haskell. - M .: DMK-Press, 2007. - 608 str. ISBN 5-94074-335-8 (ru). * pʼraktʼikʼuli da laboratʼoriuli metsadineobebis masala: <http://e-learning.tsu.ge/course(ge)> * Simon Marlow. Parallel and Concurent Programming in Haskell. 2012. * [Learn You a Haskell for Great Good!: A Beginner's Guide](http://learnyouahaskell.com/chapters) by Miran Lipovaca <http://learnyouahaskell.com/chapters> * Richard Bird. Thinking Functionally With Haskell. Cambridge University Press. 2015 |
| **Specific course information** |
| This course introduces the student to the functional programming paradigm. Functional programming techniques are illustrated in the Haskell language. A significant portion of the learning occurs through the programming examples that are discussed in class. |
| **Prerequisites or co-requisites** |
| Basics Of Programming CS 101 |
| **Course status** |
| Elective |
| Outcomes |
| At the end of the course the student will be able to:   1. Write programs in a functional style; 2. Reason formally about functional programs; 3. Use polymorphism and higher-order functions; 4. Reason about the time and space complexity of programs. 5. explain on a simple problem how functional programming differs from imperative and object-oriented programming, and differs from logic programming. |
| **Student Outcomes from Criterion 3 covered by this Course** |
| 1.1, 1.2, 2.1, 3.1, 4.1 |
| **List of Topics Covered** |
| Fundamental concepts: functions, relations, recursion, tail-recursion, type systems, polymorphism, datatypes, recursive datatypes, lambda calculus, higher-order functions, data abstraction, module system. Programming in a functional programming language, such as Haskell . Similarities and differences with imperative, object-oriented, and logic programming. Lazy evaluation. |