

Основи програмирања

Вежбе 3

Исидора Грујић
isidora@uni.kg.ac.rs

Лазар Илић
iliclazar15@gmail.com

Факултет инжењерских наука
Универзитет у Крагујевцу

октобар 2023.

- 1 Кратак преглед
- 2 Нескаларни објекти - наставак
- 3 Петље - наставак
- 4 Додатак - учитавање података
- 5 Задаци

- 1 Кратак преглед
- 2 Нескаларни објекти - наставак
- 3 Петље - наставак
- 4 Додатак - учитавање података
- 5 Задаци

- ниске:
 - форматирање и исписивање на стандардном излазу
 - оператори над нискама
 - извлачење делова ниски: индексирање и одсецање
- гранање - **if** наредба: структура, сложено гранање, угнеждено гранање
- понављање: **while** петља

- 1 Кратак преглед
- 2 Нескаларни објекти - наставак**
- 3 Петље - наставак
- 4 Додатак - учитавање података
- 5 Задаци

Подела типова објеката

Позната подела:

- скаларни објекти
- нескаларни (сложени) објекти

Нова подела:

- непроменљиви

У ову категорију спадају сви скаларни типови објеката, као и два нескаларна типа: **ниске** и **поворке**.

- променљиви

У ову категорију спадају разни нескаларни типови објеката: **низови**, **скупови**, **речници**...

- енгл. **tuple** - поворка, н-торка
- дефинисање поворке:
 - () - празна поворка
 - (<члан>,)
 - (<члан1>, <члан2>, <члан3>)
- Члан поворке може бити објекат било ког типа!

Пример 1

Разни примери поворки.

Операције над поворкама

- "аритметички" оператори:
 - сабирање: `<поворка1> + <поворка2>`
 - множење целим бројем: `<цео_број> * <поворка>`
- извлачење делова поворке:
 - индексирање: `<поворка>[<индекс>]`
 - одсецање: `<поворка>[<почетак>:<крај>:<корак>]`
- уграђена функција `len()`: дужина поворке
- оператори поређења
- оператори чланства:
 - `<члан> in <поворка>`
 - `<члан> not in <поворка>`

Пример 2

Примери операција над поворкама.

- енгл. **list** - низ, листа
- дефинисање низа:
 - [] - празан низ
 - [<члан>]
 - [<члан1>, <члан2>, <члан3>]
- Члан низа може бити објекат било ког типа!

Пример 3

Разни примери низова.

Операције над низовима

- "аритметички" оператори:
 - сабирање: $\langle \text{низ1} \rangle + \langle \text{низ2} \rangle$
 - множење целим бројем: $\langle \text{цео_број} \rangle * \langle \text{низ} \rangle$
- извлачење делова низа:
 - индексирање: $\langle \text{низ} \rangle [\langle \text{индекс} \rangle]$
 - одсецање: $\langle \text{низ} \rangle [\langle \text{почетак} \rangle : \langle \text{крај} \rangle : \langle \text{корак} \rangle]$
- уграђена функција **len()**: дужина низа
- оператори поређења
- оператори чланства:
 - $\langle \text{члан} \rangle$ **in** $\langle \text{низ} \rangle$
 - $\langle \text{члан} \rangle$ **not in** $\langle \text{низ} \rangle$

Пример 4

Примери операција над низовима.

У чему је онда разлика између поворки и низова?

Непроменљиви и променљиви објекти

- Поворке су **непроменљиви** објекти.
 - Прослеђивањем непроменљивог објекта као параметар функције прослеђујемо његову вредност. Ствара се локална променљива којој се додељује ова вредност. Позивањем ове променљиве можемо да приступимо вредности(ма) локално дефинисаног објекта.
 - Не можемо променити вредност "оригиналних" објеката унутар неке функције.
 - Немогуће је променити вредност појединачних чланова сложеног непроменљивог објекта, као ни његову дужину.
- Низови су **променљиви** објекти.
 - Прослеђивањем променљивог објекта заправо прослеђујемо сам тај објекат, а не његову вредност, односно копију.
 - Могуће је променити вредност "оригиналних" објеката унутар неке функције.
 - Могуће је стварати више путања према једном истом објекту - **алијас** (енгл. **alias**).
 - Могућа је манипулација над самим објектом - промена вредности чланова, скраћивање, додавање нових чланова и слично.

Пример 5

Понашање прослеђених објеката унутар функције.

Низови као променљиви објекти

Додатне операције над низовима:

- промена члана: `<низ>[<индекс>] = <нови_члан>`
- додавање нових чланова: функција **append()**
`<низ>.append(<нови_члан>)`
- брисање чланова: функција **remove()**
`<низ>.remove(<постојећи_члан>)`

Пример 6

Промена члана низа; алијас, клонирање.

Пример 7

Коришћење функције `append()`.

Пример 8

Коришћење функције `remove()`.

- 1 Кратак преглед
- 2 Нескаларни објекти - наставак
- 3 Петље - наставак**
- 4 Додатак - учитавање података
- 5 Задачи

Код `while` петљи важи:

- понављачка променљива се дефинише ван тела петље
- променљива се проверава пре сваког циклуса петље (**итерације**)
- променљива се такође мора мењати унутар саме петље

Додатак: Коришћење наредбе **break**

Пример 9

Илустрација коришћења наредбе **break** кроз задатак.

Структура **for** петље

- За разлику од **while** петље, механизам **for** петље је такав да број итерација, као и понављачка променљива, зависе од проласка кроз неки сложени објекат.
- Постављање услова:
 - **for** члан **in** <сложени_објекат>:
 - **for** члан **in range**(<крај>):
 - **for** члан **in range**(<почетак>, <крај>):
 - **for** члан **in range**(<почетак>, <крај>, <корак>):

Пример 10

Понављање применом **for** петље.

- Наредба **continue**

Пример 11

Илустрација коришћења наредбе **continue** кроз задатак.

- 1 Кратак преглед
- 2 Нескаларни објекти - наставак
- 3 Петље - наставак
- 4 Додатак - учитавање података
- 5 Задаци

Учитавање више података са стандардног улаза

- уграђена функција за ниске: `split()`
`<ниска>.split(<подниска>, <број_одсецања>)`

Пример 12

Коришћење функције `split()`.

- функција `map()`
`map(<функција>, <сложени_објекат>)`

Пример 13

Коришћење функције `map()`.

- Комбиновањем ових функција можемо да извршимо учитавање више променљивих са стандардног улаза одједном.

Пример 14

Учитавање више променљивих са стандардног улаза.

- 1 Кратак преглед
- 2 Нескаларни објекти - наставак
- 3 Петље - наставак
- 4 Додатак - учитавање података
- 5 **Задаци**

Задатке са трећих вежби можете пронаћи на [страници предмета](#).