FAKULTET INFORMATIKE
I DIGITALNIH TEGNOLOGIJA
Preddiplomski studij informatike

Projektni zadatak iz kolegija

PROGRAMSKE PARADIGME I JEZICI

Funkcijsko Programiranje: Elementary Cellular Automaton

Mentori: prof. dr. sc. Ana Meštrović
Karlo Babić, mag. inf.

Autor: Bojan Radulović

U Rijeci, Studeni 2022.

Sadržaj

1. Opis koda

1.1 Pregled funkcija

```
def length(li):

    def length_iter(inner_li, n):
        if not inner_li:
            return n
        return length_iter(inner_li[1:], n + 1)

    return length_iter(li, 0)
```

Kao argument uzima listu, iterira kroz sve njene elemente i vraća njenu duljinu.

```
def reversed_str(string):

    def reversed_str_iter(inner_str, rev_str):
        if inner_str == '':
            return rev_str
        return reversed_str_iter(inner_str[:-1], rev_str + inner_str[-1])

    return reversed_str_iter(string, '')
```

Kao argument uzima string i vraća novi string s obrnutim redoslijedom znakova.

```
def list_to_string(li):

    def list_to_string_iter(inner_li, new_string):
        if not inner_li:
            return new_string
        return list_to_string_iter(inner_li[1:], new_string +
str(inner_li[0]))

    return list_to_string_iter(li, "")
```

Kao argument uzima listu i spaja sve elemente u novi string te vraća taj string.

```
def bits_to_int(li):
    return int(list_to_string(li), 2)
```

Kao argument uzima listu čiji elementi predstavljaju niz bitova te pretvara taj niz bitova u integer.

```
def int_to_bits(n):
    return '{0:08b}'.format(n)
```

Kao argument uzima integer te vraća 8 bitnu reprezentaciju tog integera.

```
def ceiling(n):
    return int(-1 * n // 1 * -1)
```

Kao argument uzima float te ga zaokružuje na veću vrijednosti i vraća tu vrijednost u obliku integera.

```
def matrix_to_string(matrix):

    def matrix_to_string_iter(inner_matrix, new_string):
        if not inner_matrix:
            return new_string[:-1]
        return matrix_to_string_iter(inner_matrix[1:], new_string +
list_to_string(inner_matrix[0]) + '\n')

    return matrix_to_string_iter(matrix, "")
```

Kao argument uzima dvodimenzionalnu listu te vraća string u kojem svaki redak predstavlja jednu podlistu dane dvodimenzionalne liste.

```
def pretty_string(old_string):
    return old_string.replace("0", " ").replace("1", "■")
```

Kao argument uzima string nula i jedinica te zamijenjuje sve nule s razmakom i sve jedinice sa "■" radi ljepšeg prikaza.

```
def elementary_cellular_automaton(rule, width=33, height=16, first_row=None):

    def elementary_cellular_automaton_iter(h, inner_rule, matrix):
        if h <= 0:
            return matrix
        return elementary_cellular_automaton_iter(
            h - 1,
            inner_rule,
            matrix + [generate_row(matrix[-1], inner_rule)]
        )

    if not 0 <= rule <= 255:
        return "Rule number must be between 0 and 255"

    if width < 3:
        return "Minimum width is 3"

    if height < 1:
        return "Minimum height is 1"

    if first_row:
        return elementary_cellular_automaton_iter(
            height - 1,
            int_to_bits(rule),
            [first_row]
        )
    return elementary_cellular_automaton_iter(
        height - 1,
        int_to_bits(rule),
        [generate_first_row(width)]
    )
```

Kao argumente uzima pravilo koje želimo generirati (integer od 0 do 255), željenu širinu outputa (integer veći ili jednak tri), željenu visinu outputa (integer veći ili jednak jedan) i opcionalni argument kojim se postavlja prvi redak outputa (lista bitova). Ako se ovaj argument ostavi prazan, algoritam će sam generirati prvi redak čiji će broj elemenata biti jednak širini outputa, gdje su svi elementi jednaki nuli osim središnjeg elementa koji je jednak jedan. Vraća listu lista bitova u kojoj svaka lista bitova predstavlja jedan redak outputa. Ova funkcija sadrži više pomoćnih funkcija:

```
def rule_check(bit_li, inner_rule):
    return list(reversed_str(inner_rule))[bits_to_int(bit_li)]
```

Kao argumente uzima listu od 3 bita i pravilo te vraća vrijednost koju će središnji od ta 3 bita poprimiti u sljedećem retku (integer 1 ili 0).

```
def generate_first_row(w):
    return [0] * int(w / 2) + [1] + [0] * int(w / 2 - 1 + ceiling(w % 2))
```

Kao argument uzima integer koji predstavlja širinu željenog outputa te generira prvi redak outputa na gore definiran način. Vraća listu bitova koja predstavlja prvi redak outputa.

```
def generate_row(old_row, inner_rule):

    def generate_row_iter(inner_old_row, new_row, iter_rule):
        if length(inner_old_row) <= 2:
            return new_row
        return generate_row_iter(
            inner_old_row[1:],
            new_row + [rule_check([inner_old_row[0], inner_old_row[1],
inner_old_row[2]], iter_rule)],
            iter_rule
        )
```

Kao argumente uzima listu koja predstavlja prošli redak outputa i pravilo. Vraća listu koja predstavlja sljedeći redak outputa.

1.2 Pregled rada koda

Prvo se poziva funkcija elementary_cellular_automaton sa željenim argumentima. Funkcija prvo provjerava točnost unosa. Zatim, ako prvi redak nije unesen, on se automatski generira. Nakon toga se generiraju svi ostali redci do željenog broja redaka (visine). Radi ljepšeg ispisa, output ove funkcije prosljeđuje se funkciji matrix_to_string te se njen output dalje prosljeđuje funkciji pretty_string. Na kraju se konačni output prosljeđuje ugrađenoj funkciji print kako bi se ispisao na ekran.

2. Primjena deklarativnog stila

- Vrijednosti varijabla se ne mijenjaju kroz program

- Kod opisuje što, a ne kako program treba raditi
- Ne koriste se petlje (nego rekurzije)
- Koriste se funkcije višeg reda (map)
- Podatci i funkcije su u potpunosti razdvojeni
- Korištenje generičkih funkcija
- Referencijalna prozirnost (funkcije nemaju popratnih pojava)
- Kod je kraći i jasniji

3. Prednosti i mane deklarativnog stila

Vjerujem kako deklarativni stil, u ovom konkretnom slučaju kao i općenito, ima mnoge prednosti. Kod pisan deklarativnim stilom je kraći, jasniji i ekspresivniji od onog pisanog imperativnim stilom. Funkcije nemaju popratnih pojava te uvijek za isti input vraćaju isti output što ih čini lakšima za testirati i debuggirati. Funkcije pisane na generički način mogu se upotrebljavati u raznim okolnostima itd.

Neke od mana deklarativnog stila su da se umjesto petlja koriste rekurzije koje znaju biti sporije i memorijski intenzivnije. Osim toga, funkcije za input/output u/iz programa nije moguće implementirati deklarativno te je zato te funkcije potrebno što je više moguće izolirati od ostatka programa. Nadalje, izbjegavao sam korištenje gotovih funkcija u Pythonu jer mnoge od njih koriste imperativni stil.

4. Izvori

Elementary Cellular Automaton: https://mathworld.wolfram.com/ElementaryCellularAutomaton.html

Stephen Wolfram, A New Kind of Science: https://www.wolframscience.com/nks/