Poročilo naloge:

Command prompt extended

Avtor: Luka Orlić Razred: 1.A

Datum: 12.12.2018 Mentorica: Marina Trost

Contents

[2. Opis naloge 3](#_Toc533089969)

[a) Opis: 3](#_Toc533089971)

[b) Načrti: 3](#_Toc533089972)

[3. Elementi 3](#_Toc533089973)

[4. Dokaz o naučenem 4](#_Toc533089974)

[5. Zaključek 5](#_Toc533089975)

# Opis naloge



## Opis:

Programerski problem se sooča s shranjevanjem in iskanjem po nizih znakov, ki so odgovori raznih ukazov vneseni v program. Program za osnovo uporablja ukaze iz ukazne vrstice ter 13 samostojnih funkcij.

## Načrti:

Načrt pred začetkom naloge:

|  |  |
| --- | --- |
| Kaj že vem? | Česa še ne vem?/naloge |
| Glavni modul (del programa) | Implementacija funkcije Nslookup |
| Osnovni help ukaz | Implementacije funkcije Ping |
|  | Implementacija shranjevanja datotek |
|  | Implementacija search funkcije |
|  | Prirejanje podrobnih opisov za ukaz help |

Načrt po zaključku naloge:

|  |  |
| --- | --- |
| Kaj že vem? | Česa še ne vem?/naloge |
| Glavni modul (del programa) |
| Osnovni help ukaz |
| Implementacija funkcije Nslookup |
| Implementacije funkcije Ping |
| Implementacija shranjevanja datotek |
| Implementacija search funkcije  Prirejanje podrobnih opisov za ukaz help |
|

# Elementi

Program vsebuje 40 funkcij, 6 uvozov (import-ov), 1 "while"[[1]](#footnote-1) zanka ter 2 "For"[[2]](#footnote-2) zanki in 17 "if"[[3]](#footnote-3) izjav. Program deluje tako da ima 40 različnih funkcij, ki jih kliče operativni del programa. Nekatere funkcije temeljijo na povratnih informacijah ukazne vrstice[[4]](#footnote-4). Ostale se nanašajo na nize črk[[5]](#footnote-5), ki jih povrnejo funkcije, katere temeljijo na ukazni vrstici, to je s tem namenom, da se program lažje prilagodi drugim string-om in uporabam. Program vsebuje tudi glavni oziroma operativni del[[6]](#footnote-6), ki preverja vnose ukazov in po tem izvede pravilno funkcijo.

# Dokaz o naučenem

|  |  |  |
| --- | --- | --- |
| Koncept zaporednosti  izvajanja ukazov | Primer nakazuje na pomembnost zaporedja, saj se najprej mora definirati spremenljivka x da se lahko definira spremenljivka name | x = current\_time()  name = str(cmd)+str(x)+".txt"[[7]](#footnote-7) |
| Koncept vejitve (če...) | Vejitev če v tem primeru pogleda, če je program vrnil vrednost nula oziroma vse poteka kakor je potrebno, izpiše string, drugače napiše da ukaz ni uspel. | if addr[0] == 0:  nslookupr = addr[1]  print("URL: " + URL + " - Returned information: \n\n" + addr[1])  else:  print("Nslookup failed - Returned status:" + addr[0])[[8]](#footnote-8) |
| Koncept zanke | Zanka celotnega operativnega dela skrbi, da program se ne ustavi in čel čas čaka vnos ukaza, se ponavlja do vnosa ukaza "Exit" ali temu podobnega, ki prekine program. | while work:  command = input("Enter Command: ").lower()  . . .  else :  elsecommand(command)[[9]](#footnote-9) |
| Vhod | Funkcija, ki potrebuje naknadni vhod informacij. | nslookup(input("Enter URL: ")) |
| Izhod | Formatirano izpiše, da ukat ni uspel. | print("\n\tInput command not valid. \n\tInput commnad: " + str(inpc)) |
| Koncept spremenljivke | Spremenljivka, ki se spremeni vsakič, ko je vpisan ukaz sysinfo, spremenljivka vsebuje formatirani string, ki ga pridobi iz funkcije platform[[10]](#footnote-10) | platform = "" |
| Koncept podprograma  oziroma funkcije | Podprogram imenovan platform, ki izpiše sistemske informacije | def platform(platcom): #izpise sistemske informacije  . . .  platform +='\t ' + pr2 + '\n'[[11]](#footnote-11) |
| Tabelarična  spremenljivka (seznam) | Je seznam, ki vsebuje krajše definicije ukazov in pojasnitve le teh. | helpt = [  "ping - Ping computer system",  . . .  "exit - Exit program"  ][[12]](#footnote-12) |

# Zaključek

Program bi lahko nadgradili s posamičnimi samostoječimi ukazi, ki bi izvajala težja oziroma manj dostopnejše procese kakor ukazna vrstica. Namesto če ukazov bi lahko uporabili ukaz "Switch", ter namesto funkcij naredili "class"-e. Najtežji del programa so mi bile funkcije iskanja in shranjevanje. Problem sem rešil z pomočjo posveta s znanci in programerji ter veliko poizkušanja in razmišljanja.

1. While zanka – (oziroma dokler zanka) ponavlja vse znotraj zanke doklet trditev ni več resnična (angleško false). [↑](#footnote-ref-1)
2. For zanka – (oziroma za zanka) se ponavlja tolikokrat kolikokrat je obseg (angleško range). [↑](#footnote-ref-2)
3. If izjava – (ozrioma če izjava) pogleda če trditev drži in izvede podrejene ukaze, drugače se nadaljuje pod ukazi drugače če nato drugače. [↑](#footnote-ref-3)
4. Ukazna vrstica – (Angleško command prompt) v nadeljevanju tudi command prompt. [↑](#footnote-ref-4)
5. Niz črk – (Angleško string) v nadeljevanju tudi string. [↑](#footnote-ref-5)
6. Operativni del – (v programu : GCF oziroma global command function) je glavni del programa, ki se do konca vrti in čaka na vnos ukaza. [↑](#footnote-ref-6)
7. Izvirna koda je v programu v vrsticah 663-664. [↑](#footnote-ref-7)
8. Izvirna koda je v programu v vrsticah 200 – 204. [↑](#footnote-ref-8)
9. Izvirna koda je v programu v vrsticah 733-766; v tabeli ". . ." ponazarja vmesno kodo, ki ni relevantna za zanko. [↑](#footnote-ref-9)
10. Definicija funkcije najdena v vrsticah 515-606. [↑](#footnote-ref-10)
11. Definicija funkcije najdena v vrsticah 515-606; v tabeli ". . ." ponazarja vmesno kodo, ki ni relevantna za zanko. [↑](#footnote-ref-11)
12. Izvirna koda je v programu v vrsticah 288-304. [↑](#footnote-ref-12)