NAPOJ

UČNA PRIPRAVA

Osnovni pojmi algoritmike

DATUM NAPOJ

OSNOVNI PODATKI

Šola:
Letnik: 1. Letnik
Datum:
Predmet: Informatika
Učna tema: Abstrakcija, algoritem
Učna enota: abstrakcija, programerski problem, program, algoritem, načini zapisa algoritma, pravilnost in ustavljivost algoritma
Učne oblike: - frontalno, individualno
Učne metode: - razlaga, pogovor, demonstracija, reševanje problemov
Predznanje: - poznajo pojme tabela, funkcija
Operativni učni cilji Ob koncu učne ure učenec: - Razume kaj je ter pomen abstrakcije. - Razuma kaj je algoritem. - Zna zapisati algoritem na različne načine.
Učna sredstva: - Učila: delovni list, računalnik - Učni pripomočki: računalnik, Python, zvezek, tabla
Didaktične etape učnega procesa: 1. Pripravljanje ali uvajanje 2. Obravnava nove učne snovi ali usvajanje 3. Urjenje 4. Ponavljanje 5. Preverjanje
Medpredmetne povezave: slovenščina, angleščina
Literatura: - B. Haberman, O. Muller, Teaching abstraction to novices: Pattern-based and adt-based problem-solving processes, in: Frontiers in Education Conference, 2008. FIE 2008. 38th Annual, IEEE, 2008, pp. F1C–7. - B. Habermana, Z. Scherzb, Connectivity between abstraction layers in declarative adt-based problem-solving processes, Informatics in Education 8 (1) (2009) 3. - N. Dale, H. M. Walker, Abstract data types: specifications, implementations, and applications, Jones & Bartlett Learning, 1996. - G. Anželj, J. Brank, A. Brodnik, P. Bulić, M. Ciglarič, M. Đukić, L. Furst, M. Kikelj, A. Krapež, H. Medvešek, N. Mori, M. Pančur, P. Sterle, Računalništvo in informatika 1, Založba Univerze na Primorskem and Založba Fakultete za računalništvo in informatiko and Založba Fakultete za elektrotehniko and računalništvo in informatiko, 2015. URL https://lusy.fri.uni-

lj.si/ucbenik/book/index.html.
Novi pojmi: - Abstrakcija, programerski problem, program, algoritem, psevdokoda, zapis algoritma v naravnem jeziku, pravilnost algoritma, ustavljivost algoritma.
Priloga: - python modul Sklad - rešitev naloge Gumb Nazaj - delovni list - delovni list - rešitve

POTEK UČNE URE

UVODNI DEL: UVAJANJE

ČAS	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI
Uvod 8 min	*Pozdravi učence, zapiše manjkajoče učence in po potrebi prosi dežurnega učenca, da pobriše tablo.*	Odzdravijo, naštejejo manjkajoče učence, dežurni učenec pobriše tablo.	Frontalno.
Motivacija 2 min	Danes bomo spoznali, kako problem iz realnega življenja skozi postopek pretvorimo v programerski problem, katerega lahko rešimo s pomočjo računalnika. To znanje je pomembno, ker se v praksi tako tudi rešuje probleme.	Poslušajo.	Frontalno, razlaga.

GLAVNI DEL: OBRAVNAVANJE UČNE SNOVI

VSEBINSKI POUDARKI	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI
Naloga 2 min	Današnjo snov bomo spoznali s pomočjo naloge. Naloga je sledeča: Smo programerji, ki programiramo nov spletni brskalnik. Naša trenutna naloga je v spletnem brskalniku sprogramirati delovanje gumba »Nazaj«. To je gumb, ki nam odpre zadnjo stran, katero smo obiskali.	Poslušajo in si zapisujejo.	Frontalno, razlaga.
Model	V našem primeru je naloga hkrati tudi **programerski problem**, saj je to izziv, ki ga rešimo s		Frontalno,

abstrakcije 33 minut	pomočjo računalnika. Reševanje naloge se bomo lotili na podlagi modela abstrakcije. To je model, s katerim se učimo abstrakcije pri procesu reševanja problemov. Model se deli na 2 dela – analiza problema ter izdelava rešitve problema, vsak del pa se deli še na 3 poddele. Začnimo s prvim delom: **I. Razumevanje in analiza problema** 1. *Konceptualizacija*: tu moramo razumeti in opisati problem, ki ga definira naloga. V našem primeru lahko definiramo 3 glavne dele naloge. Ali lahko katerega sami določite? i. Hranjenje vseh prejšnjih spletnih strani, ki jih je uporabnik obiskal. ii. Dodajanje novih obiskanih spletnih strani. iii. Pridobitev zadnje obiskane spletne strani. 2. *Posplošitev*: tu se z mislimi odmaknemo od naše naloge in jo posplošimo na splošen problem, neodvisen od konteksta. Katerega od prej naštetih delov lahko posplošimo? Posplošitvi rečemo lahko tudi **abstrakcija**, to je postopek, kjer zanemarimo nebistvene okoliščine. 3. *Izbira abstraktnega modela*: Tu izberemo ustrezno abstraktno podatkovno strukturo. To ni nič drugega kot pravila v kakšni obliki zapišemo podatke. Nekaj podatkovnih struktur: i. Vrsta, kjer so podatki zapisani po vrsti tako kot so bili dodani (primer: vrsta za kino). ii. Sklad, kjer so podatki zapisani od najstarejšega do najnovejšega (primer: kup krožnikov). iii. Graf, kjer so poleg podatkov opisane tudi povezave med podatki (primer: letalske povezave med mesti). iv. Ostale strukture, ki jih bomo spoznali v naslednjih tednih.	Spremljajo razlago in si zapišejo v zvezek. Odgovorijo, učitelj jih napelje na prave odgovore. Odgovorijo: hranimo lahko poljubno vrsto podatkov, ne zgolj nizov. Odgovorijo: sklad,	razlaga, pogovor.

	Katerega od naštetih struktur po vašem nam najbolj ustreza? Preden gremo na drugi del, dajmo na hitro obrazložiti izbrano podatkovno strukturo sklad. Lahko jo razumemo kot zaporedje podatkov, v katerem se elementi tako dodajajo kot odvzemajo samo na koncu/vrhu. Predstavljamo si ga lahko kot skladovnico krožnikov, saj krožnik načeloma vedno postavimo na vrh, z vrha pa ga tudi vzamemo. Se spomnite še kakšen primere uporabe sklada? Pojdimo sedaj na drugi del modela abstrakcije. **I. Izdelava ustrezne rešitve** 4. *Formalizacija*: tu definiramo povezave med problemom iz naloge in izbrano podatkovno strukturo. Program mora podpirati naslednje funkcije: i. Ko uporabnik požene spletni brskalnik, program ustvari prazen seznam povezav. ii. Vsakič, ko uporabnik dostopa do nove spletne strani, se trenutna spletna povezava shrani v seznam povezav. iii. Ko uporabnik klikne na gumb »nazaj«, program pridobi zadnjo povezavo. 5. *Realizacija in testiranje*: tu lahko končno začnemo s pisanjem programa in ga na koncu testiramo - preverimo **pravilnost** programa (da za vsak vhod podatkov proizvede pričakovan pravilen izhod) in **ustavljivost** programa (da se program vedno konča).	saj nam ustreza hranjenje podatkov od najnovejšega do najstarejšega. Učenci naštejejo primere uporabe podatkovne strukture sklad.	
Reševanje naloge 25 min	Sedaj se osredotočimo na realizacijo – samo programiranje naše rešitve. Odločili smo se za uporabo podatkovne strukture sklad. Implementacijo le-tega sem, da se naloga ne preveč zakomplicira, napisal sam, vsebuje pa	Sledijo razlagi in si zapisujejo.	Frontalno, razlaga, pogovor, individualno ali skupinsko delo, reševanje problemov,

	funkcije:
	
	1. ustvari() za kreiranje novega praznega seznama,
	2. dodaj() za dodajanje elementa v sklad,
	3. vzemi() za pridobitev zadnje dodanega elementa iz sklada,
	4. jePrazen() za preverjanje, ali je sklad prazen ter
	5. izpisi(), ki izpise vsebino sklada za potrebe testiranja.
	
	Te funkcije bomo uporabili v našem programu. Datoteko najdete na spletni učilnici. Vaša naloga je, da jo kopirate v mapo, kjer boste imeli tudi datoteko rešitve naše naloge.
	
	Pri konceptualizaciji smo našteli 3 funkcionalnosti, ki jih mora program podpirati. Kar začnimo s prvim - hranjenje vseh prejšnjih spletnih strani, ki jih je uporabnik obiskal.
	
	Spletne strani bomo hranili kot spletne povezave v skladu. Sprogramirajmo torej prvi del naloge - kreiranje novega praznega sklada, ki bo vseboval spletne povezave. Za vsak del naloge bomo zapisali algoritem v 3 različnih načinih – v naravnem jeziku, s psevdokodo ter s funkcijo. S tem se bomo naučili **različne zapise algoritma**. Vi si zapišite rešitev v Pythonu na računalnik, da bomo na koncu preverili pravilnost naše rešitve.
	
	Del 1: Kreiranje novega praznega sklada.
	
	Zapis v naravnem jeziku:
	
	Ustvari nov prazen sklad.
	Vrni sklad.
	
	Zapis s psevdokodo:
	
	Povezave = ustvari sklad
	Vrni povezave
	
	Zapis s funkcijo:
	
	`def ustvariPrazenSeznamPovezav():`
	`  povezave = ustvari()`
	`  return povezave`
	
	Del 2: Dodajanje novih obiskanih spletnih strani.
	
	Zapis v naravnem jeziku:
	
	Dodaj spletno povezavo v sklad.
		Poberejo iz spletne strani datoteko sklad.py in jo kopirajo v ustrezno mapo.
		
		Si zapišejo funkcijo v Python datoteko.
			demonstracija.

	Zapis s psevdokodo:
	
	Dodaj v sklad(povezava).
	
	Kako bi to zapisali kot funkcijo v Pythonu?
	
	Zapis s funkcijo:
	
	`def dodajNovoStran(povezave, povezava):`
	`  dodaj(povezave, povezava)`
	
	Del 3: Pridobitev zadnje spletne povezave iz sklada, ki se zgodi, ko uporabnik klikne na gumb »nazaj«.
	
	Zapis v naravnem jeziku:
	
	Če sklad ni prazen, vzemi ven zadnjo dodano spletno povezavo ter jo vrni.
	
	Zapis s psevdokodo:
	
	Če sklad ni prazen, potem:
	povezava = vzemi povezavo iz sklada
	vrni povezava
	
	Kako bi to zapisali kot funkcijo v Pythonu?
	
	Zapis s funkcijo:
	
	`def pridobiZadnjoPovezavo(povezave):`
	`  if not jePrazen(povezave):`
	`  povezava = vzemi(povezave)`
	`  return povezava`
	
	To je tudi vsak koda, ki jo potrebujemo. Za konec lahko še testiramo naš program. Tudi samo preverite pravilnost naše rešitve.
	
	`povezave = ustvariPrazenSeznamPovezav()`
	`dodajNovoStran(povezave, "www.google.com")`
	`dodajNovoStran(povezave, "www.fb.com")`
	`povezava = pridobiZadnjoPovezavo(povezave)`
	`izpisi(povezave)`
		Odgovorijo, si zapišejo funkcijo v Python datoteko.
		
		Odgovorijo, si zapišejo funkcijo v Python datoteko.
		
		Preverijo pravilnost programa.	

ZAKLJUČNI DEL: ZAKLJUČNO PONAVLJANJE / PREVERJANJE

ČAS	UČITELJ	UČENEC	UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI
15 min	*Ponovi nove pojme in razdeli delovne liste, za katerega na koncu skupaj z učenci preveri rešitve.*	Poslušajo, rešijo delovni list, pregledajo rešitve..	Frontalno, individualno, razlaga, pogovor, reševanje problemov.
5 min	*Se zahvali učencem za sodelovanje, jim da napotke za domačo nalogo. Dežurni učenec naj po potrebi pobriše tablo. Učenci naj ugasnejo računalnike.*	Poslušajo, ugasnejo računalnike, dežurni učenec pobriše tablo.	Frontalno.

Priloge

Priloga 1: Python modul Sklad

```
def ustvari():
    return []

def dodaj(sklad, element):
    sklad.append(element)

def vzemi(sklad):
    return sklad.pop()

def jePrazen(sklad):
    return (len(sklad)==0)

def izpisi(sklad):
    print(sklad)
```

Priloga 2: Rešitev naloge Gumb Nazaj.

```
from sklad import *

def ustvariPrazenSeznamPovezav():
    povezave = ustvari()
    return povezave

def dodajNovoStran(povezave, povezava):
    dodaj(povezave, povezava)

def pridobiZadnjoPovezavo(povezave):
    if not jePrazen(povezave):
        povezava = vzemi(povezave)
        return povezava


'''''''''''''''''''''''''''
testiranje nase resitve
'''''''''''''''''''''''''''
povezave = ustvariPrazenSeznamPovezav()

dodajNovoStran(povezave, "www.google.com")
dodajNovoStran(povezave, "www.fb.com")

povezava = pridobiZadnjoPovezavo(povezave)


izpisi(povezave)
```

Priloga 3: delovni list

Delovni list – osnovni pojmi algoritmike

1. Kaj je abstrakcija?

2. Ali imajo lahko različne naloge isti osnoven problem? Zakaj?

3. Spoznali smo model abstrakcije, ki se dela na 2 dela, vsak del pa na 3 poddele. Napišite oba dela in okvirno kaj pri njih naredimo.

4. Povedali smo, da za programerski problem obstajata 2 lastnosti, da bo le-ta bil pravilen. Prva je pravilnost programa - da za vsak primerek problema proizvede pričakovan izhod. Katera je druga lastnost? Opiši jo.

5. Poznamo 3 možne zapisa algoritma. Naštejte jih ter dodaj še krajši opis vsakega.

6. Naslednji algoritem zapiši s programsko kodo.

Za vsak vnos uporabnika preveri, ali je enak nizu »algoritem« ter če je, izpiši »Heureka!«

Priloga 4: delovni list - rešitve

Delovni list – osnovni pojmi algoritmike - Rešitve

1. Kaj je abstrakcija?

Abstrakcija je posplošitev problema, izločitev nebistvenih informacij.

2. Ali imajo lahko različne naloge isti osnoven problem? Zakaj?

Da, ker je bistvo problema enako. Primer: urejanje tekačev od najhitrejšega do najpočasnejšega ter urejanje živali od najhitrejše do najpočasnejše.

3. Spoznali smo model abstrakcije, ki se dela na 2 dela, vsak del pa na 3 poddele. Napišite oba dela in okvirno kaj pri njih naredimo.

Razumevanje in analiza problema – konceptualiziramo in posplošimo nalogo ter izberemo ustrezno podatkovno strukturo.

Izdelava ustrezne rešitve – formalizacija, realizacija in testiranje.

4. Povedali smo, da za programerski problem obstajata 2 lastnosti, da bo le-ta bil pravilen. Prva je pravilnost programa - da za vsak primerek problema proizvede pričakovan izhod. Katera je druga lastnost? Opiši jo.

Ustavljivost programa. Ali se program vedno ustvari.

5. Poznamo 3 možne zapisa algoritma. Naštejte jih ter dodaj še krajši opis vsakega.

Zapis v naravnem jeziku – opišemo rešitev v jeziku ljudi.

Zapis s programsko kodo – napišemo rešitev v programskem jeziku.

Zapis s psevdokodo – opišemo rešitve v na pol naravnem ter na pol programskem jeziku.

6. Naslednji algoritem zapiši s programsko kodo.

Za vsak vnos uporabnika preveri, ali je enak nizu »algoritem« ter če je, izpiši »Heureka!«.

```
while True:
    niz = input()
    if niz == »algoritem«:
        print(»Heureka!«)
```