Poročilo:

O izvedeni vaji:

11\_lav03\_vaja

Laboratorijske vaje

Vaja 11

Maj Mohar, G3A

Ljubljana, december 2024

**Naloga 1**

V zagonski(driver) metodi razreda MojeMetodeZaTabele obstaja delaracija tabele z inicializacijo elementov:

int[] ti = {12,3,5,9,-2,4};

Spišite razredni metodi vrniPrvega/1 in vrniZadnjega/1, ki bosta kot argument prejeli tabelo, kot je podana in: prva naj vrne vrednost prvega elementa tabele, druga vrednost zadnjega elementa tabele. Za vsak slučaj: vrniPrvega/1 interpretirate kot (verjetno static) int vrniPrvega(int[] tab) ….

**Naloga 2**

V razred dodajte še (dve) enakovredni metodi, ki bosta cilje iz naloge 1 izvedli na tabeli znakov, kot npr.:

char[] tc = new char[]{'a','h','H','j','X'};

(metodi opravljata enako delo kot tisti iz Naloga1. Zaslužita enako ime? )

**Naloga 3**

* Razredna metoda vrniVsoto/1 vrne vsoto elementov take tabele, kot je definirana v nalogi 1
* Razredna metoda vrniVsoto/2 vrne vsoto (prvih) n elementov tabele, kot je podana v nalogi 1

premislite : bo vrnjena vsota vrste int ali long ? Kdaj boste uporabili prvega, kdaj drugega. Pri vrniVsoto/2 je drugi argument število elementov, ki jih upoštevamo pri izračunu vsote. Če je ta nič, je vsota pač nič. Kaj pa če je njegova vrednost negativna? Hm, recimo, da potem upošteva v vsoti z-a-d-n-j-i-h n elementov v tabeli. In, če je elementov manj kot n, jih je v vsoti upoštevanih pač manj kot n.

* Razredna metoda vrniVsoto/3 vrne vsoto n zaporednih elementov tabele, začenši s k-tim elementom tabele. (glejte predhodne opombe)

**Naloga 4**

Metodi razreda MojeMetodeZaTabele z imenoma getIntTab/1 in genCharTab/1 vrneta tabelo ustreznega tipa (glej ime metode) in sicer tako dolgo, kot je vrednost argumenta tabele. Vrnjeni tabeli sta napolnjeni z naključnimi vrednostmi iz obsega int oz. iz množice znakov/simbolov za črke in številke.

Za kontrolo delovanja v razred dodajte še metodi za izpis vrednosti elementov tabel: izpisiTab/1. Pri prvi je tako argument tabela int-ov, pri drugi tabela char-ov.

Preskusite še : System.out.println( java.util.Arrays.toString(tab) );

**Naloga 5**

(linearno oz. zaporedno iskanje) Spišite razredno metodo aliJeVTabeli/2 oz. static boolean aliJeVTabeli(int[] t, int iskana); . Metoda vrne vrednost true, če se vrednost argumenta iskana nahaja v tabeli, sicer vrne vrednost false; . Dodajte tudi metodo za enako iskanje po tabeli znakov. Metodi naj bosta del razreda iz Naloga 1.

**Naloga 6**

Poigrajte se z metod-o/ama iz Naloga 5; naredite kopijo metod v aliJeVTabeliTest/2. V kopijah dodajte števec, ki bo štel število primerjav pri iskanju ( izvedb if stavkov) in število izpisal pred izhodom iz metode (pred stavkom return). Nato naredite tabelo 12-tih elementov in jo izpišite na zaslon. In:

1. Izvedite iskanje elementa, ki se nahaja na prvem mestu v tabeli. Zapišite število primerjav.
2. Izvedite iskanje elementa, ki se nahaja na zadnjem metu v tabeli. Zapišite število primerjav.
3. Izvedite iskanje elementa, ki se nahaja na sredini tabele(recimo na mestu 6). Zapišite število primerjav.
4. Spišite test (metodo?), ki bo 1000x izvedla iskanje naključnega elementa iz tabele in izračunajte povprečno število primerjav. Zapišite to število.

Vprašanja, na katera morate znati odgovoriti so:

Kateri element najdalj iščemo v zaporedju (najdalj je: za katerega 'porabimo' največ primerjav(if-ov)? Katerega iščemo najmanj časa? Kolikšno je povprečje pri iskanju ? Kako je število primerjav odvisno od števila elementov v zaporedju ? logaritemsko, linearno, kvadratično, exponencialno

**Naloga 7**

Predelajte mehanizma metod iz Naloga5 v metodi kjeVTabeliJe/2, kjer pa ti nalogi vrneta poziciji iskanih elementov v tabeli. Metodi naj vrneta: -1 v primeru, da številskega elementa ni v tabeli, znak na poziciji 0 v kodni tabeli, v primeru iskanja znakovnega elementa.

**Naloga 8**

Metodi razreda iz Naloga 1 z imenom skrajsajTabeloNa/2, skrajša z argumentom podano tabelo na dolžino, podano z vrednostjo drugega argumenta in skrajšano vrne. Predpostavite, da v tabeli pri krajšanju zavržemo konec v tabelo zapisanega zaporedja (ohranimo prve elemente tabele, tiste na nižjih indeksih).

Metodi zanemariPrviZadnji/1 vrneta tabelo brez prvega in zadnjega elementa z argumentom podane tabele.

**Naloga 9**

Metoda razreda tableLeftTrim/2 naj bo kopija skrajševanja z 'leve' strani tabele. Dodajte razredu metodi tableRightTrim/2, ki bosta počeli enako, le z 'desnega' konca tabele, ter metodi tableMidTrim/3, ki bo ohranila zgolj n-elementov od k-tega elementa v tabeli dalje

**Naloga 10**

Spišite metode in jih dodajte v razred MojeMetodeZaTabele:

* Vrne tabelo lihih elementov podane tabele (lihi so tisti, ki so številsko lihi ali znaki na lihih mestih v kodni tabeli znakov.
* Vrne tabelo sodih elementov tabele.
* Iz znakovne tabele vrne tabelo brez samoglasnikov.

**Naloga 11**

Metoda razreda statZnakov/1 na podani tabeli izvede statistiko pojavitev posameznih črk v tabeli. Statistiko izmed vseh znakov zapisanih v tabeli izvajate zgolj za črkovne znake, pri čemer 'a' in 'A' predstavljata isto črko. Rezultat izvedbe naj bo tabela, ki za vsako zahtevano črko vsebuje število njenih pojavitev. Metoda naj to metodo vrne klicočemu delu programa. Lahko uporabite poenostavitev in uporabite zgolj angleško abecedo.

/uporabite (direktno) razprševanje : črk je v ang. Abecedi 26, torej potrebujete tabelo števil te velikosti; A je prva črka in njene pojavitve zapisujemo na 1. el tabele (indeks 0), B druga itd. .. A in a preslikamo v isti element (o)… torej je logika : if(crka=='A' || or crka=='a') tabelaStevil[ 0 ]++;

Rešitev:

**import** java.util.\*;

public class MojeMetodeZaTabele{

public static void main(String[] args){

int[] ti = {12,3,5,9,-2,4,3,6,9,2,-2,-14,2};

char[] ti2 = {'a', 'b','c','g','f','a','a'};

System.out.println(Arrays.toString(vrniLiheElemente(ti)));

System.out.println(Arrays.toString(vrniSodeElemente(ti)));

System.out.println(Arrays.toString(prestejCrke(ti2)));

}

public static int vrniPrvega(int[] tab){

**return** tab[0];

}

public static int vrniZadnjega(int[] tab){

**return** tab[tab.length-1];

}

public static char vrniPrvega(char[] tab){

**return** tab[0];

}

public static char vrniZadnjega(char[] tab){

**return** tab[tab.length-1];

}

public static int vrniVsoto(int[] tab){

int vsota =0;

**for** (int i =0; i<tab.length-1; i++)vsota+=tab[i];

**return** vsota;

}

public static int vrniVsoto(int[] tab, int stevilo){

int vsota =0;

**if** (stevilo>0){

stevilo=(stevilo>tab.length-1)**?**tab.length-1**:**stevilo;

**for** (int i = 0; i<stevilo; i++) vsota+=tab[i];

}

**else** **if**(stevilo<0){

stevilo=(stevilo<-tab.length+1)**?**-tab.length+1**:**stevilo;

**for** (int i = tab.length-1; i>stevilo;i--)vsota+=tab[i];

}

**return** vsota;

}

public static int vrniVsoto(int[] tab, int stevilo, int zacetek){

int vsota =0;

**try**{

**if** (zacetek>tab.length);

**else**{

**if** (stevilo>0){

stevilo=(stevilo>tab.length-1)**?**tab.length-1**:**stevilo;

**for** (int i = zacetek; i<stevilo; i++) vsota+=tab[i];

}

**else** **if**(stevilo<0){

stevilo=(stevilo<-tab.length+1)**?**-tab.length+1**:**stevilo;

**for** (int i = zacetek; i>stevilo;i--)vsota+=tab[i];

}

}

}**catch**(ArrayIndexOutOfBoundsException exception){}

**return** vsota;

}

public static int[] getIntTab(int dolzina){

int[] tab = **new** int[dolzina];

**for** (int i =0; i<dolzina; i++)tab[i]=(int)(Math.random()\*Integer.MAX\_VALUE);

**return** tab;

}

public static char[] getCharTab(int dolzina){

char[] tab = **new** char[dolzina];

char[] abeceda = {'a','b','c','č','d','e','f','g','h','i','j','k','l','m','n','o','p','r','s','š','t','u','v','z','ž','A','B','C','Č','D','E','F','G','H','I','J','K','L','M','N','O','P','R','S','Š','T','U','V','Z','Ž'};

**for** (int i=0;i<dolzina;i++) tab[i]=abeceda[(int)(Math.random()\*abeceda.length-1)];

**return** tab;

}

public static boolean aliJeVTabeli(int[] tab, int iskano){

boolean a = **false**;

**for** (int i = 0; i<tab.length-1;i++)

**if** (tab[i] == iskano){

a=**true**;

**break**;

}

**return** a;

}

public static boolean aliJeVTabeli(char[] tab, char iskano){

boolean a = **false**;

**for** (int i = 0; i<tab.length-1;i++)

**if** (tab[i] == iskano){

a=**true**;

**break**;

}

**return** a;

}

public static boolean aliJeVTabeliTest(int[] tab, int iskano){

boolean a = **false**;

int b =0;

**for** (int i = 0; i<tab.length-1;i++){

**if** (tab[i] == iskano){

a=**true**;

**break**;

}

b++;

}

**return** a;

}

public static boolean aliJeVTabeliTest(char[] tab, char iskano){

boolean a = **false**;

int b=0;

**for** (int i = 0; i<tab.length-1;i++){

**if** (tab[i] == iskano){

a=**true**;

**break**;

}

b++;

}

System.out.println("Primerjanih je bilo: "+b);

**return** a;

}

public static boolean aliJeVTabeliTestRand(char[] tab, char iskano){

boolean a = **false**;

int b=0;

**for** (int i = 0; i<1000;i++){

int rand = (int)(Math.random()\*tab.length-1);

**if** (tab[rand] == iskano){

a=**true**;

**break**;

}

b++;

}

System.out.println("Primerjanih je bilo: "+b);

**return** a;

}

public static boolean aliJeVTabeliTestRand(int[] tab, int iskano){

boolean a = **false**;

int b=0;

**for** (int i = 0; i<1000;i++){

int rand = (int)(Math.random()\*tab.length-1);

**if** (tab[rand] == iskano){

a=**true**;

**break**;

}

b++;

}

System.out.println("Primerjanih je bilo: "+b);

**return** a;

}

public static int kjeVTabeliJe(char[] tab, char iskano){

int b=0,c=-1;

**for** (int i = 0; i<tab.length-1;i++){

**if** (tab[i] == iskano){

c=i;

**break**;

}

b++;

}

**return** tab[0];

}

public static int kjeVTabeliJe(int[] tab, int iskano){

int b=0,c=-1;

**for** (int i = 0; i<1000;i++){

int rand = (int)(Math.random()\*tab.length-1);

**if** (tab[rand] == iskano){

c=i;

**break**;

}

b++;

}

System.out.println("Primerjanih je bilo: "+b);

**return** c;

}

public static int[] skrajsajTabeloNa(int[] tab, int dolzina){

int[] n\_tab = **new** int[dolzina];

**for** (int i =0; i<dolzina-1; i++){

n\_tab[i]=tab[i];

}

**return** n\_tab;

}

public static char[] skrajsajTabeloNa(char[] tab, char dolzina){

char[] n\_tab = **new** char[dolzina];

**for** (int i =0; i<dolzina-1; i++){

n\_tab[i]=tab[i];

}

**return** n\_tab;

}

public static char[] zanemariPrviZadnji(char[] tab){

int x=0;

char[] n\_tab = **new** char[tab.length-2];

**for** (int i =1; i<n\_tab.length; i++){

n\_tab[x]=tab[i];

x++;

}

**return** n\_tab;

}

public static int[] zanemariPrviZadnji(int[] tab){

int[] n\_tab = **new** int[tab.length-2];

int x=0;

**for** (int i =1; i<n\_tab.length; i++){

n\_tab[x]=tab[i];

x++;

}

**return** n\_tab;

}

public static int[] tableLeftTrim(int[] tab, int dolzina){

int[] n\_tab = **new** int[dolzina];

int x= tab.length-dolzina;

**for** (int i =0; i<n\_tab.length; i++){

n\_tab[i] = tab[x];

x++;

}

**return** n\_tab;

}

public static char[] tableLeftTrim(char[] tab, int dolzina){

char[] n\_tab = **new** char[dolzina];

int x= tab.length-dolzina;

**for** (int i =0; i<n\_tab.length; i++){

n\_tab[i] = tab[x];

x++;

}

**return** n\_tab;

}

public static int[] tableRightTrim(int[] tab, int dolzina){

int[] n\_tab = **new** int[dolzina];

**for** (int i= 0; i<dolzina; i++) n\_tab[i] = tab[i];

**return** n\_tab;

}

public static char[] tableRightTrim(char[] tab, int dolzina){

char[] n\_tab = **new** char[dolzina];

**for** (int i= 0; i<dolzina; i++) n\_tab[i] = tab[i];

**return** n\_tab;

}

public static int[] tableMidTrim(int[] tab, int dolzina, int k){

int[] n\_tab=**new** int[dolzina];

int a=k;

**for** (int i=0; i<dolzina; i++){

n\_tab[i] = tab[k];

k++;

}

**return** n\_tab;

}

public static char[] tableMidTrim(char[] tab, int dolzina, int k){

char[] n\_tab=**new** char[dolzina];

int a=k;

**for** (int i=0; i<dolzina; i++){

n\_tab[i] = tab[k];

k++;

}

**return** n\_tab;

}

public static int[] vrniLiheElemente(int[] tab){

int dolzina=0, a=0;

**for** (int i=0;i<tab.length; i++){

**if** (i%2==1 || tab[i]%2==1) dolzina++;

}

int[] n\_tab = **new** int[dolzina];

int x=0;

**for** (int i=0;i<tab.length; i++){

**if** (i%2==1 || tab[i]%2==1){

n\_tab[x] = tab[i];

x++;

}

}

**return** n\_tab;

}

public static int[] vrniSodeElemente(int[] tab){

int dolzina=0, a=0;

**for** (int i=0;i<tab.length; i++){

**if** (i%2==0 || tab[i]%2==0) dolzina++;

}

int[] n\_tab = **new** int[dolzina];

int x=0;

**for** (int i=0;i<tab.length; i++){

**if** (i%2==0 || tab[i]%2==0){

n\_tab[x] = tab[i];

x++;

}

}

**return** n\_tab;

}

public static char[] vrniLiheElemente(char[] tab){

int dolzina = (int)tab.length/2;

**if** (tab.length%2 == 1) dolzina++;

char[] n\_tab = **new** char[dolzina];

int j=0;

**for** (int i =0; i<tab.length; i+=2){

n\_tab[j] = tab[i];

j++;

}

**return** n\_tab;

}

public static char[] vrniBrezSamoglasnikov(char[] tab) {

char[] samoglasniki = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'};

int a = tab.length;

**for** (int i = 0; i < tab.length; i++) {

**for** (int j = 0; j < samoglasniki.length; j++) {

**if** (tab[i] == samoglasniki[j]) {

a--;

**break**;

}

}

}

char[] n\_tab = **new** char[a];

int k = 0;

**for** (int i = 0; i < tab.length; i++) {

boolean jeSamoglasnik = **false**;

**for** (int j = 0; j < samoglasniki.length; j++) {

**if** (tab[i] == samoglasniki[j]) {

jeSamoglasnik = **true**;

**break**;

}

}

**if** (!jeSamoglasnik) {

n\_tab[k] = tab[i];

k++;

}

}

**return** n\_tab;

}

public static int[] prestejCrke(char[] tab){

char[] crke = {

'a', 'A', 'b', 'B', 'c', 'C', 'd', 'D', 'e', 'E',

'f', 'F', 'g', 'G', 'h', 'H', 'i', 'I', 'j', 'J',

'k', 'K', 'l', 'L', 'm', 'M', 'n', 'N', 'o', 'O',

'p', 'P', 'q', 'Q', 'r', 'R', 's', 'S', 't', 'T',

'u', 'U', 'v', 'V', 'w', 'W', 'x', 'X', 'y', 'Y',

'z', 'Z'

};

int[] n\_tab = **new** int[crke.length];

**for** (int i=0; i<tab.length; i++){

**for** (int j=0; j<crke.length; j+=2){

**if**(tab[i] == crke[j]){

n\_tab[i]+=1;

**break**;

}

**else** **if** (tab[i] == crke[j+1]){

n\_tab[i+1]+=1;

**break**;

}

}

}

**return** n\_tab;

}

}