Pole (Array) v Jave

Statické a dynamické polia

Abstrakt

Cieľom práce je podrobne oboznámiť čitateľa s dátovou štruktúrou pole. Práca sa v úvode zameriava na jednu z najzákladnejších dátových štruktúr v jazyku Java, tzv. statické pole. Vysvetľuje nielen pole ako pojem, ale zameriava sa aj na samotnú syntax tohto jazyka pre prácu so statickými poľami. Ďalej sa venuje tzv. viacrozmerným poliam. Uvádzajú sa rozdiely voči konvenčným poliam. Záver tvorí porovnanie dvoch dátových štruktúr, a to statického a dynamického poľa.

Definícia

Pole ako dátová štruktúra je v skutočnosti kontajner objektov, ktorý umožňuje zjednotiť v jeden celok množstvo premenných. Tie sú radené postupne za sebou. Každú takúto premennú nazývame prvok poľa, pričom všetky prvky daného poľa musia byť rovnakého vopred určeného typu. Typom môže byť ako základná dátová štruktúra tak aj vlastná dátová štruktúra (objekt), dokonca aj pole. V poslednom prípade sa z poľa stáva viacrozmerné pole, čomu sa budeme venovať neskôr.

Statické pole sa vyznačuje vopred alokovaným miestom v pamäti. Preto majú statické polia vopred určenú a pevnú veľkosť, tj. počet prvkov poľa.

Medzi základné funkčnosti poľa v Jave patria metódy pre manipuláciu s prvkami a získavanie ich hodnoty, radenie, prehľadávanie a taktiež kopírovanie.

Deklarácia a inicializácia

```
object[] pole = new object[dlzka];                    (1)
```

Táto deklarácia(1) vytvorí premennú *pole*, ktorá odkazuje na dané pole. Typ poľa je definovaný pred premennou, podobne ako u iných dátových štruktúr, pričom slovom *objekt* je myslený typ premenných, ktoré budú ukladané do poľa. Za premennou nasleduje kľúčové slovo *new*. To vytvorí nový objekt typu pole, ktorý je priradený hodnotou do premennej. Podobne ako pri deklarácii aj pri inicializácii musí byť typ poľa definovaný, pričom naviac v hranatých zátvorkách je celé kladné číslo určujúce počet prvkov uložených v poli. Touto inicializáciou sa alokuje miesto v pamäti. Toto miesto je vopred presne definovateľné, keďže počet prvkov a objektov, ktoré budú v poli uložené, je známy.

```
object[] pole = {hodnota, ..., hodnota};             (2)
```

Prístupovanie

```
pole[index] = hodnota;                                (3)
```

V tomto prípade sa prvku poľa priradenému premennej *pole* s pozíciou *index* priradí hodnota.

```
... .println(pole[index]);                            (4)
```

Pri získaní prvku z poľa môžeme používať tento prvok ako regulárny odkaz na objekt. Tým pádom môžeme volať priamo metódy daného objektu, resp. upravovať alebo získavať jeho vlastnosti.

```
pole_objektov[index].getClass();                      (5)
```

Viacrozmerné polia

Viacrozmerné polia môžeme chápať taktiež ako pole polí. Tieto polia sú jedinečné tým, že ich prvky neobsahujú konvenčné objekty, ale polia. Pritom musí byť dodržaná vlastnosť poľa, že všetky prvky musia byť rovnakého typu, z čoho logicky vyplýva, že taktiež všetky podpolia musia prijímať prvky toho istého typu.

```
object[][] pole = new object[dlzka][sirka];           (6)
```

```
object[][] pole = {{hodnota, ..., hodnota}, ..., {hodnota, ..., hodnota}};   (7)
```

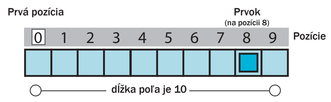
Tento zápis je zaujímavý tým, že nám umožňuje vytvoriť viacrozmerné pole, ktorého podpolia nemajú rovnakú dĺžku, čo nie je zvyčajná vlastnosť programovacích jazykov.

Statické vs. dynamické pole

Doteraz spomínané polia boli tzv. statické polia. Druhým typom polí sú tzv. dynamické polia, ktoré sa od statických líšia viacerými vlastnosťami.

Zatiaľ čo statické polia majú vopred alokované miesto v pamäti, čím je ich veľkosť v priebehu programu nemenná, dynamické polia môžu svoju veľkosť priebežne meniť. Táto zmena veľkosti je však pomerne časovo náročná, čo nemusí byť pri niektorých riešeniach vhodné.

Dynamické polia nepatria medzi základné dátové štruktúry. Sú implementované pomocou viacerých štruktúr. Najpoužívanejšia štruktúra je ArrayList, pričom sa často používa preddefinovaný interface List.



Deklarácia

```
public class Stack {
    private int[] pole;
    private int index = -1;//prazdny
    private int dlzka = 0;

    public Stack() {
        this.pole = new int[100];
        this.dlzka = 100;
    }

    public Stack(int dlzka) {
        this.pole = new int[dlzka];
        this.dlzka = dlzka;
    }

    public boolean isFull(){
        if(index >= dlzka-1){
            return true;
        } else {
            return false;
        }
    }

    public boolean isEmpty(){
        if(index == -1){
            return true;
        } else {
            return false;
        }
    }

    public void push(int x){
        if(isFull()){
            for(int a = 1; a < dlzka; a++){
                this.pole[a-1] = this.pole[a];
            }
            this.pole[index] = x;
        } else {
            this.index++;
            this.pole[this.index] = x;
        }
    }

    public int pop() throws Exception{
        if(!isEmpty()){
            index--;
            return this.pole[index+1];
        } else {
            throw new Exception("Chyba!\nMetoda: pop()\nZasobnik je prazdny!");
        }
    }

    public int getTopValue() throws Exception{
        if(!isEmpty()){
            return this.pole[index];
        } else {
            throw new Exception("Chyba!\nMetoda: getTopValue()\nZasobnik je prazdny!");
        }
    }

    @Override
    public String toString(){
        if(!isEmpty()){
            String str = "";
            for(int a = 0; a <= index; a++){
                str += this.pole[a] + " ";
            }
            return str;
        } else {
            return "Zasobnik je prazdny!";
        }
    }
}
```

Inicializácia

```
public Stack(int dlzka) {
    this.pole = new int[dlzka];
    this.dlzka = dlzka;
}
```

Inicializácia

```
public void push(int x){
    if(isFull()){
        for(int a = 1; a < dlzka; a++){
            this.pole[a-1] = this.pole[a];
        }
        this.pole[index] = x;
    } else {
        this.index++;
        this.pole[this.index] = x;
    }
}
```

Viacrozmerné pole

```
public static void rotate(final int[][] pole, int kolko){
    int[] pole_pomocne;
    int velkost = 0;

    for(int a = 0; a<pole.length;a++){
        velkost += pole[a].length;
    }

    pole_pomocne = new int[velkost];

    velkost = 0;
    for(int a = 0; a<pole.length;a++){
        for(int b = 0; b<pole[a].length; b++){
            pole_pomocne[velkost] = pole[a][b];
            velkost++;
        }
    }

    velkost = 0;
    for(int a = 0; a<pole.length;a++){
        for(int b = 0; b<pole[a].length; b++){
            int index = velkost - kolko;

            if(index >= 0 && index < pole_pomocne.length){
                pole[a][b] = pole_pomocne[index];
            } else if(index < 0){
                index += pole_pomocne.length;
                pole[a][b] = pole_pomocne[index];
            } else {
                index = index-pole_pomocne.length;
                pole[a][b] = pole_pomocne[index];
            }
            velkost++;
        }
    }
}

public static void vypis(int[][] pole) {
    for (int i = 0; i < pole.length; i++) {
        for (int j = 0; j < pole[i].length; j++)
            System.out.printf("%02d  ", pole[i][j]);
        System.out.println();
    }
}

// TODO: metody isDiagonal, detDiagonal, rotate

public static void main(String[] args) {
// Testovani ukolu A, B
    double[][] diag1 = {{1, 0, 0}, {0, 2, 0}, {0, 0, 7}};
    double[][] diag2 = {{1, 0, 0}, {0, 0, 0}, {0, 0, 7}, {0, 0, 0}};
    double[][] diag3 = {{0.5, 0, 0, 0}, {0, -5, 0, 0}, {0, 0, 7, 0}};

    double[][] notDiag1 = {{1, 0}, {0, 2, 0}, {0, 0, 7}};

    ...
    ...
```


Adam Netocny
Letný semester 2008/2009
Fel CVUT
netocada@fel.cvut.cz