Politechnika Warszawska
Wydział Elektryczny
Algorytmy i struktury danych

Zadanie nr 4

Wykonała:

Alicja Puacz, nr albumu 235434

Informatyka, gr. 2

rok akademicki 2015/2016

Opis zadania

Celem zadania było zastosowanie struktury B-drzewa oraz Wektora na nim opartego i stworzenie struktury do przechowania macierzy rzadkich oraz wykonywania na nich operacji dodawania i mnożenia.

W tym celu do poprzedniego projektu utworzono klasę Macierz. Klasa ta zawiera elementy: „wierszy" i „kolumn", które definiują wymiar macierzy oraz tablicę wskaźników na klasę Wektor. Oznacza to, że macierz jest tablicą Wektorów, gdzie indeks tablicy jest numerem wiersza macierzy i przechowuje on indeks kolumny macierzy. Indeks wiersza wskazuje na Wektor, zaś kolumny na element w tym Wektorze (wartość). Czyli Wektory przechowują pary numer kolumny-wartość macierzy.

Klasa Macierz posiada również metody:

- pobierz(int wiersz, int kolumna) – pobierająca ze struktury element o podanym położeniu;
- wstaw(int wiersz, int kolumna, double wartosc) – zapisująca daną liczbę do struktury na dane położenie;
- czytajPlik(File filename) – odczytującą strukturę macierzy z pliku i zapisującą dane do macierzy. Struktura pliku była następująca:

X Y
W K Liczba
…

Gdzie X- to ilość wierszy w macierzy, Y – ilość kolumn, następnie podawane były elementy niezerowe macierzy: w- nr wiersza, w którym występuje dany element, k- nr kolumny oraz Liczba – wartość;

- dodajMacierze(Macierz a, Macierz b) – następuje wtedy dodawanie macierzy reprezentującej pierwszy argument do macierzy reprezentującej drugi argument. Funkcja zwraca nową macierz – wynik. Algorytm dodawania macierzy jest dość

łatwy, następuje tu dodawanie elementów macierzy na tych samych pozycjach i zapisanie sumy w nowej macierzy na takiej samej pozycji;
- mnozMacierze(Macierz a, Macierz b) – mnożenie macierzy jedna przez drugą jest już nieco bardziej skomplikowane. Wykorzystano w tym celu wektor pomocniczy, który jest sumą iloczynów odpowiadających sobie elementów w wierszu pierwszej macierzy i kolumnie drugiej macierzy. Wprowadzony jest tu trzeci pomocniczy indeks „n", dzięki któremu wykonywana jest pętla „for". „n" jest mniejsze od ilości kolumn pierwszej macierzy, czyli także od ilości wierszy drugiej macierzy.
Klasa Mcierz zawiera również dwie inne klasy – wyjątki:
- ElementPozaZakresemException – gdy zostanie odczytany element o indeksach większych niż zdefiniowane w pliku jako wymiary macierzy;
- WadliwyRozmiarException – gdy przy dodawaniu lub mnożeniu wymiary macierzy nie będą sobie odpowiadać.
Funkcja Testująca zawiera oprócz funkcji main klasę drukujMacierz(Macierz f) wypisującą na standardowe wyjście strukturę macierzy. Funkcja main wczytuje dwa pliki podane jako argumenty oraz znak – który definiuje operację do wykonania. W wyniku jej działania otrzymujemy wydrukowane struktury macierzy oraz macierzy wynikowej.

Kody programu

Macierz.java

```
import java.io.*;


/**
 * Klasa Macierz.Tworzy macierz i wykonuje na nich operacje.
 * @author Alicja Puacz
 */
public class Macierz {
   int wierszy; //ile wierszy
   int kolumn; //ile kolumn
   //przechowuje indeksy
   Wektor[] macierz;


   public Macierz(){
   }

   public Macierz(int w, int k){
      this.wierszy = w;
      this.kolumn = k;
      macierz = new Wektor[kolumn];
      int i;
      for (i=0; i<wierszy; i++){
        macierz[i] = new Wektor();
```

```
    }
}


/**
* Zwraca element z macierzy o wskazanej pozycji.
* @param wiersz numer wiersza szukanego elementu macierzy
* @param kolumna numer kolumny szukanego elementu macierzy
*/
double pobierz(int wiersz, int kolumna) throws ElementPozaZakresemException{
    if(wiersz>this.wierszy || kolumna>this.kolumn )
        throw new ElementPozaZakresemException("Element ma złe nr indeksów");

    if (this.macierz[wiersz].wek[kolumna]!= null)
        return this.macierz[wiersz].wezZWek(kolumna);
    else
        return 0;
}


/**
* Zapisuje liczbę do macierzy na wskazaną pozycję.
* @param wiersz numer wiersza szukanego elementu macierzy
* @param kolumna numer kolumny szukanego elementu macierzy
* @param wartosc liczba, którą zapisujemy
*/
void wstaw(int wiersz, int kolumna, double wartosc){
    this.macierz[wiersz].wstawDoWek(kolumna, wartosc);
}


/**
* Odczytuje plik, a w nim: rozmiar macierzy i jej niezerowe elementy
* i wstawia je do macierzy
* @param filename ścieżka do pliku, który zawiera wczytywaną strukturę
* macierzy
*/
void czytajPlik(File filename) throws ElementPozaZakresemException{
   try{
        //otworz plik
        BufferedReader reader = new BufferedReader(new FileReader(filename));
        String line= reader.readLine();
        String[] lineL = line.split("\\s+");
        //Ustaw rozmiar macierzy
        this.wierszy = Integer.parseInt(lineL[0]);
        this.kolumn = Integer.parseInt(lineL[1]);
        this.macierz = new Wektor[this.wierszy];
        //Inicjalizacja Wektorów wierszy naszej macierzy
        for (int l=0; l<this.wierszy; l++){
            this.macierz[l] = new Wektor();
```

```
        }
        while ((line= reader.readLine()) != null){
           //Jeśli pusta linia - oznacza koniec pliku, wyjdź z pętli.
           if (line == null)
              break;
           //Rozbijamy wyrazy w linii i przypisujemy do tablicy
           String[] lineS = line.split("\\s+");
           int i = Integer.parseInt(lineS[0]);
           int j = Integer.parseInt(lineS[1]);
           //Sprawdzamy poprawność pliku z
           if(i>this.wierszy || j>this.kolumn)
              throw new ElementPozaZakresemException(
                    "Element ma złe nr indeksów");
           Double h = Double.parseDouble(lineS[2]);
           wstaw(i, j, h) ;
           //Nie dodawaj do wektora elementów o indeksach mniejszych od 0
           // i większych od 100000.
           if (i<0 || i>100000 || j<0 || j>100000){
              System.err.format("Nie wczytano v[" + i + "]=" + h +
                    ", błędny indeks\n");
              continue;
           }
        }
        reader.close();
     //Wyłapuje błędy przy wczytywaniu pliku z danymi.
     }catch (IOException e){
        System.err.format("Blad przy wczytywaniu pliku\n");
        e.printStackTrace();
     }
  }

  /**
  * Dodaje do siebie dwie macierze.
  * @param a pierwsza macierz
  * @param b druga macierz
  */
  void dodajMacierze(Macierz a, Macierz b)throws WadliwyRozmiarException{
     //Gdy wymiary macierzy są niezgodne - zgłaszamy wyjątek
     if(a.kolumn != b.kolumn || a.wierszy != b.wierszy)
        throw new WadliwyRozmiarException("Wymiary macierzy są różne");
     //Ustawiamy rozmiar macierzy wynikowej: wybieramy maksymalny rozmiar
     //kolumny i wiersza
     this.wierszy = a.wierszy>b.wierszy ? a.wierszy : b.wierszy;
     this.kolumn = a.kolumn>b.kolumn ? a.kolumn : b.kolumn;
     this.macierz = new Wektor[this.wierszy];
     for (int l=0; l<this.wierszy; l++){
        this.macierz[l] = new Wektor();
     }
     for(int i=0; i<this.wierszy; i++){
        for(int j=0; j<this.kolumn; j++){
```

```
            //Klasa Wektor przechowuje tylko elementy niezerowe - jeśli taki
            //napotkamy, przypiszmy mu "0"
            if(a.macierz[i].wek[j] == null)
                a.macierz[i].wek[j]=0.0;
            if(b.macierz[i].wek[j] == null)
                b.macierz[i].wek[j]=0.0;
            this.macierz[i].wstawDoWek(j, a.macierz[i].wek[j] + b.macierz[i].wek[j]);
        }
    }
}

/**
* Mnoży przez siebie dwie macierze.
* @param a pierwsza macierz
* @param b druga macierz
*/
void mnozMacierze(Macierz a, Macierz b) throws WadliwyRozmiarException{
    //Gdy wymiary macierzy są niezgodne - zgłaszamy wyjątek
    if(a.kolumn != b.wierszy)
        throw new WadliwyRozmiarException("Liczba kolumn pierwszej macierzy "
                + "nie jest równa liczbie wierszy drugiej macierzy");
    //Ustawiamy rozmiar macierzy wynikowej
    this.wierszy = a.wierszy;
    this.kolumn = b.kolumn;
    this.macierz = new Wektor[this.wierszy];
    for (int l=0; l<this.wierszy; l++){
        this.macierz[l] = new Wektor();
    }

    int p = a.kolumn;
    for(int i=0; i<this.wierszy; i++){
        for(int j=0; j<this.kolumn; j++){
            //Wektor pomocniczy
            double temp = 0.0;
            for(int n=0; n<p; n++){
                //Klasa Wektor przechowuje tylko elementy niezerowe - jeśli
                //taki napotkamy, przypiszmy mu "0", nawet jeśli liczba
                //wierszy w pierwszej macierzy nie jest równa liczbie kolumn
                //w drugiej
                if(a.macierz[i].wek[n] == null)
                    a.macierz[i].wek[n]=0.0;
                if(b.macierz[n].wek[j] == null)
                    b.macierz[n].wek[j]=0.0;
                //Sumujemy odpowiednie iloczyny elementów
                temp += a.macierz[i].wek[n] * b.macierz[n].wek[j];
            }
            //Klasa Wektor przechowuje tylko elementy niezerowe
            if(temp==0)
                continue;
            this.macierz[i].wstawDoWek(j, temp);
```

```
        }
      }
   }


   /**
   * Wyjątek dla przypadku, gdy indeks elementu przekracza rozmiar macierzy
   */
   class ElementPozaZakresemException extends Exception{
      public ElementPozaZakresemException(String message) {
         super(message);
      }
   }


   /**
   * Wyjątek dla przypadku, gdy przy mnożeniu macierzy ich rozmiary są różne
   */
   class WadliwyRozmiarException extends Exception{
      public WadliwyRozmiarException(String message) {
         super(message);
      }
   }
}
```

Test.java

```
import java.io.File;
/**
 * Funkcja testująca klasę Macierz.
 * @author Alicja Puacz
 */
public class Test {


   /**
   * Funkcja main, wymagane są 3 argumenty
   * @param f macierz, któej strunturę drukujemy
   */
   public static void drukujMacierz(Macierz f) throws
         Macierz.ElementPozaZakresemException{
      for (int i=0; i<f.wierszy; i++){
         for (int j=0; j<f.kolumn; j++){
               System.out.print("\t" + f.pobierz(i, j) + "\t");
         }
         System.out.print("\n");
      }
   }
```

```
   /**
   * Funkcja main, wymagane są 3 argumenty
   */
   public static void main(String args[]) throws
         Macierz.ElementPozaZakresemException, Macierz.WadliwyRozmiarException{
      //Gdy liczba argumentów się nie zgadza, kończymy program
      if (args.length != 3){
         System.out.println("\nZła liczba argumentów, powinno być 3\n");
         return;
      }
      //Czytamy macierze z plików
      Macierz a = new Macierz();
      a.czytajPlik(new File(args[0]));
      Macierz b = new Macierz();
      b.czytajPlik(new File(args[2]));
      //Drukujemy macierze
      System.out.println("\nPierwsza macierz:");
      drukujMacierz(a);
      System.out.println("\nDruga macierz:");
      drukujMacierz(b);
      //Wykonujemy operację i drukujemy wynik
      switch (args[1]) {
         //Dodawanie
         case "+":
            {
               //dodaj i pokaz wynik
               System.out.println("\nWynik dodawania:");
               Macierz c = new Macierz();
               c.dodajMacierze(a, b);
               drukujMacierz(c);
               break;
            }
         //Mnożenie
         case "x":
            {
               //pomnoz i pokaz wynik
               System.out.println("\nWynik mnożenia:");
               Macierz c = new Macierz();
               c.mnozMacierze(a, b);
               drukujMacierz(c);
               break;
            }
         //Gdy nie rozpoznano znaku
         default:
            System.out.println("Nieznana operacja");
            break;
      }
   }
}
```

Otrzymane wyniki

plik macierzA.txt

16 5
0 0 20
1 2 20
1 4 20
5 0 20
6 4 20
7 0 20
9 4 20
12 0 20
14 4 20

plik macierzB.txt

5 5
0 0 30
0 4 30
1 0 30
1 1 30
1 2 30
3 3 30
4 4 30

plik macierzC.txt

16 5
0 0 30
0 4 30
1 0 30
1 1 30
1 2 30
3 3 30
4 4 30

Wyniki:
macierzA.txt * macierzB.txt

```
Pierwsza macierz:
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             20.0            0.0             20.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        0.0             0.0             0.0             0.0             0.0

Druga macierz:
        30.0            0.0             0.0             0.0             30.0
        30.0            30.0            30.0            0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             30.0            0.0
        0.0             0.0             0.0             0.0             30.0

Wynik mnożenia:
        600.0           0.0             0.0             0.0             600.0
        0.0             0.0             0.0             0.0             600.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        600.0           0.0             0.0             0.0             600.0
        0.0             0.0             0.0             0.0             600.0
        600.0           0.0             0.0             0.0             600.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             600.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        600.0           0.0             0.0             0.0             600.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             600.0
        0.0             0.0             0.0             0.0             0.0
```

macierzA.txt + macierzB.txt

```
Pierwsza macierz:
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             20.0            0.0             20.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        0.0             0.0             0.0             0.0             0.0
Exception in thread "main" cw4.Macierz$WadliwyRozmiarException: Wymiary macierzy są różne
        0.0             0.0             0.0             0.0             0.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        0.0             0.0             0.0             0.0             0.0

Druga macierz:
        30.0            0.0             0.0             0.0             30.0
        30.0            30.0            30.0            0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             30.0            0.0
        0.0             0.0             0.0             0.0             30.0

Wynik dodawania:
        at cw4.Macierz.dodajMacierze(Macierz.java:116)
        at cw4.Test.main(Test.java:54)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

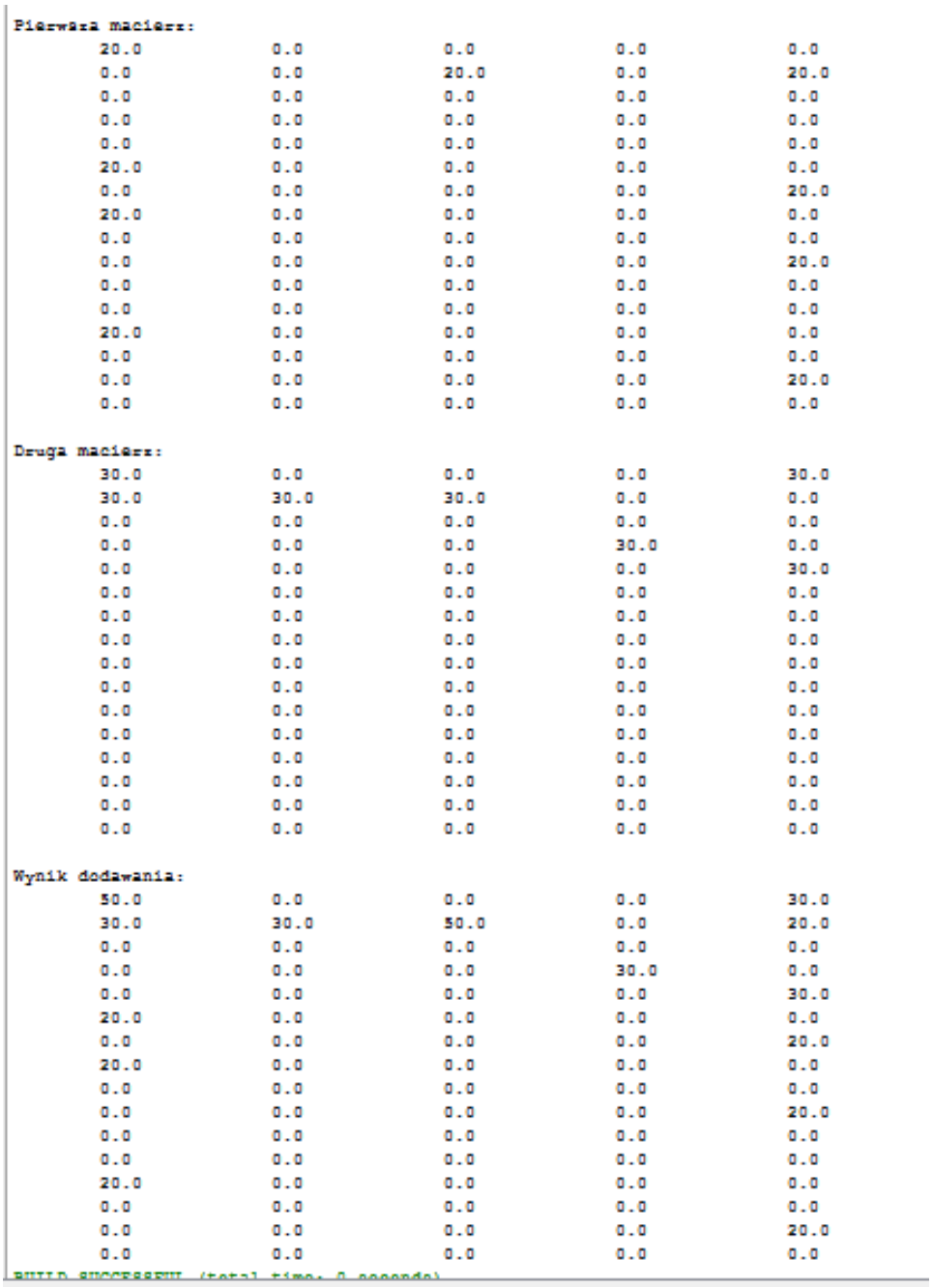
macierzA.txt x macierzC.txt

```
Pierwsza macierz:
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             20.0            0.0             20.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        0.0             0.0             0.0             0.0             0.0

Druga macierz:
Exception in thread "main" cw4.Macierz$WadliwyRozmiarException: Liczba kolumn pierwszej macierzy nie jest równa liczbie wierszy drugiej macierzy
        30.0            0.0             0.0             0.0             30.0
        30.0            30.0            30.0            0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             30.0            0.0
        0.0             0.0             0.0             0.0             30.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0

Wynik mnożenia:
        at cw4.Macierz.mnozMacierze(Macierz.java:146)
        at cw4.Test.main(Test.java:64)
Java Result: 1
```

macierzA.txt + macierzC.txt



```
Pierwsza macierz:
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             20.0            0.0             20.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        0.0             0.0             0.0             0.0             0.0

Druga macierz:
        30.0            0.0             0.0             0.0             30.0
        30.0            30.0            30.0            0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             30.0            0.0
        0.0             0.0             0.0             0.0             30.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0

Wynik dodawania:
        50.0            0.0             0.0             0.0             30.0
        30.0            30.0            50.0            0.0             20.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             30.0            0.0
        0.0             0.0             0.0             0.0             30.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        20.0            0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             0.0
        0.0             0.0             0.0             0.0             20.0
        0.0             0.0             0.0             0.0             0.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

<u>Wnioski</u>

Wymagania zadania zostały spełnione, program wykonuje działania poprawnie. Program poprawnie drukuje strukturę i wyniki macierzy.