Lekcja 12: Tablice, pętle pliki

Tablica to struktura, która przechowuje elementy tego samego typu. W C# indeksy są kontrolowane przez klasę implementującą tablicę, tutaj – tablica to nie tylko fragment pamięci z adresem do pierwszego elementu zapamiętanym we wskaźniku jak w C i C++, tylko obiekt instancja klasy *System. Array*. Dzięki temu możliwe jest kontrolowanie tego , czy indeksy nie są mniejsze od zera lub większe od rozmiaru tablicy (minus jeden).

Deklarując tablicę tworzymy zmienną referencyjną która wskazuje na obiekt tablicę, definiujemy ją czyli rezerwujemy pamięć dla jej elementów, a następnie zainicjować czyli zapełnić wartościami.

„Deklaracja tablicy”: czyli składnia referencji do tablicy z elementami typu int:

*int [] ti;*

typ int[] to typ tablicy, która przechowuje wartości typu int. A oto przykład deklaracji referencji wraz utworzeniem obiektu tablicy (rezerwowana jest pamięć na stercie):

*int[] ti = new int[3];*

Po utworzeniu obiektu tablicy (tj. w istocie instancji klasy System.Array) jest ona automatycznie inicjowana wartościami domyślnymi dla danego typu, czyli w przypadku typu int – zerami. Możemy także samodzielnie zainicjować elementy tablicy, podając je w nawiasach klamrowych:

*int[] ti =new int[3] {1,2,4};*

To polecenie można zresztą skrócić:

*int[] ti={1,2,4};*

Ponieważ kompilator potrafi z listy elementów odczytać rozmiar tablicy. Graficznie :

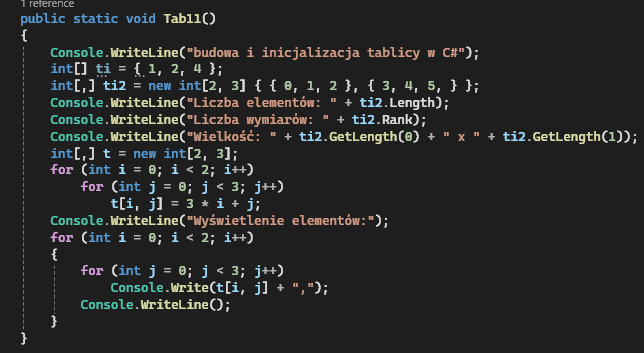
|  |  |  |  |
| --- | --- | --- | --- |
| *Indeksy* | *0* | *1* | *2* |
| *Wartości* | *1* | *2* | *4* |

Elementy są indeksowane od 0. Ostatni element to indeks *rozmiar -1*. Rozmiar tablicy można odczytać z jej właściwości Length. Rozmiaru tablicy nie można zmienić. Dostęp do tablicy jest możliwy dzięki operatorowi [] z indeksem elementu jako argumentem. Umożliwia ona odczytanie wartości tego elementu tablicy, jak i jego zmianę. Oznacza to że może stać zarówno z lewej jak i z prawej strony operatora przypisania

Tablica wielowymiarowa – obiekt klasy System.Array, którego elementy są odpowiednio poukładane, przykład:

*int[,]ti2 = new int[2,3] {{0,1,2},{3,4,5}};*

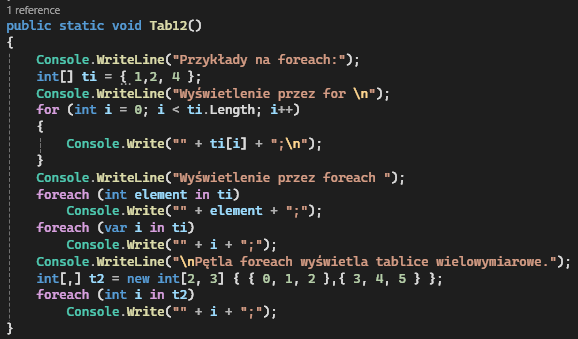
Wielkość tablicy odczytana za pomocą właściwości ti2.Length, równa jest 6, ale dostęp do elementów możliwy jedynie przez umieszczenie wewnątrz nawiasów kwadratowych dwóch indeksów. Dla przykładu ostatni element w tablicy to ti2[1,2]. Liczbę wymiarów tablicy można odczytać z własności Rank, a rozmiar w każdym wymiarze za pomocą metody GetLength z numerem wymiaru przekazanym jako argument (też numerowane od zera).:



W drugiej połowie metody mamy inicjalizację tablicy t w pętlach for zewnętrznej i wewnętrznej przy pomocy zmiennych sterujących pętlami i . j.

**Pętla Foreach**

Jeżeli zainicjowaliśmy tablice to łatwiej jest użyć do np.: wyświetlenia tych elementów pętli foreach.:



1. Mamy zmienną która w kolejnych iteracjach pętli wskazuje na kolejne elementy tablicy.
2. Do deklaracji możemy użyć słowa var (kompilator sam się domyśli jaki typ przedstawiają przechowywane wartości. (lepiej jednak stosować typ, powoduje to czytelność kodu).
3. Pętla ta nie nadaje się do inicjacji i zmiany wartości jedynie tylko do prezentacji.
4. Pętla foreach obsługuje tablice wielowymiarowe,(ostatnie linijki mojego kodu.

**Wybór elementów z tablicy**

1. Odczytanie elementów z tablicy jest możliwe dzięki operatorowi [] w środek należy wstawić indeks.

*int[] ti = {2,4,8,16,32,64,128,256,512,1024};*

*Console.WriteLine(ti[3});*

*Console.WriteLine(ti[6]};*

Zostaną wyświetlone 16,128 czyli czwarty i siódmy (pierwszy element ma indeks0).

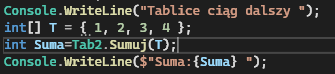
1. Zmiana wartości czyli:

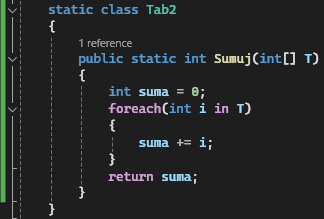
ti[2] = 9;

Tablica jako argument metody (tablice jako argumenty metod, jak i zwracane przez ich wartości.)

**Exercise 1.**

Napisać funkcję, która będzie przyjmować tablicę liczb całkowitych, a zwracać sumę jej elementów.

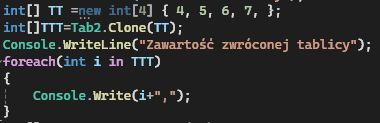
Wywołanie metody: 

Metoda: 

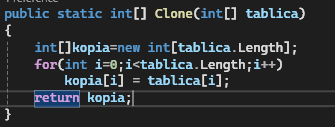
**Exercise 2.**

Napisać metodę która pobiera tablicę ale również ją zwraca. Zadaniem metody jest sklonowanie tablicy, czyli stworzenie nowej , tego samego typu o tej samej długości a następnie skopiowanie do niej wszystkich elementów oryginalnej tablicy.

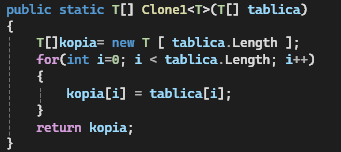
Wywołanie metody:



Metoda:

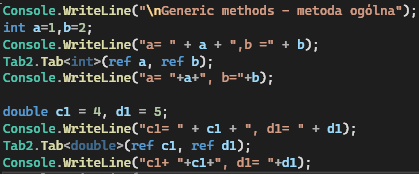


Metodę powyższą możemy zmienić w metodę ogólną. Dzięki temu metoda ta będzie mogła kopiować tablicę z elementami dowolnego typu.

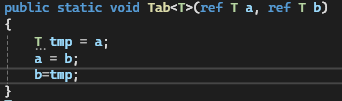


|  |
| --- |
| **Generic methods** czyli metoda ogólna w nagłówku po nazwie metody a przed argumentami w nawiasach ostrych podajemy symbol oznaczający typ: *nazwa<T>(argumenty)* Symbolu tego możemy następnie używać zarówno definiując parametry metody i zwracaną wartość, jak i w jej ciele. Symbol musi być określony w momencie wywołania metody. Do czego również wykorzystuje się ostre nawiasy: nazwa\_metody<np.:\_int> |

Wywołanie:

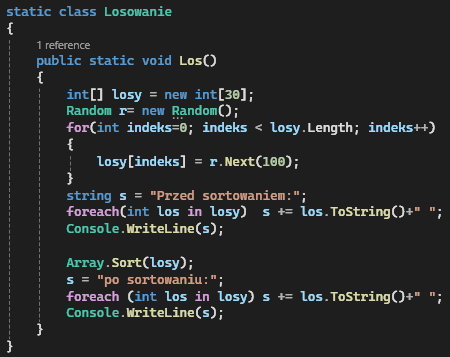


metoda:



**Exercise 3**

Sortowanie tabeli liczb pseudolosowych (posortowanie liczb całkowitych, rzeczywistych łańcuchów. Służy do tego metoda statyczna Array.Sort();)

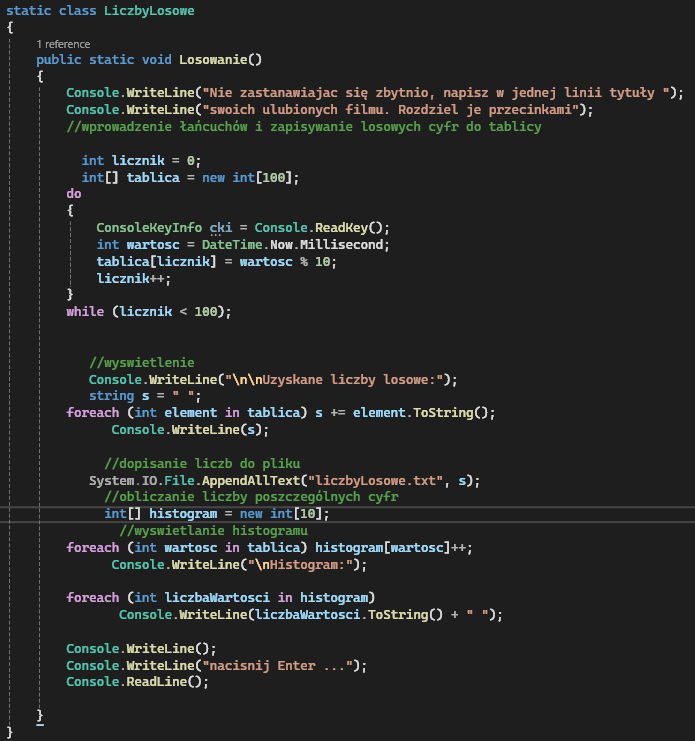


**Exercise 4**

Chcemy uzyskać zbiór stu w pełni losowych liczb z zakresu od 0 do 9. Generowanie losowego zbioru liczb zmusimy użytkownika. Każemy mu wprowadzić łańcuchy. Ważne będzie tempo, w jakom to zrobi. Będziemy pobierać ostatnie cyfry w liczbie milisekund z momentu wprowadzenia każdej litery. A ponieważ tak uzyskany zbiór liczb staje się cenny przez wysiłek włożony, zapiszemy go w pliku tekstowym.

1. Ścieżka dostępu do utworzonego przez ten program pliku liczbtLosowe.txt





1. Pętla do … while() oczekuje na znaki wprowadzone przez użytkownika. (odczyt – metoda Console.ReadKey). Jeżeli jej argumentem jest false wpisany znak (od 0 do 999) z bieżącego czasu odczytywanego z własnego DataTime.Now.
2. Obliczamy resztę z dzielenia tej liczby przez 10, co jest równe ostatniej cyfrze tej liczby. To będą nasze losowe liczby. Zapisujemy je do tablicy tablica. Pętla towarzyszy zdefiniowany przez nas indeks o nazwie licznik. Pętla jest kontynuowana dopóty, dopóki jego wartość jest mniejsza od 100. Kolejny fragment kodu (pętla foreach) przenosi odczytane cyfry do łańcucha, który jest wyświetlany w konsoli, a także zapisywany do pliku.
3. Zapis realizujemy **metodą AppendAllText**, która nie tworzy nowego pliku jeżeli ten już istnieje, a dodaje nowy tekst do istniejącego. Dzięki temu kolejne uruchomienia programu będą powiększały zbiór losowych cyfr. Statyczna metoda AppendAllText zdefiniowana jest w klasie File z przestrzeni System.IO.
4. Oblicz histogram - 10 elementową tablicę w której będziemy gromadzili liczebności każdej z cyfr. Następnie przebiegamy cały zbiór wylosowanych liczb, tj tablicę tablica zwiększając o jeden odpowiednie elementy z tablicy tablica jako indeksy tablicy histogram w zależności od kolejnych wartości. Możemy to łatwo zrobić traktując wartości z tablicy tablica jako indeksy tablicy histogram.

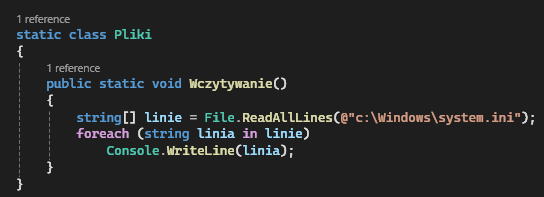
Pliki tekstowe.

Poznaliśmy klasę StreamReader – dzięki ,której odczytaliśmy plik tekstowy linijka po linijce. Do dyspozycji mamy analogiczną klasę StreamWrite umożliwiającą zapisywanie plików.

Jeśli pliki nie są duże, wygodniej jest korzystać z zestawu statycznych metod zdefiniowanych w klasie System.IO,File. Pierwsza o nazwie **ReadAllLines**, służy do odczytania całego pliku. Zachowuje przy tym podziała na linie/akapity, które otrzymujemy rozdzielone w tablicy łańcuchów string[].

**Exercise 5**

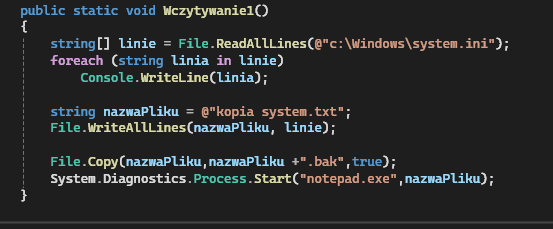
W zadaniu wczytujemy zawartość pliku i wyświetlamy go w konsoli.



File.WriteAllLines -to funkcja , która pozwala zapisać tablicę łańcuchów do pliku.

**Exercise 6**

Wczytujemy zawartość pliku wyświetlamy, kopiujemy do lokalnego folderu: (ze zmienioną nazwą). Kopiujemy plik bez zmiany jego zawartości File.Copy .Process.Start która pozwala na uruchomienie zewnętrznego programu tu uruchamiamy notatnik żeby pokazać skopiowany plik.



**Powtórzenie**

**Zadanie 1**

|  |  |
| --- | --- |
| *using (StreamReader sr = new StreamReader(“dane.txt”))*  *{*  *string linia;*  *while((linia=sr.ReadLine()) !=null)*  *{*  *console.WriteLine(linia);*  *}*  *}* | //utworzenie zmiennej sr klasy StreamReader (pozwala czytać z pliku tak jak ze strumienia), sr przestanie istnieć po zakończeniu tego wątku.  Deklaracja zmiennej linia w której będziemy zapisywać linijki pobrane z pliku.  Linia przyjmuje linie z pliku dopóki będą jeżeli nie będzie nastąpi wyjście z pętli.  Wyświetlanie linijek z pliku. |

*System.IO.File.AppendAllText(„nazwaPliku.txt)* -jeżeli plik nie istnieje to zostanie stworzony o podanej nazwie , jeżeli istnieje to tam będą zapisywane liczby.

*string[] linie=File.ResdAllLines(„ścieżka i nazwa pliku”)* – Służy do odczytywania całego pliku, zachowuje podział na linie/akapity, które otrzymujemy rozdzielone w tablicy łańcuchów.

*string[] linie = FileWriteAllLines(„ścieżka i nazwa pliku”)* - pozwala zapisać tablicę łańcuchów do pliku.

*File.Copy* – najprostrze kopiowanie zawartości pliku bez zmiany zawartości.

*Process.Start* -pozwala na uruchomienie zewnętrznego programu np. notatnik.

*System.Diagnostics.Process.Start(„notepad.exe, nazwaPliku);*

**Exercise 7 TRUDNE**

Użyć metodę ReadAllLines i WriteAllLines do zapisania i odczytania z plików tekstowych dwuwymiarowych tablic liczb rzeczywistych w formacie przypominającym CSV. Utworzyć dwie metody realizujące to zadanie: zapiszTablicę, i czytajTablicę Obydwie metody opierają się na użyciu podwójnych pętli for. W metodzie zapiszTablicę budowana jest w ten sposób jednowymiarowa tablica łańcuchów. Pętla zewnętrzna indeksowana zmienną j, odpowiedzialna jest za zbieranie kolejnych łańcuchów do tablicy, natomiast pętla wewnętrzna z indeksem i – za tworzenie linii ze skonwertowanych do łańcuchów wartości z jednego wiersza tablicy. Elementy w linii rozdzielane są znakiem podanym przez użytkownika. Domyślnie jest to średnik.

Kod główny (main):

*Console.WriteLine("Czytanie i zapisywanie do pliku");*

*double[,] tablica = new double[4, 8];*

*int licznik = 0;*

*for (int j = 0; j < tablica.GetLength(1); j++)*

*for (int i = 0; i < tablica.GetLength(0); i++)*

*tablica[i, j] = licznik++;*

*Tab7.zapiszTablice("tablica.csv", tablica);*

*Console.WriteLine();*

*double[,] tablica\_kopia = Tab7.czytajTablice("tablica.csv");*

*Console.WriteLine("Rozmiar tablicy: "+tablica\_kopia.GetLength(0)+"x"+tablica\_kopia.GetLength(1));*

*for(int j = 0;j< tablica\_kopia.GetLength(1); j++)*

*{*

*for (int i = 0; i < tablica\_kopia.GetLength(0); i++)*

*Console.Write(tablica[i, j].ToString() + "\t");*

*Console.WriteLine();*

*}*

Metody wywoływane:

*static class Tab7*

*{*

*public static void zapiszTablice(string sciezkaPlik, double[,] tablica, char separator = ';')*

*{*

*int liczbaWierszy = tablica.GetLength(1);*

*string[] linie =new string[liczbaWierszy];*

*for(int j=0;j<liczbaWierszy;j++)*

*{*

*string linia = " ";*

*for(int i = 0; i < tablica.GetLength(0); i++)*

*{*

*linia += tablica[i, j].ToString() + separator;*

*}*

*linie[j] = linia.TrimEnd(separator);*

*}*

*File.WriteAllLines(sciezkaPlik, linie);*

*}*

*public static double[,] czytajTablice (string sciezkaPliku, char separator = ';')*

*{*

*string[] linie = File.ReadAllLines(sciezkaPliku);*

*int liczbaWierszy=linie.Length;*

*int liczbaKolumn = linie[0].Split(separator).Length;*

*double[,] tablica = new double[liczbaKolumn, liczbaWierszy];*

*for (int j = 0; j < liczbaWierszy; j++)*

*{*

*string[] elementy = linie[j].Split(separator);*

*for (int i= 0; i < liczbaKolumn; i++)*

*{*

*tablica[i, j] = double.Parse(elementy[i]);*

*}*

*}*

*return tablica;*

*}*

*}*

*}*

**Lekcja13: Tablice dla typów referencyjnych**

**Inicjacja elementów tablicy**

W tablicy zawierającej typy referencyjne domyślne wartość elementów tablicy to null. Konieczna jest jawa inicjacja elementów przez wywołanie operatora new dla każdego elementu tablicy i utworzenie odpowiedniej grupy obiektów. Nie należy w tym przypadku mylić wykorzystanie operatora new w celu utworzenia tablicy z użyciem go do utworzenia jej elementów.



„Komunikat:” + te[0].Message” pokaże domyślny komunikat pierwszego wyjątku z tablicy.

*Console.WriteLine("wykorzystanie foreach");*

*foreach (Exception ei in te)*

*{*

*ei.HelpLink = "http://www.fizyka.umk.pl"; ;*

*Console.WriteLine(ei.Message);*

*}*

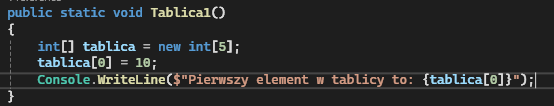
Co prawda pętla foreach nie pozwala, ale w przypadku typów referencyjnych jest inaczej. W tablicy są przechowywane referencje do obiektów i one rzeczywiście nie mogą być zmieniane – obiektu którego referencja zapisana jest w tablicy nie można w pętli foreach zastąpić innym. Natomiast nic nie stoi na przeszkodzie aby modyfikować w niej obiekty typów referencyjnych własności i metod (jednak bez zmiany samej referencji.

**Płytkie i głębokie kopiowanie**

**Lekcja 14: Zadania Tablice**

Exercise 1

Zadeklaruj i zainicjalizuj tablicę elementów typu całkowitego. Przypisz pierwszemu elementowi tablicy dowolną wartość. Wyświetl zawartość tego elementu na ekranie.



Sprawdź przy pomocy pętli foreach co zawierają inne komórki w tablicy.

*public static void Tablica1()*

*{*

*int[] tablica = new int[5];*

*tablica[0] = 10;*

*Console.WriteLine($"Pierwszy element w tablicy to: {tablica[0]}");*

*foreach(int i in tablica)*

*{*

*Console.WriteLine(i);*

*}*

*}*

Exercise 2.

Zadeklaruj i zainicjuj tablicę 10-elementową. Spróbuj przypisać elementowi o indeksie 10 *dowolną liczbę całkowitą.*

*public static void Tablica2()*

*{*

*int[] tablica = new int[10];*

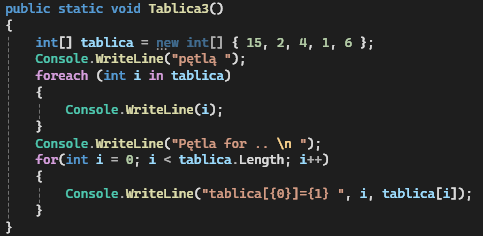
*tablica[10] = 1;*

*Console.WriteLine("Element o indeksie 10 to: " + tablica[10]);*

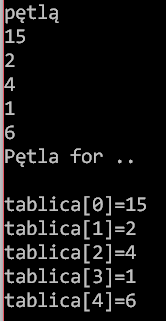
Program można skompilować ale przy urychomieniu na ekranie zobaczymy okno z informacją o wystąpieniu błędu. Może ono mieć różną postać w zależności od tego, w jakiej wersji systemu została uruchomiona aplikacja. Błąd zostanie wychwycony

Exercise 3

Zadeklaruj tablicę 5-elementową typu int i zainicjalizuj ją liczbami od całkowitymi. Zawartość tablicy wyświetl. Metoda z dwiema pętlami :



Odp:



Exercise 4

Utwórz tablicę zawierającą pewną liczbę wartości całkowitych. Zawartość tablicy wyświetl na ekranie za pomocą pętli for. Do określenia rozmiaru tablicy użyj właściwości Length.

*public static void Tablica4()*

*{*

*int[] Tablica = new int[6] { 2, 4, 6, 8, 10, 1 };*

*for(int i = 0; i < Tablica.Length; i++)*

*{*

*Console.WriteLine("Tablica [" + i + "]=" + Tablica[i]);*

*}*

*}*

**Exercise 5**

Użyj pętli for do zapisania w 10-elementowej tablicy 10 kolejnych liczb całkowitych.

*public static void Tablica5()*

*{*

*int[] Tablica = new int[10];*

*for(int i=0; i < Tablica.Length; i++)*

*{*

*Tablica[i] = i + 1;*

*}*

*Console.WriteLine("Zawartość tablicy: ");*

*for(int i=0; i < Tablica.Length; i++)*

*{*

*Console.WriteLine("Tablica[{0}]={1}", i, Tablica[i]);*

*//w C# 6.0 można jeszcze tak: zapis roboczy*

*Console.WriteLine($"Tablica[{i}]={Tablica[i]}");*

*}*

*}*

Pętla Foreach – Tablice:

Jest wygodna do obsługi tablic, umożliwia prostą iterację po wszystkich elementach tablicy; nie trzeba wtedy wprowadzać dodatkowej zmiennej iteracyjnej. Struktura zapisu:

*Foreach (typ identyfikator in wyrażenie)*

*{*

*//instrukcje*

*}*

Jeżeli zatem mamy tablicę o nazwie *tab* zawierającą liczby typu *int* możemy zastosować konstrukcję:

*Foreach(int val in tab)*

*{*

*//instrukcje*

*}*

Wtedy w kolejnych przebiegach pętli pod val będą podstawiane kolejne elementy tablicy. Słowo val jest tu identyfikatorem odczytanej wartości

**Exercise 6**

Wypełnij tablicę losowymi liczbami całkowitymi typu int. Wykorzystaj pętlę foreach do sprawdzenia, ile jest w tej tablicy liczb parzystych.

*public static void Tablica6()*

*{*

*int[] Tablica = new int[100];*

*int parzyste = 0, nieparzyste = 0;*

*Random rand = new Random();*

*for (int i = 0; i < Tablica.Length; i++)*

*{*

*Tablica[i]= rand.Next();*

*}*

*foreach(int i in Tablica)*

*{*

*if(i%2==0)*

*{*

*parzyste++;*

*}*

*else*

*{*

*nieparzyste++;*

*}*

*}*

*Console.WriteLine("Parzyste: {0}",parzyste);*

*Console.WriteLine("nieparzyste {0}", nieparzyste);*

*}*

**Tablice wielowymiarowe**

Wymiarów może być więcej

Deklaracja tablicy dwuwymiarowej:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0,0 | 0,1 | 0,2 | 03, | 0,4 |
| 1,0 | 1,1 | 1,2 | 1,3 | 1,4 |

*typ\_tablicy[,] nazwa\_tablicy={{wartość\_1, wartość\_2, … wartość\_N}{wartość\_1, wartość\_2, …, wartość\_N}}*

Zapis ten można rozbić na kilka linii w celu zwiększenia czytelności:

*Typ\_tablicy[,] nazw\_tablicy=*

*{*

*{war\_1, war\_2, …, war\_n}*

*{war\_1, war\_2, …, war\_n}*

*}*

Drugi sposób utworzenia tablicy dwuwymiarowej:

***typ\_tablicy [,] nazwa\_tablicy=new typ\_tablicy[liczba\_wierszy, liczba\_kolumn];***

np.:

*int[,] tablica = new int[2,5];*

**Exercise 7**

Zadeklaruj tablicę dwuwymiarową typu int o dwóch wierszach i pięciu kolumnach i zainicjalizuj ją kolejnymi liczbami całkowitymi. Zawartość wyświetl na ekranie.

*public static void Tablica7()*

*{*

*int[,] Tablica = new int[2, 5] { { 0, 1, 2, 3, 4 },{ 5, 6, 7, 8, 9 } };*

*for(int i = 0;i < 2; i++)*

*{*

*for (int j = 0; j < 5; j++)*

*{*

*Console.WriteLine("tablica[{0},{1}]={2}", i, j, Tablica[i, j]);*

*}*

*}*

*}*

Wprowadzenie wartości do tablicy w pierwszych pętlach

*public static void Tablica8()*

*{*

*int[,] Tab=new int[2,5];*

*int licznik = 0;*

*for(int i = 0; i < 2; i++)*

*{*

*for(int j = 0;j < 5; j++)*

*{*

*Tab[i,j] = licznik++;*

*}*

*}*

*for(int i = 0; i < 2; i++)*

*{*

*for(int j=0;j< 5; j++)*

*{*

*Console.WriteLine("Tablica [{0},{1}]={2}", i, j, Tab[i,j]);*

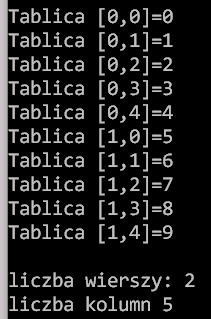
*}*

*}*

*int liczbaKolumn = Tab.GetLength(1);*

*int liczbaW=Tab.GetLength(0);*

*Console.WriteLine("liczba wierszy "+liczbaWierszy+"\n liczba kolumn: "+liczbaK);*



Tablice tablic

*int []* – oznacza tablicę liczb typu int,

*int[][]* – oznacza tablicę tablic typu int, czyli tablicę składającą się z innych tablic typu int Czyli:

*int[][]* tablica=new int[n][] – zadeklarowanie i utworzenie n-elementowej tablicy, której elementami będą mogły być inne tablice liczb typu int. Można to zrobić oddzielnymi instrukcjami:

*tablica[0]=new int[m];*

*tablica[1] = new int[m]*

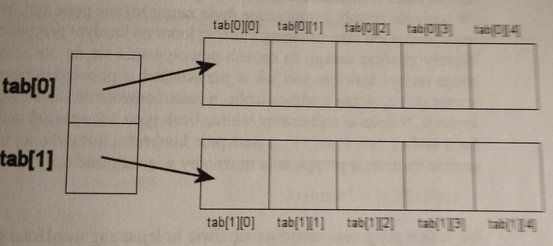
lub też używajac pętli:

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

*tab[i]=new int[m];*

*}*

W ten sposób powstanie n-elementowa tablica zawierajaca m – elementowe tablice liczb typu int. Zilustrowano:



**Exercise 8**

Zbuduj dwie tablice zawierające liczby typu int wprowadź wartości i wyświetl wyniki.

*public static void Tablica9()*

*{*

*int[][] tab = new int[2][];*

*for(int i = 0; i < 2; i++)*

*{*

*tab[i] = new int[5];*

*}*

*int licznik = 0;*

*for (int i=0;i< 2; i++)*

*{*

*for (int j = 0; j < 5; j++)*

*{*

*tab[i][j] = licznik++;*

*}*

*}*

*for(int i = 0;i< 2; i++)*

*{*

*for(int j = 0;j< 5; j++)*

*{*

*Console.WriteLine("tab[{0}][{1}]={2}", i, j, tab[i][j]);*

*}*

*}*

*}*

Najperw zadeklarowano tablicę której wartością przechowywaną będą tablice potem w pętli zadeklarowano dwie tablice pięcio-elementowe. W zagnieżdżonych pętlach wpisano wartości od jednego do 10. Pierwsza zewnętrzna, odpowiada za iterację po indeksach wierszy tablicy. Druga – za iterację po indeksach kolumn.

Najpierw została tu zadeklarowana i utworzona tablica główna o odpowiednim typie (int[][]tab=new int[2][]). Następnie w pętli for poszczególnym jej komórkom zostały przypisane utworzone za pomocą operatora new nowe tablice liczb typu int (tab[i]=new int[5];).

Zamiast pętli można by też użyć dwóch instrukcji:

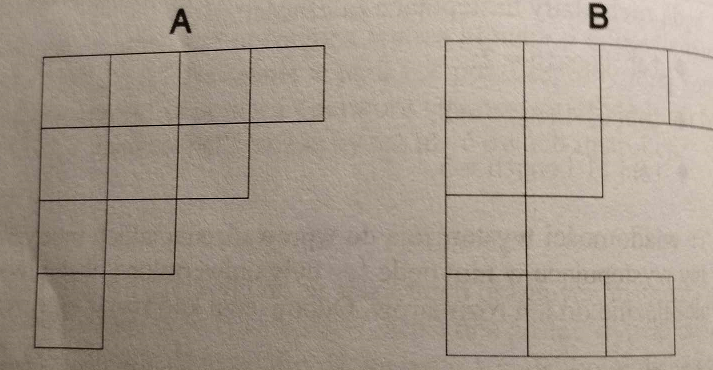
*tab[0]=new int[5];*

*tab[1]=new int[5];*

Skutek byłby taki sam. Tak więc po wykonaniu wszystkich opisanych instrukcji powstała tablica tablic o strukturze takiej jak na rysunku wyżej.

**Tablice nieregularne**

Struktury tablic mogą być nieregularne np.:



Wiemy, że tablice wielowymiarowe to tablice tablic jednowymiarowych. To znaczy że tablica dwuwymiarowa to tablica jednowymiarowa zawierająca szereg tablic jednowymiarowych. Tablica trójwymiarowa to tablica jednowymiarowa zawierająca w sobie tablice dwuwymiarowe itd.

Deklaracja : *int[][]tab=new int[4][];*

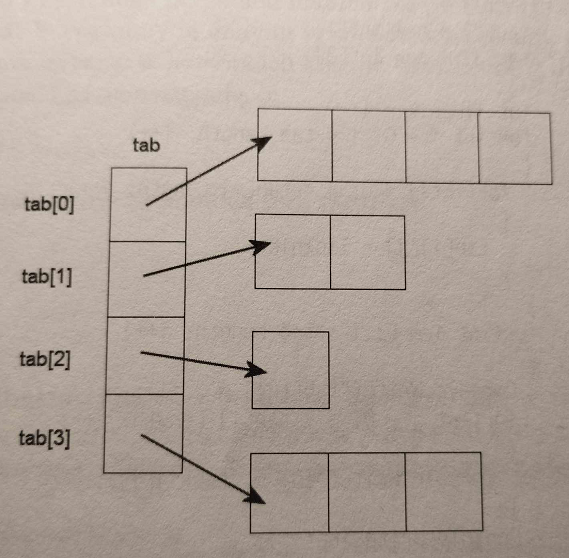
Następnie kolejnym elementom: tab[0], tab[1], tab[2], tab[3], przypiszemy nowoutworzone tablice jednowymiarowe liczb typu int, tak aby w komórce tab[0] znalazła się tablica 4-elementowa, tab[1] – 2-elementowa, tab[2] jednoelementowa tab[3] trzyelementowa Czyli należy:

*tab[0]=new int[4];*

*tab[1]=new int[2];*

*tab[2]=new int[1];*

*tab[3]=new int[3];*



Chcemy wypełnić kolejne komórki liczbami od 1 do 10 –

pierwszy wiersz – 1,2,3,4

drugi wiersz – 5,6;

trzeci wiersz -7

czwarty wiersz – 8,9,10

**Exercise 9**

Wypełnić powyższą strukturę i wyświetlić liczbami od 1 do 10.

*public static void Tablica10()*

*{*

*int[][] tab = new int[4][];*

*tab[0] = new int[4];*

*tab[1] = new int[2];*

*tab[2] = new int[1];*

*tab[3] = new int[3];*

*int licznik = 1;*

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

*{*

*for(int j=0; j< tab[i].Length; j++)*

*{*

*tab[i][j] = licznik++;*

*}*

*}*

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

*{*

*Console.Write("tab{0}= ", i);*

*for(int j = 0; j < tab[i].Length; j++)*

*{*

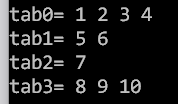
*Console.Write("{0} ", tab[i][j]);*

*}*

*Console.WriteLine(" ");*

*}*

*}*



**Exercise 10**

Utwórz tablicę o trójkątnym kształcie.

*public static void Tablica11()*

*{*

*int[][]tab= new int[4][];*

*tab[0] = new int[4];*

*tab[1] = new int[3];*

*tab[2] = new int[2];*

*tab[3] = new int[1];*

*int licznik = 1;*

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

*{*

*for (int j = 0; j < tab[i].Length; j++)*

*{*

*tab[i][j] = licznik++;*

*}*

*}*

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

*{*

*Console.Write("tab{0}", i);*

*for(int j = 0; j < tab[i].Length; j++)*

*{*

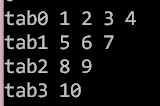
*Console.Write("{0} ", tab[i][j]);*

*}*

*Console.WriteLine(" ");*

*}*

*}*



**Exercise 11**

Zmodyfikuj kod z poprzedniej tablicy tak aby pisać liczby od 10 do 1 zastosuj pętle for do stworzenia tablic wewnętrznych.

*public static void Tablica12()*

*{*

*int[][] tab = new int[4][];*

*for (int i = 0; i < 4; i++)*

*{*

*tab[i] = new int[4 - i];*

*}*

*int licznik = 10;*

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

*{*

*for (int j = 0; j < tab[i].Length; j++)*

*{*

*tab[i][j] = licznik--;*

*}*

*}*

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

*{*

*Console.WriteLine("tab{0} ", i);*

*Console.WriteLine("ilość elementów w tablicy: "+tab[i].Length);*

*for (int j = 0; j < tab[i].Length; j++)*

*{*

*Console.Write("[{0}] ", tab[i][j]);*

*}*

*Console.WriteLine();*

*}*

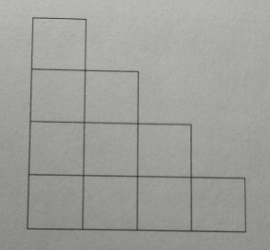
*} }*

**Exercise 12**

Utwórz tablicę liczb typu int zaprezentowaną na rysunku. Wypełnij kolejne komórki wartościami malejącymi od 10 do 1. Do utworzenia tablicy i wypełniania jej danymi wykorzystaj pętlę typu for.

*public static void Tablica13()*

*{*

 *int[][] tab = new int[4][];*

*for(int i = 0;i < 4; i++)*

*{*

*tab[i]=new int[1 + i];*

*}*

*int licznik=10;*

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

*{*

*for(int j = 0;j < tab[i].Length; j++)*

*{*

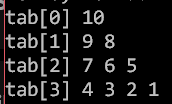
*tab[i][j] = licznik--;*

*// Console.Write(tab[i][j]);*

*}*

*}*

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

* {*

*Console.Write("tab[{0}] ", i);*

*for(int j = 0;j< tab[i].Length; j++)*

*{*

*Console.Write("{0} ", tab[i][j]);*

*}*

*Console.WriteLine();*

*}*

*}*

**Exercise 13.**

Utwórz tablicę dwuwymiarową typu bool o rozmiarze 5x8 komórek. Wypełnij ją danymi w taki sposób aby komórki o parzystych indeksach wiersza i kolumny zawierały wartość true, a pozostałe wartość false (przyjmij że 0 jest wartością parzystą).

*public static void Tablica14()*

*{*

 *bool[,] tab = new bool[5, 8];*

*for(int i = 0; i < 5; i++)*

*{*

*for(int j = 0; j < tab.GetLength(1); j++)*

*{*

*if (i == 0 || j == 0)*

*{*

*tab[i,j] = true;*

*}else if(i % 2 == 0 && j % 2 == 0)*

*{*

*tab[i,j] = true;*

*}*

*else*

*{*

*tab[i,j] = false;*

*}*

*}*

*}*

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

*{*

*for (int j = 0; j < tab.GetLength(0); j++)*

*{*

*if (tab[i, j] == true)*

*{*

*Console.Write(1 + " ");*

*}else if (tab[i, j]==false)*

*{*

*Console.Write(0 + " ");*

*}*

*}*

*Console.WriteLine() ;*

*}*

*}*

**Exercise 14**

Utwórz przykładową tablicę trójwymiarową i wypełnij ją przykładowymi danymi. Zawartość wyświetl.

**Exercise 15**

Zabawa dwoma kostkami. Sto razy rzucamy dwiema kostkami do gry. Interesować nas będą sumy oczek na obydwóch kostka. Napisać metodę, która zwr óci tablicę zawierającą sumy dwóch liczb wylosowanych z zakresu 1 do 6

Wołanie z main:

*TabR.Tab2();*

*int[] wartosc = TabR.zbiorSumOczek(100);*

*foreach(int w in wartosc)*

*{*

*Console.Write(" " + w + ";");*

*}*

Metoda:

*public static int[] zbiorSumOczek(int x=100)*

*{*

*Random r = new Random();*

*int[] wyniki = new int[x];*

*for(int i = 0; i < x; ++i)*

*{*

*int pierwszaKostka = r.Next(1, 7);*

*int drugaKostka = r.Next(1, 7);*

*wyniki[i] = pierwszaKostka + drugaKostka;*

*}*

*return wyniki;*

*}*

**Exercise 16**

Do powyższego zadania należy dopisać jeszcze jedną metodę która będzie obliczała sumę wszystkich wylosowanych liczb (agregacja danych czyli obliczanie różnych statystyk na podstawie uzyskanych danych).

Należy dopisać do main:

*double[]tablica=Array.ConvertAll<int, double>(wartosc,i=>(double)i);*

*Console.WriteLine("\nLiczba elementów: "+tablica.Length);*

*Console.WriteLine("\nSuma: "+TabR.Suma(tablica));*

Druga metoda:

*public static double Suma(double[] wartosc)*

*{*

*double suma = 0;*

*foreach (double w in wartosc) suma += w;*

*return suma;*

*}*

Metoda Array.ConvertAll przekonwertuje dane typu int na dane typu double

*Array.ConvertAll<int, double>(wartosc,i=>(double)i);*

Drugim argumentem metody Array.ConvertAll jest wyrażenie lambda – przepis na konwersję elementów z typu int na typ double. W naszym przypadku jest to po prostu użycie operatora konwersji (na każdym elemencie tablicy osobno).

Oblicz, obok sumy średnią tych liczb: suma/ilość elementów.

Wywołanie w main:

*Console.WriteLine("Srednia wylosowanych liczb to: " + TabR.srednia(tablica));*

Metoda srednia:

*public static double srednia(double[] wartosc)*

*{*

*if (wartosc == null) throw new ArgumentNullException("przesłano pusty obiekt");*

*if (wartosc.Length == 0) throw new ArgumentException("w tablicy nie ma elementów");*

*return Suma(wartosc) / wartosc.Length;*

*}*

**Exercise 17**

*Przeszukiwanie zbioru.*

Napisz metodę, która ustali, jaka jest minimalna i maksymalna wartość elementów tablicy. Do tego nie potrzebujemy agregacji tylko przeszukania danych. Przebiegając wszystkie elementy tablicy będziemy szukać tych, które mają minimalną i maksymalną wartość. Zadeklarujmy zmienne lokalne minimum i maksimum i zainicjujmy wartością pierwszego elementu tablicy. Następnie będziemy sprawdzać pozostałe elementy tablicy w relacji do tych dwóch zmiennych lokalnych czyli mniejsza albo większa. Jeżeli są to kopiujemy je do odpowiedniej zmiennej lokalnej większa do maksimum mniejsza do minimum , kontynuując przeszukiwanie.

Cdalszy zadania tablica[100]elementowa wykonana w poprzednim zadaniu funkcja main:

*int indeksMin, indeksMax;*

*TabR.ekstrema(tablica, out indeksMin, out indeksMax);*

*Console.WriteLine("wartość od " + tablica[indeksMin] + " do " + tablica[indeksMax]);*

*Console.WriteLine("Zakres: " + TabR.zakres(tablica));*

Dwie funkcje 1 porządkująca i ustalajaca zakres indeksów od … do.:

*public static void ekstrema(double[] wartosc, out int indeksMin, out int indeksMax)*

*{*

*if (wartosc == null) throw new ArgumentNullException("przesłano pusty obiekt");*

*if (wartosc.Length == 0) throw new ArgumentException("w tablicy nie ma elementów");*

*indeksMax = 0;*

*indeksMin = 0;*

*double minimum = wartosc[0];*

*double maximum = wartosc[0];*

*for (int i = 1; i < wartosc.Length; ++i)*

*{*

*if (wartosc[i] < minimum)*

*{*

*minimum = wartosc[i];*

*indeksMin = i;*

*}*

*if (wartosc[i] > maximum)*

*{*

*maximum = wartosc[i];*

*indeksMax = i;*

*}*

*}*

*}*

*public static double zakres(double[] wartosc)*

*{*

*int indeksMin, indeksMax;*

*ekstrema(wartosc,out indeksMin,out indeksMax);*

*return wartosc[indeksMax] - wartosc[indeksMin];*

*}*

Metoda ekstream zwraca przez argumenty indeksy elementów tablicy min i max .Jednak dysponując indeksami łatwo pobrać odpowiednie wartości, a indeksy mogą być przydatne do wprowadzania jakichś zmian w tablicy.

**Exercise 18**

Powyższe obliczeni:

a możemy zastąpić gotowymi metodami które znajdują się w .NET w przestrzeni nazw System.Linq

Wywołanie: *Linq1.Obliczenia();*

