Arytmetyka w programowaniu - zadania

Zadanie 2.1.

Napisz program przeliczający temperaturę w stopniach Celsjusza na temperaturę w stopniach Fahrenheita. Program ma prosić użytkownika o podanie temperatury w stopniach Celsjusza. Wzór: $F = 32 + \frac{9}{5}C$.

Zadanie 2.2.

Napisz program, który oblicza deltę dla równania kwadratowego $ax^2 + bx + c = 0$. Program ma prosić użytkownika o podanie współczynników równania a, b oraz c. Wzór: $\Delta = b^2 - 4ac$.

Zadanie 2.3.

Napisz program, który oblicza wskaźnik masy ciała BMI. Program ma prosić użytkownika o podanie wagi w kilogramach oraz wzrostu w metrach. Wzór: $BMI = \frac{masa}{wzrost^2}$.

Sterowanie działaniem programu – instrukcje warunkowe i pętle - zadania

Zadanie 3.1.

Napisz program, który sprawdza, czy podany rok jest rokiem przestępnym. Rok przestępny dzieli się bez reszty przez 4, nie dzieli się przez 100 (za wyjątkiem lat podzielnych przez 400).

Zadanie 3.2.

Napisz program pobierający od użytkownika dwie liczby całkowite. Program powinien wyświetlać informację, czy druga liczba jest dzielnikiem pierwszej.

Zadanie 3.3.

Napisz program pobierający od użytkownika 3 liczby. Program ma wyświetlić wartość największej z nich.

Zadanie 3.4.

Napisz program – prosty kalkulator, który wczytuje od użytkownika wartości dwóch zmiennych typu *double* oraz znak operacji (+ lub – lub * lub /), a następnie wyświetla wynik operacji dla podanych wartości. Przykładowo użytkownik wprowadził znak „+" i liczby 1,5 oraz 2,5, program powinien wyświetlić sumę obu liczb, czyli 4,0.

Zadanie 3.5.

Napisz program obliczający liczbę pierwiastków równania kwadratowego. Program ma prosić użytkownika o podanie współczynników równania, a następnie ma wyświetlić stosowny komunikat.

Zadanie 3.6.

Napisz program, który oblicza wskaźnik masy ciała BMI. Program ma prosić użytkownika o podanie wagi w kg oraz wzrostu w metrach. Wzór: $BMI=\frac{masa}{wzrost^2}$ (treść zadania 2.3).

a) Po obliczeniu wskaźnika BMI program powinien wyświetlać stosowną informację w zależności od wartości wskaźnika:

- $< 18{,}5$ – niedowaga,
- 18,5–24,99 – wartość prawidłowa,
- $\geq 25{,}0$ – nadwaga.

b) Korzystając z Wikipedii rozszerz program, tak aby wyświetlał komentarz według poszerzonej klasyfikacji zakresów wskaźnika BMI.

Zadanie 3.7.

Wykonaj program z przykładu 3.8 (str. 63) z użyciem instrukcji *switch..case* (zamiast *if..else*).

Zadanie 3.8.

Pobierz od użytkownika wartość średniej ocen. Program ma wyświetlać informacje o wysokości przysługującego stypendium zgodnie z poniższą tabelą:

Średnia ocen		Kwota stypendium
Od	Do	
2,00	3,99	0,00 zł
4,00	4,79	350,00 zł
4,80	5,00	550,00 zł

Zadanie 3.9.

Napisz program w czterech wariantach (a, b, c i d), którego efektem działania będzie „figura" utworzona ze znaku gwiazdki (*) przedstawiona na danym rysunku.

a	b	c	d
*	****	*	****
**	***	**	* *
***	**	***	* *
****	*	****	****

(Liczbę wyświetlanych wierszy podaje użytkownik).

Zadanie 3.10.

Napisz program obliczający n! (*n* silnia), gdzie *n* jest podane przez użytkownika.

Zadanie 3.11.

Napisz program obliczający ile kolejnych liczb całkowitych (rozpoczynając od wartości 1) należy dodać do siebie, aby suma przekroczyła wartość 100.

Zadanie 3.12.

Napisz program pobierający od użytkownika liczby całkowite. Program ma pobierać te liczby do czasu, gdy użytkownik wprowadzi wartość 0 (zero). Wynikiem działania programu ma być informacja o sumie wprowadzonych przez użytkownika liczb.

Zadanie 3.13.

Napisz program obliczający sumę szeregu $W(n)=1 - 2 + 3 - 4 + ...\pm n$, gdzie *n* jest dowolną liczbą naturalną, którą program ma wczytać.

Zadanie 3.14.

Liczba *N* jest doskonała, gdy jest równa sumie swych podzielników mniejszych od niej samej np. 6=1+2+3=6 – jest liczbą doskonałą. Napisz program znajdujący liczby doskonałe w przedziale <1,*n*>, gdzie *n* podaje użytkownik.

Zadanie 3.15.

Dysponując monetami 1 zł, 2 zł, 5 zł sprawdź, na ile różnych sposobów można wypłacić 10 zł. Napisz program, który wyświetli w oknie konsoli wszystkie możliwe kombinacje.

Operacje na typach referencyjnych – tablice i typ string – zadania

Zadanie 4.1.

Napisz program, który pozwoli zapełnić *n*–elementową tablicę jednowymiarową liczb całkowitych wartościami podanymi przez użytkownika. Na początku działania programu użytkownik podaje liczbę elementów tablicy, a następnie poszczególne wartości jej elementów. Po wypełnieniu całej tablicy program powinien wypisać je w oknie konsoli.

Zadanie 4.2.

Napisz program kopiujący z danej tablicy liczb całkowitych *tab1* do nowej tablicy *tab2* wyłącznie wartości dodatnie. Obie tablice mają być jednowymiarowe o rozmiarze równym 10 (czyli 10-elemetowe). Elementy pierwszej tablicy (*tab1*) należy wpisać w trakcie deklaracji tej tablicy.

Zadanie 4.3.

Napisz program wyświetlający informacje o wypełnionej przez użytkownika tablicy *n*-elementowej:

- wartość i numer pozycji największego elementu,
- wartość i numer pozycji najmniejszego elementu,
- średnia wartości wszystkich elementów tablicy,
- liczba dodatnich elementów tablicy.

Zadanie 4.4.

Napisz program, który podaje, ile jest liczb pierwszych w tablicy 100 elementowej typu *int*. Tablicę należy wypełnić losowymi wartościami. Wskazówka: Poniższy fragment programu pokazuje działanie klasy *Random* (która zawiera generator liczb pseudolosowych) – w pętli zostanie wyświetlonych 100 liczb wybranych losowo z zakresu 1 – 999 (o zakresie decydują argumenty podane w wywołaniu metody *Next()*[43]).

```
Random rand = new Random();
for (int i = 0; i < 100; i++)
    Console.Write("{0,8}", rand.Next(1, 1000));
```

Zadanie 4.5.

Dana jest *n*-elementowa tablica liczb całkowitych *tab1*. Napisz program kopiujący wartości elementów tablicy *tab1* do tablicy *tab2* (o tym samym rozmiarze) z przesunięciem o jedną pozycje. To znaczy, że element w tablicy źródłowej o indeksie 0 powinien znaleźć się w tablicy docelowej pod indeksem 1, element o indeksie 1 ma być w tablicy docelowej pod indeksem 2 itd. Element ostatni tablicy źródłowej ma być elementem o indeksie 0 w tablicy docelowej.

Zadanie 4.6.

Napisz program, który deklaruje i inicjalizuje dwuwymiarową tablicę liczb rzeczywistych o rozmiarze 5 x 5. Program ma wyświetlić elementy tablicy (wiersz po wierszu), a następnie wyświetlić sumę elementów znajdujących się na głównej przekątnej tablicy (główna przekątna – od elementu o indeksach 0,0 do elementu o indeksach *n*,*n*).

Zadanie 4.7.

Napisz program, który dodaje dwie macierze o rozmiarze 2 x 3. Elementy macierzy należy umieścić w tablicach dwuwymiarowych w trakcie deklaracji. Program ma wyświetlić macierz wynikową. Wskazówka: Dodawanie macierzy – macierz wynikowa C zawiera elementy, które stanowią sumę elementów macierzy A i B o odpowiednich indeksach, tzn. element w zerowym wierszu i zerowej kolumnie macierzy A jest dodawany do elementu o tych samych indeksach macierzy B, element A [0,1] do B [0,1]... itd.

Zadanie 4.8.

Uzupełnij poniższy kod programu o wszystkie dni tygodnia i przy użyciu pętli wyświetl zawartość tablicy: w każdym wierszu dany dzień tygodnia w trzech językach (polskim, angielskim, niemieckim).

```
string[,] dniTygodnia;
dniTygodnia = new string[2, 3];     // pamiętaj o zmianie rozmiaru tablicy
dniTygodnia[0, 0] = "poniedzialek";
dniTygodnia[1, 0] = "wtorek";
dniTygodnia[0, 1] = "monday";
dniTygodnia[1, 1] = "tuesday";
dniTygodnia[0, 2] = "montag";
dniTygodnia[1, 2] = "dienstag";
```

Zadanie 4.9.

Napisz program obliczający liczbę wyrazów w łańcuchu znaków wprowadzonym przez użytkownika. Należy przyjąć, że wyrazy to ciągi znaków rozdzielone spacją.

Zadanie 4.10.

Napisać program, który pobierze datę w formacie DD-MM-RRRR, z której pobierze miesiąc i wyświetli jego nazwę słownie.

Zadanie 4.11.

Napisz program analizujący częstość występowania poszczególnych znaków w łańcuchu znaków wprowadzonym przez użytkownika. Np. dla wprowadzonego tekstu „abrakadabra" program powinien wyświetlić informacje: a – 5, b – 2, r – 2, k – 1, d – 1.

Zadanie 4.12.

Napisz program, który dla zadeklarowanej niżej zmiennej łańcuchowej wyświetli jej zawartość, poda liczbę wierszy oraz poda liczbę znaków w każdym wierszu.

```
// fragment powieści A. A. Milne, "Kubuś Puchatek"
string tekst = "W parę godzin później, gdy noc zbierała się do odejścia,\n" +
        "Puchatek obudził się nagle z uczuciem dziwnego przygnębienia.\n" +
        "To uczucie dziwnego przygnębienia miewał już nieraz i wiedział,\n" +
        "co ono oznacza. Był głodny. Więc poszedł do spiżarni,\n" +
        "wgramolił się na krzesełko, sięgnął na górną półkę, ale nic nie znalazł.";
```

Zadanie 4.13.

Napisz program, który przeanalizuje dany łańcuch pod kątem wielokrotnego występowania słów w tekście. Przykładowo dla zmiennej łańcuchowej o zawartości: „Kiedy idzie się po miód z balonikiem, to trzeba się starać, żeby pszczoły nie wiedziały, po co się idzie – odpowiedział Puchatek" – program powinien wypisać raport o słowach powielonych w tym tekście: idzie – 2 razy, po – 2 razy, się – 3 razy.

Zadanie 4.14.

W danej firmie środki trwałe mają identyfikatory złożone z kilku liter, myślnika oraz czterech cyfr. Te cztery cyfry to rok zakupu danego środka trwałego. Przykładowe identyfikatory to: KOMG-2002, BH-2010. Napisz program, który deklaruje 5-cio elementową tablicę typu *string* dla środków trwałych, którą należy zainicjalizować przykładowymi identyfikatorami w czasie deklaracji. Program ma dla każdego środka trwałego podać liczbę lat, jakie upłynęły od jego zakupu.

Zadanie 4.15.

Napisz program, który szyfruje podany przez użytkownika tekst prostym szyfrem podstawieniowym zwanym „gaderypoluki". Nazwa pochodzi od jednego z najczęściej używanych kluczy GA-DE-RY-PO-LU-KI. Klucz ten zawiera pary znakowych zamienników umieszczonych między myślnikami. Litery, których nie ma w kluczu pozostawia się w szyfrowanym tekście bez zmian. Przykładowo tekst „DRZEWO" po zaszyfrowaniu ma postać „EYZDWP".

0	1	2	3	4	5	6	7	8	9	10	11
G	A	D	E	R	Y	P	O	L	U	K	I

Wskazówki: Można zastosować w programie klucz z pominięciem myślników (GADERYPOLUKI). Wówczas można przyjąć, że znaki na parzystych pozycjach (numerując od 0) mają zamiennik po prawej stronie, a znaki na nieparzystych pozycjach mają swój zamiennik z lewej strony (np. litera O jest na pozycji numer 7 i ma swój zamiennik z lewej strony, czyli P). Zostaje zatem sprawdzenie, czy dany znak szyfrowanego tekstu występuje w kluczu, a jeśli tak to, na której pozycji klucza – parzystej czy nieparzystej. W tym celu możesz wykorzystać poznaną w tym rozdziale metodę *IndexOf()*. Dla zaszyfrowanego tekstu najlepiej będzie zadeklarować nową zmienną łańcuchową i zainicjalizować ją wartością pustą. Zalecamy, aby zrobić to przy pomocy statycznego pola publicznego *String.Empty* zawierającego łańcuch pusty (czyli ""), np.: `string tekstZaszyfrowany = String.Empty;`. A następnie w pętli dodawać kolejny znak z tekstu źródłowego – ten sam lub zamieniony zgodnie z kluczem (jeśli jest w kluczu).

Metody – zadania

Zadanie 5.1.

Napisz program zawierający metodę statyczną obliczającą temperaturę w stopniach Fahrenheita na temperaturę w stopniach Celsjusza. Metoda ma przyjmować jeden argument (temperaturę w stopniach Fahrenheita) i zwracać temperaturę w stopniach Celsjusza.

Zadanie 5.2.

Napisz program wczytujący 3 liczby rzeczywiste *a*, *b*, *x*, a następnie wywołujący metodę, która sprawdza, czy liczba *x* należy do przedziału obustronnie otwartego *(a, b)*. Metoda sprawdzająca ma zwrócić wartość logiczną, którą należy zinterpretować w metodzie *Main()* z podaniem stosownego komunikatu.

Zadanie 5.3.

Napisz program, który ma znaleźć współrzędne punktu po przesunięciu o dany wektor. W metodzie *Main()* wczytaj od użytkownika współrzędne punktu *A* oraz zadeklaruj współrzędne wektora *wek* [3, 2], a następnie wywołaj metodę o nazwie *Przesun()*, która ma przesunąć punkt *A* o wektor *wek* (dodać odpowiednie współrzędne). Współrzędne punktu (jako dwie zmienne typu *double*) mają zostać przesłane do tej metody przez referencję, a współrzędne wektora (także jako dwie zmienne typu *double*) przez wartość. Metoda *Przesun()* ma nic nie zwracać (*void*), aktualne współrzędne punktu mają być pamiętane dzięki użyciu argumentów przesyłanych przez referencje. Program ma wyświetlić współrzędne punktu po przesunięciu o wektor *wek*. Przykładowo, gdyby użytkownik podał początkowe współrzędne punktu *A* (2, 1), to wówczas program znajdzie położenie punktu *A* po przesunięciu w miejscu o współrzędnych (5, 3) (czyli 2+3, 1+2).

Zadanie 5.4.

Napisz program, który mnoży elementy tablicy jednowymiarowej przez zadaną liczbę. Mnożenie ma być wykonane w metodzie statycznej przyjmującej jako argumenty tablicę typu *int* oraz liczbę całkowitą (mnożnik).

Wykonaj zadanie w dwóch wariantach:

a) Wewnątrz metody tworzona jest nowa tablica wynikowa, która ma być zwrócona przez metodę.

b) Wyniki mnożenia elementów tablicy mają zostać umieszczone w tablicy będącej argumentem metody (w tym wariancie metoda ma niczego nie zwracać).

Przykładowo dla tablicy o elementach {1,4,6,8,2} oraz mnożniku 2 program powinien wyświetlić tablicę {2,8,12,16,4}.

Zadanie 5.5.

Napisz program, który wypisze na ekranie znaki w kształcie prostokąta. Program ma prosić użytkownika o podanie rozmiaru prostokąta: długość i szerokość, a następnie znak, którym ma być zapełniony prostokąt. Napisz metodę *Rysuj()*, która wypisze na konsoli prostokąt, przesyłając jako argument długość, szerokość oraz znak wypełnienia. Wywołaj metodę dwa razy, za drugim razem podaj na odwrót argumenty dla długości i szerokości. W wyniku działania programu powinny zostać wyświetlone dwa prostokąty, jeden „leżący" oraz drugi „stojący". Przykładowy przebieg działania programu na rysunku:

```
Podaj dlugość: 6
Podaj szerokość: 3
Podaj znak: *

******
******
******

***
***
***
***
***
***
```

Zadanie 5.6.

Uzupełnij program z zadania 5.4 (dowolny wariant) o metodę przeładowaną przyjmującą tablicę typu *string* oraz mnożnik typu *int*. W tym przypadku metoda ma powielać łańcuch znaków (konkatenować tyle razy, ile wynika z mnożnika). Przykładowo dla tablicy o elementach {"ala", "kot", "dom"} oraz mnożniku 2 program powinien wyświetlić tablicę {"alaala", "kotkot", "domdom"}.

Zadanie 5.7.

Napisz statyczną metodę, która oblicza wyrażenie:

W = (x+1) + (x+2) + (x+3) +.....+ (x+n). W metodzie *Main()* wywołaj funkcję dla x i n (liczb naturalnych) wczytanych z klawiatury.

Zadanie 5.8.

Napisz metodę, która oblicza sumę cyfr liczby naturalnej x. W programie głównym wywołaj funkcję dla x wczytanego z klawiatury. Przykładowo jeśli użytkownik wpisze 125, to metoda powinna zwrócić wartość 8 (1+2+5=8).

Zadanie 5.9.

Wykonaj program znajdujący n-ty wyraz ciągu Fibonacciego według wzoru

$$F_n = \begin{cases} n & dla\ n = 0 \vee n = 1 \\ F_{n-1} + F_{n-2} & dla \quad n > 1 \end{cases}$$

Wykonaj program w dwóch wariantach: w jednym metoda znajdująca wyraz ciągu ma być rekurencyjna, a w drugim ma wykorzystać iteracyjne podejście (z użyciem pętli).

Zadanie 5.10.

Jaki będzie rezultat metody *Oblicz()* wywołanej z parametrem n = 5? Zadanie wykonaj najpierw bez udziału kompilatora, a dopiero później uruchom program i sprawdź otrzymany wynik.

```
static int Oblicz(int n)
{
   if (n <= 1) return (1);
   else return (n + Oblicz(n - 1));
}
```

Wprowadzenie do tworzenia klas – zadania

Zadanie 6.1.

Napisz program, który tworzy klasę *Prostokat*, zawierającą dwie prywatne dane składowe: *dlugosc*, *szerokosc*, dwie prywatne metody: *powierzchnia()*, *obwod()* oraz metodę publiczną – *Prezentuj()* (która wyświetla powierzchnię i obwód prostokąta) i konstruktor inicjalizujący. W metodzie *Main()* zdefiniuj obiekt i uruchom dla niego metodę *Prezentuj()*.

Zadanie 6.2.

Uzupełnij program z poprzedniego zadania o definicje tablicy obiektów dla prostokątów. W metodzie *Main()* wyświetl powierzchnie oraz obwód wszystkich prostokątów w tablicy używając (wewnątrz pętli *foreach*) metody publicznej *Prezentuj()*.

Zadanie 6.3.

Uzupełnij program z poprzedniego zadania o definicję metody statycznej, która podaje powierzchnię największego prostokąta.

Zadanie 6.4.

Zdefiniuj klasę, która pozwoli na rejestrację zużycia energii elektrycznej. Klasa powinna pozwalać na:

- rejestrację początkowego i bieżącego stanu licznika energii,
- uzyskanie danych o początkowym oraz bieżącym stanie licznika,
- obliczanie zużytej energii.

Zadanie 6.5.

Napisz program tworzący klasę *Punkt* do obsługi punktów na płaszczyźnie. Klasa ta ma zawierać: konstruktor, którego argumentami będą współrzędne punktu, metodę składową *Przesun()*, realizującą przesunięcie o zadane wielkości oraz metodę składową *Wyswietl()* wypisującą aktualne współrzędne punktu. Współrzędne punktu mają być zdefiniowane poprzez właściwości.

Zadanie 6.6.

Napisz program (używając klasy *Punkt* zdefiniowanej w poprzednim zadaniu), który przechowuje dane o trzech punktach w tablicy obiektów i sprawdza przy pomocy statycznej metody, czy leżą one na jednej prostej.

Zadanie 6.7.

Zdefiniuj klasę *Odcinek* składającą się z dwóch punktów klasy *Punkt*. W klasie Odcinek zdefiniuj metodę, która obliczy długość odcinka.

Zadanie 6.8.

Zdefiniuj klasę *Prostopadloscian*, która pozwoli na reprezentację danych opisujących długość, szerokość i wysokość prostopadłościanu. W klasie zaimplementuj metody pozwalające na obliczenie objętości prostopadłościanu, oraz porównanie objętości dwóch prostopadłościanów.

Zadanie 6.9.

Wykonaj zadania 6.1 oraz 6.2 z użyciem struktury (zamiast klasy).

Zadanie 6.10.

Napisz program z użyciem struktury *KandydatNaStudia*, która ma posiadać następujące pola: *nazwisko, punktyMatematyka, punktyInformatyka, punktyJezykObcy*. W trzech ostatnich polach mają być zapisane punkty za przedmioty zdawane na maturze (dla uproszczenia uwzględniamy tylko jeden poziom zdawanej matury, np. podstawowy). Jeden punkt to jeden procent (tj. student, który ma 55% z matematyki ma mieć 55 punktów z tego przedmiotu). Struktura ma posiadać metodę obliczającą łączną liczbę punktów kandydata według przelicznika: 0,6 punktów z matematyki + 0,5 punktów z informatyki + 0,2 punktów z języka obcego. W metodzie *Main()* utwórz obiekty dla struktury (jako elementy tablicy) dla kilku kandydatów i pokaż listę kandydatów, zawierającą nazwisko i obok, w tej samej linii, obliczoną łączną liczbę punktów.

Zadanie 6.11.

Napisz z użyciem klas menu programu zawierające kilka opcji dla programów wykonanych w rozdziale 3. Przykładowe opcje: Kalkulator, Wskaźnik BMI, Liczby doskonałe. Po wybraniu przez użytkownika danej opcji program ma wykonać działanie określone dla danego zadania (np. kalkulator z zadania 3.4). Po naciśnięciu klawisza *Esc* należy zakończyć działanie programu[71].