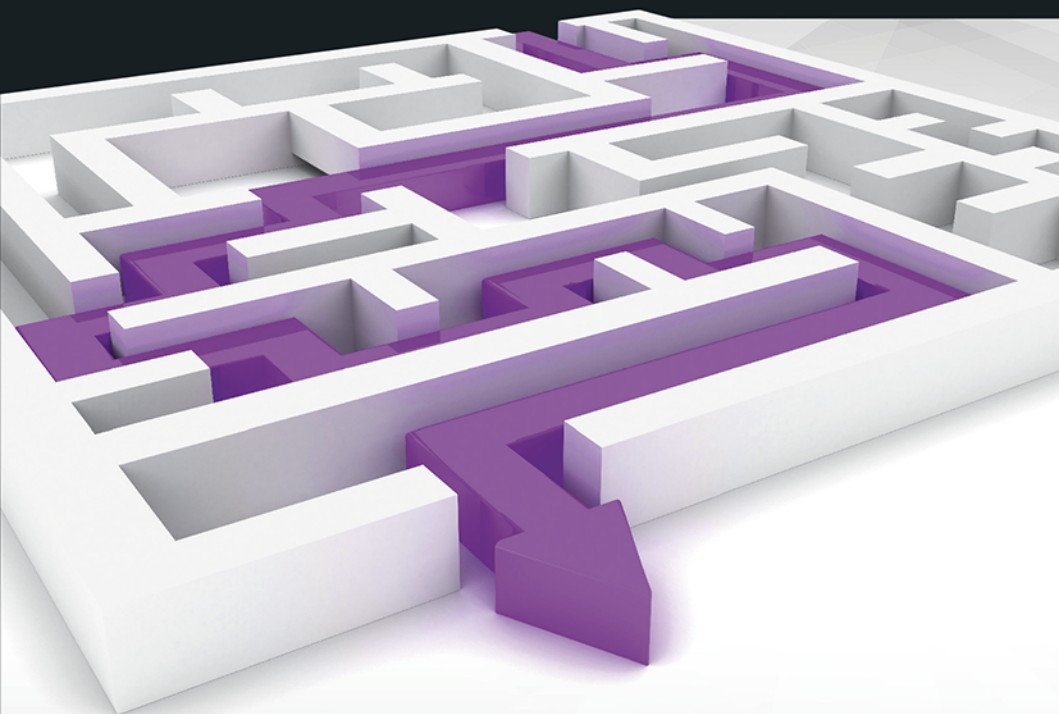


Mirośław J. Kubiak



C#

**Zadania z programowania
z przykładowymi rozwiązaniami**

WYDANIE II

Helion 

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Opieka redakcyjna: Ewelina Burska

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/cshza2>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-4143-2

Copyright © Helion 2018

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

Od autora	5
Rozdział 1. Jak język C# komunikuje się z użytkownikiem?	9
Informacje ogólne	9
Obsługa sytuacji wyjątkowych	19
Rozdział 2. Instrukcje sterujące przebiegiem programu — instrukcje wyboru	23
Instrukcje wyboru	23
Instrukcja if ... else	24
Instrukcja switch ... case	24
Rozdział 3. Instrukcje sterujące przebiegiem programu — instrukcje iteracyjne	37
Instrukcje iteracyjne	37
Pętla for	38
Pętla do ... while	39
Pętla while	39
Rozdział 4. Tablice i kolekcje	75
Tablice	75
Kolekcje	75
Tablice jednowymiarowe	76
Tablice dwuwymiarowe	80
Pętla foreach	98
Działania na macierzach	105
Łącuchy tekstowe	114
Konkatenacja	116

Programowanie uogólnione i klasy generyczne	118
Proste metody generyczne	119
Proste klasy generyczne	121
Listy generyczne	124
Rozdział 5. Elementy programowania obiektowego	127
Informacje ogólne	127
Klasy, pola, metody	128
Rekurencja	140
Klasa Osoba	146
Dziedziczenie	148
Rozdział 6. Pliki tekstowe i pliki o dostępie swobodnym	153
Informacje ogólne	153
Pliki tekstowe	153
Pliki o dostępie swobodnym	167
Serializacja	170
Rozdział 7. Wprowadzenie do współbieżności	173
Informacje ogólne	173
Wprowadzenie do programowania równoległego	174
Wielowątkowość	184
Mój pierwszy wątek	184
Praca z wątkami	188
Priorytety wątków	194
Klasa Task	197
Moje pierwsze zadanie	197
Praca z zadaniami	198
Synchronizacja zadań	200
Polecana literatura	205

Rozdział 2.

Instrukcje sterujące przebiegiem programu — instrukcje wyboru

W tym rozdziale przedstawiłem typowe zadania, wraz z przykładowymi rozwiązaniami, wykorzystujące instrukcje wyboru.

Instrukcje wyboru

Instrukcje sterujące przebiegiem programu są jednym z najważniejszych elementów w każdym języku programowania. Instrukcje te, w połączeniu z wyrażeniami, umożliwiają zapisanie dowolnego algorytmu programu.

Instrukcje sterujące w języku C# można podzielić na:

- ◆ instrukcje wyboru,
- ◆ instrukcje iteracyjne (znane jako pętle),
- ◆ instrukcje skoku.

W niniejszym rozdziale przedstawiam typowe zadania z wykorzystaniem instrukcji wyboru, w rozdziale 3. natomiast zadania z wykorzystaniem instrukcji iteracyjnych.

W języku C# istnieją dwie instrukcje wyboru, które służą do przeprowadzania operacji na podstawie wartości wyrażenia:

- ◆ instrukcja `if ... else`,
- ◆ instrukcja `switch ... case`.

Instrukcja `if ... else`

Instrukcja `if ... else` służy do sprawdzania poprawności wyrażenia warunkowego i — w zależności od tego, czy dany warunek jest prawdziwy, czy nie — pozwala wykonać różne bloki programu.

Jej ogólna postać jest następująca:

```
if (warunek)
{
    ..... // instrukcje do wykonania, kiedy warunek jest prawdziwy
}
else
{
    ..... // instrukcje do wykonania, kiedy warunek jest fałszywy
}
```

Blok `else` jest opcjonalny, a instrukcja warunkowa w wersji skróconej ma postać:

```
if (warunek)
{
    ..... // instrukcje do wykonania, kiedy warunek jest prawdziwy
}
```

Instrukcja `switch ... case`

Instrukcja `switch ... case` pozwala w wygodny i przejrzysty sposób sprawdzić ciąg warunków i wykonać kod w zależności od tego, czy są one prawdziwe, czy fałszywe. Jej ogólna postać jest następująca:

```
switch (wyrażenie)
{
    case wartość_1 : instrukcje_1;
    break;
    case wartość_2 : instrukcje_2;
    break;
    .....
    case wartość_n : instrukcje_n;
```

```
        break;
    default : instrukcje;
}
```

Instrukcja `break` przerywa wykonanie całego bloku `case`. **UWAGA:** jej brak może doprowadzić do uzyskania nieoczekiwanych wyników i pojawienia się błędów w programie.

Zadanie

2.1

Napisz program, który dla trzech liczb, `a`, `b`, `c`, wprowadzonych z klawiatury sprawdza, czy tworzą one trójkę pitagorejską.



W teorii liczb trójka pitagorejska to takie trzy liczby całkowite dodatnie `a`, `b`, `c`, które spełniają równanie Pitagorasa: $a^2 + b^2 = c^2$.

Listing 2.1. Przykładowe rozwiązanie

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Zadanie_21 // Zadanie 2.1
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, b, c;

            Console.WriteLine("Program sprawdza, czy wczytane liczby a,
            ↪b, c to trójka pitagorejska.");
            Console.WriteLine("Podaj a.");
            a = int.Parse(Console.ReadLine());
            Console.WriteLine("Podaj b.");
            b = int.Parse(Console.ReadLine());
            Console.WriteLine("Podaj c.");
            c = int.Parse(Console.ReadLine());

            if ((a*a + b*b) == c*c)
            {
                Console.Write("Liczby ");
                Console.Write("a = " + a + ", ");
                Console.Write("b = " + b + ", ");
                Console.Write("c = " + c);
            }
        }
    }
}
```

```

        Console.WriteLine(" są trójką pitagorejską.");
    }
    else
    {
        Console.Write("Liczby ");
        Console.Write("a = " + a + ", ");
        Console.Write("b = " + b + ", ");
        Console.Write("c = " + c);
        Console.WriteLine(" nie są trójką pitagorejską.");
    }

    Console.Write("Naciśnij dowolny klawisz.");
    Console.ReadKey(true);
}
}
}
}

```

Sprawdzenie twierdzenia Pitagorasa dla wczytanych liczb a, b, c zostało zawarte w następujących liniach kodu:

```

if ((a*a + b*b) == c*c)
{
    Console.Write("Liczby ");
    Console.Write("a = " + a + ", ");
    Console.Write("b = " + b + ", ");
    Console.Write("c = " + c);
    Console.WriteLine(" są trójką pitagorejską.");
}
else
{
    Console.Write("Liczby ");
    Console.Write("a = " + a + ", ");
    Console.Write("b = " + b + ", ");
    Console.Write("c = " + c);
    Console.WriteLine(" nie są trójką pitagorejską.");
}
}

```

Łatwo sprawdzić, że liczby $a = 3$, $b = 4$ i $c = 5$ tworzą trójkę pitagorejską (spełniają twierdzenie Pitagorasa) i na ekranie pojawi się komunikat: *Liczby ... są trójką pitagorejską*, natomiast liczby $a = 1$, $b = 2$ i $c = 3$ nie tworzą trójki pitagorejskiej (nie spełniają twierdzenia Pitagorasa) i na ekranie pojawi się komunikat: *Liczby ... nie są trójką pitagorejską*.

Rezultat działania programu dla $a = 9$, $b = 12$ i $c = 15$ można zobaczyć na rysunku 2.1.

Rysunek 2.1.

Efekt działania programu
Zadanie 2.1

```
Program sprawdza, czy wczytane liczby a, b, c to trójka pitagorejska.  
Podaj a.  
9  
Podaj b.  
12  
Podaj c.  
15  
Liczby a = 9, b = 12, c = 15 są trójką pitagorejską.
```

Zadanie**2.2**

Napisz program, który z wykorzystaniem instrukcji `if` oblicza pierwiastki równania kwadratowego $ax^2 + bx + c = 0$, w którym zmienne `a`, `b`, `c` to liczby rzeczywiste wprowadzane z klawiatury. Wszystkie zmienne wyświetlamy na ekranie z dokładnością do dwóch miejsc po przecinku.

Listing 2.2. Przykładowe rozwiązanie

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace Zadanie_22 // Zadanie 2.2  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            double a, b, c, delta, x1, x2;  
  
            Console.WriteLine("Program oblicza pierwiastki równania  
↳ax^2 + bx + c = 0.");  
            Console.WriteLine("Podaj a.");  
            a = double.Parse(Console.ReadLine());  
  
            if (a == 0)  
            {  
                Console.WriteLine("Niedozwolona wartość współczynnika a.");  
            }  
            else  
            {  
                Console.WriteLine("Podaj b.");  
                b = double.Parse(Console.ReadLine());  
                Console.WriteLine("Podaj c.");  
                c = double.Parse(Console.ReadLine());
```

```
delta = b*b-4*a*c;

if (delta < 0)
{
    Console.WriteLine();
    Console.Write("Dla ");
    Console.Write("a = {0}, ", a);
    Console.Write("b = {0}, ", b);
    Console.Write("c = {0} ", c);
    Console.Write("brak pierwiastków rzeczywistych.");
}
else
{
    if (delta == 0)
    {
        x1 = -b/(2*a);
        Console.WriteLine();
        Console.Write("Dla ");
        Console.Write("a = {0}, ", a);
        Console.Write("b = {0}, ", b);
        Console.Write("c = {0} ", c);
        Console.WriteLine("trójmian ma jeden pierwiastek
↳podwójny x1 = {0:##.##}.", x1);
    }
    else
    {
        x1 = (-b-Math.Sqrt(delta))/(2*a);
        x2 = (-b+Math.Sqrt(delta))/(2*a);
        Console.WriteLine();
        Console.Write("Dla ");
        Console.Write("a = {0}, ", a);
        Console.Write("b = {0}, ", b);
        Console.Write("c = {0} ", c);
        Console.WriteLine("trójmian ma dwa pierwiastki:
↳x1 = {0:##.##}, x2 = {1:##.##}.", x1, x2);
    }
}

Console.Write("Naciśnij dowolny klawisz.");
Console.ReadKey(true);
}
}
```

W pierwszej części programu sprawdzamy, czy wartość współczynnika a jest równa zero. Ilustrują to następujące linijki kodu:

```
if (a == 0)
{
    Console.WriteLine("Niedozwolona wartość współczynnika a.");
}
else
{
    .....
}
```

Jeśli wartość współczynnika $a = 0$, to zostanie wyświetlony komunikat: *Niedozwolona wartość współczynnika a* i program zostanie zakończony. Dla a różnego od zera program będzie oczekiwał na wprowadzenie wartości b i c . Po ich wprowadzeniu zostanie obliczona Δ według wzoru:

$$\Delta = b^2 - 4ac;$$

Jeśli $\Delta < 0$, to zostanie wyświetlony komunikat: *...brak pierwiastków rzeczywistych*.

Dla $\Delta = 0$ równanie kwadratowe ma jeden pierwiastek podwójny, który obliczymy ze wzoru:

$$x_1 = -b/(2a);$$

Dla $\Delta > 0$ równanie kwadratowe ma dwa pierwiastki, które obliczymy ze wzorów:

$$x_1 = (-b - \text{Math.Sqrt}(\Delta))/(2a);$$

$$x_2 = (-b + \text{Math.Sqrt}(\Delta))/(2a);$$

Dla na przykład $a = 1$, $b = 5$ i $c = 4$ wartości pierwiastków równania wynoszą odpowiednio: $x_1 = -4$ i $x_2 = -1$.

Dla na przykład $a = 1$, $b = 4$ i $c = 4$ trójmian ma jeden pierwiastek podwójny: $x_1 = -2$.

Dla na przykład $a = 1$, $b = 2$ i $c = 3$ trójmian nie ma pierwiastków rzeczywistych.

Rezultat działania programu dla $a = 1$, $b = 5$ i $c = 4$ można zobaczyć na rysunku 2.2.

Zadanie

2.3

Napisz program, który z wykorzystaniem instrukcji `switch` oblicza pierwiastki równania kwadratowego $ax^2 + bx + c = 0$, w którym zmienne a , b , c to liczby rzeczywiste wprowadzane z klawiatury. Wszystkie zmienne wyświetlamy z dokładnością do dwóch miejsc po przecinku.

Rysunek 2.2.

Efekt działania programu
Zadanie 2.2

Program oblicza pierwiastki równania $ax^2 + bx + c = 0$.

Podaj a.

1

Podaj b.

5

Podaj c.

4

Dla $a = 1$, $b = 5$, $c = 4$ trójmian ma dwa pierwiastki: $x_1 = -4$, $x_2 = -1$.



Wskazówka

Należy wprowadzić do programu zmienną pomocniczą `liczba_pierwiastkow`.

Listing 2.3. Przykładowe rozwiązanie

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Zadanie_23 // Zadanie 2.3
{
    class Program
    {
        static void Main(string[] args)
        {
            double a, b, c, delta, x1, x2;
            byte liczba_pierwiastkow = 0;

            Console.WriteLine("Program oblicza pierwiastki równania
↳  $ax^2 + bx + c = 0$ .");
            Console.WriteLine("Podaj a.");
            a = double.Parse(Console.ReadLine());

            if (a == 0)
            {
                Console.WriteLine("Niedozwolona wartość współczynnika a.");
            }
        }
    }
}
```

```
else
{
    Console.WriteLine("Podaj b.");
    b = double.Parse(Console.ReadLine());
    Console.WriteLine("Podaj c.");
    c = double.Parse(Console.ReadLine());

    delta = b*b-4*a*c;

    if (delta < 0) liczba_pierwiastkow = 0;
    if (delta == 0) liczba_pierwiastkow = 1;
    if (delta > 0) liczba_pierwiastkow = 2;

    switch (liczba_pierwiastkow)
    {
        case 0:
            {
                Console.WriteLine();
                Console.Write("Dla ");
                Console.Write("a = {0}, ", a);
                Console.Write("b = {0}, ", b);
                Console.Write("c = {0} ", c);
                Console.Write("brak pierwiastków
↳ rzeczywistych.");
            }
            break;
        case 1:
            {
                x1 = -b/(2*a);
                Console.WriteLine();
                Console.Write("Dla ");
                Console.Write("a = {0}, ", a);
                Console.Write("b = {0}, ", b);
                Console.Write("c = {0} ", c);
                Console.WriteLine("trójmian ma jeden
↳ pierwiastek podwójny x1 = {0: ##.##}.", x1);
            }
            break;
        case 2:
            {
                x1 = (-b-Math.Sqrt(delta))/(2*a);
                x2 = (-b+Math.Sqrt(delta))/(2*a);
                Console.WriteLine();
                Console.Write("Dla ");
                Console.Write("a = {0}, ", a);
                Console.Write("b = {0}, ", b);
                Console.Write("c = {0} ", c);
                Console.Write("trójmian ma dwa pierwiastki
↳ x1 = {0: ##.##} i ", x1);
                Console.WriteLine("x2 = {0: ##.##}.", x2);
            }
    }
}
```

```

        break;
    }
}
Console.WriteLine("Naciśnij dowolny klawisz.");
Console.ReadKey(true);
}
}
}

```

Zmienna pomocnicza `liczba_pierwiastkow` przyjmuje trzy wartości w zależności od znaku zmiennej `delta`. Ilustrują to następujące linijki kodu:

```

if (delta < 0) liczba_pierwiastkow = 0;
if (delta == 0) liczba_pierwiastkow = 1;
if (delta > 0) liczba_pierwiastkow = 2;

```

Rezultat działania programu dla $a = 1$, $b = 4$ i $c = 4$ można zobaczyć na rysunku 2.3.

Rysunek 2.3.

Efekt działania programu
Zadanie 2.3

Program oblicza pierwiastki równania $ax^2 + bx + c = 0$.

Podaj a.

1

Podaj b.

4

Podaj c.

4

Dla $a = 1$, $b = 4$, $c = 4$ trójmian ma jeden pierwiastek podwójny $x_1 = -2$.

Zadanie

2.4

Napisz program, który oblicza wartość niewiadomej x z równania $ax + b = c$. Wartości a , b , c należą do zbioru liczb rzeczywistych i są wprowadzane z klawiatury. Dodatkowo należy zabezpieczyć program na wypadek sytuacji, kiedy wprowadzona wartość $a = 0$. Wszystkie zmienne wyświetlamy z dokładnością do dwóch miejsc po przecinku.

Listing 2.4. Przykładowe rozwiązanie

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Zadanie_24 // Zadanie 2.4
{

```

```
class Program
{
    static void Main(string[] args)
    {
        double a, b, c, x;

        Console.WriteLine("Program oblicza wartość x z równania
↳ liniowego  $ax + b = 0$ .");
        Console.WriteLine("Podaj a.");
        a = double.Parse(Console.ReadLine());

        if (a == 0)
        {
            Console.WriteLine("Niedozwolona wartość współczynnika a.");
        }
        else
        {
            Console.WriteLine("Podaj b.");
            b = double.Parse(Console.ReadLine());
            Console.WriteLine("Podaj c.");
            c = double.Parse(Console.ReadLine());

            x = (c-b)/a;

            Console.WriteLine();
            Console.Write("Dla ");
            Console.Write("a = {0:##.##}, ", a);
            Console.Write("b = {0:##.##}, ", b);
            Console.Write("c = {0:##.##} ", c);
            Console.WriteLine("wartość x = {0:##.##}.", x);
        }
        Console.Write("Naciśnij dowolny klawisz.");
        Console.ReadKey(true);
    }
}
```

Rezultat działania programu można zobaczyć na rysunku 2.4.

Rysunek 2.4.

*Efekt działania
programu
Zadanie 2.4*

```
Program oblicza wartość x z równania liniowego  $ax + b = 0$ .
Podaj a.
1
Podaj b.
6
Podaj c.
2

Dla a = 1, b = 6, c = 2 wartość x = -4.
```

Zadanie**2.5**

Napisz program, w którym użytkownik zgaduje całkowitą liczbę losową z przedziału od 0 do 9 generowaną przez komputer.

**Wskazówka**

W języku C# liczby pseudolosowe generujemy za pomocą klasy:

```
Random r = new Random(); .
```

Listing 2.5. Przykładowe rozwiązanie

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Zadanie_25 // Zadanie 2.5
{
    class Program
    {
        static void Main(string[] args)
        {
            Random r = new Random();
            double losuj_liczbe, zgadnij_liczbe;

            Console.WriteLine("Program losuje liczbę od 0 do 9. Zgadnij
↳jā.");

            losuj_liczbe = Math.Round(10*(r.NextDouble()));
            zgadnij_liczbe = double.Parse(Console.ReadLine());

            if (zgadnij_liczbe == losuj_liczbe)
            {
                Console.WriteLine("Gratulacje! Zgadłeś liczbę!");
            }
            else
            {
                Console.WriteLine("Bardzo mi przykro, ale wylosowana
↳liczba to {0}.", losuj_liczbe);
            }

            Console.Write("Naciśnij dowolny klawisz.");
            Console.ReadKey(true);
        }
    }
}
```


Funkcja `Round()` w poniższej linijce kodu:

```
losuj_liczbe = Math.Round(10*(r.NextDouble()));
```

umożliwia zaokrąglenie liczby zmiennoprzecinkowej do liczby całkowitej.

Rezultat działania programu można zobaczyć na rysunku 2.5.

Rysunek 2.5.

Efekt działania

programu

Zadanie 2.5

Program losuje liczbę od 0 do 9. Zgadnij ją.

9

Gratulacje! Zgadłeś liczbę!

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

Wśród wielu obiektowych języków programowania C# zajmuje miejsce szczególne. Ma przejrzystą strukturę, jasne zasady i jest wciąż rozwijany. Nie znajdziesz wygodniejszego narzędzia programistycznego dla platformy .NET. C# sprawdza się w najróżniejszych projektach, a jego zalety sprawiają, że jest niezwykle popularny. Jeśli znasz podstawy tego języka, ale nie czujesz się w nim zbyt pewnie i nie zawsze umiesz przewidzieć, jak zachowa się Twój program, czas na profesjonalne szkolenie! Zadania z tego zbioru pomogą Ci opanować bardziej zaawansowane zagadnienia, a także zrozumieć, jak to wszystko działa.

W zaktualizowanym i rozszerzonym wydaniu cenionej książki Mirosława J. Kubiaka znajdziesz informacje, zalecenia i konkretne ćwiczenia programistyczne z różnych obszarów. Nauczysz się efektywnie i poprawnie stosować instrukcje sterujące, używać tablic i tworzyć kolekcje do przechowywania swoich obiektów. Dowiesz się, do czego służą klasy, pola, metody oraz dlaczego warto używać rekurencji. Sprawdzisz, do czego przydaje się dziedziczenie, i popracujesz na plikach tekstowych. Po ukończeniu wszystkich zadań ze zbioru będziesz mógł już swobodnie programować w C#!

- Sposoby komunikowania się języka C# z użytkownikiem
- Instrukcje sterujące przebiegiem programu (wyboru oraz iteracyjne)
- Tablice i kolekcje
- Elementy programowania obiektowego
- Pliki tekstowe i pliki o dostępie swobodnym
- Wprowadzenie do współbieżności

Programowanie w C# — szybkie, bezpieczne, eleganckie!

 Helion	<i>Sprawdź nasze szkolenia!</i> SZKOLENIA  AKADEMIA IT & BUSINESS WWW.SZKOLENIA.HELION.PL	KOD KORZYŚCI <i>Sięgnij po więcej!</i> ► 
 helion.pl		ISBN 978-83-283-4143-2  9 788328 341432
 HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl		INFORMATYKA W NAJLEPSZYM WYDANIU Cena: 34,90 zł