Materiały do przedmiotu:

Podstawy programowania

Laboratorium nr 2

Autor:

Patrycja Miazek

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Instrukcje sterujące. Operacje wejścia/wyjścia

Wprowadzenie, tematyka zajęć

Wprowadzenie

Instrukcje sterujące stanowią kluczowy element każdego programu w C#, ponieważ decydują o tym, w jaki sposób wykonywany jest kod. Bez nich program działałby sekwencyjnie – linijka po linijce. Dzięki konstrukcjom sterującym możemy podejmować decyzje, powtarzać operacje i reagować na różne sytuacje w trakcie działania aplikacji.

Instrukcje warunkowe

Najczęściej używaną instrukcją decyzyjną w C# jest **if**, która pozwala wykonać fragment kodu tylko wtedy, gdy spełniony zostanie określony warunek logiczny.

```
int liczba = 12;

if (liczba > 10)
{
    Console.WriteLine("Liczba jest większa niż 10");
}
```

Aby obsłużyć alternatywny przypadek, stosuje się **if** wraz z **else**:

```
int temperatura = 18;

if (temperatura >= 20)
{
    Console.WriteLine("Ciepło");
}
else
{
    Console.WriteLine("Chłodno");
}
```

Jeśli potrzebujemy rozpatrzyć więcej niż dwie możliwości, używamy rozszerzonej formy **if – else if – else**:

```
int ocena = 2;
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
if (ocena == 5)
{
    Console.WriteLine("Bardzo dobrze");
}
else if (ocena == 4)
{
    Console.WriteLine("Dobrze");
}
else if (ocena == 3)
{
    Console.WriteLine("Dostatecznie");
}
else
{
    Console.WriteLine("Nie zaliczono");
}
```

Instrukcja switch

W przypadkach, gdy sprawdzamy wartość tej samej zmiennej wobec wielu możliwych opcji, znacznie wygodniejszym rozwiązaniem jest instrukcja **switch**. Zwiększa ona czytelność kodu i eliminuje nadmierne zagnieżdżanie.

```
int miesiac = 4;
switch (miesiac)
{
    case 1:
        Console.WriteLine("Styczeń");
        break;
    case 2:
        Console.WriteLine("Luty");
        break;
    case 3:
        Console.WriteLine("Marzec");
        break;
    case 4:
        Console.WriteLine("Kwiecień");
        break;
    default:
        Console.WriteLine("Inny miesiąc");
        break;
}
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

W nowoczesnym C# dostępna jest również skrócona postać tzw. **switch expression**, dzięki której kod jest bardziej zwięzły:

```
int miesiac = 7;

string nazwa = miesiac switch
{
    1 => "Styczeń",
    2 => "Luty",
    3 => "Marzec",
    7 => "Lipiec",
    _ => "Nieznany miesiąc"
};

Console.WriteLine(nazwa);
```

switch warto stosować, gdy analizujemy różne przypadki jednej zmiennej. W sytuacjach, gdzie potrzebne są złożone warunki logiczne (np. porównania zakresów), lepiej sprawdza się if.

Pętle w C#

Drugą grupę konstrukcji sterujących stanowią pętle, które umożliwiają wielokrotne wykonywanie tego samego fragmentu kodu.

Pętla while

Instrukcja **while** wykonuje się dopóki spełniony jest warunek logiczny:

```
int licznik = 0;

while (licznik < 4)
{
    Console.WriteLine($"Licznik: {licznik}");
    licznik++;
}
```

Pętla for

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Pętla **for** jest użyteczna, gdy z góry wiemy, ile razy chcemy wykonać określoną operację. Składa się z trzech części: inicjalizacji, warunku i modyfikacji zmiennej sterującej.

```
for (inicjalizacja; warunek; zmiana)
{
    // kod wykonywany w każdej iteracji
}
```

Inicjalizacja uruchamia się tylko raz, na początku pętli. Tworzy zmienną licznikową (np. int i = 1;).
Warunek sprawdzany przed każdą iteracją. Jeśli jest true, pętla się wykonuje. Jeśli false, pętla się kończy.
Zmiana wykonuje się po każdej iteracji. Najczęściej zwiększa licznik (i++). Np.

```
for (int i = 1; i <= 5; i++)
{
    Console.WriteLine($"Powtórzenie nr {i}");
}
```

Pętla foreach

Do przetwarzania elementów kolekcji (np. tablicy, listy) używa się **foreach**. Dzięki niej nie musimy odwoływać się do indeksów.

```
string[] zwierzeta = { "Kot", "Pies", "Zebra" };

foreach (string zwierze in zwierzeta)
{
    Console.WriteLine($"Zwierzę: {zwierze}");
}
```

Dla kolekcji typu List<T> można wykorzystać metodę **ForEach()** z wyrażeniem lambda:

```
List<string> miasta = new List<string> { "Warszawa", "Kraków", "Poznań" };
miasta.ForEach(miasto => Console.WriteLine(miasto));
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Pętla nieskończona

Czasami chcemy, aby pętla działała bez końca, aż do momentu wystąpienia konkretnego warunku. Wtedy stosuje się **while(true)** wraz z break.

```
while (true)
{
    Console.Write("Podaj hasło (exit kończy): ");
    string input = Console.ReadLine();
    if (input == "exit")
        break;
}
```

Instrukcje break, continue, return i goto
W pętlach można kontrolować przepływ programu.
break - przerywa działanie pętli,
continue - przechodzi do kolejnej iteracji,
return - kończy metodę,
goto - przeskakuje do oznaczonej etykiety (stosowane bardzo rzadko).

```
for (int x = 0; x < 8; x++)
{
    if (x == 2)
        continue; // pomija 2
    if (x == 6)
        break; // zatrzymuje pętlę
    Console.WriteLine(x);
}
```

Zagnieżdżone pętle i warunki umożliwiają tworzenie bardziej złożonych operacji, np. przeglądanie tablic dwuwymiarowych:

```
for (int a = 0; a < 2; a++)
{
    for (int b = 0; b < 2; b++)
    {
        Console.WriteLine($"Pozycja: [{a}, {b}]");
    }
}
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Operacje wejścia i wyjścia

Operacje wejścia/wyjścia (ang. I/O) pozwalają programowi komunikować się z użytkownikiem oraz danymi zapisanymi w plikach.

Klasa Console

Najprostszy sposób interakcji w aplikacjach konsolowych to klasa Console.

Console.Write() - wypisuje dane bez przejścia do nowej linii,
Console.WriteLine() - wypisuje dane i przechodzi do nowej linii.

```
Console.Write("Podaj imię: ");
string imie = Console.ReadLine();
Console.WriteLine($"Witaj, {imie}!");
```

Jeśli chcemy wczytać liczbę, trzeba przekonwertować tekst na typ liczbowy:

```
Console.Write("Podaj liczbę: ");
string dane = Console.ReadLine();
int liczba = int.Parse(dane);
Console.WriteLine($"Liczba razy dwa: {liczba * 2}");
```

Bezpieczniejszą metodą jest **TryParse(),** która zapobiega błędom konwersji:

```
Console.Write("Wpisz wiek: ");
string tekst = Console.ReadLine();

if (int.TryParse(tekst, out int wiek))
{
    Console.WriteLine($"Za 10 lat będziesz mieć {wiek + 10} lat.");
}
else
{
    Console.WriteLine("Nieprawidłowy format liczby!");
}
```

Można też reagować na pojedyncze klawisze dzięki metodzie **ReadKey():**

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
Console.WriteLine("Naciśnij dowolny klawisz, aby kontynuować...");
ConsoleKeyInfo klawisz = Console.ReadKey(true);
Console.WriteLine($"Wcisnąłeś: {klawisz.Key}");
```

Operacje na plikach (System.IO)

W przestrzeni nazw System.IO znajdują się narzędzia umożliwiające zapis i odczyt danych z plików.

Odczyt i zapis tekstu

```
File.WriteAllText("info.txt", "To jest testowy zapis do pliku.");
string tekstPliku = File.ReadAllText("info.txt");
Console.WriteLine(tekstPliku);
```

Dopisywanie danych

```
File.AppendAllText("info.txt", "\nNowa linia została dodana.");
```

Odczyt i zapis linia po linii

```
using (StreamWriter writer = new StreamWriter("lista.txt"))
{
    writer.WriteLine("Linia 1");
    writer.WriteLine("Linia 2");
}
```

```
using (StreamReader reader = new StreamReader("lista.txt"))
{
    string wiersz;
    while ((wiersz = reader.ReadLine()) != null)
    {
        Console.WriteLine(wiersz);
    }
}
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Sprawdzanie istnienia pliku

```
if (File.Exists("info.txt"))
    Console.WriteLine("Plik istnieje");
else
    Console.WriteLine("Plik nie istnieje");
```

Dane binarne

```
using (BinaryWriter bw = new BinaryWriter(File.Open("dane.bin",
FileMode.Create)))
{
    bw.Write(42);
    bw.Write("Przykładowy tekst");
}

using (BinaryReader br = new BinaryReader(File.Open("dane.bin",
FileMode.Open)))
{
    int liczba = br.ReadInt32();
    string tekst = br.ReadString();
    Console.WriteLine($"{liczba} - {tekst}");
}
```

Operacje asynchroniczne

C# pozwala na asynchroniczny zapis i odczyt plików, co nie blokuje działania programu:

```
using (StreamWriter writer = new StreamWriter("async.txt", true))
{
    await writer.WriteLineAsync("Zapis asynchroniczny");
}

using (StreamReader reader = new StreamReader("async.txt"))
{
    string linia;
    while ((linia = await reader.ReadLineAsync()) != null)
    {
        Console.WriteLine(linia);
    }
}
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Obsługa wyjątków

Podczas pracy z plikami warto zabezpieczyć kod blokiem try-catch-finally, aby uniknąć awarii programu w przypadku błędu:

```
StreamReader reader = null;

try
{
    reader = new StreamReader("brak.txt");
    Console.WriteLine(reader.ReadToEnd());
}
catch (Exception ex)
{
    Console.WriteLine("Błąd podczas odczytu pliku: " + ex.Message);
}
finally
{
    reader?.Close();
}
```

Cel ćwiczenia/laboratorium

Celem zajęć jest wprowadzenie studentów w zagadnienia związane z podstawowymi instrukcjami sterującymi oraz operacjami wejścia i wyjścia w języku C#. Uczestnicy poznają zasady działania instrukcji warunkowych, pętli i innych elementów kontrolujących przebieg programu. Nauczą się także wczytywać dane od użytkownika, przetwarzać je i prezentować wyniki w czytelny sposób. Zajęcia mają na celu rozwinięcie praktycznych umiejętności programistycznych, zrozumienie przepływu danych w programie oraz tworzenie prostych aplikacji interaktywnych. Stanowią one przygotowanie do dalszej nauki i pracy z bardziej złożonymi zagadnieniami programowania w C#.

Instrukcja laboratoryjna/ćwiczeniowa

Przykład 2.1. Sprawdzenie, czy liczba jest wielokrotnością 3

Opis problemu:

Program sprawdza, czy podana liczba jest podzielna przez 3. Ćwiczenie uczy użycia operatora modulo i instrukcji warunkowej.

Przykładowe rozwiązanie:

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
Console.Write("Podaj liczbę: ");
int liczba = int.Parse(Console.ReadLine());
if (liczba % 3 == 0)
    Console.WriteLine("Liczba jest wielokrotnością 3.");
else
    Console.WriteLine("Liczba nie jest wielokrotnością 3.");
```

Modyfikacje do samodzielnego testowania:

1. Sprawdź podzielność przez 4 lub 6.
2. Wyświetl komunikat, jeśli liczba wynosi 0.

Przykład 2.2. Ocena z testu

Opis problemu:

Program prosi użytkownika o wynik procentowy i wyświetla ocenę na podstawie przedziału.

Przykładowe rozwiązanie:

```
Console.Write("Podaj wynik testu w %: ");
int wynik = int.Parse(Console.ReadLine());
if (wynik < 50)
    Console.WriteLine("Ocena: niedostateczna");
else if (wynik < 70)
    Console.WriteLine("Ocena: dostateczna");
else if (wynik < 90)
    Console.WriteLine("Ocena: dobra");
else
    Console.WriteLine("Ocena: bardzo dobra");
```

Modyfikacje do samodzielnego testowania:

1. Dodaj ocenę „celującą” dla 100%.
2. Sprawdź zachowanie programu przy wartościach >100.

Przykład 2.3. Sprawdzenie rodzaju znaku

Opis problemu:

Program określa, czy podany znak to cyfra, litera, czy inny symbol.

Przykładowe rozwiązanie:

```
Console.Write("Podaj znak: ");
char znak = char.Parse(Console.ReadLine());
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
if (char.IsDigit(znak))
    Console.WriteLine("To cyfra.");
else if (char.IsLetter(znak))
    Console.WriteLine("To litera.");
else
    Console.WriteLine("To inny symbol.");
```

Modyfikacje do samodzielnego testowania:

1. Dodaj rozróżnienie na duże i małe litery.
2. Sprawdź, co się stanie przy wprowadzeniu spacji.

Przykład 2.4. Obliczanie silni liczby

Opis problemu:

Program oblicza silnię liczby n przy użyciu pętli for.

```
Console.Write("Podaj liczbę całkowitą: ");
int n = int.Parse(Console.ReadLine());
long silnia = 1;
for (int i = 1; i <= n; i++)
    silnia *= i;
Console.WriteLine("Silnia liczby " + n + " wynosi " + silnia);
```

Modyfikacje:

1. Obsłuż przypadek n = 0.
2. Wypisz pośrednie wyniki (1!, 2!, 3! …).

Przykład 2.5. Kalkulator średniej dwóch liczb

Opis problemu:

Program oblicza średnią arytmetyczną z dwóch liczb.

Przykładowe rozwiązanie:

```
Console.Write("Podaj pierwszą liczbę: ");
double a = double.Parse(Console.ReadLine());
Console.Write("Podaj drugą liczbę: ");
double b = double.Parse(Console.ReadLine());

double srednia = (a + b) / 2;
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
Console.WriteLine("Średnia: " + srednia);
```

Modyfikacje do samodzielnego testowania:

1. Dodaj trzecią liczbę.
2. Zaokrąglij wynik do dwóch miejsc po przecinku.

Przykład 2.6. Liczba do dnia tygodnia

Opis problemu:

Program zamienia liczbę od 1 do 7 na nazwę dnia tygodnia.

```
Console.Write("Podaj numer dnia (1-7): ");
int dzien = int.Parse(Console.ReadLine());
switch (dzien)
{
    case 1: Console.WriteLine("Poniedziałek"); break;
    case 2: Console.WriteLine("Wtorek"); break;
    case 3: Console.WriteLine("Środa"); break;
    case 4: Console.WriteLine("Czwartek"); break;
    case 5: Console.WriteLine("Piątek"); break;
    case 6: Console.WriteLine("Sobota"); break;
    case 7: Console.WriteLine("Niedziela"); break;
    default: Console.WriteLine("Niepoprawny numer dnia."); break;
}
```

Modyfikacje do samodzielnego testowania:

1. Dodaj język angielski.
2. Dodaj obsługę liczb spoza zakresu.

Przykład 2.7. Najmniejsza z trzech liczb

Opis problemu:

Program wskazuje najmniejszą z trzech podanych liczb.

Przykładowe rozwiązanie:

```
Console.Write("Podaj pierwszą liczbę: ");
int a = int.Parse(Console.ReadLine());
Console.Write("Podaj drugą liczbę: ");
int b = int.Parse(Console.ReadLine());
Console.Write("Podaj trzecią liczbę: ");
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
int c = int.Parse(Console.ReadLine());

int min = a;
if (b < min) min = b;
if (c < min) min = c;
Console.WriteLine("Najmniejsza liczba: " + min);
```

Modyfikacje do samodzielnego testowania:

1. Wyświetl również największą liczbę.
2. Sprawdź, czy wszystkie liczby są równe.

Przykład 2.8. Obliczanie pola prostokąta

Opis problemu:

Program oblicza pole prostokąta na podstawie boków.

Przykładowe rozwiązanie:

```
Console.Write("Podaj długość boku A: ");
double a = double.Parse(Console.ReadLine());
Console.Write("Podaj długość boku B: ");
double b = double.Parse(Console.ReadLine());

Console.WriteLine("Pole prostokąta: " + (a * b));
```

Modyfikacje do samodzielnego testowania:

1. Dodaj obwód prostokąta.
2. Sprawdź przypadek, gdy bok = 0.

Przykład 2.9. Dzielenie i reszta

Opis problemu:

Program dzieli dwie liczby całkowite i pokazuje wynik oraz resztę.

Przykładowe rozwiązanie:

```
Console.Write("Podaj dzielną: ");
int a = int.Parse(Console.ReadLine());
Console.Write("Podaj dzielnik: ");
int b = int.Parse(Console.ReadLine());

if (b != 0)
{
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
    Console.WriteLine("Wynik: " + (a / b));
    Console.WriteLine("Reszta: " + (a % b));
}
else
{
    Console.WriteLine("Nie można dzielić przez zero!");
}
```

Modyfikacje do samodzielnego testowania:

1. Dodaj obsługę liczb ujemnych.
2. Zapisz wynik jako double.

Przykład 2.10. Zmiana znaku temperatury

Opis problemu:

```
Program odwraca znak podanej temperatury (np. z -5 na +5).
Przykładowe rozwiązanie:
Console.Write("Podaj temperaturę: ");
int temp = int.Parse(Console.ReadLine());
Console.WriteLine("Temperatura o przeciwnym znaku: " + (-temp));
```

Modyfikacje do samodzielnego testowania:

1. Dodaj jednostkę °C.
2. Sprawdź, czy temperatura = 0.

Przykład 2.11. Operacje na trzech liczbach

Opis problemu:

Program oblicza sumę, średnią i iloczyn trzech liczb.

Przykładowe rozwiązanie:

```
Console.Write("Podaj trzy liczby oddzielone spacjami: ");
string[] dane = Console.ReadLine().Split(' ');
double a = double.Parse(dane[0]);
double b = double.Parse(dane[1]);
double c = double.Parse(dane[2]);

Console.WriteLine("Suma: " + (a + b + c));
Console.WriteLine("Średnia: " + ((a + b + c) / 3));
Console.WriteLine("Iloczyn: " + (a * b * c));
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Modyfikacje do samodzielnego testowania:

1. Dodaj największą z liczb.
2. Wypisz różnicę między największą a najmniejszą.

Przykład 2.12. Sprawdzenie liczby ułamkowej

Opis problemu:

Program sprawdza, czy liczba rzeczywista jest dodatnia, ujemna czy zerowa.

Przykładowe rozwiązanie:

```
Console.Write("Podaj liczbę: ");
double x = double.Parse(Console.ReadLine());

if (x > 0)
    Console.WriteLine("Dodatnia");
else if (x < 0)
    Console.WriteLine("Ujemna");
else
    Console.WriteLine("Zero");
```

Modyfikacje do samodzielnego testowania:

1. Dodaj komunikat dla wartości mniejszych niż 1.
2. Sprawdź, czy liczba ma część ułamkową.

Przykład 2.13. Potęgowanie liczby

Opis problemu:

Program oblicza potęgę liczby.

Przykładowe rozwiązanie:

```
Console.Write("Podaj liczbę: ");
double podstawa = double.Parse(Console.ReadLine());
Console.Write("Podaj wykładnik: ");
int wykladnik = int.Parse(Console.ReadLine());

Console.WriteLine("Wynik: " + Math.Pow(podstawa, wykladnik));
```

Modyfikacje do samodzielnego testowania:

1. Dodaj pierwiastkowanie (np. dla wykładnika 0.5).

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

2. Obsłuż liczby ujemne.

Przykład 2.14. Powitanie zależne od dnia tygodnia

Opis problemu:

Program wita użytkownika inaczej w zależności od dnia tygodnia.

Przykładowe rozwiązanie:

```
Console.Write("Podaj dzień tygodnia (1–7): ");
int dzien = int.Parse(Console.ReadLine());

if (dzien <= 5)
    Console.WriteLine("Miłego dnia w pracy!");
else
    Console.WriteLine("Udanych dni wolnych!");
```

Modyfikacje do samodzielnego testowania:

1. Dodaj nazwę dnia.
2. Sprawdź przypadek niepoprawnej liczby.

Przykład 2.15. Sprawdzenie liczby parzystej i dodatniej

Opis problemu:

Program sprawdza dwa warunki jednocześnie: parzystość i dodatniość.

Przykładowe rozwiązanie:

```
Console.Write("Podaj liczbę: ");
int liczba = int.Parse(Console.ReadLine());

if (liczba > 0 && liczba % 2 == 0)
    Console.WriteLine("Liczba jest dodatnia i parzysta.");
else
    Console.WriteLine("Liczba nie spełnia obu warunków.");
```

Modyfikacje do samodzielnego testowania:

1. Dodaj sprawdzenie dla liczb ujemnych.
2. Rozdziel komunikaty dla każdego warunku.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Przykład 2.16. Obliczanie różnicy godzin

Opis problemu:

Program liczy różnicę między dwiema godzinami (bez daty).

Przykładowe rozwiązanie:

```
Console.Write("Podaj godzinę startową: ");
int start = int.Parse(Console.ReadLine());
Console.Write("Podaj godzinę końcową: ");
int koniec = int.Parse(Console.ReadLine());
int roznica = koniec - start;
Console.WriteLine("Różnica: " + roznica + " godzin");
```

Modyfikacje do samodzielnego testowania:

1. Obsłuż sytuację, gdy koniec < start (np. po północy).
2. Wyświetl wynik w minutach.

Przykład 2.17. Zamiana jednostek masy (kg -> g)

Opis problemu:

Program zamienia kilogramy na gramy.

Przykładowe rozwiązanie:

```
Console.Write("Podaj masę w kilogramach: ");
double kg = double.Parse(Console.ReadLine());
Console.WriteLine(kg + " kg = " + (kg * 1000) + " g");
```

Modyfikacje do samodzielnego testowania:

1. Dodaj konwersję na tony.
2. Dodaj walidację dla wartości ujemnych.

Przykład 2.18. Porównanie dwóch napisów

Opis problemu:

Program sprawdza, czy dwa wprowadzone słowa są takie same.

Przykładowe rozwiązanie:

```
Console.Write("Podaj pierwsze słowo: ");
string s1 = Console.ReadLine();
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
Console.Write("Podaj drugie słowo: ");
string s2 = Console.ReadLine();

if (s1 == s2)
    Console.WriteLine("Słowa są identyczne.");
else
    Console.WriteLine("Słowa różnią się.");
```

Modyfikacje do samodzielnego testowania:

1. Porównaj bez względu na wielkość liter.
2. Sprawdź długość słów.

Przykład 2.19. Obliczanie ceny z rabatem

Opis problemu:

Program liczy cenę po obniżce procentowej.

Przykładowe rozwiązanie:

```
Console.Write("Podaj cenę produktu: ");
double cena = double.Parse(Console.ReadLine());
Console.Write("Podaj rabat (%): ");
double rabat = double.Parse(Console.ReadLine());

double poRabracie = cena - (cena * rabat / 100);
Console.WriteLine("Cena po rabacie: " + poRabracie);
```

Modyfikacje do samodzielnego testowania:

1. Wyświetl kwotę zaoszczędzoną.
2. Zaokrąglij wynik do groszy.

Przykład 2.20. Zapis imion do pliku

Modyfikacje do samodzielnego testowania:

Program zapisuje wprowadzone imiona do pliku imiona.txt.

Przykładowe rozwiązanie:

```
using System.IO;
Console.Write("Ile imion chcesz zapisać? ");
int n = int.Parse(Console.ReadLine());
using (StreamWriter sw = new StreamWriter("imiona.txt"))
```

```
{
    for (int i = 0; i < n; i++)
    {
        Console.Write("Podaj imię: ");
        sw.WriteLine(Console.ReadLine());
    }
}
Console.WriteLine("Imiona zapisane do pliku imiona.txt");
```

Modyfikacje do samodzielnego testowania:

1. Dopisuj nowe imiona do istniejącego pliku.
2. Dodaj numerację przed każdym imieniem.

Przykład 2.21. Odczyt imion z pliku

Opis problemu:

Program odczytuje i wypisuje imiona zapisane w pliku imiona.txt.

Przykładowe rozwiązanie:

```
using System.IO;

if (!File.Exists("imiona.txt"))
{
    Console.WriteLine("Plik nie istnieje!");
    return;
}
string[] imiona = File.ReadAllLines("imiona.txt");
Console.WriteLine("Zawartość pliku:");
foreach (string imie in imiona)
{
    Console.WriteLine(imie);
}
```

Modyfikacje do samodzielnego testowania:

1. Wyświetl liczbę imion.
2. Wypisz imiona w kolejności odwrotnej.

Ćwiczenia samodzielne

Zadanie 2.1.

1. Poproś użytkownika o podanie 7 temperatur (w °C).
2. Dla każdej wypisz, czy jest ujemna, dodatnia czy równa zero.
3. Oblicz średnią temperatur i wypisz ją.
4. Wypisz, ile dni było z temperaturą powyżej średniej.

Zadanie 2.2.

1. Poproś użytkownika o wprowadzenie nazw sprzedawców i ich wyników miesięcznych (5 osób).
2. Oblicz i wypisz najwyższy oraz najniższy wynik sprzedaży.
3. Oblicz średni wynik i pokaż, kto miał sprzedaż powyżej średniej.
4. Jeśli ktoś ma dokładnie 0, wypisz komunikat „Brak sprzedaży!”.

Zadanie 2.3.

1. Poproś użytkownika o wprowadzenie boków prostokąta i trójkąta.
2. Oblicz pole każdego z nich.
3. Porównaj, które pole jest większe.
4. Jeśli są równe, wypisz „Figury mają takie samo pole!”.

Zadanie 2.4.

1. Wprowadź wiek 4 pracowników.
2. Dla każdego określ, czy jest poniżej 30, między 30–50 czy powyżej 50 lat.
3. Oblicz średni wiek grupy.
4. Wypisz, ilu pracowników jest w każdej kategorii wiekowej.

Zadanie 2.5.

1. Poproś użytkownika o 6 ocen (1–6).
2. Oblicz średnią ocen.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

3. Wypisz ocenę semestralną na podstawie średniej:
   <3.0 – niedostateczny, 3.0–3.9 – dostateczny, 4.0–4.9 – dobry, ≥5.0 – bardzo dobry.
4. Jeśli średnia to 6.0, wypisz „Wzorowy uczeń!”.

Zadanie 2.6.

1. Poproś użytkownika o podanie 8 liczb całkowitych.
2. Wypisz tylko te liczby, które mieszczą się w zakresie 10–100.
3. Zlicz, ile liczb mieści się w zakresie, a ile jest poza nim.
4. Dla każdej liczby spoza zakresu wypisz „Poza limitem!”.

Zadanie 2.7.

1. Poproś użytkownika o kwotę w złotówkach.
2. Zapytaj, na jaką walutę chce przeliczyć: EUR, USD lub GBP.
3. Przelicz kwotę według stałego kursu (np. EUR=4.3, USD=4.0, GBP=5.0).
4. Jeśli użytkownik poda nieznany symbol waluty, wypisz „Nieznana waluta!”.

Zadanie 2.8.

1. Poproś użytkownika o wpisanie dowolnego tekstu.
2. Policz, ile zawiera liter, cyfr i spacji.
3. Wypisz długość tekstu.
4. Jeśli tekst zawiera wyraz „C#”, wypisz „Programista zauważony!”.

Zadanie 2.9.

1. Poproś użytkownika o 3 liczby rzeczywiste.
2. Oblicz sumę, średnią i iloczyn tych liczb.
3. Wypisz, które liczby są większe od średniej.
4. Jeśli wszystkie są równe, wypisz „Wszystkie wartości są identyczne!”.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Zadanie 2.10.

1. Przygotuj 5 pytań o zwierzętach (np. „Czy słoń jest największym ssakiem lądowym?”).
2. Poproś użytkownika o odpowiedzi (tak/nie).
3. Policz, ile odpowiedzi było poprawnych.
4. Wypisz ocenę: 5 – „Perfekcyjnie!”, 3–4 – „Nieźle!”, 0–2 – „Musisz się podszkolić!”.

Zadanie 2.11.

1. Wczytaj 9 liczb całkowitych.
2. Oddziel je do trzech list: dodatnich, ujemnych i zer.
3. Policz, ile jest w każdej grupie.
4. Wypisz listy na ekran.

Zadanie 2.12.

1. Poproś użytkownika o podanie punktów zdobytych w 7 grach.
2. Dla każdej określ, czy wynik jest niski (<50), średni (50–100), czy wysoki (>100).
3. Policz, ile wyników jest w każdej kategorii.
4. Jeśli wszystkie wyniki są powyżej 100, wypisz „Gracz doskonały!”.

Zadanie 2.13.

1. Poproś użytkownika o 10 liczb całkowitych w jednej linii.
2. Znajdź najdłuższy ciąg malejący wśród tych liczb.
3. Wypisz długość i elementy tego ciągu.
4. Jeśli nie ma żadnego ciągu dłuższego niż 1, wypisz „Brak ciągu malejącego”.

Zadanie 2.14.

1. Poproś użytkownika o podanie imienia, nazwiska i wieku.
2. Zapisz te dane do pliku tekstowego dane.txt (każda informacja w nowej linii).
3. Następnie odczytaj zawartość pliku i wyświetl ją w konsoli.
4. Jeśli plik nie istnieje, wyświetl komunikat: „Plik nie został znaleziony!”.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Zadanie 2.15.

1. Poproś użytkownika o podanie nazwy pliku tekstowego do analizy.
2. Odczytaj cały tekst z pliku i policz, ile zawiera słów.
3. Wypisz liczbę słów na ekran.
4. Jeśli plik jest pusty, wypisz komunikat: „Plik nie zawiera żadnych danych.”

Zadanie 2.16.

1. Poproś użytkownika o podanie dwóch nazw plików: źródłowego i docelowego.
2. Sprawdź, czy plik źródłowy istnieje.
3. Jeśli tak, skopiuj jego zawartość do nowego pliku.
4. Po zakończeniu wypisz komunikat: „Plik został skopiowany pomyślnie!”.

Zadanie 2.17.

1. Utwórz plik log.txt, jeśli jeszcze nie istnieje.
2. Poproś użytkownika o wpisanie dowolnego tekstu (np. komentarza lub notatki).
3. Dopisz tekst do pliku, nie nadpisując istniejącej zawartości.
4. Po każdym dopisaniu dodaj datę i godzinę wpisu.

Zadanie 2.18.

1. Poproś użytkownika o podanie słowa kluczowego do wyszukania.
2. Otwórz plik tekstowy tekst.txt i przeszukaj go linia po linii.
3. Wypisz wszystkie linie, które zawierają to słowo.
4. Jeśli żadne linie nie pasują, wypisz komunikat: „Nie znaleziono wystąpień.”

Pytania pomocnicze

1. Jak działa instrukcja if i kiedy warto używać else if oraz else?
2. W jakich sytuacjach lepiej zastosować switch zamiast wielu zagnieżdżonych if?

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

3. Czym różni się tradycyjny switch od tzw. switch expression w nowszych wersjach C#?
4. Jak działają operatory logiczne &&, || i ! w warunkach i kiedy je stosować?
5. Jak poprawnie wczytać dane od użytkownika z klawiatury i przekonwertować je na typ liczbowy?
6. Jakie są różnice między Console.Write() i Console.WriteLine()?
7. Jak działają pętle for, while i foreach i w jakich sytuacjach najlepiej je stosować?
8. Jak używać instrukcji break, continue i return w pętlach i funkcjach?
9. Jak w C# sprawdzić, czy plik istnieje, i jak bezpiecznie go odczytać lub zapisać?
10. Jak obsługiwać błędy podczas operacji wejścia–wyjścia przy użyciu try-catch i dlaczego jest to ważne?

Podsumowanie

Instrukcje warunkowe w języku C# (if, if–else, else if oraz switch) służą do podejmowania decyzji i kierowania działaniem programu w zależności od spełnienia określonych warunków. Konstrukcja switch sprawdza się szczególnie dobrze, gdy trzeba obsłużyć wiele możliwych wartości jednej zmiennej. Pętle (for, while, foreach) pozwalają wykonywać ten sam fragment kodu wielokrotnie, a dzięki instrukcjom break i continue można przerwać lub pominąć kolejne iteracje, co daje większą kontrolę nad przebiegiem programu. Do komunikacji z użytkownikiem służą operacje wejścia i wyjścia w konsoli, takie jak Console.ReadLine(), Console.WriteLine() czy Console.ReadKey(). Natomiast klasy i metody z przestrzeni nazw System.IO, np. File.ReadAllText(), File.WriteAllText(), StreamReader i StreamWriter, umożliwiają wygodny odczyt i zapis danych do plików. Podczas pracy z plikami warto stosować blok try-catch, aby obsłużyć ewentualne błędy na przykład brak dostępu, nieistniejący plik czy błędny format danych. Umiejętne wykorzystanie instrukcji sterujących oraz operacji wejścia-wyjścia pozwala tworzyć programy elastyczne, bezpieczne i łatwe w utrzymaniu.

Literatura

Burns, Sean (2019). Hands-On Network Programming with C# and .NET Core: Build Robust Network Applications with C# and .NET Core. Wydanie 1. Birmingham: Packt Publishing, Limited.

Alls, Jason; Meryk, Radosław (tłum.) (2021). Czysty kod w C# : techniki refaktoryzacji i najlepsze praktyki. Gliwice: Helion.

Michaelis, Mark; Lippert, Eric (red.); Walczak, Tomasz (tłum.); Torgersen, Mads (przedm.) (2020). C# 7.0 : kompletny przewodnik dla praktyków. Gliwice: Helion.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Alls, Jason; Meryk, Radosław (tłum.) (2021). Czysty kod w C# : techniki refaktoryzacji i najlepsze praktyki. Gliwice: Helion.

https://learn.microsoft.com/pl-pl/troubleshoot/developer/visualstudio/csharp/language-compilers/file-io-operation

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00