

Materiały do przedmiotu:

Podstawy programowania

Laboratorium nr 2

Autor:

Patrycja Miazek



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Spis treści

Wprowadzenie, tematyka zajęć -----	3
Cel ćwiczenia/laboratorium -----	3
Instrukcja laboratoryjna/ćwiczeniowa -----	3
Ćwiczenia samodzielne -----	21
Pytania pomocnicze -----	28
Podsumowanie -----	29
Literatura-----	29



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Instrukcje sterujące. Operacje wejścia/wyjścia

Wprowadzenie, tematyka zajęć

Podczas laboratorium studenci zapoznają się z podstawowymi instrukcjami sterującymi w języku C#, które pozwalają kontrolować przepływ programu i podejmować decyzje w zależności od napotkanych warunków. Ćwiczenia pokazują, jak instrukcje warunkowe (if, if-else, else if, switch) oraz pętle (for, while, foreach) umożliwiają powtarzanie operacji i definiowanie różnych ścieżek wykonania programu. Równocześnie studenci uczą się wykonywać operacje wejścia i wyjścia, odczytywać dane wprowadzane przez użytkownika oraz przetwarzać informacje zapisane w plikach tekstowych i binarnych, a także prezentować wyniki w konsoli lub zapisywać je do plików. Dzięki integracji sterowania przepływem programu z obsługą wejścia/wyjścia studenci poznają praktyczne zastosowanie tych zagadnień w tworzeniu interaktywnych i elastycznych programów.

Cel ćwiczenia/laboratorium

Celem zajęć jest umożliwienie studentom zdobycia praktycznych umiejętności w zakresie stosowania instrukcji sterujących oraz operacji wejścia i wyjścia w języku C#. Studenci uczą się, jak podejmować decyzje w programie, jak powtarzać operacje za pomocą pętli oraz jak integrować te mechanizmy z odczytem i zapisem danych w plikach. Zajęcia pozwalają zrozumieć przepływ danych w programie, przetwarzać dane wprowadzane przez użytkownika, obsługiwać pliki tekstowe i binarne oraz prezentować wyniki w czytelny sposób. Dzięki ćwiczeniom studenci rozwijają umiejętności tworzenia prostych aplikacji interaktywnych, które stanowią podstawę do dalszej nauki i pracy z bardziej zaawansowanymi zagadnieniami programowania w C#.

Instrukcja laboratoryjna/ćwiczeniowa

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)
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



```
{  
    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;  
  
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.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



```
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 miesiac");
        break;
}
```

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 miesiac"
};

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**.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



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 do...while

Pętla do...while działa podobnie jak while, ale warunek jest sprawdzany **po wykonaniu** ciała pętli. Oznacza to, że kod wewnątrz pętli wykona się **przynajmniej raz**, niezależnie od warunku.

```
int liczba;
do
{
    Console.Write("Podaj liczbę większą od zera: ");
    liczba = int.Parse(Console.ReadLine());
} while (liczba <= 0);

Console.WriteLine("Prawidłowa liczba: " + liczba);
```

To tzw. „pętla zaporowa”. Przydatna, gdy jest potrzeba, aby użytkownik podał poprawne dane.

Pętla for



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



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



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.WriteLine("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 pętli 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




```
}
```

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!");  
}
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



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

```
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



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}");  
}
```

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  
{
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



```
reader?.Close();  
}
```

Uwaga! W typach wyliczeniowych wartości domyślnie są numerowane od 0. W przykładach często liczby zaczynają się od 1, dlatego przy korzystaniu z typów wyliczeniowych może być konieczne dostosowanie indeksów i warunków w pętlach lub instrukcjach switch.

Przykład 2.1

Sprawdzenie, czy liczba jest wielokrotnością 3.

Przykładowe rozwiązanie:

```
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

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");
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Modyfikacje do samodzielnego testowania:

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

Przykład 2.3

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

Przykładowe rozwiązanie:

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

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 do samodzielnego testowania:

1. Obsłuż przypadek n = 0.
2. Wypisz pośrednie wyniki [1!, 2!, 3! ...].

Przykład 2.5

Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



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

Modyfikacje do samodzielnego testowania:

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

Przykład 2.6

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

Przykładowe rozwiązanie:

```
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

Program wskazuje najmniejszą z trzech podanych liczb.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



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: ");  
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

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

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: ");
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



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

if (b != 0)
{
    Console.WriteLine("Wynik: " + (a / b));
    Console.WriteLine("Reszta: " + (a % b));
}
else
{
    Console.WriteLine("Nie można dzielić przez zero!");
}
```

Modyfikacje do samodzielnego testowania:

1. Obsłużyć przypadek, gdy dzielna lub dzielnik są liczbami ujemnymi.
2. Zapisz wynik jako double.

Przykład 2.10

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

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));
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską




```
Console.WriteLine("Iloczyn: " + (a * b * c));
```

Modyfikacje do samodzielnego testowania:

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

Przykład 2.12.

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

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).
2. Obsłuż liczby ujemne.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Przykład 2.14

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

Program sprawdza dwa warunki jednocześnie: parzystość liczby oraz czy jest dodatnia.

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.

Przykład 2.16

Program liczy różnicę między dwiema godzinami (bez daty).

Przykładowe rozwiązanie:



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



```
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

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

Program sprawdza, czy dwa wprowadzone słowa są takie same.

Przykładowe rozwiązanie:

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



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



2. Sprawdź długość słów.

Przykład 2.19

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

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.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Przykład 2.2

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

Napisz program, który pobierze wiek użytkownika i wypisze komunikat: „Jesteś pełnoletni” lub „Nie jesteś pełnoletni”, korzystając z operatora ?:.

Zadanie 2.2

Napisz program, który pobierze od użytkownika dwie liczby całkowite, a następnie wyświetli informację, która z nich jest większa (lub że są równe). Do rozwiązania zastosuj operator warunkowy ?:.

Zadanie 2.3

Przyjmij od użytkownika siedem temperatur w stopniach Celsjusza, dla każdej wypisz, czy jest ujemna, dodatnia czy równa zero, oblicz średnią temperaturę i wypisz, ile dni miało temperaturę powyżej średniej.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Zadanie 2.4

Przyjmij od użytkownika sześć ocen w skali 1–6, oblicz średnią ocen i wypisz ocenę semestralną według skali: poniżej 3 – niedostateczny, 3–3,9 – dostateczny, 4–4,9 – dobry, 5 i więcej – bardzo dobry, a jeśli średnia wynosi dokładnie 6, wypisz komunikat „Wzorowy uczeń!”.

Zadanie 2.5

Przyjmij od użytkownika osiem liczb całkowitych, wypisz tylko te, które mieszczą się w zakresie od 10 do 100, zlicz ile liczb jest w zakresie i ile poza nim, a dla każdej liczby spoza zakresu wypisz komunikat „Poza limitem!”.

Zadanie 2.6

Przyjmij od użytkownika trzy liczby rzeczywiste, oblicz ich sumę, średnią i iloczyn, wypisz, które liczby są większe od średniej, a jeśli wszystkie są identyczne, wypisz komunikat „Wszystkie wartości są identyczne!”.

Zadanie 2.7

Przyjmij od użytkownika nazwy pięciu sprzedawców i ich miesięczne wyniki sprzedaży, wypisz najwyższy i najniższy wynik, oblicz średni wynik oraz pokaż, którzy sprzedawcy mieli wynik powyżej średniej, a jeśli wynik sprzedawcy wynosi dokładnie zero, wypisz komunikat „Brak sprzedaży!”.

Zadanie 2.8

Przyjmij od użytkownika dowolny tekst, policz i wypisz liczbę liter, cyfr i spacji, podaj długość tekstu, a jeśli tekst zawiera wyraz „C#”, wypisz komunikat „Programista zauważony!”.

Zadanie 2.9

Przygotuj pięć pytań o zwierzętach i przyjmij od użytkownika odpowiedzi „tak” lub „nie”, zlicz liczbę poprawnych odpowiedzi i wypisz ocenę: 5 – „Perfekcyjnie!”, 3–4 – „Nieźle!”, 0–2 – „Musisz się podszkolić!”.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Zadanie 2.10

Przyjmij od użytkownika długości boków prostokąta i trójkąta, oblicz pola obu figur i wypisz, która figura ma większe pole lub że pola są równe.

Zadanie 2.11

Przyjmij od użytkownika wiek czterech pracowników, określ kategorię wieku każdego pracownika (<30 , $30-50$, >50), oblicz średni wiek grupy i wypisz liczbę pracowników w każdej kategorii.

Zadanie 2.12

Przyjmij od użytkownika kwotę w złotych i zapytaj, na jaką walutę chce ją przeliczyć (EUR, USD, GBP), przelicz kwotę według stałego kursu (EUR=4,3, USD=4,0, GBP=5,0), a jeśli użytkownik poda nieznany symbol waluty, wypisz komunikat „Nieznana waluta!”.

Zadanie 2.13

Przyjmij od użytkownika liczbę całkowitą, oblicz sumę i iloczyn jej cyfr oraz znajdź największą i najmniejszą cyfrę, wykorzystując operator $\% 10$ i $/ 10$ do pobierania kolejnych cyfr.

Zadanie 2.14

Przyjmij od użytkownika dziewięć liczb całkowitych, podziel je na trzy listy: dodatnie, ujemne i zera, zlicz ile jest w każdej grupie i wypisz wszystkie listy.

Zadanie 2.15

Przyjmij od użytkownika punkty zdobyte w siedmiu grach, określ dla każdej, czy wynik jest niski (<50), średni ($50-100$) czy wysoki (>100), zlicz wyniki w każdej kategorii, a jeśli wszystkie są powyżej 100, wypisz komunikat „Gracz doskonały!”.

Zadanie 2.16



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Przyjmij od użytkownika dziesięć liczb całkowitych w jednej linii, znajdź najdłuższy ciąg malejący wśród tych liczb, wypisz długość i elementy tego ciągu, a jeśli nie ma ciągu dłuższego niż 1, wypisz komunikat „Brak ciągu malejącego”.

Zadanie 2.17

Przyjmij od użytkownika imię, nazwisko i wiek, zapisz te dane do pliku dane.txt każdą informację w nowej linii, odczytaj zawartość pliku i wypisz ją w konsoli, a jeśli plik nie istnieje, wypisz komunikat „Plik nie został znaleziony!”.

Zadanie 2.18

Przyjmij od użytkownika nazwę pliku tekstowego, odczytaj cały tekst z pliku, policz i wypisz liczbę słów, a jeśli plik jest pusty, wypisz komunikat „Plik nie zawiera żadnych danych.”

Zadanie 2.19

Przyjmij od użytkownika dwie nazwy plików: źródłowego i docelowego, sprawdź, czy plik źródłowy istnieje, jeśli tak, skopiuj jego zawartość do pliku docelowego i wypisz komunikat „Plik został skopiowany pomyślnie!”.

Zadanie 2.20

Przyjmij od użytkownika dowolny tekst, dopisz go do pliku log.txt nie nadpisując istniejącej zawartości i po każdym dopisaniu dodaj datę i godzinę wpisu.

Zadanie 2.21

Przyjmij od użytkownika słowo kluczowe do wyszukania w pliku tekst.txt, wypisz wszystkie linie zawierające to słowo, a jeśli nie znaleziono żadnych wystąpień, wypisz komunikat „Nie znaleziono wystąpień.”

Zadanie 2.22



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Przyjmij od użytkownika wysokość dochodu, oblicz podatek według progów: poniżej 30 000 zł – 12%, od 30 000 do 100 000 zł – 20%, powyżej 100 000 zł – 32% i wypisz kwotę podatku oraz dochód netto.

Zadanie 2.23

Przyjmij od użytkownika długość trasy w kilometrach i oblicz cenę biletu: do 50 km – 1,5 zł/km, 51–200 km – 1,2 zł/km, powyżej 200 km – 1 zł/km i wypisz łączną cenę.

Zadanie 2.24

Przyjmij od użytkownika trzy długości odcinków, sprawdź, czy można z nich zbudować trójkąt, a jeśli tak, określ, czy jest prostokątny.

Zadanie 2.25

Przyjmij od użytkownika współrzędne punktu (x, y) , określ, w której ćwiartce leży punkt, sprawdź, czy punkt leży na okręgu jednostkowym $(x^2+y^2=1)$ oraz czy należy do koła o zadanym promieniu i środku.

Zadanie 2.26

Przyjmij od użytkownika środki i promienie dwóch okręgów, sprawdź, czy się przecinają, są styczne lub rozłączne.

Zadanie 2.27

Przyjmij od użytkownika liczbę rzeczywistą, sprawdź, czy posiada część ułamkową, a jeśli tak, wypisz jej wartość.

Zadanie 2.28

Przyjmij od użytkownika liczbę całkowitą, wypisz jej cyfry od końca, policz liczbę cyfr, liczbę zer końcowych, sprawdź, czy liczba jest palindromem i policz wystąpienia każdej cyfry.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Zadanie 2.29

Przyjmij od użytkownika liczby dodatnie zakończone zerem, oblicz średnią liczb, znajdź najdłuższy ciąg rosnący i malejący, wypisz długości i elementy tych ciągów oraz łączną liczbę wprowadzonych liczb.

Zadanie 2.30

Napisz program, który pobiera od użytkownika liczbę całkowitą, a następnie obliczy sumę pierwszej i ostatniej cyfry tej liczby. Jeżeli liczba jest ujemna, przed obliczeniami należy zastosować `Math.Abs()`. Przyjmujemy, że dla liczby jednocyfrowej suma jest równa tej liczbie.

Zastosuj pętlę:

a) `while`,

b) `for`

Zadanie 2.31

Napisz program, który przyjmie od użytkownika liczbę całkowitą, a następnie wydrukuje ją od końca. Na przykład dla liczby 12345 wynik działania programu powinien wynosić 54321. Program powinien wykorzystywać pętlę (`while` lub `for`)Napi.

Przygotuj dwie wersje programu:

- 1) z wyznaczaniem liczby o odwróconej kolejności cyfr (zapisanie jej w nowej zmiennej),
- 2) bez wyznaczania liczby o odwróconej kolejności cyfr (tylko wypisanie cyfr w odwróconej kolejności).

Zadanie 2.32

Napisz program, który wczyta od użytkownika 10 liczb całkowitych i zapisze je w tablicy. Za pomocą pętli `foreach` oblicz średnią arytmetyczną tych liczb, a następnie wypisz, ile liczb jest większych od tej średniej.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Zadanie 2.33

Napisz program, który pobiera od użytkownika liczbę całkowitą z przedziału [0;25], oznaczającą liczbę zdobytych punktów z kolokwium. Program powinien ponawiać prośbę o podanie liczby, dopóki użytkownik nie wprowadzi poprawnej wartości. Zastosuj pętlę:

- a) while
- b) do/while.

Na podstawie wprowadzonej liczby punktów program powinien określić i wyświetlić ocenę z kolokwium według następujących reguł:

Liczba punktów	Ocena
[0; 12]	2
(12; 17]	3
(17; 22]	4
(22; 25]	5

Program powinien obliczyć ocenę wykorzystując:

- 1) instrukcję if - else if - else ,
- 2) instrukcję switch,
- 3) wyrażenie switch expression.

Zadanie 2.34

Napisz program, który pobiera od użytkownika dwie liczby całkowite: n – rozmiar kwadratu oraz k – grubość ramki. Program powinien narysować w konsoli kwadrat o boku n , którego ramka zbudowana jest z gwiazdek $*$ o grubości k . Wnętrze kwadratu powinno być wypełnione spacjami. Zakładamy, że $n \geq 2k$ oraz $k \geq 1$.

Przykład dla $n = 10$ oraz $k = 2$:

```

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

```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Zadanie 2.35

Napisz program, który pobierze od użytkownika wysokość trójkąta n , a następnie wyświetli równoramienny trójkąt z gwiazdek $*$ o podanej wysokości.

Dla $n = 5$ program powinien wyświetlić:

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

Zadanie 2.36 (**)

Przyjmij od użytkownika ścieżkę do folderu, znajdź wszystkie pliki tekstowe .txt w tym folderze, dla każdego policz liczbę wierszy, słów i znaków, znajdź plik z największą i najmniejszą liczbą słów, oblicz łączną liczbę słów we wszystkich plikach i zapisz wynik w pliku raport.txt w formie tabeli z możliwością sortowania po liczbie słów lub znaków.

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`?
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?



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



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

Podczas laboratorium studenci zapoznali się z instrukcjami warunkowymi w języku C#, takimi jak if, if-else, else if oraz switch, które pozwalają podejmować decyzje i kierować przepływem programu w zależności od spełnienia określonych warunków. Ćwiczenia umożliwiły zrozumienie, jak stosować te konstrukcje w praktyce, aby program reagował na różne sytuacje. Studenci poznali również pętle for, while i foreach, które umożliwiają powtarzanie operacji na danych w sposób kontrolowany. Kolejnym elementem zajęć były operacje na plikach – odczyt i zapis danych w plikach tekstowych i binarnych z wykorzystaniem klas File, StreamReader i StreamWriter. Studenci nauczyli się integrować instrukcje sterujące z operacjami na plikach, stosować blok try-catch w celu obsługi błędów oraz tworzyć programy reagujące na różne sytuacje i przetwarzające dane zapisane w plikach w sposób elastyczny i bezpieczny.

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.

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

