Dane z klawiatury i obliczenia

Czas: 2 × 45 min | praca indywidualna | Wykład 1.02–1.03

Cel: Napiszesz dwa programy obliczeniowe, które bezpiecznie wczytują liczby z klawiatury i odrzucają błędne dane.

Po zajęciach potrafisz:

- wczytuje dane metodą `Console.ReadLine` i zamienia je na liczby metodą `TryParse`
- powtarza pytanie do skutku w pętli przy błędnych danych
- wykonuje obliczenia na typie `double` i formatuje wynik
- testuje program na danych poprawnych, granicznych i błędnych

Sprzęt i materiały

- komputer z Windows 11 i dostępem do internetu
- pendrive lub konto w repozytorium do zapisania projektu

Oprogramowanie

- **Visual Studio Community 2026 (obciążenie „Programowanie aplikacji klasycznych platformy .NET")** — darmowe dla uczniów i nauczycieli; w pracowni zainstalowane visualstudio.microsoft.com/pl/downloads
- **.NET 10 SDK (LTS)** — instalowany razem z Visual Studio; sprawdzenie: `dotnet --version` dotnet.microsoft.com/pl-pl/download/dotnet/10.0

Bezpieczeństwo

- Zapisuj projekt w swoim folderze (np. `C:\Users\<login>\source\repos`), a na koniec zajęć skopiuj go na pendrive lub wyślij do repozytorium.
- Nie uruchamiaj programów i pakietów z nieznanych źródeł; pakiety NuGet instaluj tylko te, które wskazuje zadanie.

Przebieg zadania

1 Koszt przejazdu

Program pyta o długość trasy (km), spalanie (l/100 km), cenę paliwa (zł/l) i liczbę osób. Wypisuje zużycie paliwa, koszt całkowity i koszt na osobę.

1. Utwórz projekt konsolowy `Obliczenia` i zapisz wzór: paliwo = km × spalanie / 100, koszt = paliwo × cena.
2. Wczytaj dane metodą `double.TryParse(Console.ReadLine(), out double x)`.
3. Wynik wypisz z dwoma miejscami po przecinku (`{koszt:F2}`) i jednostką.
4. Sprawdź wynik na kalkulatorze dla danych: 250 km, 6,5 l, 6,19 zł, 3 osoby.

2 Odporność na błędy

Wydziel wczytywanie liczby do funkcji lokalnej, która pyta tak długo, aż dane będą poprawne.

```
double Wczytaj(string pytanie, double min, double max)
{
    while (true)
    {
        Console.Write(pytanie);
        if (double.TryParse(Console.ReadLine(), out double x) && x >= min && x <= max)
            return x;
        Console.WriteLine($"  Błąd: podaj liczbę od {min} do {max}.");
    }
}
```

1. Napisz funkcję `Wczytaj(string pytanie, double min, double max)` zgodnie z szablonem.
2. Dodaj sensowne zakresy (np. trasa 1–5000 km, spalanie 1–40 l).
3. Sprawdź działanie dla tekstu `abc`, pustego wiersza, liczby ujemnej i liczby z kropką zamiast przecinka.
4. Zapisz w tabeli, jak program zareagował na każdy przypadek.

3 Kalkulator BMI

W tym samym programie dopisz drugą część: BMI = masa / wzrost² (wzrost w metrach).

1. Wczytaj masę w kilogramach i wzrost w centymetrach funkcją `Wczytaj`; zamień centymetry na metry.
2. Wypisz BMI z jednym miejscem po przecinku.
3. Dodaj kategorię: poniżej 18,5 — niedowaga, poniżej 25 — waga prawidłowa, poniżej 30 — nadwaga, w przeciwnym razie otyłość (operator `?:`).
4. Przetestuj wartości graniczne: BMI dokładnie 18,5, 25 i 30 (dobierz masę dla wzrostu 200 cm).
5. Zatwierdź pracę w Git (`git commit`).

W sprawozdaniu umieść

- ☐ kod programu
- ☐ tabela testów z wynikami
- ☐ zrzut ekranu programu dla błędnych i poprawnych danych
- ☐ wyjaśnienie, dlaczego `double.Parse` jest gorszym wyborem niż `TryParse`

Pytania kontrolne

1. Co zwraca metoda `TryParse` i co trafia do parametru `out`, gdy konwersja się nie uda?
2. Dlaczego w polskim Windows liczba `6.5` może nie zostać rozpoznana?
3. Jakie dane testowe nazywamy granicznymi? Podaj przykład z tego zadania.
4. Dlaczego warto wydzielić wczytywanie do osobnej funkcji?

Kryteria oceny

Kryterium	Punkty
Koszt przejazdu — poprawne obliczenia i format	5
Funkcja Wczytaj z zakresem i pętlą	4
BMI z kategoriami	4
Tabela testów (poprawne, graniczne, błędne)	4
Sprawozdanie i pytania	3
Razem	**20**

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).