Programowanie w C#

Paweł Biesiada

Zadanie 1.1

- Napisz program, który wypisze „Hi ” + imię przekazane jako zmienna
  - Przekaż imię jako parametr do metody Console.WriteLine
    - PrintHi
  - Przekaż imię jako argument wejściowy programu

Zadanie 1.2

- Napisz program, do którego możemy wprowadzić długości trzech boków trójkąta
- sprawdź, czy mogą one stworzyć trójkąt.

Zadanie 1.3

- Napisz program, który znajduje najmniejszą liczbę Fibonacciego większą od:
  - 1000
  - **Parametru z args**
  - **1,1,2,3,5,8,13,21,34,55**

Zadanie 1.4

- Napisz program, który sprawdza czy zadana liczba jest liczbą pierwszą
  - Wydziel nową metodę, która zwraca wartość bool
  - Argumentem wejściowym jest rozpatrywana liczba

Zadanie 1.5

- Sprawdź ile dni minęło od dnia Twoich urodzin do teraz.
  - Wprowadź datę z konsoli (Console.ReadLine())
  - Wypisz datę urodzin (sam dzień) w formacie polskim
    - CultureInfo
  - Wynik wypisz do konsoli
  - **Policz czas również w godzinach.**

Zadanie 1.6

- Napisz program, który na podstawie następującego tekstu „*Lorem ipsum dolor sit amet, consectetur adipiscing elit.*”:
  - Zlicza liczbę wyrazów (Split)
  - Wypisuje fragment tekstu do przecinka (bez przecinka) (IndexOf, Substring)
    - wypisuje cały tekst za przecinkiem (bez przecinka)
  - Wypisuje tylko trzeci wyraz (Split)
  - Łączy w jeden string co drugi wyraz (Split, Concat)
  - Zlicza wystąpienie litery ‘e’ (foreach or Split, or IndexOf and Substring)

Zadanie 1.7

- Porównaj prędkość działania:
  - Konkatenacji stringów
  - StringBuilder'a

Zadanie 2.1

- Napisz klasę *Robot* z następującymi trzema zmiennymi instancji:
  - *Name(string)*, *Age(ushort)* i *IsOn(bool)*, nadaj im Gettery i Settery.
  - Pozwól użytkownikowi na przypisywanie wartości Name oraz Age w konstruktorze, gdy tworzony jest obiekt *Robot*. Zapewnij również istnienie konstruktora domyślnego
  - właściwość IsOn jest inicjowana domyślnie z wartością true
  - Dodaj nowa metode SayHi() ktora, jesli wlasciwosc IsOn jest true wypisuje na konsolę “Say Hi {Name}”

Zadanie 2.1b

- Napisz klasę *AppConfiguration*, która:
  - której do konstruktora można przesłać ścieżkę do pliku konfiguracyjnego
  - w pliku konfiguracyjnym jest informacja o języku używanym w aplikacji
  - Pozwól osobom z zewnątrz odczytać informację o języku
  - napisz sygnaturę metody, która będzie odpowiedzialna za wczytywanie pliku - bez konkretnej implementacji
  - wczytaj konfigurację podczas tworzenia obiektu klasy

Zadanie 2.2

- Napisz klasę *RainFall* zawierającą jednowymiarową tablicę z 12 elementami reprezentującymi miesięczne pomiary opadów deszczu. Uwzględnij w niej następujące cechy:
  - Klasa powinna udostępniać użytkownikom metodę GetMonthyRainFall ma dostęp do elementów tablicy poprzez indeks typu *int*, ale miesiąc pierwszy powinien być dostępny przez indeks 1
  - Dołącz właściwość o nazwie *Average*, która policzy średni roczny opad miesięczny
  - Napisz metodę *AddRainFall*, ktora dodaje opad do danego miesiaca
  - Napisz metodę *ImportRainFall* która przyjmuje jako parametr typ *RainFall*, która dodaje do obecnej instancji wszystkie opady z przekazanej zmiennej (można również zrobić przeciążenie operatora '+' w tym celu)

Zadanie 2.3

- Napisz cztery klasy o nazwach Employee, Secretary, Director oraz Programmer.
  - klasa Employee jest klasą bazową zawierającą właściwość Name oraz metodę CalculateSalary, która wypisuje „Obliczam wypłatę dla [nazwaOsoby]”
  - klasy pochodne nadpisują CalculateSalary() dopisując nową linię z wysokością pensji
  - ponadto niech Director zawiera metodę GetBonus() (Niech wypisze na konsolę „Wypłacam bonus.”)
  - Napisz metodę, która tworzy po 2 obiekty Secretary oraz Programmer i 1 obiekt Director. Zapisz je w dowolnej kolekcji
  - Napisz metodę, która jako parametr przyjmuje tą kolekcję i wykonuje dla wszystkich obiektów metodę CalculateSalary(), a dla Director również GetBonus();

Zadanie 2.4

- Przeciąż metodę ToString() tak, aby wszystkie klasy dziedziczące po Employee wypisywały treść:
  - “My name is {Name}, Id: {Id}”

Zadanie 2.5

- Przeciąż metodę porównującą obiekty (Equals()) tak, aby:
  - zwracała true, jeżeli porównywane obiekty typu Employee mają takie samo Id
  - W innych przypadkach powinna zwracać false
  - Pamiętaj o przeciążeniu GetHashCode()!

Zadanie 2.6

- Napisz klasę pozwalającą na wczytanie konfiguracji z pliku konfiguracyjnego aplikacji:
  - Elementy konfiguracji aplikacji to:
  - poziom logowania (enum – Debug, Warning, Error)
  - język aplikacji (CultureInfo)
  - Klasa posiada właściwości z getterami i setterami
  - Właściwości są inicjalizowane podczas tworzenia instancji klasy

Zadanie 2.7

- Dla Klasy przykładowej FamilyCar napisz nowy typ wyjątku CapacityExceededException:
  - Do wyjątku przekaż informację o maksymalnej pojemności i wartości, którą próbowano przypisać
  - Przechwyć ten wyjątek w metodzie, w której inicjalizujemy obiekt i wywołujemy metodę
  - Wypisz informację z wyjątku na konsolę

Zadanie 2.9

- Napisz metodę rozszerzającą dla:
  - Klasy string pozwalającej liczyć wyrazy w tekście
  - Klasy string zliczającą wystąpienie danej litery w tekście
  - Klasy FamilyCar, która będzie wykonywała metodę LoadTrunk dla każdego elementu z listy przekazanej jako parametr. Metoda zwraca void.

Zadanie 3.1a

- Wykorzystaj klasę Users, do następujących operacji wykorzystujących LINQ (pobierz kolekcję za pomocą CreateCollection.GetUsers():
  - Sprawdź czy kolekcja posiada jakikolwiek element (*Any*)
  - Policz wszystkie nie nullowe elementy (*Where, Count*)
  - Wypisz wszystkich użytkowników, których imię zaczyna się na literę “M” (Where)
  - wyciągnij 5 pierwszych użytkowników posortowanych po imieniu (*Take, OrderBy*)
    - wypisz drugą piątkę (*Take*, *Skip*)
  - Wypisz wszystkie imiona bez powtórzeń (Select + Distinct)
  - Wypisz wszystkie imiona, które się powtarzają (GroupBy)

Zadanie 3.1b

- Wykorzystaj klasę Users, do następujących operacji wykorzystujących LINQ (pobierz kolekcję za pomocą CreateCollection.GetUsers():
  - Wypisz tylko obiekty typu SuperUser (OfType)
  - Wypisz wszystkich użytkowników, którzy są aktywnymi administratorami (OfType, Where)
  - Pobierz tylko jednego użytkownika, który jest administratorem. (OfType, Where, FirstOrDefault)
  - Utwórz obiekt Dictionary<string, int> gdzie kluczem jest imię użytkownika, a wartością liczba użytkowników o takim samym imieniu (GroupBy, ToDictionary)

Zadanie 3.2

- Napisz parser argumentów do command line'a:
  - Dane wejściowe w formacie „parametr=wartość -switch"
  - Parser powinien przechowywać listę switchy i parametrów z wartościami
  - Napisz metodę HasSwitch(), która sprawdza, czy dany switch został przekazany
  - Dodaj indekser, który zwraca wartość dla zadanego parametru

Zadanie 3.3

- Napisz metodę, która za parametr przyjmuje ścieżkę do pliku (*FileInfo*) i wypisuje informacje o nim:
  - Nazwę (bez pełnej ścieżki)
  - Rozszerzenie (bez kropki)
  - Ścieżkę do folderu
  - Jego wielkość w KB (.*Length*)
  - Datę utworzenia
  - Datę modyfikacji

Zadanie 3.4

- Napisz program, który eksportuje listę do pliku Users.csv:
  - Utwórz nowy plik i zapisz w nim wszystkich użytkowników pobranych za pomocą metody CreateCollection.GetUsers():
  - Pierwszy wiersz w pliku powinien mieć odpowiedni nagłówek
  - Sprawdź czy dany plik już istnieje, jeśli tak to go nadpisz
  - Użyj zapisu za pomocą strumienia (nie wiadomo jak duża może być lista)
  - Wczytaj zapisany wcześniej plik i wypisz jego zawartość na konsolę

Użyj metod WriteAllText, lub WriteAllLines

Zadanie 3.5

- Napisz program, który:
  - Przyjmuje dwa argumenty input, output
  - wczytuje plik .csv z dysku (format danych z poprzedniego zadania)
  - wypisuje jego zawartość w konsoli (oddziel kolumny tabulatorem)
  - zapisuje kopię pliku w innym miejscu zdefiniowanym przez output
  - Po zamknięciu pliku dopisz do niego jeszcze jeden nowy wiersz (może być jako hard-coded string)

Zadanie 3.6

- Zmodyfikuj poprzednie zadanie tak, aby:
  - Użyć strumienia, jeśli użyty odczyt/zapis był przy pomocy stringów
  - tekst (ten z tabulatorami) był zapisywany bezpośrednio do nowego pliku przekazanego w parametrze output
  - Użyj tego samego strumienia ponownie do wypisania tekstu w konsoli

Zadanie 3.7

- Napisz kilka testów jednostkowych do programu :
  - Parsowania argumentów
    - Sprawdza liczbę wczytanych parametrów
    - sprawdza istnienie Switchy
    - sprawdza zgodność parametrów kluczy z ich wartościami
    - Sprawdza czy rzucony jest odpowiedni wyjątek (*ArgumentNullException*) przy inicjalizacji z parametrem typu null.
    - Sprawdza czy inicjalizacja się powiodła, jeżeli przekazana jest pusta kolekcja

Zadanie 3.8

- Użyj klasy Task do implementacji rozwiązania gdzie:
  - zdanie "Hello {Name}!!!" jest wyświetlane na konsoli co sekundę
  - Kiedy jakiś przycisk na klawiaturze jest wciśnięty, przestań wypisywać "Hello {Name}!!!“, wypisz raz “Bye”

  - użyj CancellationToken.

Zadanie 3.9

- Napisz aplikację wielowątkową która:
  - Generuje losowo 50 liczb z przedziału 100-1000
  - Dla wygenerowanych liczb wykonywane jest współbieżnie obliczenie, czy dana liczba, jest liczbą pierwszą (Task<bool>)
  - Zlicz liczbę znalezionych liczb pierwszych w zadanym zbiorze i wynik wypisz na konsoli.
  - Można użyć statycznej metody z klasy Task do synchronizacji. (WaitAll lub WhenAll)

  - Możesz użyć Parallel.Foreach, Zadań z puli wątków, lub PLINQ

Zadanie 4.1

- Utwórz aplikację WPF do odczytu i zapisu plików tekstowych:
  - W nowym oknie dodaj kontrolki:
    - TextBox – do przekazania świeżki do pliku
    - TextBox – do wyświetlenia zawartości pliku
    - Button – do odczytania pliku
    - Button – do zapisu pluku
  - Do odczytywania i zapisywania plików używaj metody asynchronicznych

Zadanie 4.2

- Stwórz schemat bazy danych dla firmy, o następujących parametrach.
  - W firmie mamy dwie fabryki w różnych lokalizacjach
  - Mamy pracowników, którzy są przypisani do fabryki
  - Mamy samochody/pojazdy, które przypisane są do fabryki
  - Potrzebujemy tabelę z grafikiem przydziału samochodu (dzień/osoba/samochód)
    - Co jeśli samochód jest przypisany do osoby
  - Stwórz tabelę Logs z kolumnami Date, Level, oraz Message.

Zadanie 4.3

- Napisz widok, który wyświetla:
  - Liczbę osób pracujących w każdej fabryce

Zadanie 4.4

- Napisz procedurę składowaną, która:
  - Pozwala zarezerwować samochód na dany dzień
  - Rezerwować można tylko samochody, które pochodzą z tej samej fabryki, co pracownik, który dokonuje rezerwacji

Zadanie 4.5

- Napisz program, który (SqlConnect):
  - Łączy się z bazą danych
  - pobiera listę grup oraz użytkowników
  - Nieaktywnych użytkowników zapisuje do pliku
  - Nieaktywnych użytkowników usuwa z bazy danych za
  - DELETE FROM Users WHERE Id =2
  - pomocą procedury składowanej (z parametrem Id)
  - Pobierz dane z wielu tabel (rozszerz podpunkt b) oraz wypisz je na konsolę

Zadanie 4.6

- Napisz metodę, która za pomocą EF wykonuje następującą akcję:
  - Dodaje nową grupę do bazy
  - Usuwa użytkowników nieaktywnych
  - Wypisz na konsole użytkowników i grupy do nich przypisane

Zadanie 4.7

- Napisz program, który: (użyj programu **3.2**)
  - Po przekazaniu switcha -import, wczytuje plik .csv i zapisuje go do bazy danych z użyciem EF
  - Po przekazaniu switcha -export oraz ścieżki, pobiera dane z tabeli i zapisuje w pliku .csv

Zadanie 5.0a

- Stwórz aplikację do zarządzania uprawnieniami użytkowników
  - Tabele:
    - Users (Id, Name, Password, IsActive),
    - Groups (Id, Name),
    - Permissions (Id, Name, Description)
  - W pierwszym oknie można wybrać, co będziemy robić, po czym otworzy się kolejne okno
  - Pierwsze okienko z DataGridView do zarządzania użytkownikami (Load, Reset, Save)
    - Następnie nadawanie użytkownikom uprawnień – przydzielanie ich do ról (osobne okno)
    - Niech w DataGridView za pmocą combinacji ctrl+r zaznaczone wiersze zostaną powielone w gridze

Zadanie 5.0b

- Aplikacja do zarządzania uprawnieniami użytkowników
  - Dodaj Menu z opcjami
    - Zamknij
    - Zapisz
    - Eksportuj (tylko użytkowników w tabeli z użytkownikami)
  - Kolejny widok umożliwia zarządzania rolami i nadawanie im uprawnień
    - Mamy dwie listy:
      - pierwsza (ListBox) z rolami
      - druga (CheckedListBox) z uprawnieniami
    - lista z uprawnieniami jest odświeżana przy zmianie wyboru roli
  - Ekran logowania, jeśli czas pozwoli