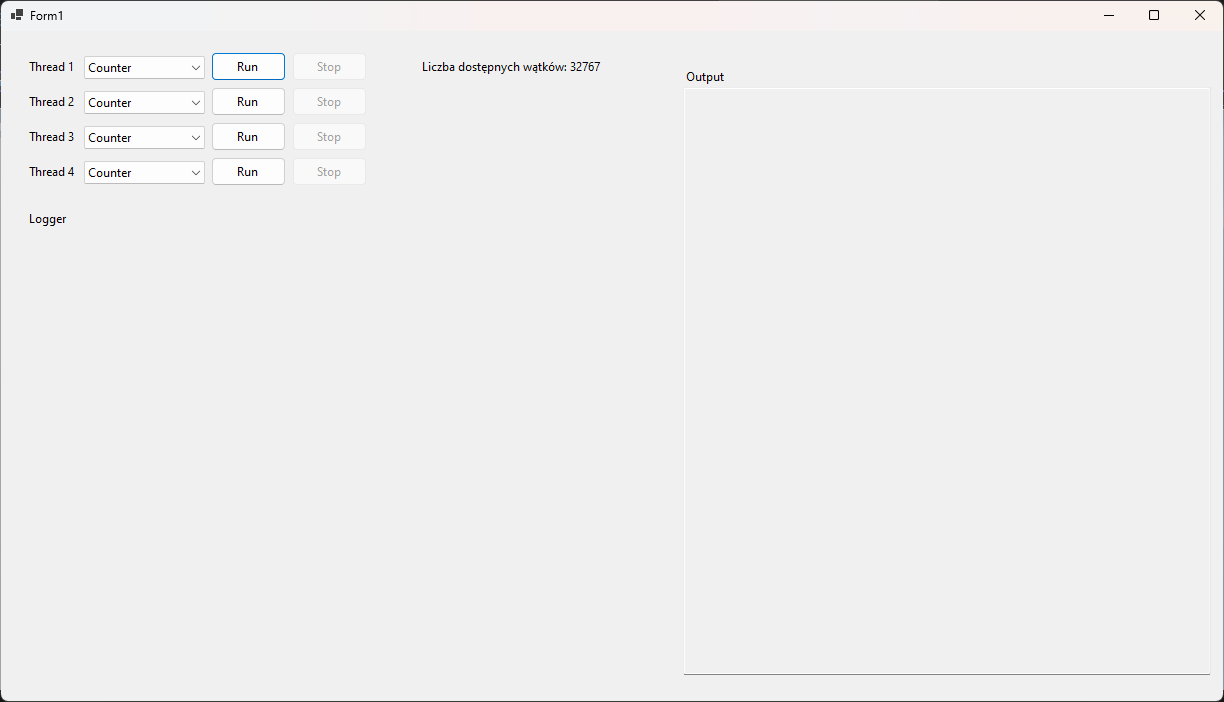
Pojęcia

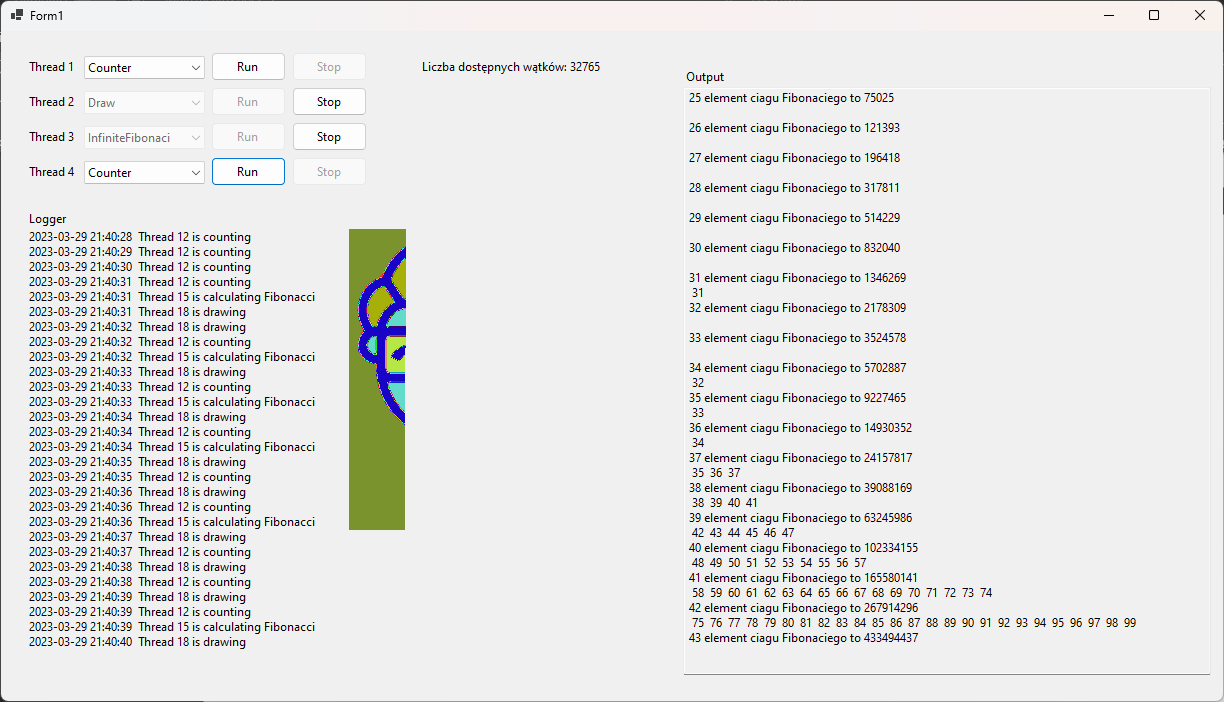
Wielowątkowość jest to technika programowania, która umożliwia jednoczesne wykonywanie wielu wątków w ramach jednego procesu. Wątki są to sekwencje wykonywania kodu programu, które mogą działać równolegle i niezależnie od siebie.

Threads i Task są dwoma różnymi sposobami realizowania wielowątkowości.  
Thread jest to podstawowy obiekt wątku w C#. Tworząc nowy wątek za pomocą klasy Thread, trzeba samodzielnie zarządzać jego cyklem życia, włączając tworzenie, startowanie, zatrzymywanie i usuwanie wątku. Klasa Thread daje programiście pełną kontrolę nad procesem wielowątkowym, ale wymaga bardziej skomplikowanego kodu i może prowadzić do problemów z synchronizacją dostępu do zasobów współdzielonych.  
Task jest to klasa, która ułatwia tworzenie i zarządzanie wątkami. Task automatycznie zarządza cyklem życia wątków i umożliwia programiście asynchroniczne wykonywanie operacji.  
  
Metoda .Invoke()isłuży do wykonania operacji w wątku który utworzył daną operacje z wątku, który go nie utworzył. Gdy operacje są wykonywane w wątku np. nie-UI, używając metody .Invoke(), można bezpiecznie zaktualizować kontrolki UI. Metoda ta zapewnia, że kod wykonywany jest na wątku UI i chroni przed wywołaniem wyjątku związanego z dostępem do kontrolki z innego wątku niż ten, w którym został utworzony.

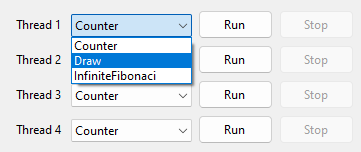
Zastosowanie Programu

Program służy do demonstracji programowania z użyciem wielowątkowości w c#. Program generuje bitmapę z losowymi kolorami na podstawie podanego pliku bmp. Wypisuje kolejne liczby ciągu Fibonacciego oraz tworzy prosty licznik od 0 do 100.

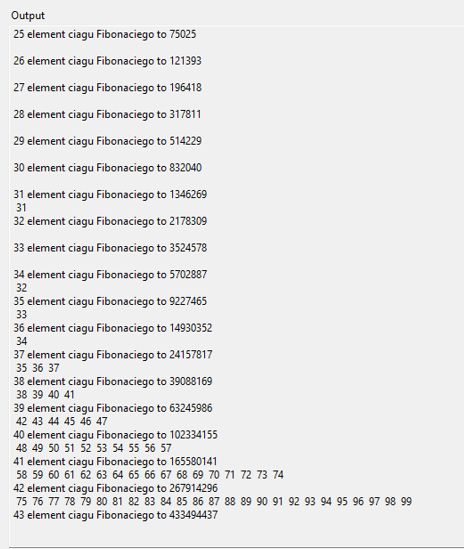
Wygląd Programu



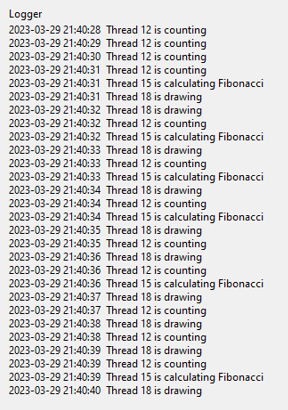
Opis Programu

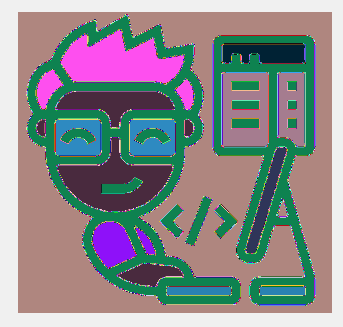


Wybieramy zadania dla każdego wątku z listy combobox, po kliknieciu Run przycisk staje się nie dostępny do ponownego klikniecia, z kolei przycisk stop staje się dostępny – zatrzymuje on wybrany watek.

Output:

TextBox - wypisywane są tu wyniki obliczeń.

Logger – Textbox – tutaj są wypisywane logi z informacja jakie id wątku jakie wykonuje zadanie.



pictureBox – tutaj rysowany jest rysunek.  
  
threadLabel1, threadLabel2, threadLabel3, threadLabel4: kontrolki Label wyświetlające nazwy wątków.

RunThread1, RunThread2, RunThread3, RunThread4: kontrolki Button uruchamiające wątki.

textBox1: TextBox kontrolka, do której wypisywane są wyniki obliczeń wątków

comboBox1, comboBox2, comboBox3, comboBox4: ComboBox kontrolki, które umożliwiają wybór jednej z kilku opcji akcji wątku.

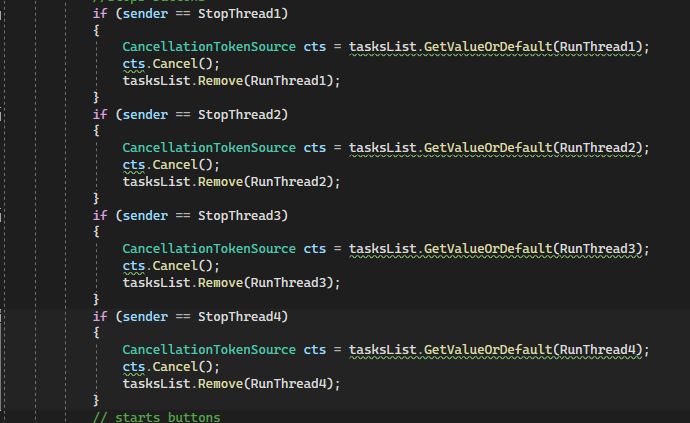
pictureBox1: PictureBox kontrolka wyświetlająca generowany obraz.

outputLabel: Label kontrolka, która wyświetla wyniki działania programu.

LoggerTextbox: TextBox kontrolka, która wyświetla dziennik zdarzeń.

StopThread1, StopThread2, StopThread3, StopThread4: Button kontrolki zatrzymującej wątki.

AvaibleThreadsLabel: Label kontrolka, która wyświetla liczbę dostępnych wątków.

Opis kodu

Ten kod odpowiada za obsługę przycisków w programie. Jeśli przycisk "StopThreadX" zostanie kliknięty, gdzie "X" to liczba od 1 do 4, to kod zatrzymuje wątek, który został uruchomiony wcześniej przez przycisk "RunThreadX". Jeśli przycisk "RunThreadX" zostanie kliknięty, to kod rozpoczyna obliczenia związane z wybraną operacją i czeka na ich zakończenie. Podczas trwania obliczeń odpowiednie przyciski są wyłączane, a po zakończeniu operacji ponownie włączane.

Ten kod definiuje metodę RunCalculation, która wywołuje jedno z trzech działań na nowym wątku: zliczanie, rysowanie lub obliczanie ciągu Fibonacciego. Metoda tworzy nowy wątek i dodaje go do listy wątków, aby móc kontrolować ich anulowanie. Następnie czeka na zakończenie zadania i usuwa wątek z listy. W końcu zwraca wartość logiczną wskazującą, czy operacja została zakończona pomyślnie.  
  
  
  
  
  
  
  
  
Ten kod definiuje metodę PrintLog, która przyjmuje wiadomość i drukuje ją w polu tekstowym LoggerTextbox z datą i bieżącą godziną. Jeśli wywołanie tej metody nie pochodzi z wątku głównego, to używa metody Invoke do przekazania drukowania wiadomości na wątek główny.