ZADANIE

Projekt 2

do samodzielnego wykonania

Dane jest poniższa implementacja algorytmu badania czy zadana liczba jest pierwsza:

```
bool IsPrime(BigInteger Num)
{
  if (Num < 2) return false;
  else if (Num < 4) return true;
  else if (Num % 2 == 0) return false;
  else for (BigInteger u = 3; u < Num / 2; u += 2)
           if (Num % u == 0) return false;
  return true;
}
```

Celem projektu jest zaproponowanie bardziej efektywnego algorytmu przy zachowaniu niezmienionego interfejsu podprogramu. Przeprowadzić analizę za pomocą instrumentacji i pomiarów czasu. Przyjąć, że operacją dominującą jest dzielenie modulo (%).

W sprawozdaniu przedstawić dla obu algorytmów:

- kod źródłowy przed instrumentacją
- kod źródłowy po instrumentacji
- zebrane wyniki w postaci tekstu i wykresów
- wnioski z analizy zebranych danych (ocena złożoności)

Badanie przeprowadzić dla następującego zbioru punktów pomiarowych (liczb pierwszych):
{ 100913, 1009139, 10091401, 100914061, 1009140611, 10091406133, 100914061337, 1009140613399 }

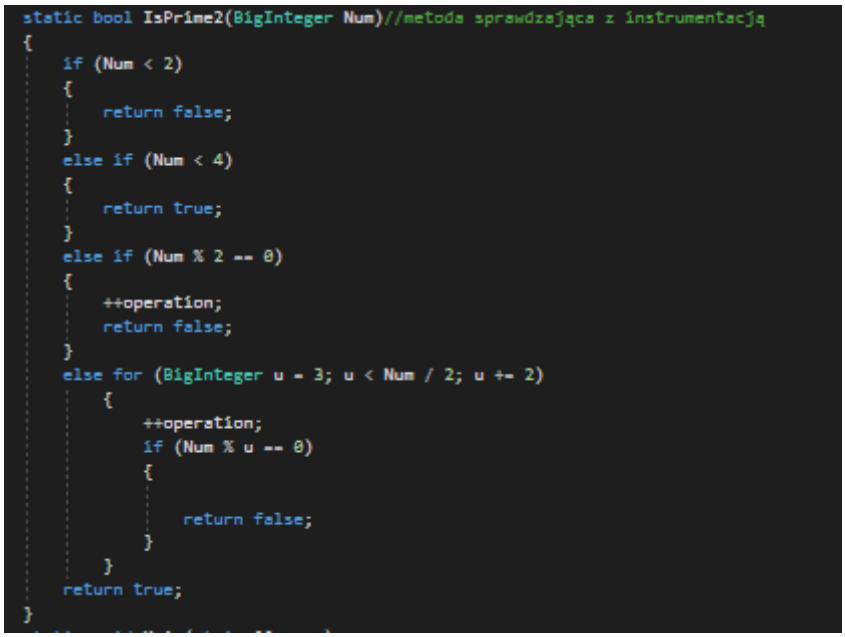
I.Metoda sprawdzająca czy dana liczba jest liczbą pierwszą(przykładowa)

Metoda przykładowa bez instrumentacji

```
static bool IsPrime(BigInteger Num)//metoda sprawdzająca bez instrumentacji
{
    if (Num < 2) return false;
    else if (Num < 4) return true;
    else if (Num % 2 == 0) return false;
    else for (BigInteger u = 3; u < Num / 2; u += 2)

            if (Num % u == 0) return false;
    return true;
}
```

Metoda przykładowa z instrumentacją



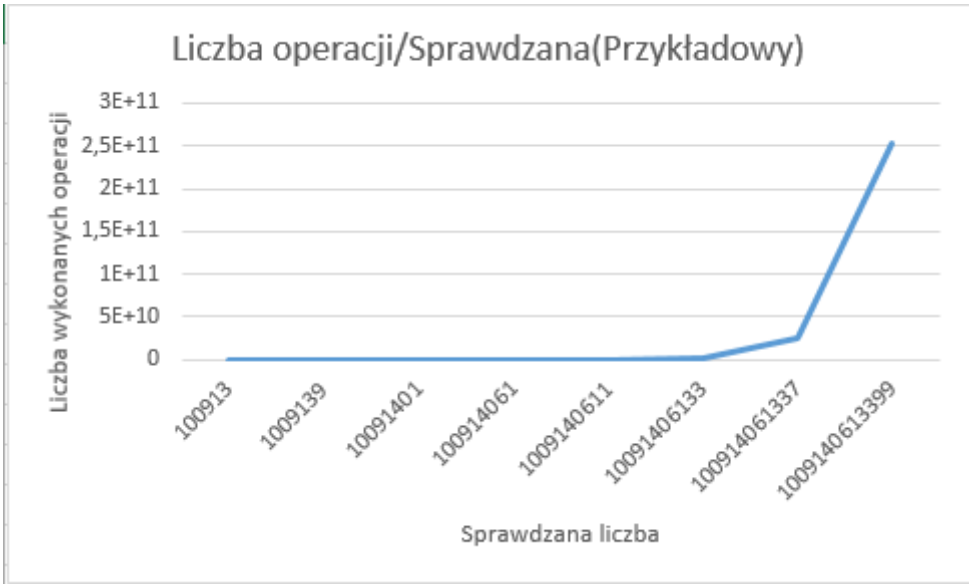
```
static bool IsPrime2(BigInteger Num)//metoda sprawdzająca z instrumentacją
{
    if (Num < 2)
    {
        return false;
    }
    else if (Num < 4)
    {
        return true;
    }
    else if (Num % 2 == 0)
    {
        ++operation;
        return false;
    }
    else for (BigInteger u = 3; u < Num / 2; u += 2)
        {
            ++operation;
            if (Num % u == 0)
            {

                return false;
            }
        }
    return true;
}
```

Dane zgromadzone z ośmiu wykonanych pomiarów

Dane dla algorytmu przykładowego			
l.p.	Sprawdzana liczba	Czas (s)	Liczba wiodących operacji
1	100913	0,00329	25227
2	1009139	0,02654	252283
3	10091401	0,26911	2522849
4	100914061	3,08172	25228514
5	1009140611	26,96215	252285151
6	10091406133	626,52217	2522851532
7	100914061337	6265,2217	25228515322
8	1009140613399	62652,21701	252285153225

Instrumentacja w porównaniu do wielkości sprawdzanej liczby



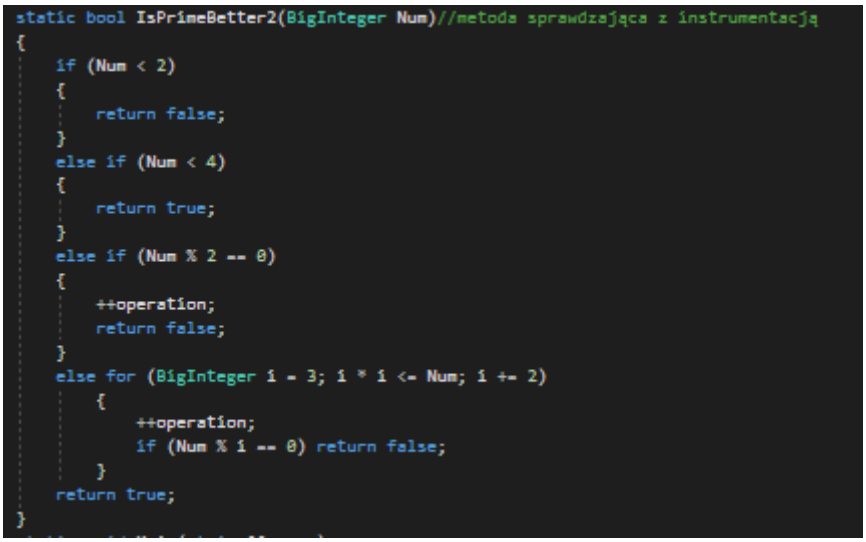
Czas w porównaniu do wielkości sprawdzanej liczby

II.Metoda sprawdzająca czy dana liczba jest liczbą pierwszą(przyzwoita)

Metoda przyzwoita bez instrumentacji

```
static bool IsPrimeBetter(BigInteger Num)//metoda sprawdzająca bez instrumentacji
{
    if (Num < 2) return false;
    else if (Num < 4) return true;
    else if (Num % 2 == 0) return false;
    for (BigInteger i = 3; i * i <= Num; i += 2)
        if (Num % i == 0) return false;
    return true;
}
```

Metoda przyzwoita z instrumentacją

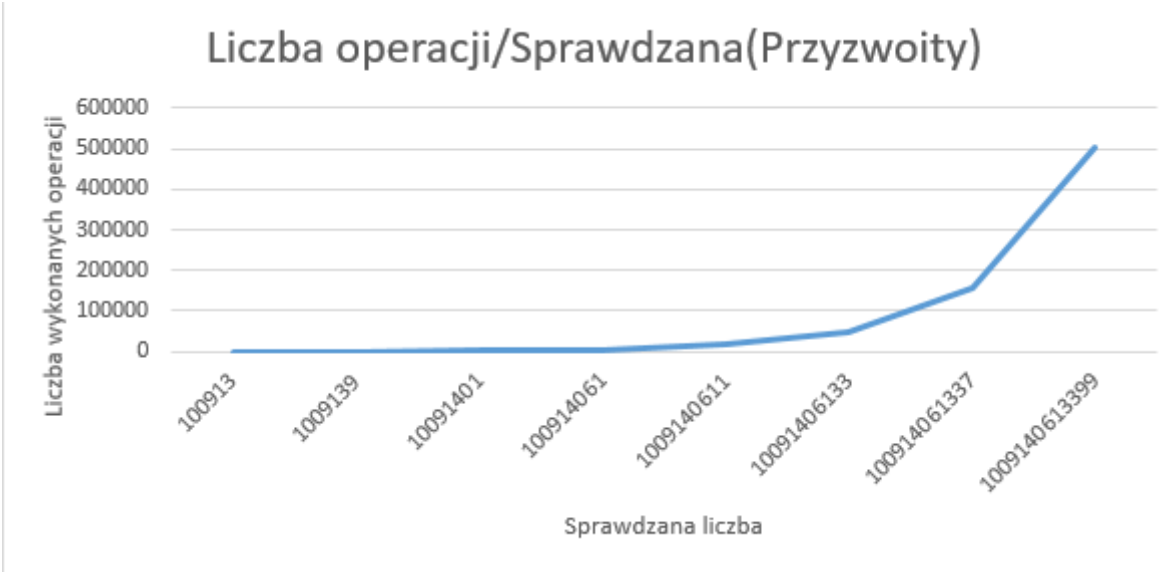


```
static bool IsPrimeBetter2(BigInteger Num)//metoda sprawdzająca z instrumentacją
{
    if (Num < 2)
    {
        return false;
    }
    else if (Num < 4)
    {
        return true;
    }
    else if (Num % 2 == 0)
    {
        ++operation;
        return false;
    }
    else for (BigInteger i = 3; i * i <= Num; i += 2)
        {
            ++operation;
            if (Num % i == 0) return false;
        }
    return true;
}
```

Dane zgromadzone z ośmiu wykonanych pomiarów

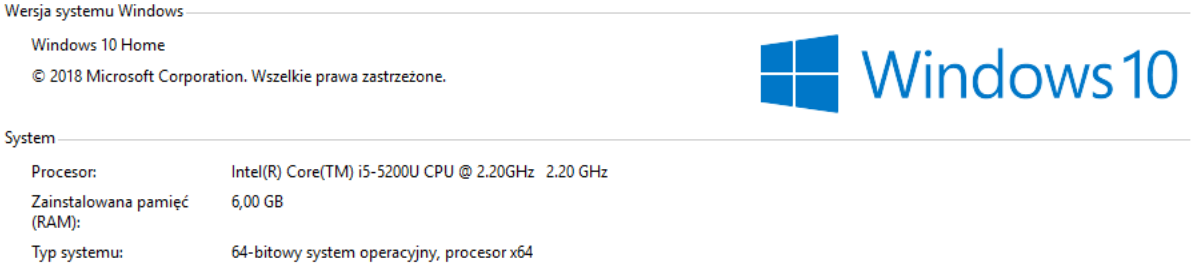
Dane dla algorytmu przyzwoitego			
l.p.	Sprawdzana liczba	Czas(s)	Liczba wiodących operacji
1	100913	0,0004344	158
2	1009139	0,0000564	501
3	10091401	0,0001701	1587
4	100914061	0,0005416	5022
5	1009140611	0,0030102	15882
6	10091406133	0,0138248	50227
7	100914061337	0,0251895	158834
8	1009140613399	0,1434589	502279

Iteracja w porównaniu do wielkości sprawdzanej liczby



Czas w porównaniu do wielkości sprawdzanej liczby

Komputer, na którym zostały wykonane pomiary



Wnioski:

- O ile nie widać znacznej poprawy czasowej między Przykładowym a Przyzwoitym to w momencie większej liczby sprawdzanej owa różnica staje się widoczna, a jak się spodziewamy program będzie również sprawdzał duże liczby.
- Zmiana algorytmu z Przykładowego na Przyzwoity nie wymaga wiele trudu więc szybkim sposobem możemy zoptymalizować działanie naszego programu.
- Liczba wiodących operacji w przypadku algorytmu Przyzwoitego jest mniejsza nawet 500tys. razy(porównując 8-sme pomiary w obu przypadkach).