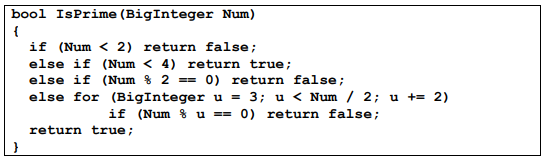
**Algorytmy i struktury danych**

**Projekt 2**

**Cel projektu:**

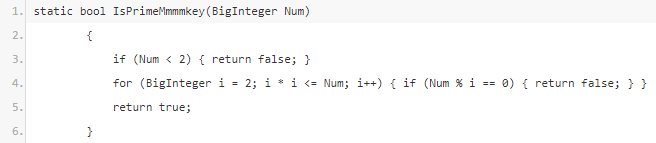
*Zaproponowanie bardziej efektywnego algorytmu, niż podany poniżej, przy zachowaniu niezmienionego interfejsu podprogramu. Przeprowadzić analizę za pomocą instrumentacji i pomiarów czasu. Przyjąć, że operacją dominującą  jest dzielenie modulo (%).*



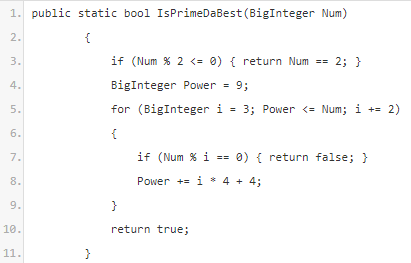
*Liczby wykorzystane w analizie: { 100913, 1009139, 10091401, 100914061, 1009140611,  10091406133, 100914061337, 1009140613399 }*

Przyjmując, iż narzucona metoda sprawdzania, czy liczba jest pierwsza, jest najwolniejsza oraz najmniej optymalna stworzone zostały dwie dodatkowe – nazwane kolejno **przyzwoita** oraz **najlepsza.**

Metoda **przyzwoita** (bez instrumentacji):

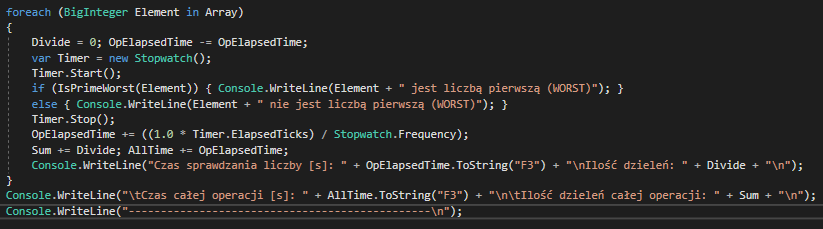


Metoda **najlepsza** (bez instrumentacji):

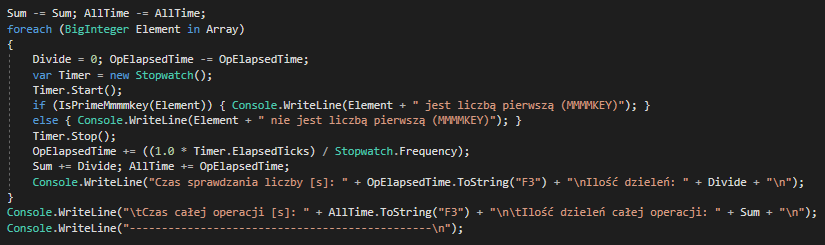


Każda z trzech metod została zaimplementowana do głównej funkcji main, gdzie, za pomocą klasy ***Stopwatch*** przeprowadzono badanie czasu działania metod.

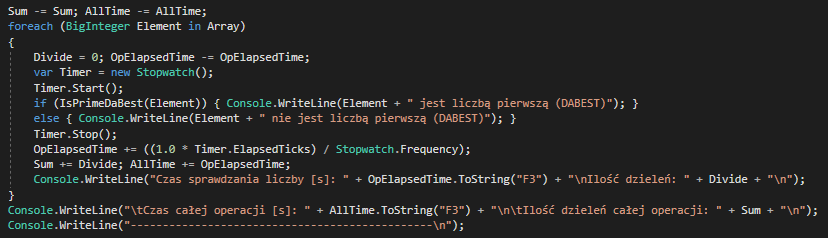
Implementacja metody **najgorszej** (z pomiarem czasu):



Implementacja metody **przyzwoitej** (z pomiarem czasu):

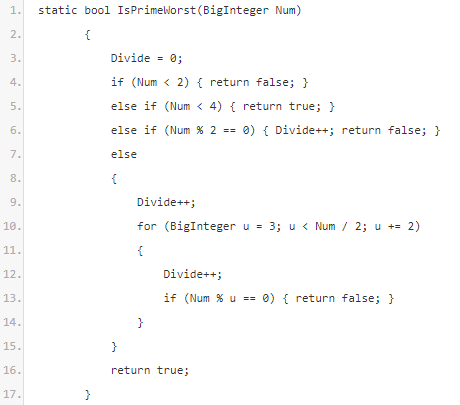


Implementacja metody **najlepszej** (z pomiarem czasu):

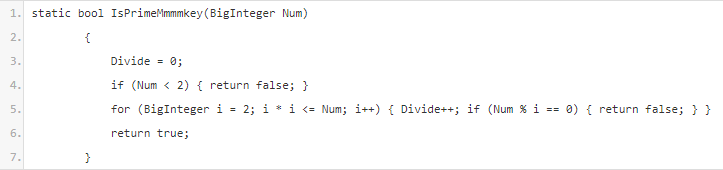


Po wprowadzeniu pomiaru czasu wykonywania się każdej z metod wprowadzono **intrumentację** do każdej z nich, dzięki której zostanie wyświetlona ilość **dzieleń** w każdej z nich. Po implementacji zmiennej ***Divide*** obliczającej tę wartość, kod źródłowy kolejnych funkcji  ***IsPrime*** przedstawia się następująco:

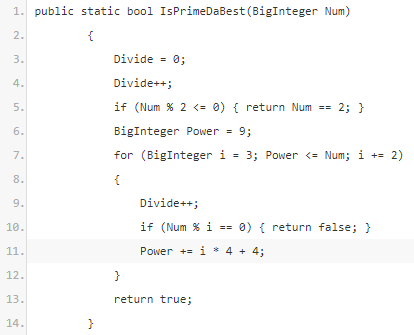
Metoda **najgorsza** (z instrumentacją):



Metoda **przyzwoita** (z instrumentacją):

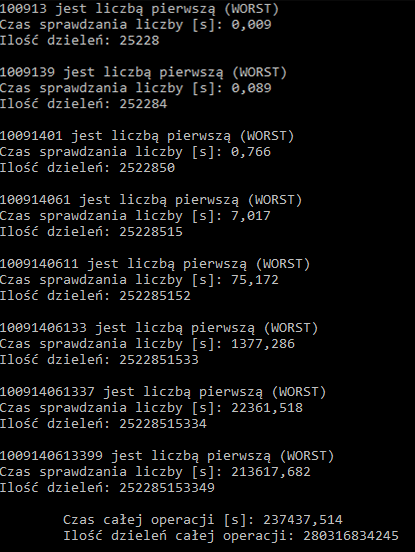


Metoda **najlepsza** (z instrumentacją):

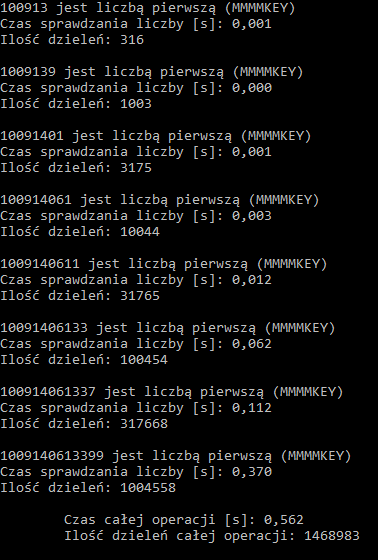


Wyniki wyświetlone w konsoli (**cmd**) po skompilowaniu kodu źródłowego:

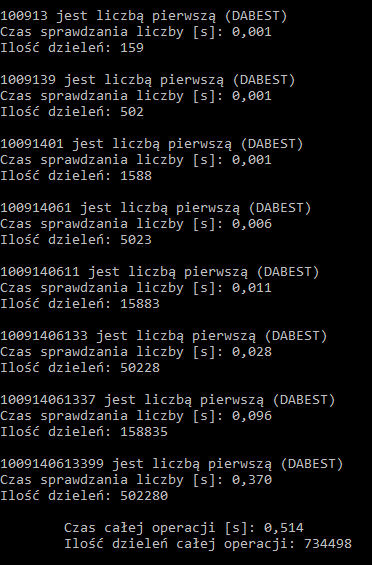
Metoda **najgorsza:**



Metoda **przyzwoita:**



Metoda **najlepsza:**



Wyniki zebrane zostały do tabeli przez wzgląd na **czas** oraz **ilość operacji** (dzieleń) dla każdej z metod:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ilość dzieleń w danej metodzie | | |
| Liczba | Najgorsza | Przyzwoita | Najlepsza |
| 100913 | 25228 | 316 | 159 |
| 1009139 | 252284 | 1003 | 502 |
| 10091401 | 2522850 | 3175 | 1588 |
| 100914061 | 25228515 | 10044 | 5023 |
| 1009140611 | 252285152 | 31765 | 15883 |
| 10091406133 | 2522851533 | 100454 | 50228 |
| 100914061337 | 25228515334 | 317668 | 158835 |
| 1009140613399 | 252285153349 | 1004558 | 502280 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ilość czasu wykonania danej metody [s] | | |
| Liczba | Najgorsza | Przyzwoita | Najlepsza |
| 100913 | 0,009 | 0,001 | 0,001 |
| 1009139 | 0,089 | 0,001 | 0,001 |
| 10091401 | 0,766 | 0,001 | 0,001 |
| 100914061 | 7,017 | 0,003 | 0,006 |
| 1009140611 | 75,172 | 0,012 | 0,011 |
| 10091406133 | 1377,286 | 0,062 | 0,028 |
| 100914061337 | 22361,518 | 0,112 | 0,096 |
| 1009140613399 | 213617,682 | 0,370 | 0,370 |

Na podstawie zebranych w tabelach wyników stwierdzić można, iż dla tak dużych liczb metoda **najgorsza** jest bardzo mało opłacalna, ponieważ dla narzuconej nam tabeli całość operacji trwała **~66 godzin (!)**, gdy pozostałe dwie **ŁĄCZNIE** nie trwały nawet sekundy. Dodać do tego trzeba astronomiczną liczbę operacji dzielenia, która, po zsumowaniu, wyniosła aż **280316834245** dzieleń. Ponownie porównując to z pozostałymi metodami wywnioskować można, iż są one o wiele bardziej optymalne (**przyzwoita** – 1468983 dzieleń, **najlepsza** – 734498 dzieleń) oraz nie wymagają one takiego procentowego wkładu procesora w wykonanie zadania, co przy kompilacji metody **najgorszej** wynosiło około 30%.

Starając się zestawić czas oraz ilość operacji dzielenia wszystkich metod na raz przy użyciu wykresów otrzymamy praktycznie nieczytelne wyniki:

Kontrastujące wyniki metody **najgorszej** sprawiają, że musimy usunąć ją z wykresu skupiając się tylko na metodzie **przyzwoitej** oraz **najlepszej**, co daje nam czytelniejsze wyniki:

Czasowe różnice między metodami **przyzwoitą** a **najlepszą** są prawie niezauważalne, lecz **najlepsza** wykonuje prawie dwa razy mniej operacji dzielenia dla liczby pokroju *1009140613399.*

***Badania przeprowadzono na komputerze wyposażonym w procesor Intel Core i5-4210U przy użyciu programu Microsoft Visual Studio 2017. Projekt został skompilowany bez debugowania z ustawieniami Release/Any CPU.***