

Dokumentacja do projektu nr 7.

Szymon Mrzygłód

Spis treści:

1. Opis problemu.
2. Opis podstawowych algorytmów
 - 2.1. Sprawdzenie poprawności wprowadzonej liczby.
 - 2.2. Sprawdzenie czy podana liczba należy do liczb pierwszych.
 - 2.3. Rozłożenie liczby na czynniki pierwsze.
 - 2.4. Wyznaczenie dzielników liczby.
 - 2.5. Wyznaczenie największego wspólnego dzielnika liczb (NWD).
3. Opis implementacji zaprojektowanego systemu
4. Instrukcja użytkownika
 - 4.1. Wymagania
 - 4.2. Opcje
 - 4.3. Przykładowe skorzystanie z aplikacji
 - 4.3.1. Opcja nr 1 – Liczby Pierwsze
 - 4.3.2. Opcja nr 2 – Rozłóż Liczbę
 - 4.3.3. Opcja nr 3 – Dzielniki Liczby
 - 4.3.4. Opcja nr 4 – Największy Wspólny Dzielnik
 - 4.3.5. Opcja ESC – Zakończ
5. Analiza działania programu oraz czas pracy.
 - 5.1. Wnioski z przeprowadzonych testów.
 - 5.1.1. Wprowadzenie niepoprawnego znaku.
 - 5.1.2. Sprawdzenie czy liczba należy do liczb pierwszych.
 - 5.1.3. Sprawdzenie poprawności rozkładu liczby na czynniki pierwsze.
 - 5.1.4. Sprawdzenie poprawności wyznaczania dzielników liczby oraz NWD dwóch liczb.

1. Opis problemu/zadania.

Program sprawdzający, czy dana liczba naturalna jest pierwsza. Program powinien wczytywać liczbę ze standardowego wejścia i drukować na standardowe wyjście odpowiedni komunikat.

Dodatkowo program rozkłada daną liczbę naturalną na czynniki pierwsze w następujący sposób. Sprawdzamy, czy liczba dzieli się przez 2. Jeżeli tak, to stwierdzamy, że dwójka występuje w jej rozkładzie na czynniki pierwsze, a samą liczbę dzielimy przez 2. Czynność tę powtarzamy, aż liczba przestanie być podzielna przez 2. Następnie powtarzamy tę samą procedurę badając podzielność przez 3, 4, itd., aż rozważana liczba stanie się równa 1.

Ponadto program, dla podanej liczby całkowitej wyświetla jej dzielniki. Przykładowo, dla liczby 21 dzielniki to: 1, 3, 7, 21.

Program ma również wyznaczać największy wspólny dzielnik dwóch liczb.

2. Opis podstawowych algorytmów:

2.1. Sprawdzenie poprawności wprowadzonej liczby.

W klasie „Program” występuje konieczność wielokrotnego wprowadzenia przez użytkownika liczb. W celu sprawdzenia czy wprowadzono prawidłową liczbę wykorzystano blok „try...catch”, który obejmuje cały blok „switch...case” znajdujący się w metodzie „Main” klasy „Program”.

W przypadku wprowadzenia nieprawidłowego znaku np. litery zostanie przechwycony wyjątek. Na konsoli wyświetli się odpowiednia informacja wraz z szczegółowym komunikatem systemowym.

Wyjątek zostanie także zgłoszony w przypadku wprowadzenia zera lub liczb ujemnych gdy należy wprowadzić liczbę naturalną dodatnią. Założono, że w przypadku liczb całkowitych nie należy wprowadzać zera, ponieważ pozbawione sensu jest np. szukanie dzielników takiej liczby – dlatego wprowadzenie takiej liczby również skutkować będzie błędem.

```
if (podanaLiczba <= 0)
{
    throw new System.FormatException();    //Wyłapanie własnego wyjątku.
}
```

2.2. Sprawdzenie czy podana liczba należy do liczb pierwszych.

Pierwszą opcją programu jest sprawdzenie czy podana przez użytkownika liczba należy do liczb pierwszych. Liczba pierwsza to liczba naturalna większa od 1, która ma dokładnie dwa dzielniki naturalne: jedynkę i siebie samą. W aplikacji sprawdza się czy występuje więcej dzielników podanej liczby niż dwa – jeśli tak to wiemy, że liczba nie należy do liczb pierwszych.

W stworzonej metodzie „SprawdzLiczbe” w klasie „LiczbaPierwsza” wykorzystano w tym celu pętlę „for”:

```
int iloscDzielnikow = 0;

for (int i = 2; i < podanaLiczba; i++) //Pominięte dwa dzielniki
                                     1 i "podanaLiczba".
{
    if (podanaLiczba % i == 0)
    {
        iloscDzielnikow++;
        break;
    }
}
```

Przypisana wartość licznika pętli „i” jest równa 2 i wykonywana jest do czasu osiągnięcia przez nią wartości mniejszej od podanej liczby. Takie działanie jest podyktowane brakiem konieczności próby dzielenia podanej liczby przez 1 i siebie samą.

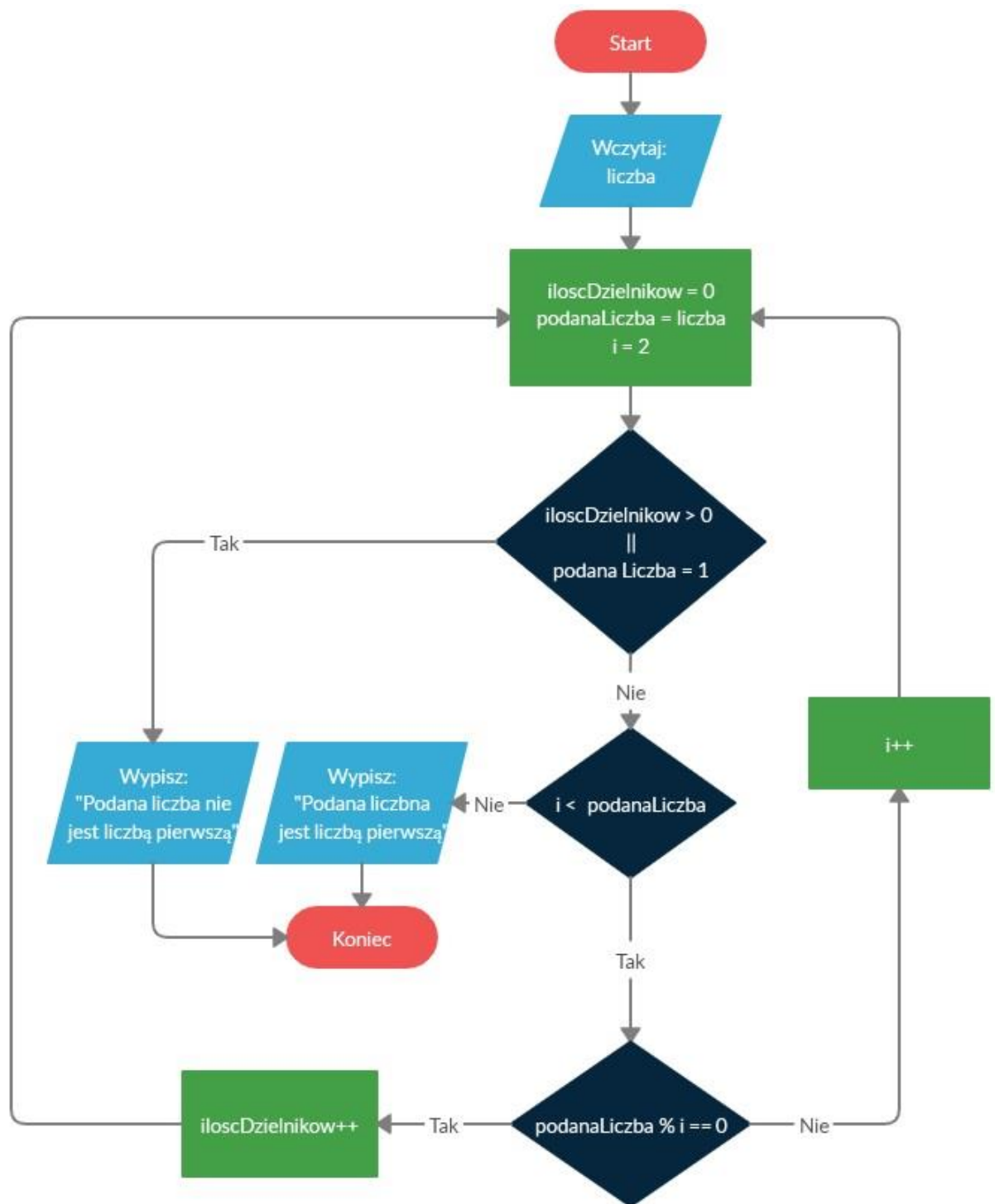
Po przeprowadzonych testach związanych z czasem pracy dopisano w instrukcji warunkowej słowo kluczowe „break”. W przypadku znalezienia pierwszego dzielnika (zmienna „iloscDzielnikow” zmieni wartość na 1) wykonywanie pętli „for” zostanie przerwane. Działanie to jest podyktowane stwierdzeniem faktu iż już przy znalezieniu pierwszego dzielnika z tego przedziału wiemy, że liczba ta nie należy do liczb pierwszych.

Za pomocą instrukcji warunkowej „if...else” sprawdza się czy znalezion dzielnik lub podana liczba jest równa 1.

```
if (iloscDzielnikow > 0 || podanaLiczba == 1)
{
    Console.WriteLine($"Podana liczba {podanaLiczba} nie jest liczbą
                       pierwszą.\n");
}
else
{
    Console.WriteLine($"Podana liczba {podanaLiczba} jest liczbą
                       pierwszą.\n");
}
```

W przypadku spełnienia warunku: „iloscDzielnikow > 0 || podanaLiczba==1” zostaje wypisany komunikat informujący, że podana liczba nie należy o liczb pierwszych. W przeciwnym wypadku zostajemy poinformowani, że wprowadzona liczba jest liczbą pierwszą.

W celu zobrazowania działania algorytmu – poniżej zamieszczono schemat blokowy:



2.3. Rozłożenie liczby na czynniki pierwsze.

Drugą funkcjonalnością programu jest rozłożenie liczby na czynniki pierwsze. W tym celu stworzono nową metodę w klasie „LiczbaPierwsza” o nazwie „RozlozLiczbe”. Wykorzystano w tym celu pętlę „while”.

```
int nowaLiczba = podanaLiczba;
int dzielnik = 2;          //Rozkład liczby zaczynamy od dzielenia przez 2.
while (nowaLiczba != 1)
{
    if (nowaLiczba % dzielnik == 0)
    {
        Console.WriteLine($"{nowaLiczba,10}|{dzielnik}\t");
        nowaLiczba = nowaLiczba / dzielnik;
    }
    else
    {
        dzielnik++;
    }
}
Console.WriteLine($"{1,10}|");
```

Tym razem podana przez użytkownika liczba jest przypisywana nowej zmiennej typu integer o nazwie „nowaLiczba”, która następnie zostaje dzielona przez liczbę 2 do momentu gdy nie ma już możliwości podzielić jej przez tę liczbę bez reszty. Taka sama sytuacja występuje w przypadku kolejnych prób dzielenia przez 3, 4 itd. - do momentu gdy „nowaLiczba” przyjmie wartość równą 1. Przy każdej udanej próbie dzielenia bez reszty zostaje wyświetlony komunikat (w „graficznej” formie), informujący przez jaką liczbę podzielono i jaka jest aktualnie przypisana wartość do zmiennej „nowaLiczba”.

2.4. Wyznaczenie dzielników liczby.

W założeniach projektu występuje konieczność wyznaczenia dzielników podanej liczby. Po wprowadzeniu przez użytkownika liczby całkowitej program wypisze jej dzielniki. Zadanie to wykonuje metoda w klasie „LiczbaPierwsza” o nazwie „WyznaczDzielnikiLiczby”, która zwraca tablicę typu integer.

```
int wielkosc = 0;          //Nowa zmienna "wielkość" potrzeba do stworzenia tablicy.
int wartoscBezwzgledna = Math.Abs(liczba); //Konieczne przy liczbach ujemnych.
for (int i = 1; i <= wartoscBezwzgledna; i++)
{
    if (wartoscBezwzgledna % i == 0)
    {
        wielkosc++;
    }
}
```

```

int[] tablica = new int[wielkosc];
int indeks = 0;

for (int i = 1; i <= wartoscBezwzgledna; i++)
{
    if (wartoscBezwzgledna % i == 0)
    {
        tablica[indeks] = i;
        Console.Write(" " + tablica[indeks]);
        indeks++;
    }
}

```

Początkowo zostaje zadeklarowana nowa zmienna typu integer o nazwie „wielkosc”. Jest ona konieczna w celu określenia rozmiaru tablicy, do której zostaną wpisane kolejne dzielniki wprowadzonej liczby. W pierwszej pętli „for” sprawdza się ile dzielników ma dana liczba i odpowiednią wartość przypisuje się do zmiennej „wielkosc”. Dzięki tej informacji zostaje utworzona tablica typu integer o odpowiedniej ilości elementów. Dzielniki liczby w niniejszym programie są wyznaczone z wartości bezwzględnej liczby podanej.

Druga pętla „for” ma za zadanie wpisać kolejno wartości dzielników do utworzonej tablicy. Stworzono w tym celu zmienną typu integer o nazwie „indeks” o wartości początkowej równej 0, który odpowiada zerowemu indeksowi tablicy. W przypadku spełnienia warunku `wartoscBezwzgledna % i == 0` zostaje wpisana do tablicy wartość dzielnika liczby pod indeksem 0, który po wykonaniu tej operacji zwiększy swoją wartość o 1. Czynność ta będzie powtarzana do momentu wypełnienia całej tablicy wartościami dzielników. Pętla ta odpowiada również za wyświetlenie na konsoli wpisanych do tablicy dzielników.

2.5. Wyznaczenie największego wspólnego dzielnika liczb (NWD).

Ostatnią funkcjonalnością programu jest wyznaczenie największego wspólnego dzielnika obu wprowadzonych liczb całkowitych. W klasie „LiczbaPierwsza” stworzono kolejną metodę o nazwie „WyznaczNWD”, która jako argumenty przyjmuje dwie tablice typu integer.

Wywołując metodę „WyznaczNWD” jako jej argumenty podano tablicę zwrócone przez metodę „WyznaczDzielnikLiczby”.

```

int NWD = pierwszaTablica[0]; //Przyjęto, że NWD to liczba z zerowego
                             indeksu.
for (int i = 0; i < pierwszaTablica.Length; i++)
{
    for (int j = 0; j < drugaTablica.Length; j++)
    {
        if (pierwszaTablica[i] == drugaTablica[j])
        {

```

```

        NWD = pierwszaTablica[i];           //Przypisanie nowej wartości NWD w
                                           przypadku znalezienia większego
                                           dzielnika.
    }
}
Console.WriteLine("\n\n NWD = " + NWD);

```

Początkowo założono, że NWD odpowiada wartości przypisanej do zerowego elementu „pierwszejTablicy”. Dzielniki w tablicach zostały wpisane rosnąco. Dzięki wykorzystaniu zagnieżdżonych pętli „for” zostaje sprawdzone czy istnieje większy od 1 wspólny dzielnik. Dla kolejnych elementów pierwszejTablicy sprawdza się czy występuje taki sam w drugiejTablicy, jeśli tak - wartość NWD zostaje nadpisana. Po wykonaniu całej operacji zostaje wyświetlony komunikat wypisujący wartość największego wspólnego dzielnika obu liczb.

3. Opis implementacji zaprojektowanego systemu.

Program został napisany w języku C#. Kod został napisany w środowisku Visual Studio Community 2019. W celu uruchomienia aplikacji niezbędny jest zainstalowany pakiet .NET Framework, który ówczśnie jest standardowo zainstalowany w systemie operacyjnym Windows firmy Microsoft. Jednak jeśli pakiet nie został zainstalowany konieczne jest jego doinstalowanie. W głównej mierze aplikacja działa na danych liczbowych typu „integer” oraz tekstowych typu „string”. Program można uruchomić poprzez uruchomienie pliku Projekt.exe lub np. w środowisku typu Visual Studio.

4. Instrukcja użytkowania.

4.1. Wymagania.

Do uruchomienia aplikacji niezbędny jest zainstalowany pakiet .NET Framework.

4.2. Opcje.

Po uruchomieniu aplikacji ukaże się menu główne.

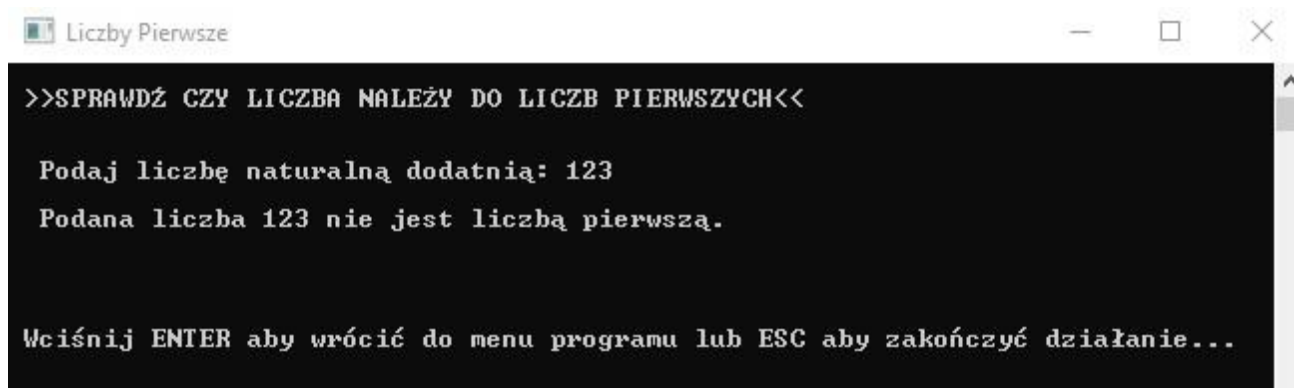


Użytkownik ma do wyboru jedną z 5 opcji, które są uruchomiane poprzez wciśnięcie odpowiedniego klawisza tj. cyfry od 1 do 4 lub ESC.

4.3. Przykładowe skorzystanie z systemu.

4.3.1. Opcja nr 1 – Liczby Pierwsze

W przypadku wybrania opcji nr 1 zostaniemy poproszeni o podanie liczby naturalnej dodatniej. Po wpisaniu odpowiedniej liczby i zatwierdzeniu klawiszem ENTER dostaniemy informację zwrotną czy podana liczba należy do liczb pierwszych.



```
>>SPRAWDŹ CZY LICZBA NALEŻY DO LICZB PIERWSZYCH<<

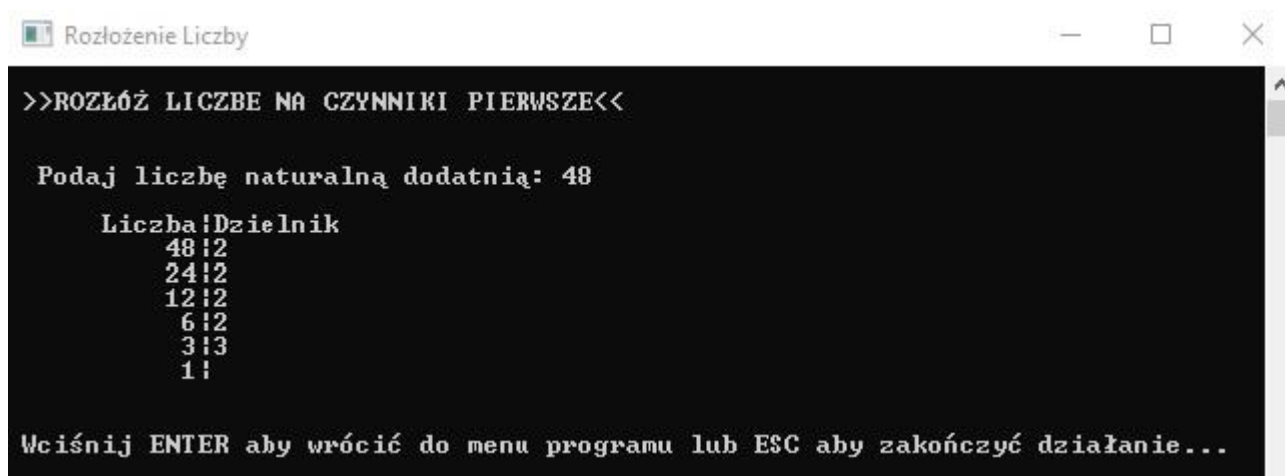
Podaj liczbę naturalną dodatnią: 123
Podana liczba 123 nie jest liczbą pierwszą.

Wciśnij ENTER aby wrócić do menu programu lub ESC aby zakończyć działanie...
```

Po wykonaniu zadania mamy do wyboru dwie opcje – wrócić do menu (ENTER) lub zakończyć działanie programu (ESC).

4.3.2. Opcja nr 2 – Rozłóż Liczbę.

Opcja druga ma za zadanie rozłożyć liczbę na czynniki pierwsze. Po jej wybraniu zostajemy poproszeni o podanie liczby naturalnej dodatniej. Podana liczba zostaje rozłożona na czynniki pierwsze a efekt tego jest widoczny w oknie konsoli.



```
>>ROZŁÓŻ LICZBE NA CZYNNIKI PIERWSZE<<

Podaj liczbę naturalną dodatnią: 48


Liczba:Dzielnik
48:2
24:2
12:2
6:2
3:3
1:1

Wciśnij ENTER aby wrócić do menu programu lub ESC aby zakończyć działanie...
```

Po wykonaniu zadania mamy do wyboru dwie opcje – wrócić do menu (ENTER) lub zakończyć działanie programu (ESC).

4.3.3. Opcja nr 3 – Wyznacz Dzielniki Liczby.

Trzecią opcją do wyboru jest wyznaczenie dzielników danej liczby całkowitej. Po jej wybraniu będzie proszeni o wprowadzenie liczby całkowitej. Po wykonaniu tego działania program wyświetli w konsoli wszystkie dzielniki danej liczby.



```
>>WYZNACZ DZIELNIKI LICZBY<<

Podaj liczbę całkowitą: 12

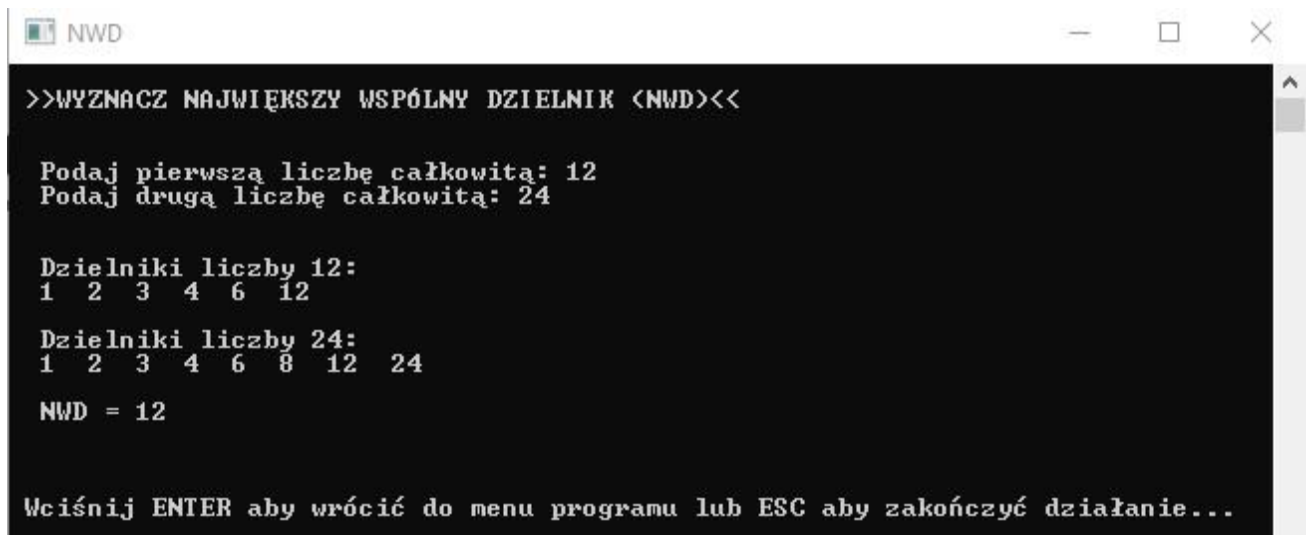
Dzielniki liczby 12:
1 2 3 4 6 12

Wciśnij ENTER aby wrócić do menu programu lub ESC aby zakończyć działanie...
```

Po wykonaniu zadania mamy do wyboru dwie opcje – wrócić do menu (ENTER) lub zakończyć działanie programu (ESC).

4.3.4. Opcja nr 4 – Największy Wspólny Dzielnik

Po wybraniu opcji nr 4 będziemy proszeni o podanie dwóch liczb całkowitych. Program wyświetli dzielniki obu liczb oraz ich największy wspólny dzielnik (NWD).



```
>>WYZNACZ NAJWIĘKSZY WSPÓLNY DZIELNIK <NWD><<

Podaj pierwszą liczbę całkowitą: 12
Podaj drugą liczbę całkowitą: 24

Dzielniki liczby 12:
1 2 3 4 6 12

Dzielniki liczby 24:
1 2 3 4 6 8 12 24

NWD = 12

Wciśnij ENTER aby wrócić do menu programu lub ESC aby zakończyć działanie...
```

Tak jak w poprzednich przypadkach, również tutaj po wykonaniu mamy dwie opcje do wyboru tj. wróć do menu lub zakończ działanie programu.

4.3.5. Opcja ESC – Zakończ

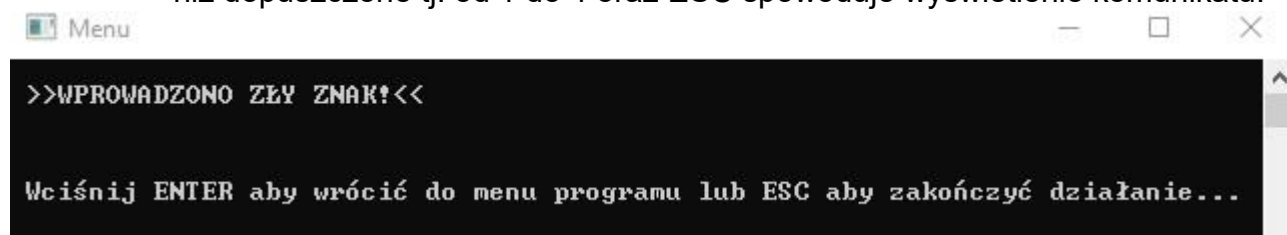
Jeśli wciśniemy klawisz ESC w menu programu to jego działanie się zakończy.

5. Analiza działania programu.

5.1. Wnioski z przeprowadzonych testów.

5.1.1. Wprowadzenie niepoprawnego znaku.

Wprowadzenie w menu programu innego znaku (litery, liczby większe od 4 itd.) niż dopuszczone tj. od 1 do 4 oraz ESC spowoduje wyświetlenie komunikatu:



Jest to spowodowane ominięciem całego bloku „switch...case” i wykonaniem instrukcji zaraz po nim, tzn:

```
//Nie wybranie odpowiedniej pozycji w menu spowoduje ominięcie instrukcji  
"Switch...Case"  
Console.Clear();  
Console.WriteLine("\n >>WPROWADZONO ZŁY ZNAK!<<");
```

WrocLubZakoncz:

```
Console.WriteLine("\n\n\n Wciśnij ENTER aby wrócić do menu programu lub ESC aby  
zakończyć działanie... ");
```

```
ConsoleKeyInfo wrocLubZakoncz = Console.ReadKey();  
if (wrocLubZakoncz.Key == ConsoleKey.Enter)  
{  
    Console.Clear();  
    goto Menu;  
}  
else if (wrocLubZakoncz.Key == ConsoleKey.Escape)  
{  
    Environment.Exit(0);  
}  
else  
{  
    goto WrocLubZakoncz;  
}
```

Błędy związane z wprowadzeniem złego znaku zostają również sygnalizowane w przypadku wejścia do danej opcji programu. Cały blok „switch...case” jest objęty instrukcją „try...catch”, która po wyłapaniu wyjątku z klasy System.Exception (np. nieprawidłowy znak, przekroczony zakres danego typu liczby itd.) poinformuje o tym oraz wyświetli obszerny komunikat systemowy.

```
Liczy Pierwsze

>>SPRAWDŹ CZY LICZBA NALEŻY DO LICZB PIERWSZYCH<<

Podaj liczbę naturalną dodatnią: A

>>PODANO NIEPRAWIDŁOWY ZNAK!<<

Komunikat systemowy: System.FormatException: Nieprawidłowy format ciągu wejściowego.
   w System.Number.StringToNumber(String str, NumberStyles options, NumberBuffer& number, NumberFormatInfo info, Boolean parseDecimal)
   w System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info)
   w System.Int32.Parse(String s)
   w Projekt.Program.Main(String[] args) w C:\Users\Użytkownik\Desktop\1-WSB (INFORMATYKA)\Podstawy programowania\PROJEKT ZALICZENIOWY\Projekt\Program.cs: wiersz 37

Wciśnij ENTER aby wrócić do menu programu lub ESC aby zakończyć działanie...
```

Założono, że nie będzie możliwości wprowadzenia liczby równej 0 w żadnej z wybranych opcji. Dodatkowo w opcji nr 1 – Liczby Pierwsze oraz opcji nr 2 – Rozłóż Liczbę błąd będzie również sygnalizowany w przypadku wprowadzenia liczb ujemnych.

```
Liczy Pierwsze

>>SPRAWDŹ CZY LICZBA NALEŻY DO LICZB PIERWSZYCH<<

Podaj liczbę naturalną dodatnią: -1

>>PODANO NIEPRAWIDŁOWY ZNAK!<<

Komunikat systemowy: System.FormatException: Format jednego ze zidentyfikowanych elementów był nieprawidłowy.
   w Projekt.LiczbaPierwsza.SprawdzLiczbe(Int32 podanaLiczba) w C:\Users\Użytkownik\Desktop\1-WSB (INFORMATYKA)\Podstawy programowania\PROJEKT ZALICZENIOWY\Projekt\LiczbaPierwsza.cs: wiersz 16
   w Projekt.Program.Main(String[] args) w C:\Users\Użytkownik\Desktop\1-WSB (INFORMATYKA)\Podstawy programowania\PROJEKT ZALICZENIOWY\Projekt\Program.cs: wiersz 40

Wciśnij ENTER aby wrócić do menu programu lub ESC aby zakończyć działanie...
```

Dzieje się tak dzięki wyłapaniu własnego wyjątku:

```
if (podanaLiczba <= 0)
{
    throw new System.FormatException(); //Wyłapanie własnego wyjątku.
}
```

oraz:

```
if (liczba == 0)
{
    throw new System.FormatException();
}
```

Po wykonaniu testów związanych z próbą wprowadzenia nieprawidłowego znaku lub nieprawidłowej liczby stwierdzam, że przewidziane przeze mnie możliwości błędnego wprowadzania znaków lub liczb zostały odpowiednio obsłużone - program działa prawidłowo.

5.1.2. Sprawdzenie czy liczba należy do liczb pierwszych.

Aby sprawdzić czy program prawidłowo wskazuje liczby pierwsze została użyta pętla „for” w celu wyświetlenia w tym przypadku wszystkich liczb pierwszych z zakresu od 1 do 100.

```
for (int podanaLiczba = 1; podanaLiczba <= 100; podanaLiczba++)
{
    LiczbaPierwsza liczba = new LiczbaPierwsza();
    liczba.SprawdzLiczbe(podanaLiczba);
}
```

Aby wyniki testu były wyświetlone w czytelniejszy sposób zmieniono oraz za komentowano część kodu odpowiedzialnego za wypisanie komunikatu informującego czy dana liczba należy bądź nie do liczb pierwszych.

```
if (iloscDzielnikow > 0 || podanaLiczba == 1)
{
    //Console.WriteLine($" \n Podana liczba {podanaLiczba} nie jest liczbą
    pierwszą.\n");
}
else
{
    Console.WriteLine($" \n Podana liczba {podanaLiczba} jest
    liczbą pierwszą.\n");
}
```

Program powinien wypisać następujące liczby:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

Wynik testu:

```
C:\Users\Uzytkownik\Desktop\1-WSB (INFORMATYKA)\Podstawy programowania\PROJEKT ZALICZENIOWY\Projekt\bin\Debug\Projekt.exe
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97,
```

Z przeprowadzonego testu, można wnioskować, że program wyznacza prawidłowo liczby pierwsze.

Do sprawdzenia czasu pracy wykorzystano metodę `Stopwatch`, a wynik zostanie zaprezentowany w tikach procesora:

```
Stopwatch czas = new Stopwatch();
czas.Start();
{
    Badany fragment kodu.
}
czas.Stop();
Console.WriteLine("Czas w tikach procesora:" + czas.ElapsedTicks);
```

Tak jak pisano w punkcie 2.2. po wykonaniu tego testu zostało dopisane słowo kluczowe „break” w instrukcji warunkowej, ponieważ wystarcza do stwierdzenia czy liczba należy do liczb pierwszych czy nie - znalezienie tylko jednego dzielnika (nie licząc liczby 1 i podanej liczby). Podyktowane jest to skróceniem czasu pracy aplikacji dla dużych liczb (powyżej 6 znaków), które okazały się nie należeć do liczb pierwszych. Przed zmianą dla liczb 9 cyfrowych ilość tików procesora potrzebna na wykonanie tego zadania oscylowała w granicach 25 000 000, po zmianie nie przekracza 2 000. Dla liczb maksymalnie 5 cyfrowych, które należą lub nie do liczb pierwszych ilość tików procesora potrzebna na wykonanie zadania nie przekracza 3 000. Dla dużych liczb pierwszych (powyżej 6 cyfrowych) widać znaczny wzrost wartości tików procesora.

5.1.3. Sprawdzenie poprawności rozkładu liczby na czynniki pierwsze.

W opcji 3 – Rozłóż Liczbę założono, że wynik programu będzie przedstawiony w formie schematycznej. W celu sprawdzenia poprawności rozkładania liczby na czynniki pierwsze sprawdzono kilka losowych liczb.

Liczba	Dzielnik	Liczba	Dzielnik	Liczba	Dzielnik
48	2	12	2	100	2
24	2	6	2	50	2
12	2	3	3	25	5
6	2	1		5	5
3	3			1	
1					

Tak jak w poprzednim punkcie do sprawdzenia czasu pracy wykorzystano metodę `Stopwatch`, a wynik zostanie zaprezentowany w tikach procesora.

Czas pracy dla bardzo dużych liczb np. 9 cyfrowych nie przekracza 21 000 tików procesora. Dla losowych liczb, które są najczęściej używane ilość tików oscyluje w okolicy 5 000.

5.1.4. Sprawdzenie poprawności wyznaczania dzielników liczby oraz NWD dwóch liczb.

W celu wyznaczenia największego wspólnego dzielnika dwóch liczb program korzysta także z metody „WyznaczDzielnikiLiczby” dlatego sprawdzenie wykonujemy razem. Metoda sprawdzenia była identyczna jak w poprzednim punkcie. Po wprowadzeniu kilku przykładowych liczb i analizie wyników nie ma zastrzeżeń co do pracy aplikacji. Dodatkowo w przypadku wyznaczania NWD program wypisuje także dzielniki liczb, które podano do wyznaczenia NWD co pozwala dodatkowo sprawdzić czy program prawidłowo wskazał największy wspólny dzielnik.

Tak jak w poprzednich przykładach do sprawdzenia ilości tików potrzebnych na wykonanie zadania wykorzystano metodę `Stopwatch`. Wyznaczenie dzielników liczby dla liczb najczęściej używanych nie przekracza 5 000 tików. Dla liczb bardzo dużych (9 cyfrowe) ilość tików znacząco wzrasta i wynosi nawet 50 000 000. Samo wyznaczenie NWD gdy znane są już dzielniki obu liczb nie przekracza 10 tików.