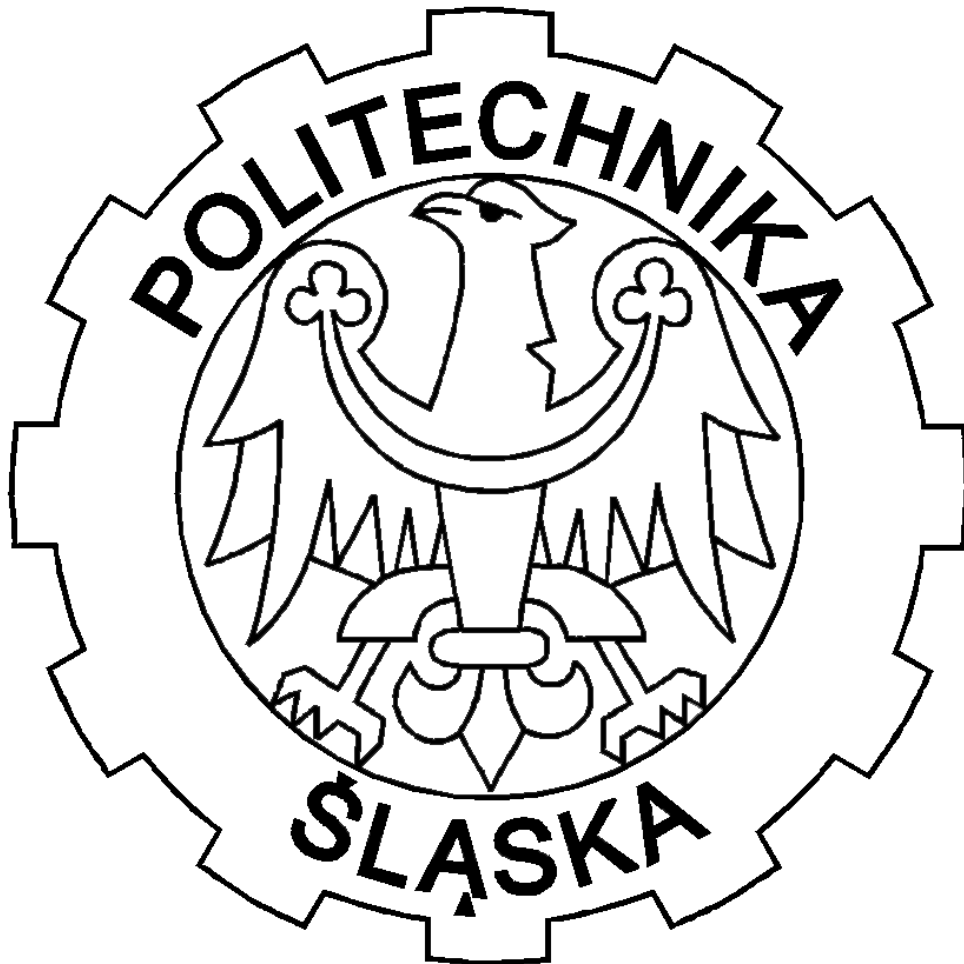


Języki skryptowe

dokumentacja projektu „!”



Krzysztof Gumiński, grupa 3/5

Kierunek Informatyka, Wydział Matematyki Stosowanej

30 grudnia 2023

Część 1

Opis programu

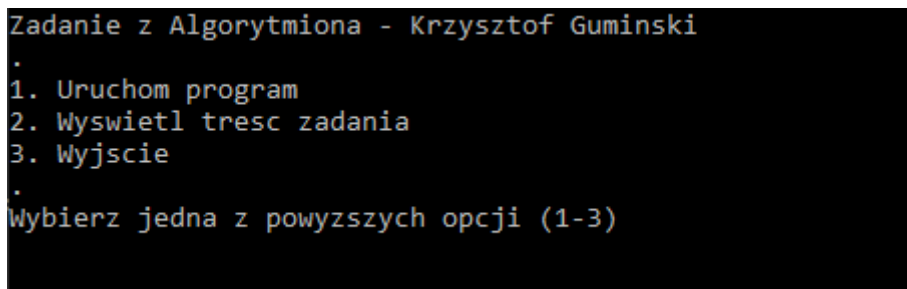
W wielu przypadkach (np. w rachunku prawdopodobieństwa) spotykamy się z potrzebą obliczania liczb np. typu $C^n_k = n!/(k!(n-k)!)$. Możemy tu napotkać następujący problem: chociaż końcowy wynik jest stosunkowo mały, to liczby występujące w liczniku i mianowniku mogą być ogromne (np. $C^{100}_4 = 3921225$, ale liczba $100!$ ma aż 158 cyfr). O ile w przykładzie tym możemy program „nauczyć” jak obliczać tego typu liczby bez wykorzystywania dodatkowych bibliotek dla dużych liczb (w tym przykładzie mamy: $C^{100}_4 = 100!/(4!96!) = 96!97 \cdot 98 \cdot 99 \cdot 100 / (2 \cdot 3 \cdot 4 \cdot 96!) = 97 \cdot 49 \cdot 33 \cdot 25 = 3921225$), to nie zawsze da się to tak łatwo zrobić, np. wyrażenie $13 \cdot 55 \cdot 2^{13} \cdot 3^5 \cdot 7^2 \cdot 100! / (83 \cdot 89 \cdot 97 \cdot 49! \cdot 67!)$ po uproszczeniu sprowadza się do liczby 409457, ale w tym przypadku „nauczenie” programu obliczania wartości takich wyrażen (w poprzedni sposób) może być trudne lub niewykonalne. Załóżmy, że w wyrażeniach będą występowały tylko liczby naturalne nie większe od 100 (tak jak w powyższym przykładzie – nie większe od 100 jako „składowe” wyrażenia).

Zadanie do zrealizowania:

Zaproponuj, opisz i zaimplementuj metodę obliczania wartości ww. wyrażen (bez korzystania z bibliotek dla dużych liczb).

Instrukcja obsługi

W celu uruchomienia programu należy wykorzystać plik wykonywalny run.exe, który wyświetli początkowy ekran (w postaci menu). Można tam zobaczyć instrukcję obsługi, która pokazuje potencjalne działania. Aby przejść dalej użytkownik musi wybrać liczbę od 1 do 3 (wybrana liczba odpowiada opcji, jaka zostanie wykonana).



```
Zadanie z Algorytmiona - Krzysztof Guminski
.
1. Uruchom program
2. Wyświetl treść zadania
3. Wyjście
.
Wybierz jedną z powyższych opcji (1-3)
```

Rysunek 1. Instrukcja obsługi programu

Użytkownik ma do wyboru następujące opcje:

1. Opcja „Uruchom program” – Wykonuje główne zadanie programu. Na podstawie danych z katalogu input zapisuje wyniki do katalogu output oraz tworzy raport.html

```
Zadanie z Algorytmiona - Krzysztof Guminski
.
1. Uruchom program
2. Wyszwiatl tresc zadania
3. Wyjscie
.
Wybierz jedna z powyzzszych opcji (1-3)1
-dane1.txt
-dane2.txt
-dane3.txt
10:31:13 30/12/2023
```

Rysunek 2. Wygląd poprawnie działającego programu (po wyborze opcji).

10:31:13 30/12/2023

input	output
$(150!)/(100!*50!)$	20128660909731932294240234380929315748140
$(13*55*2^{13}*3^5*7^2*100!)/(83*89*97*49!*67!)$	409457
$(140!)/(20!*120!)$	827163809330939321148600

Rysunek 3. Zawartość pliku raport.html

2. Opcja „Wyszwiatl tresc zadania” – Wypisuje na ekranie treść zadania

```
Zadanie z Algorytmiona - Krzysztof Guminski
.
1. Uruchom program
2. Wyszwiatl tresc zadania
3. Wyjscie
.
Wybierz jedna z powyzzszych opcji (1-3)2
W wielu przypadkach (np. w rachunku prawdopodobienstwa) spotykamy sie z potrzeba
obliczania liczb np. typu  $C_{n,k} = n!/(k!(n-k)!)$ .
Mozemy tu napotkac nastepujacy problem: chociaz koncowy wynik jest stosunkowo
maly, to liczby wystepujace w liczniku i mianowniku moga byc ogromne (np.  $C_{4,100} = 3921225$ , ale liczba  $100!$  ma az 158 cy
fr!)
O ile w przykladzie tym mozemy program "nauczyc" jak obliczac tego typu liczby bez
wykorzystywania dodatkowych bibliotek dla duzych liczb (w tym przykladzie mamy:  $C_{4,100} = 97*49*33*25$ ), to nie zawsze da
sie to tak latwo
zrobic, np. wyrazenie  $13*55*2^{13}*3^5*7^2*100!/(83*89*97*49!*67!)$ 
po uproszczeniu sprowadza sie do liczby 409457, ale
w tym przypadku "nauczzenie" programu obliczania wartosci takich wyrazen (jak w poprzedni sposob) moze byc trudne lub nie
wykonywalne
Zalozmy, ze w wyrazeniach beda wystepowaly tylko liczby naturalne nie wieksze od 100
(tak jak w powyzzszym przykladzie - nie wieksze od 100 jako "skladowe" wyrazenia).
Zaproponuj, opisz i zaimplementuj metode obliczania wartosci takich wyrazen (bez
korzystania z bibliotek dla duzych liczb).
```

Rysunek 4. Wyświetlanie treści zadania

3. Opcja „Wyjscie” – Kończy działanie programu

```

Zadanie z Algorytmiona - Krzysztof Guminski
.
1. Uruchom program
2. Wyszwietl tresc zadania
3. Wyjscie
.
Wybierz jedna z powyzzszych opcji (1-3)3
D:\Informatyka\Projekt z jezykow skryptowch>

```

Rysunek 5 – koniec działania programu

Struktura danych programu

Opisywany program składa się z następującej struktury danych, niezbędnych do prawidłowego działania aplikacji:

- run.exe – plik wykonywalny, który odpowiada za uruchamianie programu.
 - duzeLiczby.py – Skrypt Python, który zawiera w sobie najważniejszą logikę programu. W nim dochodzi do pobierania danych z plików, przetwarzania danych na odpowiedni wynik oraz do zapisania wyników działań do plików.
 - raport.py – Skrypt Python, który pobiera dane z katalogu input (dane wejścia i wyjścia) oraz za ich pomocą generujący raport.html do katalogu output.
- Raport.html – Zawiera w sobie dane wejściowe i wyjściowe oraz informację, o której godzinie został sporządzony raport.
- program.bat – Skrypt batch, który odpowiada za widok oraz wywoływanie skryptów Pythona.
 - katalog input – zawiera w sobie 3 pliki txt, które zawierają w sobie wyrażenia matematyczne.
 - katalog output – zawiera w sobie 3 pliki txt, które zawierają rozwiązania ww. wyrażeń.

```

D:\Informatyka\Projekt z jezykow skryptowch>dir /on /b
.idea
build
Dokumentacja z projektu.docx
duzeLiczby.py
input
output
projekt.bat
raport.html
raport.py
run.exe
run.py

```

Rysunek 6. Struktura danych programu

Część 2

Opis działania

W celu uproszczenia działania najpierw zajmujemy się skracaniem ze sobą silni (jest to nasza pierwsza czynność, ponieważ silnia jest funkcją rosnącą geometrycznie). W tym celu gdy mamy iloraz dwóch silni, zapisujemy większy składnik w postaci silni mniejszego składnika razy pozostałe czynniki (np. $100!/90! = (90!*91*92...\cdot 100)/90!$), co spowoduje, że duża część silni nam się uprości. Następnie doprowadzamy równanie do postaci, której elementy są liczbami całkowitymi (wykonujemy potęgowania oraz pozostałe silnie rozpisujemy). Na sam koniec skracamy ze sobą składniki, wykorzystując algorytm NWD (największy wspólny dzielnik). Przechodzimy po kolei przez kolejne elementy licznika i skracamy z elementem mianownika, o ile NWD rozważanych liczb jest większe od 1. Gdy już przeszliśmy po wszystkich elementach skracamy wszystkie elementy występujące i w liczniku i w mianowniku a następnie mnożymy przez siebie elementy licznika, następnie robimy to samo dla mianownika a na sam koniec dzielimy otrzymany licznik przez otrzymany mianownik.

Przykład rozwiązania

Rozwiążmy działanie: $(13*55*2^{13}*3^5*7^2*100!*5!)/(83*89*97*49!*67!*4!)$. Najpierw upraszczamy silnie z licznika i z mianownika, które możemy ze sobą uprościć (np. $100!=67!*68*69*\dots*100$). Czyli zostaje nam:

$$(13*55*2^{13}*3^5*7^2*68*69*\dots*100)/(83*89*97*6*7*\dots*49*4!).$$

Rozpisujemy potęgi i silnie i uzyskujemy działanie w postaci samych liczb czyli otrzymujemy:

$(13*55*8192*243*49*68*69*\dots*100)/(83*89*97*6*7*\dots*49*2*3*4)$, następnie za pomocą nwd (największego wspólnego dzielnika) skracamy ze sobą kolejne elementy aż otrzymamy maksymalnie skróconą postać (tylko liczby pierwsze, które już nie możemy ze sobą skrócić). Na podstawie tej postaci możemy wyliczyć ostateczny wynik, który wynosi 2047285.

Algorytm w formie pseudokodu

Pobieramy dane z pliku (dane w pliku są w postaci licznik w 1 linijce, mianownik w 2 linijce)

Liczby, które mają w sobie „!” zapisujemy do listy silnieLicznika lub silnieMianownika

Liczby, które mają w sobie „^” zapisujemy do listy potegiLicznika lub potegiMianownika

Pozostałe liczby zapisujemy do listy licznik lub mianownik

While długość listy silnieLicznika oraz silnieMianownika są większe od zera do

 a = największa silnia z licznika

 b = największa silnia z mianownika

 usuwamy a i b z silnieLicznika oraz silnieMianownika

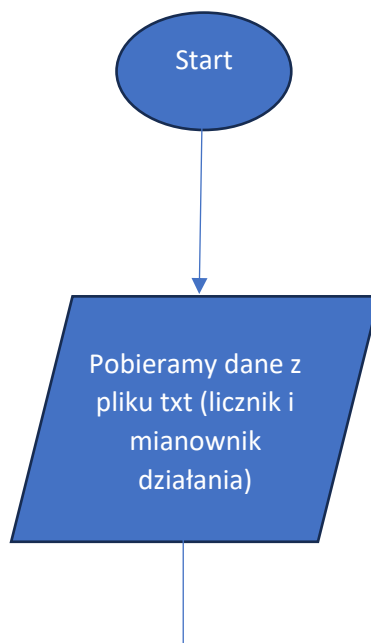
 if a >= b then

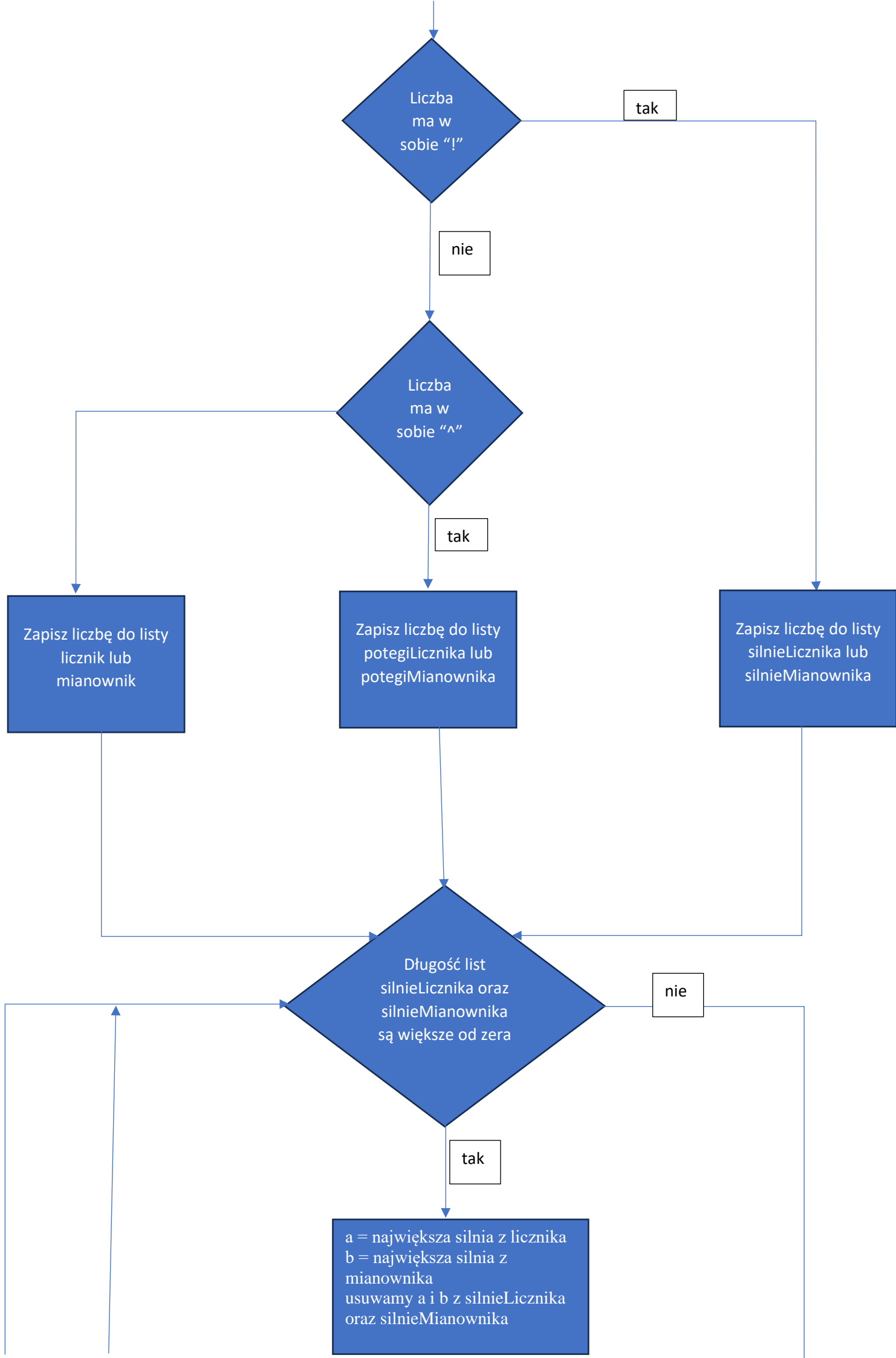
```

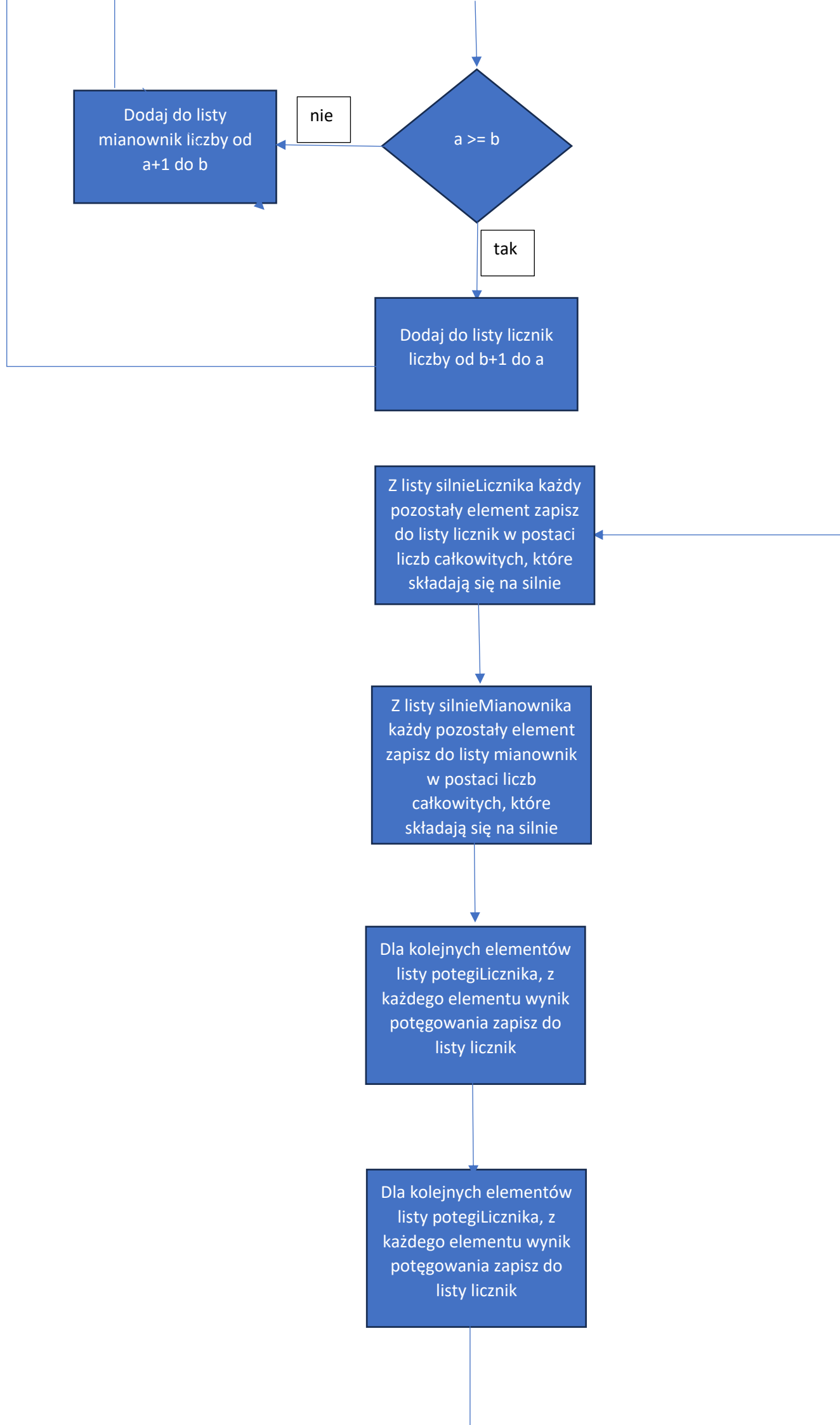
        for y = b + 1 to y = a
            dodaj y do listy licznik
    else
        for y = a + 1 to y = b
            dodaj y do listy mianownik
    for x in silnieLicznika, gdzie x to kolejne elementy listy silnieLicznika
        for i = 2 to i = a
            dodaj i do listy licznik
    for x in silnieMianownika, gdzie x to kolejne elementy listy silnieMianownika
        for i = 2 to i = a
            dodaj i do listy mianownik
    for x in potegiLicznika, gdzie x to kolejne elementy listy potegiLicznika
        dodaj do listy licznik wykonanie potęgowania
    for x in potegiMianownika, gdzie x to kolejne elementy listy potegiMianownika
        dodaj do listy mianownik wykonanie potęgowania
    Tworzymy listę liczbyPierwsze i zapisujemy do niej wszystkie liczby pierwsze z przedziału
    od 2 do 100
    For x = 0 to x = długość listy licznik – 1
        If licznik[x] nie jest w liście liczbyPierwsze then
            For y = 0 to y = długość listy mianownik – 1
                If mianownik[y] nie jest w liście liczbyPierwsze then
                    while nwd(licznik[x],mianownik[y]) > 1 do
                        dzielnik = nwd(licznik[x],mianownik[y])
                        licznik[x] = licznik[x]//dzielnik
                        mianownik[y] = mianownik[y]//dzielnik
    Usuwamy z list licznik i mianownik wszystkie elementy, które występują i w tej i w tej liście
    wartoscLicznika = 1
    wartoscMianownika = 1
    for x in licznik, gdzie x to kolejne elementy listy licznik
        wartoscLicznika = wartoscLicznika*x
    for x in mianownik, gdzie x to kolejne elementy listy mianownik
        wartoscMianownika = wartoscMianownika*x
    wynik = wartoscLicznika// wartoscMianownika
    Zapisz zmienną wynik do odpowiedniego pliku txt

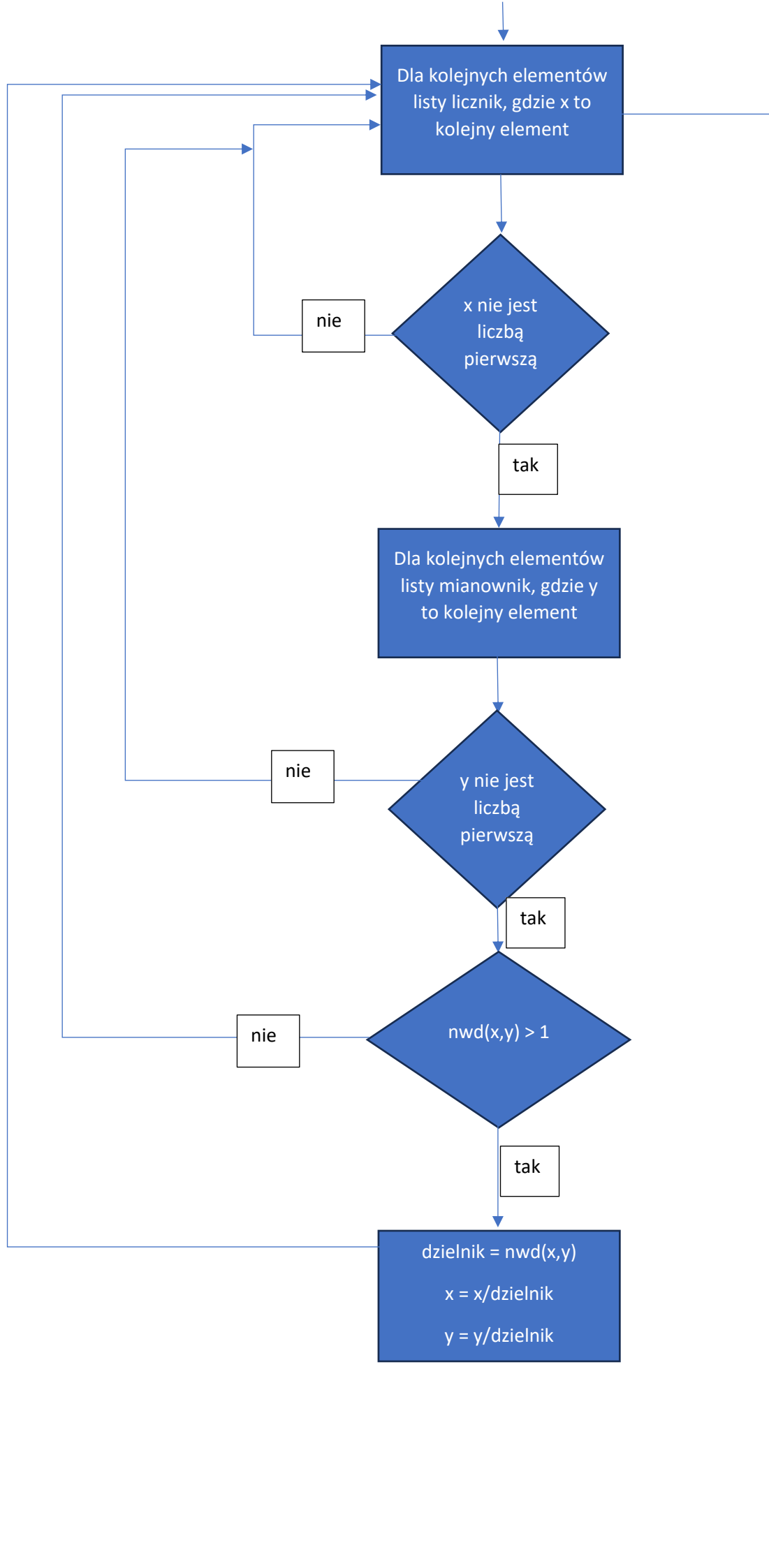
```

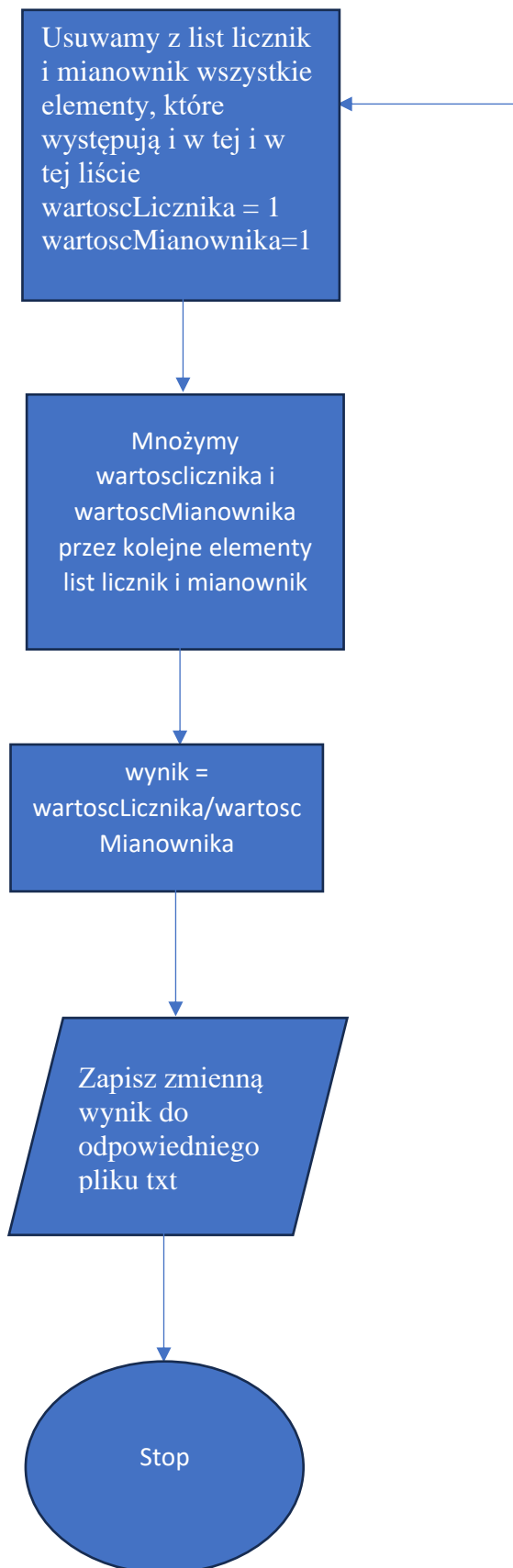
Algorytm w formie schematu blokowego



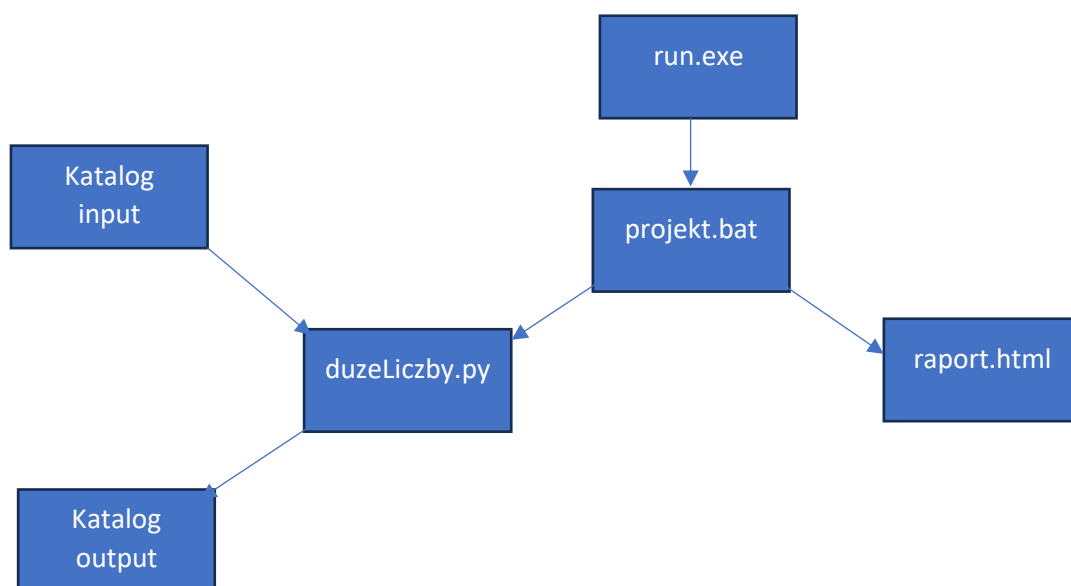








Schemat blokowy struktury programu



Testy

W celu przetestowania programu będziemy wprowadzać różne dane wyjściowe i na podstawie widoku z raport.html oraz kalkulatora, będzie można sprawdzić poprawność otrzymanych wyników.

12:58:02 30/12/2023

input	output
$(150!)/(100!*50!)$	20128660909731932294240234380929315748140
$(13*55*2^{13}*3^5*7^2*100!)/(83*89*97*49!*67!)$	409457
$(140!)/(20!*120!)$	827163809330939321148600

13:45:24 30/12/2023

input	output
$(100!)/(96!*4!)$	3921225
$(13*55*2^{13}*3^5*7^2*100!*5!)/(83*89*97*49!*67!*4!)$	2047285
$(10!)/(9!*1!)$	10

13:50:14 30/12/2023

input	output
$(100!)/(90!)$	62815650955529472000
$(10!*2!*13*55*2^{13}*3^5*7^2*100!*5!)/(9!*83*89*97*49!*67!*4!)$	40945700
$(16!)/(12!*4!)$	1820

Na podstawie przeprowadzonych testów można stwierdzić, że program wykonuje działania w poprawny sposób (co zostało sprawdzone na kalkulatorze).

Pełny kod aplikacji

duzeLiczby.py

```
import sys
import os

def czyPierwsza(liczba):
    for x in range(2,liczba):
        if (liczba % x) == 0:
            return False
    return True

def maks(lista):
    max = lista[0]
    for x in lista:
        if x > max:
            max = x
    return max

def potegaNaLiczbe(liczba):
    liczby = liczba.split("^")
    a = int(liczby[0])
    b = int(liczby[1])
    return a**b

def silniaPrzezSilnie(silnieMianownika, silnielicznika, mianownik,
licznik):
    gorneSilnie = []
    dolneSilnie = []
    for x in silnielicznika:
        gorneSilnie.append(int(x[:-1]))
    for x in silnieMianownika:
        dolneSilnie.append(int(x[:-1]))
    a = maks(gorneSilnie)
    b = maks(dolneSilnie)
    silnielicznika.remove(str(a) + "!")
    silnieMianownika.remove(str(b) + "!")
```

```

    if a >= b:
        for y in range(b+1,a+1):
            licznik.append(y)
    else:
        for y in range(a+1,b+1):
            mianownik.append(y)

def silnia(n):
    if n == 1:
        return 1
    else:
        return n*silnia(n-1)

def nwd(a,b):
    while a != 0 and b != 0:
        if a >= b:
            a = a % b
        else:
            b = b % a
    if a == 0:
        return b
    else:
        return a

os.chdir("input")
gora = []
dol = []
with open(sys.argv[1], "r") as file:
    gora = file.readline().rstrip().split("*")
    dol = file.readline().rstrip().split("*")
licznik = []
mianownik = []
potegiLicznika = []
potegiMianownika = []
silnieLicznika = []
silnieMianownika = []
a = 0
b = 0
for x in gora:
    if '^' in x and '!' in x:
        potegowanie = x.split("^")
        if "!" in potegowanie[0]:
            a = silnia(int(potegowanie[0][:-1]))
        else:
            a = int(potegowanie[0])
        if "!" in potegowanie[1]:
            b = silnia(int(potegowanie[1][:-1]))
        else:
            b = int(potegowanie[1])
        potegiLicznika.append(str(a) + "^" + str(b))
    elif '^' in x:
        potegiLicznika.append(x)
    elif '!' in x:
        silnieLicznika.append(x)
    else:
        licznik.append(int(x))
for x in dol:
    if '^' in x and '!' in x:
        potegowanie = x.split("^")

```

```

        if "!" in potegowanie[0]:
            a = silnia(int(potegowanie[0][:-1]))
        else:
            a = int(potegowanie[0])
        if "!" in potegowanie[1]:
            b = silnia(int(potegowanie[1][:-1]))
        else:
            b = int(potegowanie[1])
        potegiMianownika.append(str(a) + "^" + str(b))
    elif '^' in x:
        potegiMianownika.append(x)
    elif '!' in x:
        silnieMianownika.append(x)
    else:
        mianownik.append(int(x))
while len(silnieLicznika) > 0 and len(silnieMianownika) > 0:
    silniaPrzezSilnie(silnieMianownika, silnieLicznika, mianownik, licznik)
for x in silnieLicznika:
    a = int(x[:-1])
    for i in range(2, a+1):
        licznik.append(i)
silnieLicznika.clear()
for x in silnieMianownika:
    a = int(x[:-1])
    for i in range(2, a+1):
        mianownik.append(i)
silnieMianownika.clear()
for x in potegiLicznika:
    a = potegaNaLiczbe(x)
    licznik.append(a)
for x in potegiMianownika:
    a = potegaNaLiczbe(x)
    mianownik.append(a)
liczbyPierwsze = []
for x in range(2, 101):
    if czyPierwsza(x):
        liczbyPierwsze.append(x)
dzielnik = 0
for x in range(len(licznik)):
    if licznik[x] not in liczbyPierwsze:
        for y in range(len(mianownik)):
            if mianownik[y] not in liczbyPierwsze:
                while nwd(licznik[x], mianownik[y]) > 1:
                    dzielnik = nwd(licznik[x], mianownik[y])
                    licznik[x] = licznik[x]//dzielnik
                    mianownik[y] = mianownik[y]//dzielnik
kopiaLicznika = licznik
kopiaMianownika = mianownik
licznik = [x for x in kopiaLicznika if x not in kopiaMianownika]
mianownik = [y for y in kopiaMianownika if y not in kopiaLicznika]
wartoscLicznika = 1
wartoscMianownika = 1
for x in licznik:
    wartoscLicznika = wartoscLicznika*x
for x in mianownik:
    wartoscMianownika = wartoscMianownika*x
wynik = wartoscLicznika//wartoscMianownika
os.chdir("..")
os.chdir("output")
with open(sys.argv[1], "a+") as file2:
    file2.write(str(wynik))

```

raport.py

```
import datetime
import os
from os.path import isfile, join

now = datetime.datetime.now()
data = now.strftime("%H:%M:%S %d/%m/%Y")
with open("raport.html", "w") as file1:
    file1.write(f"""
    <html>
        <head>
            <title> Raport z obliczania </title>
        </head>
        <body>
            <h1>{data}</h1>
            <table>
                <tr>
                    <th>input</th>
                    <th>output</th>
                <tr>
                    """)
    a = ""
    b = ""
    wynik = ""
    filein = [file for file in os.listdir("input") if
isfile(join("input", file))]
    for x in range(len(filein)):
        with open(f"input/dane{x+1}.txt", "r") as file2:
            a = file2.readline().rstrip()
            b = file2.readline().rstrip()
            file1.write(f""" <tr><td>"" + "(" + a + ")/(" + b + ")")
            file1.write(f"""</td>
                <td>""")
            with open(f"output/dane{x+1}.txt", "r") as file3:
                wynik = file3.readline().rstrip()
            file1.write(wynik)
            file1.write(f"""</td>
                </tr>""")
    file1.write(f"""
    </table>
    </body>
    </html>""")
```

projekt.bat

```
@echo off

:menu
echo Zadanie z Algorytmiona - Krzysztof Guminski
echo .
echo 1. Uruchom program
echo 2. Wyszwiatl tresc zadania
echo 3. Wyjscie
echo .
```

```

set /p wybor=Wyberz jedna z powyzzszych opcji (1-3)

if %wybor% == 1 goto opcja1
if %wybor% == 2 goto opcja2
if %wybor% == 3 goto exit
echo Musisz wybrac liczbe od 1 do 3
goto menu

:opcja1
IF EXIST raport.html DEL raport.html
IF NOT EXIST output mkdir output
echo "<HTML>" >> raport.html
DEL /Q output
for /f "delims=" %%a in ('dir /b input') do (
    echo -%%a
    call python duzeLiczby.py %%a "wynik"
)
call python raport.py
goto menu

:opcja2
echo W wielu przypadkach (np. w rachunku prawdopodobienstwa) spotykamy sie z
potrzeba
echo   obliczania liczb np. typu  $C_n,k = n!/(k!(n-k)!)$ .
echo   Mozemy tu napotkac nastepujacy problem: chociaz koncowy wynik jest
stosunkowo
echo   maly, to liczby wystepujace w liczniku i mianowniku moga byc ogromne
(np.  $C_{4,100} = 3921225$ , ale liczba  $100!$  ma az 158 cyfr!)
echo   O ile w przykladzie tym mozemy program "nauczyc" jak obliczac tego
typu liczby bez
echo   wykorzystywania dodatkowych bibliotek dla duzych liczb (w tym
przykladzie mamy:  $C_{4,100} = 97*49*33*25$ ), to nie zawsze da sie to tak latwo
echo   zrobic, np. wyrazenie  $13*55*2**13*3**5*7**2*100!/(83*89*97*49!*67!)$ 
echo   po uproszczeniu sprowadza sie do liczby 409457, ale
echo   w tym przypadku "nauczenie" programu obliczania wartosci takich
wyrazen (jak w poprzedni sposob) moze byc trudne lub niewykonywalne
echo   Zalozmy, ze w wyrazeniach beda wystepowaly tylko liczby naturalne nie
wieksze od 100
echo   (tak jak w powyzzszym przykladzie - nie wieksze od 100 jako "skladowe"
wyrazenia).
echo   Zaproponuj, opisz i zaimplementuj metode obliczania wartosci takich
wyrazen (bez
echo   korzystania z bibliotek dla duzych liczb).
goto menu
:exit

```


run.py

```
import subprocess

path = r"D:\Informatyka\Projekt z jezykow  
skryptowych\projekt.bat"

subprocess.call(path, True)
```