

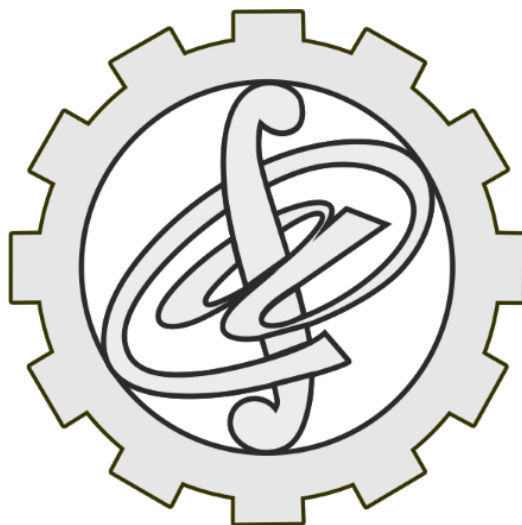
# Języki Skryptowe

Dokumentacja projektu "Kółko i Krzyżyk" - Algorytmion 2014

Patryk Gamrat, grupa 2/4

Politechnika Śląska, Wydział Matematyki Stosowanej

22 stycznia 2024



# Spis treści

<b>1</b>	<b>Informacje wprowadzające</b>	<b>3</b>
1.1	Założenia projektu . . . . .	3
1.2	Opis programu . . . . .	3
1.3	Instrukcja obsługi . . . . .	5
<b>2</b>	<b>Realizacja projektu</b>	<b>7</b>
2.1	Opis działania . . . . .	7
2.2	Algorytm obliczający wyniki . . . . .	9
2.3	Analiza wyników . . . . .	11
<b>3</b>	<b>Kod źródłowy aplikacji</b>	<b>14</b>
3.1	Skrypt Batch . . . . .	14
3.2	Skrypt Python realizujący obliczenia . . . . .	15
3.3	Skrypt Python realizujący generowanie raportu . . . . .	19

# 1. Informacje wprowadzające

## 1.1. Założenia projektu

Głównym założeniem projektu było zaimplementowanie systemu, realizującego wybrane zadanie z konkursu Algorytmion. Dla wprowadzonych danych wejściowych realizowane są obliczenia a następnie na podstawie wyników generowany jest raport. W projekcie korzystamy ze skryptów w języku Python oraz Batch, a zatem wykorzystujemy całą wiedzę zdobytą w ramach przedmiotu.

## 1.2. Opis programu

Program realizuje zadanie 2 z edycji 2014 konkursu Algorytmion - "Kółko i Krzyżyk". Zadanie to polega na odczytaniu z pliku tekstowego, które reprezentuje planszę 5x5 do gry w kółko i krzyżyk. Naszym celem jest obliczenie punktacji dla Kacpra umieszczającego "x" oraz dla Olka, który umieszcza "o". Punkty są liczone w zależności od ilości symboli w jednym wierszu kolumnie lub diagonali. Na koniec należy obliczyć sumę punktów i wyznaczyć zwycięzce.

### Pełna treść zadania

Kacper i Olek grają w "kółko i krzyżyk" na planszy 5x5, zaznaczając na przemian pola (Kacper - krzyżyki, Olek - kółka). Umówili się, że punkty będą liczyć po zakończeniu gry według poniższej reguły:

2 kółka lub krzyżyki w jednym wierszu, kolumnie lub diagonali - 1pkt,  
3 kółka lub krzyżyki w jednym wierszu, kolumnie lub diagonali - 3pkt,  
4 kółka lub krzyżyki w jednym wierszu, kolumnie lub diagonali - 7pkt,  
5 kółka lub krzyżyki w jednym wierszu, kolumnie lub diagonali - 15pkt.

Twoim zadaniem jest odczytanie z pliku tekstowego danego układu i poprawne przeliczenie punktów uzyskanych przez zawodników i wyświetlenie ich na ekranie monitora.

Przykład.

x o x o x

o x o o o

x x x o x

o o x x o

o x x o o

punkty dla gracza umieszczającego "x":  $1*7+4*3+5*1=24\text{pkt}$

punkty dla gracza umieszczającego "o":  $1*7+2*3+10*1=23\text{pkt}$

wygrał Kacper.

Uwaga!

W pliku tekstowym gra.txt znajduje się pięć wierszy, w każdym pięć znaków: "o" lub "x".

Realizacją części obliczeniowej tego zadania zajmuje się skrypt python. Pliki z układem planszy są przekazywane za pomocą skryptu batch, który działa jako menu użytkownika. Na podstawie wykonanych obliczeń, osobny skrypt generuje raport w formie pliku html, który zawiera liste odczytanych układów wraz z wynikami i zwycięzcą dla każdego z nich.

### 1.3. Instrukcja obsługi

Aby uruchomić projekt należy rozpakować dołączony plik. Następnie uruchamiamy skrypt batch nazwany **TicTacToe.bat**.

```
|=====|
| Zadanie 2 2014 - Kółko i Krzyżyk |
|=====|
|1.Wykonaj obliczenia
|2.Załaduj dane wejściowe
|3.Wygeneruj raport
|4.Otwórz raport
|5.Koniec
|=====|
|wybór:|
```

Zrzut 1: Menu

Po uruchomieniu skryptu, uzyskamy dostęp do menu. Opis poszczególnych opcji:

1. - Wykonuje skrypt python realizujący obliczenia na załadowanych danych wejściowych
2. - Wczytuje katalog z plikami .txt zawierającymi układy plansz
3. - Na podstawie wyników z katalogu *out* generuje raport
4. - Otwiera wygenerowany plik raport.html
5. - Zamyka konsolę

Aby wygenerować raport należy najpierw wybrać opcję **2.** oraz podać ścieżkę do katalogu z plikami wejściowymi. Wraz z projektem dostarczony został katalog *in*. Zawiera on przykładowe pliki wejściowe z których można skorzystać.

Po załadowaniu danych możemy wybrać opcję **1.** aby wykonać obliczenia. Na ekranie konsoli wyświetli się komunikat z informacją o przetwarzanych plikach.

Ostatnim krokiem jest wygenerowanie raportu i otworenie go za pomocą opcji **3.** oraz **4.**

## Algorytmion 2014 Zadanie 2 - "Kółko i Krzyżyk"

### Raport wykonanych obliczeń

#### Raport nr.1 (Plik out0.txt)

X	O	X	O	X
O	X	O	O	O
X	X	X	O	X
O	O	X	X	O
O	X	X	O	O

**Punkty dla gracza  
umieszczającego "x":**

$$0*15 + 1*7 + 4*3 + 5*1 = 24$$

**Punkty dla gracza  
umieszczającego "o":**

$$0*15 + 1*7 + 2*3 + 10*1 = 23$$

**Wygrał Kacper**

Zrzut 2: Wygenerowany raport

Aby program działał poprawnie nie można modyfikować plików wewnątrz katalogu *src*, należy również pamiętać aby katalog ten znajdował się w tym samym katalogu w którym znajduje się skrypt batch.

## 2. Realizacja projektu

### 2.1. Opis działania

Interakcja z użytkownikiem w programie jest realizowana za pomocą skryptu batch. W zależności od opcji wybranej przez użytkownika ładowane są odpowiednie pliki lub uruchamiany konkretny skrypt python.



Poniżej znajduje się fragment kodu realizujący uruchamianie skryptu obliczeniowego na plikach w wybranym wcześniej katalogu.

```
1 :opcja1
2 if defined input (
3     set /a i=0
4     cd /d "!input!"
5     if not exist "%~dp0out\" mkdir "%~dp0out"
6     for /r %%x in (*.txt) do (
7         py "%~dp0\src\SolveBoard.py" "%%x" "%~dp0out\out!i!.txt"
8         if exist "%~dp0out\out!i!.txt" set /a i=!i!+1
9     )
10    cd /d "%~dp0"
11    echo Wykonano obliczenia na plikach z katalogu %input%
12 ) else (
13     echo Nie załadowano katalogu z danymi wejściowymi
14 )
15 pause
16 goto :menu
```

Po wybraniu opcji **1.** w menu skrypt sprawdza czy został podany katalog z plikami wejściowymi, jeśli tak to skrypt przechodzi do tego katalogu. Za pomocą pętli dla wszystkich plików z rozszerzeniem .txt w katalogu wykonywany jest skrypt **SolveBoard.py**, który rozwiązuje układ planszy. Wyniki zapisywane są do katalogu *out*.

Skrypt **SolveBoard.py** wczytuje plik wejściowy oraz plik do zapisu jako argumenty. Pierwszym krokiem jaki wykonuje kod, jest załadowanie układu planszy z pliku. Ponieważ skrypt batch nie zajmuje się walidacją poprawności pliku, dopiero w tym kroku sprawdzane jest czy podany układ planszy jest odpowiednio zapisany. W przypadku niepoprawnego pliku wejściowego, wyrzucany jest odpowiedni wyjątek.

```

1 def loadBoard(filename: str) -> list:
2     loaded_board = []
3     try:
4         with open(filename, encoding='utf=8') as board_file:
5             lines = board_file.readlines()
6             lines = [l.strip() for l in lines]
7             for line in lines:
8                 loaded_board.append(line.split(' '))
9
10            # Sprawdzanie poprawności wczytania danych
11            if len(loaded_board) != 5:
12                raise IOError
13            for l in range(5):
14                if len(loaded_board[l]) != 5 or not all(s == 'x' or s
15                == 'o' for s in loaded_board[l]):
16                    raise IOError
17            except IOError:
18                print(f"Plik {filename} jest niepoprawny!")
19
20            print(f"Pomyślnie załadowano plik {filename} z planszą")
21            return loaded_board

```

Wyniki działania skryptu zapisywane są w plikach **out.txt**. Poniżej znajduje się przykład takowego pliku

```

1 x o x o x o x o o o x x x o x o o x x o o x x o o
2 0*15 + 1*7 + 4*3 + 5*1 = 24
3 0*15 + 1*7 + 2*3 + 10*1 = 23
4 Wygrał Kacper

```

Po wybraniu odpowiedniej opcji w menu, pliki **out.txt** są przetwarzane przez skrypt **RaportGen.py**, który wstrzykuje dane z pliku do stworzonego wcześniej szablonu dokumentu html. Poniżej znajduje się schemat prezentujący sposób działania programu



## 2.2. Algorytm obliczający wyniki

Poniżej znajduje się pseudokod algorytmu, który oblicza ilość punktów dla obu graczy.

```
Data: Tablica tab z układem planszy
Result: Ilość punktów dla obu graczy
Zainicjuj tablicę trójwymiarową usedPatterns wartościami false
punktyKacper = 0
punktyOlek = 0
foreach symbol in tab do
    foreach kierunek do
        n = 0
        while Przesuwając się w danym kierunku napotykamy
            aktualny symbol do
                n += 1
                Ustaw wartość usedPatterns sprawdzanych symboli i dla
                obecnego kierunku na True
            end
            Przelicz n na punkty według punktacji z treści zadania
            if symbol = 'x' then
                | Dodaj punkty do punktyKacper
            end
            else
                | Dodaj punkty do punktyOlek
            end
        end
    end
end
```

Najbardziej kluczowym elementem algorytmu jest tablica trójwymiarowa **usedPatterns[5][5][4]**. Pierwsze dwa indeksy są pozycję na planszy 5x5, natomiast ostatni określa o jaki kierunek chodzi (0 - wiersze, 1 - kolumny, 2 - diagonale lewe, 3 - diagonale prawe). W tablicy przechowujemy informację, czy dany symbol był już wykorzystany w wybranym kierunku.

Implementacja powyższego algorytmu w skrypcie python:

```
1 board = loadBoard(input_file)
2 if board is not None:
3     used_symbols = [[[False for _ in range(4)] for _ in range
4         (5)] for _ in range(5)]
5     kacper_points = [0 for _ in range(4)]
6     olek_points = [0 for _ in range(4)]
7     for i in range(5):
8         for j in range(5):
9             symbol = board[i][j]
10
11             # Wiersze
12             n = 0
13             while (j + n < 5) and (board[i][j + n] == symbol) and (
14                 used_symbols[i][j + n][0] == False):
15                 used_symbols[i][j + n][0] = True
16                 n += 1
17             if symbol == 'x':
18                 addPoints(n, kacper_points)
19             else:
20                 addPoints(n, olek_points)
21
22             # Kolumny
23             n = 0
24             while (i + n < 5) and (board[i + n][j] == symbol) and
25                 (used_symbols[i + n][j][1] == False):
26                 used_symbols[i + n][j][1] = True
27                 n += 1
28             if symbol == 'x':
29                 addPoints(n, kacper_points)
30             else:
31                 addPoints(n, olek_points)
32
33             # Diagonale lewe
34             n = 0
35             while (i + n < 5) and (j - n >= 0) and (board[i + n][j
36                 - n] == symbol) and (used_symbols[i + n][j - n][2] ==
37                 False):
38                 used_symbols[i + n][j - n][2] = True
39                 n += 1
40             if symbol == 'x':
41                 addPoints(n, kacper_points)
42             else:
43                 addPoints(n, olek_points)
```

```

40     # Diagonale prawe
41     n = 0
42     while (i + n < 5) and (j + n < 5) and (board[i + n][j +
n] == symbol) and (used_symbols[i + n][j + n][3] == False
):
43         used_symbols[i + n][j + n][3] = True
44         n += 1
45     if symbol == 'x':
46         addPoints(n, kacper_points)
47     else:
48         addPoints(n, olek_points)

```

### 2.3. Analiza wyników

Sprawdzenie poprawności wyników dla przykładowych układów planszy

#### Przykład 1

Dla przykładu z treści zadania

```

x o x o x
o x o o o
x x x o x
o o x x o
o x x o o

```

Otrzymujemy wyniki

punkty dla gracza umieszczającego "x":  $0*15 + 1*7 + 4*3 + 5*1 = 24\text{pkt}$

punkty dla gracza umieszczającego "o":  $0*15 + 1*7 + 2*3 + 10*1 = 23\text{pkt}$

Wygrał Kacper

Wynik jest zgodny z wynikiem podanym w treści zadania

#### Przykład 2

Dla pliku wejściowego

```

x o x o o
x o x o o
x o x o x
x o o x o
o o x x x

```

Otrzymujemy wyniki

punkty dla gracza umieszczającego "x":  $0*15 + 1*7 + 4*3 + 1*1 = 20\text{pkt}$   
punkty dla gracza umieszczającego "o":  $1*15 + 1*7 + 1*3 + 10*1 = 35\text{pkt}$   
Wygrał Olek

Zweryfikujmy wynik i policzmy punkty Kacpra

Wiersze:

```
x o x o o
x o x o o
x o x o x
x o o x o
o o x x x
```

$1*3\text{pkt}$

Kolumny:

```
x o x o o
x o x o o
x o x o x
x o o x o
o o x x x
```

$1*7+1*3+1*1=11\text{pkt}$

Diagonale lewe:

```
x o x o o
x o x o o
x o x o x
x o o x o
o o x x x
```

$1*3=3\text{pkt}$

Diagonale prawe:

```
x o x o o
x o x o o
x o x o x
x o o x o
o o x x x
```

$1*3=3\text{pkt}$

Suma:  $3 + 11 + 3 + 3 = 20\text{pkt}$

Ilość punktów Kacpra jest zgodna z ilością punktów obliczoną przez algorytm. Teraz obliczymy punkty Olka.

Wiersze:

x o x o o

x o x o o

x o x o x

x o o x o

o o x x x

4\*1=4pkt

Kolumny:

x o x o o

x o x o o

x o x o x

x o o x o

o o x x x

1\*15+1\*3+1\*1=19pkt

Diagonale lewe:

x o x o o

x o x o o

x o x o x

x o o x o

o o x x x

1\*7+2\*1=9pkt

Diagonale prawe:

x o x o o

x o x o o

x o x o x

x o o x o

o o x x x

3\*1=3pkt

Suma: 4 + 19 + 9 + 3 = 35pkt

Ilość punktów Olka również jest zgodna z oczekiwanym wynikiem. A zatem wyniki obliczane przez algorytm są poprawne.

## 3. Kod źródłowy aplikacji

### 3.1. Skrypt Batch

```
1 echo offsetlocal EnableDelayedExpansionchcp 1250
2 :menu
3     cls
4     echo ^|=====^|
5     echo ^| Zadanie 2 2014 - Kółko i Krzyżyk ^|
6     echo ^|=====^|
7     echo ^|1.Wykonaj obliczenia ^|
8     echo ^|2.Załaduj dane wejściowe ^|
9     echo ^|3.Wygeneruj raport ^|
10    echo ^|4.Otwórz raport ^|
11    echo ^|5.Koniec ^|
12    echo ^|=====^|
13    set /p choice=^|Wybór:
14    if "%choice%"=="1" ( goto :opcja1 )
15    if "%choice%"=="2" ( goto :opcja2 )
16    if "%choice%"=="3" ( goto :opcja3 )
17    if "%choice%"=="4" ( goto :opcja4 )
18    if "%choice%"=="5" ( goto :eof) else ( goto :menu )
19
20 :opcja1
21     if defined input (
22         set /a i=0
23         cd /d "!input!"
24         if not exist "%~dp0out\" mkdir "%~dp0out"
25         for /r %%x in (*.txt) do (
26             py "%~dp0\src\SolveBoard.py" "%%x" "%~dp0out\out!i!.txt"
27             "
28             if exist "%~dp0out\out!i!.txt" set /a i=!i!+1
29         )
30         cd /d "%~dp0"
31         echo Wykonano obliczenia na plikach z katalogu %input%
32     ) else (
33         echo Nie załadowano katalogu z danymi wejściowymi
34     )
35     pause
36     goto :menu
37 :opcja2
38     set /p input=Podaj ścieżkę katalogu z danymi wejściowymi:
39     if exist !input! (
40         echo Pomyślnie załadowano katalog %input%
```

```

40 ) else (
41     echo Nie udało się załadować katalogu %input%
42     set input=
43 )
44 pause
45 goto :menu
46 :opcja3
47 py "%~dp0\src\RaportGen.py" "%~dp0out" "raport.html"
48 pause
49 goto :menu
50 :opcja4
51 if exist raport.html (
52     echo Otwieram plik raport.html
53     start raport.html
54 ) else (
55     echo Nie znaleziono raportu
56 )
57 pause
58 goto :menu

```

### 3.2. Skrypt Python realizujący obliczenia

```

1 import sys
2
3
4 # Ładuje planszę
5 def loadBoard(filename: str) -> list:
6     loaded_board = []
7     try:
8         with open(filename, encoding='utf=8') as board_file:
9             lines = board_file.readlines()
10            lines = [l.strip() for l in lines]
11            for line in lines:
12                loaded_board.append(line.split(' '))
13
14            # Sprawdzanie poprawności wczytania danych
15            if len(loaded_board) != 5:
16                raise IOError
17            for l in range(5):
18                if len(loaded_board[l]) != 5 or not all(s == 'x'
19                or s == 'o' for s in loaded_board[l]):
20                    raise IOError
21            except IOError:
22                print(f"Plik {filename} jest niepoprawny!")

```

```

22
23     print(f"Pomyślnie załadowano plik {filename} z planszą")
24     return loaded_board
25
26
27 def addPoints(count: int, points: list):
28     match count:
29         # 1 punkt
30         case 2:
31             points[0] += 1
32         # 3 punkty
33         case 3:
34             points[1] += 1
35         # 7 punktów
36         case 4:
37             points[2] += 1
38         # 15 punktów
39         case 5:
40             points[3] += 1
41
42
43 if __name__ == '__main__':
44     board = None
45     try:
46         input_file = sys.argv[1]
47         output_file = sys.argv[2]
48     except IndexError:
49         raise SystemExit("Nie podano ścieżki do plików!")
50
51     board = loadBoard(input_file)
52     if board is not None:
53         used_symbols = [[[False for _ in range(4)] for _ in
54 range(5)] for _ in range(5)]
55         kacper_points = [0 for _ in range(4)]
56         olek_points = [0 for _ in range(4)]
57         for i in range(5):
58             for j in range(5):
59                 symbol = board[i][j]
60
61                 # Wiersze
62                 n = 0
63                 while (j + n < 5) and (board[i][j + n] ==
64 symbol) and (used_symbols[i][j + n][0] == False):
65                     used_symbols[i][j + n][0] = True
66                     n += 1

```



```

65         if symbol == 'x':
66             addPoints(n, kacper_points)
67         else:
68             addPoints(n, olek_points)
69
70         # Kolumny
71         n = 0
72         while (i + n < 5) and (board[i + n][j] ==
symbol) and (used_symbols[i + n][j][1] == False):
73             used_symbols[i + n][j][1] = True
74             n += 1
75             if symbol == 'x':
76                 addPoints(n, kacper_points)
77             else:
78                 addPoints(n, olek_points)
79
80         # Diagonale lewe
81         n = 0
82         while (i + n < 5) and (j - n >= 0) and (board
[i + n][j - n] == symbol) and (used_symbols[i + n][j - n
][2] == False):
83             used_symbols[i + n][j - n][2] = True
84             n += 1
85             if symbol == 'x':
86                 addPoints(n, kacper_points)
87             else:
88                 addPoints(n, olek_points)
89
90         # Diagonale prawe
91         n = 0
92         while (i + n < 5) and (j + n < 5) and (board[
i + n][j + n] == symbol) and (used_symbols[i + n][j + n
][3] == False):
93             used_symbols[i + n][j + n][3] = True
94             n += 1
95             if symbol == 'x':
96                 addPoints(n, kacper_points)
97             else:
98                 addPoints(n, olek_points)
99
100        # Sumowanie punktów
101        sumKacper = kacper_points[3] * 15 + kacper_points[2]
* 7 + kacper_points[1] * 3 + kacper_points[0] * 1
102        sumOlek = olek_points[3] * 15 + olek_points[2] * 7 +
olek_points[1] * 3 + olek_points[0] * 1

```

```

103
104     # Wypisanie wyników
105     try:
106         with open(output_file, "w", encoding='utf-8') as
out_file:
107             for row in board:
108                 print(*row, end=' ', file=out_file)
109                 print(f"\n{kacper_points[3]}*15 + {
kacper_points[2]}*7 + {kacper_points[1]}*3 + {
kacper_points[0]}*1 = {sumKacper}pkt", file=out_file)
110                 print(f"{olek_points[3]}*15 + {olek_points
[2]}*7 + {olek_points[1]}*3 + {olek_points[0]}*1 = {
sumOlek}pkt", file=out_file)
111                 if sumKacper > sumOlek:
112                     print("Wygrał Kacper", file=out_file)
113                 elif sumOlek > sumKacper:
114                     print("Wygrał Olek", file=out_file)
115                 else:
116                     print("Jest remis", file=out_file)
117                 print(f"Zapisano wynik do pliku {output_file}")
118             except IOError:
119                 print(f"Nie udało się zapisać do pliku {
output_file}!")

```

### 3.3. Skrypt Python realizujący generowanie raportu

```
1 import sys
2 import os
3
4
5 # Tworzy tablice z planszą
6 def create_table(symbols: str) -> str:
7     table = '''<table class="board"><tr>'''
8     for j, symbol in enumerate(symbols):
9         if j % 5 == 0 and j != 0:
10             table += "</tr><tr>"
11             if symbol == 'x':
12                 player = "kacper"
13             else:
14                 player = "olek"
15             table += f'''<td class="{player}">{symbol}</td>'''
16     table += "</table>"
17     return table
18
19
20 if __name__ == '__main__':
21     try:
22         input_dir = sys.argv[1]
23         output_file = sys.argv[2]
24     except IndexError:
25         raise SystemExit("Nie podano ścieżki do plików!")
26
27     # Odczytywanie plików wejściowych i tworzenie raportów
28     raport_sections = []
29     try:
30         for i, filename in enumerate(os.listdir(input_dir)):
31             if filename.endswith(".txt"):
32                 file = os.path.join(input_dir, filename)
33                 with open(file, encoding="utf8") as raport:
34                     lines = raport.read().split('\n')
35                     board = lines[0].split()
36                     score_kacper = lines[1]
37                     score_olek = lines[2]
38                     result = lines[3]
39
40                     section_header = f'''
41 <hr><section><div class="left">
42 <h3>Raport nr.{i+1} <span style="font-weight: normal">(Plik {
43     filename})</span></h3>'''
```

```

43         section_table = create_table(board)
44         section_text = f'''
45 </div><div class="right">
46 <div class="score"><span class="kacper">Punkty dla gracza
    umieszczającego "x"</span>:<br>{score_kacper}</div>
47 <div class="score"><span class="olek">Punkty dla gracza
    umieszczającego "o"</span>:<br>{score_olek}</div>
48 <div class="result">{result}</div></div>
49 </section>'''
50         raport_sections.append(section_header +
    section_table + section_text)
51     except IOError:
52         raise SystemExit("Błąd podczas odczytywania plików z
    raportami!")
53
54     # Zapisywanie raportów do pliku .html
55     try:
56         with open(output_file, "w", encoding="utf=8") as html
57             :
58                 html.write(''''<!DOCTYPE html>
59 <html lang="pl">
60 <head>
61 <meta charset="UTF-8">
62 <title>Zadanie 2 2014 - Kółko i Krzyżyk</title>
63 <meta name="viewport" content="width=device-width,initial-
64 scale=1">
65 <style>
66 body {
67 background-color: #ffffcc;
68 text-align: center;
69 font-size: 1.1em
70 }
71 header h1 {
72 font-weight: bold;
73 color: red
74 }
75 h3 {
76 font-size: 1.5em
77 }
78 hr {
79 border: none;
80 height: 2px;
81 background-color: black
82 }
83 section {

```

```

82 display: flex;
83 align-items: center;
84 justify-content: space-around
85 }
86 .right {
87 text-align: left
88 }
89 table {
90 font-size: 2em
91 }
92 th ,td {
93 text-align: center;
94 font-size: 2em;
95 border: 3px solid black;
96 width: 2em
97 }
98 .kacper {
99 color: blue
100 }
101 .olek {
102 color: green
103 }
104 .score {
105 text-align: center;
106 font-size: 1.5em
107 }
108 .score span {
109 font-weight: bold
110 }
111 .result {
112 text-align: center;
113 font-size: 1.5em;
114 font-weight: bold
115 }
116 </style>
117 </head>
118 <body>
119 <header>
120 <h1>Algorytmion 2014 Zadanie 2 - "Kółko i Krzyżyk"</h1>
121 <h2>Raport wykonanych obliczeń</h2>
122 </header>'''
123         for section in raport_sections:
124             html.write(section)
125         html.write(''')
126 </body>

```

```
127 </html>''')
```

```
128     print(f"Utworzono raport w pliku {output_file}")
```

```
129 except IOError:
```

```
130     raise SystemExit("Błąd podczas tworzenia raportu .
```

```
    html!")
```