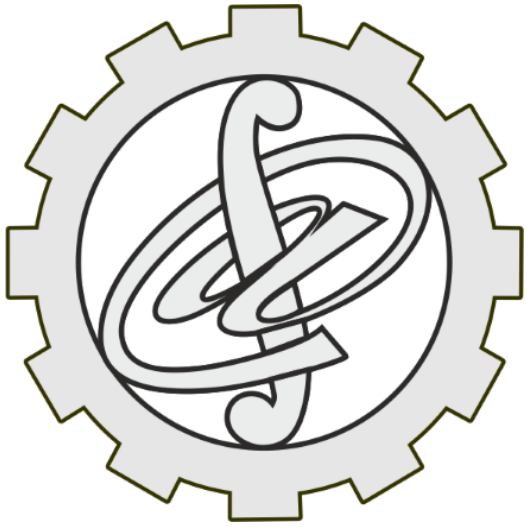
Języki Skryptowe

Dokumentacja projektu "Kółko i Krzyżyk" - Algorytmion 2014

Patryk Gamrat, grupa 2/4
Politechnika Śląska, Wydział Matematyki Stosowanej

21 stycznia 2024



Spis treści

1. Informacje wprowadzające

1.1. Założenia projektu

Głównym założeniem projektu bylo zaimplementowanie systemu, realizującego wybrane zadanie z konkursu Algorytmion. Dla wprowadzonych danych wejściowych realizowane są obliczenia a następie 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

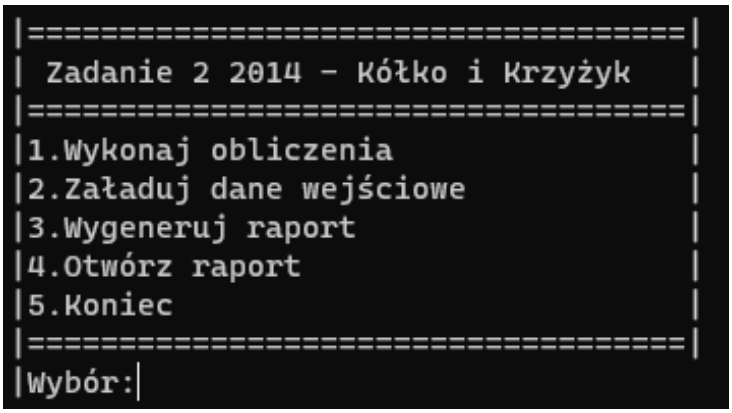
```
Kacper i Olek grają w "kółko i krzyżyk" na planszy 5x5, zaznaczając
na przemian pola (Kacper - krzyżyki, Olek - kółka). Umówili się, że
punkty będą liczyć po zakończeniu gry według poniższej reguły:

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=24pkt
punkty dla gracza umieszczającego "o": 1*7+2*3+10*1=23pkt
wygrał Kacper.
Uwaga!
W pliku tekstowym gra.txt znajduje się pięć wierszy, w każdym
pięć znaków: "o" lub "x".
```

Realizacją częsci 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**.



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 scieżkę 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 otworzenie go za pomocą opcj **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 plkiów 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 ralizowana 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.

```
:opcja1
if defined input (
  set /a i=0
  cd /d "!input!"
  if not exist "%~dp0out\" mkdir "%~dp0out"
  for /r %%x in (*.txt) do (
    py "%~dp0\src\SolveBoard.py" "%%x" "%~dp0out\out!i!.txt"
    if exist "%~dp0out\out!i!.txt" set /a i=!i!+1
  )
  cd /d "%~dp0"
  echo Wykonano obliczenia na plikach z katalogu %input%
) else (
  echo Nie załadowano katalogu z danymi wejściowymi
)
pause
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** wczytyuje 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.

```
def loadBoard(filename: str) -> list:
  loaded_board = []
  try:
    with open(filename, encoding='utf=8') as board_file:
      lines = board_file.readlines()
      lines = [l.strip() for l in lines]
      for line in lines:
        loaded_board.append(line.split(' '))

    # Sprawdzanie poprawności wczytania danych
    if len(loaded_board) != 5:
      raise IOError
    for l in range(5):
      if len(loaded_board[l]) != 5 or not all(s == 'x' or s
   == 'o' for s in loaded_board[l]):
        raise IOError
  except IOError:
    print(f"Plik {filename} jest niepoprawny!")

  print(f"Pomyślnie załadowano plik {filename} z planszą")
  return loaded_board
```

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

```
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
0*15 + 1*7 + 4*3 + 5*1 = 24
0*15 + 1*7 + 2*3 + 10*1 = 23
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 szabolnu dokumentu html. Poniżej znajduje się schemat prezentujący sposób działania programu

2.2. Algorytym 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
```

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:

```
board = loadBoard(input_file)
if board is not None:
  used_symbols = [[[False for _ in range(4)] for _ in range
   (5)] for _ in range(5)]
  kacper_points = [0 for _ in range(4)]
  olek_points = [0 for _ in range(4)]
  for i in range(5):
    for j in range(5):
      symbol = board[i][j]

      # Wiersze
      n = 0
      while (j + n < 5) and (board[i][j + n] == symbol) and (
   used_symbols[i][j + n][0] == False):
        used_symbols[i][j + n][0] = True
        n += 1
      if symbol == 'x':
        addPoints(n, kacper_points)
      else:
        addPoints(n, olek_points)

      # Kolumny
      n = 0
        while (i + n < 5) and (board[i + n][j] == symbol) and
    (used_symbols[i + n][j][1] == False):
          used_symbols[i + n][j][1] = True
          n += 1
      if symbol == 'x':
        addPoints(n, kacper_points)
      else:
        addPoints(n, olek_points)

      # Diagonale lewe
      n = 0
      while (i + n < 5) and (j - n >= 0) and (board[i + n][j
   - n] == symbol) and (used_symbols[i + n][j - n][2] ==
   False):
        used_symbols[i + n][j - n][2] = True
        n += 1
      if symbol == 'x':
        addPoints(n, kacper_points)
      else:
        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

3. Kod źródłowy aplikacji

3.1. Skrypt Batch

3.2. Skrypt Python realizujący obliczenia

3.3. Skrypt Python realizujący generowanie raportu