

Języki skryptowe

dokumentacja projektu - Algorytmion 2012 zadanie 5 "Tabliczka"

Mirosław Michalik, grupa 1/2

6 stycznia 2023

Część I

Opis programu

Zadaniem tego programu jest wygenerowanie tabliczki o wymiarze 4 na 4, zawierającej liczby od 1 do 15 i jedno puste pole oznaczane przez 0. Liczby w tabliczce muszą być ułożone tak, aby dało się ją rozwiązać w następujący sposób: zamieniamy puste pole z jednym z jego sąsiadów, robimy to tak długo, aż dojdziemy do stanu uporządkowanego - tabliczka będzie miała postać (1, 2, 3, ..., 14, 15, 0).

Stan uporządkowany:

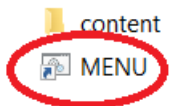
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Przykładowe rozwiązanie:

10	9	11	1
7	13		12
6	4	15	3
8	14	5	2

Instrukcja obsługi

Aby uruchomić program należy otworzyć plik "MENU.bat"



Po uruchomieniu zobaczymy menu z czterema opcjami:

```

C:\> MENU
MENU
1 - Uruchom program
2 - Backup
3 - Informacje o projekcie
4 - Wyjście

Twój wybór:
```

1 - uruchamia program, pokazuje nam efekt swojej pracy zapisuje dane do pliku "output.txt" i tworzy raport

```
C:\ MENU
14 10 3 0
2 6 1 7
13 11 9 4
15 5 12 8

Żeby rozwiązać tabliczkę potrzeba: 75 ruchów

Press any key to continue . . .
```

2 - tworzy backup (w folderze "backup") danych, które są aktualnie zapisane w plikach "output.txt" i "raport.html"

```
C:\ MENU
output\output.txt
1 File(s) copied
D:\raport.html
1 File(s) copied
Backup wykonany pomyślnie
Press any key to continue . . .
```

3 - wypisuje nam informacje o projekcie (autor, opis działania)

```
autor: Mirosław Michalik, grupa 1/2

Zadaniem tego programu jest wygenerowanie tabliczki 4 na 4, zawierającej liczby od 1 do 15 i jedno puste pole oznaczane przez 0.
Liczby w tabliczce muszą być ułożone tak, aby dało się ją rozwiązać w następujący sposób: zamieniamy puste pole z jednym z jego sąsiadów, robimy to tak długo, aż dojdziemy do stanu początkowego - tabliczka będzie miała postać (1, 2, 3, ..., 14, 15, 0).

Press any key to continue . . .
```

4 - zamyka menu

Struktura danych programu

Program ma następującą strukturę danych:

MENU.bat - skrypt batch będący menu całego programu. Przy jego pomocy możemy uruchomić program, stworzyć backup i wyświetlić informacje o projekcie.

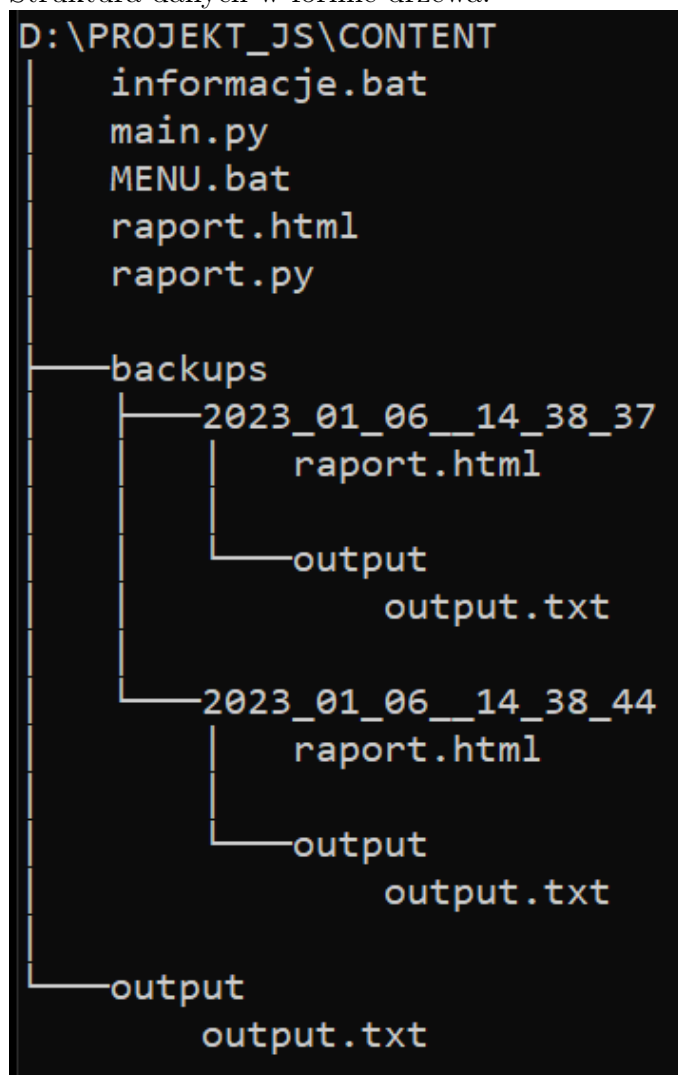
main.py - skrypt python generujący rozwiązanie programu zapisujący je do pliku "output".

informacje.bat - plik zawierający podstawowe informacje o projekcie.

raport.py - skrypt python pobierający dane z pliku "output.txt" i generujący z niego raport w formie pliku raport.html.

raport.html - plik .html zawierający raport wszystkich danych.

Struktura danych w formie drzewa:



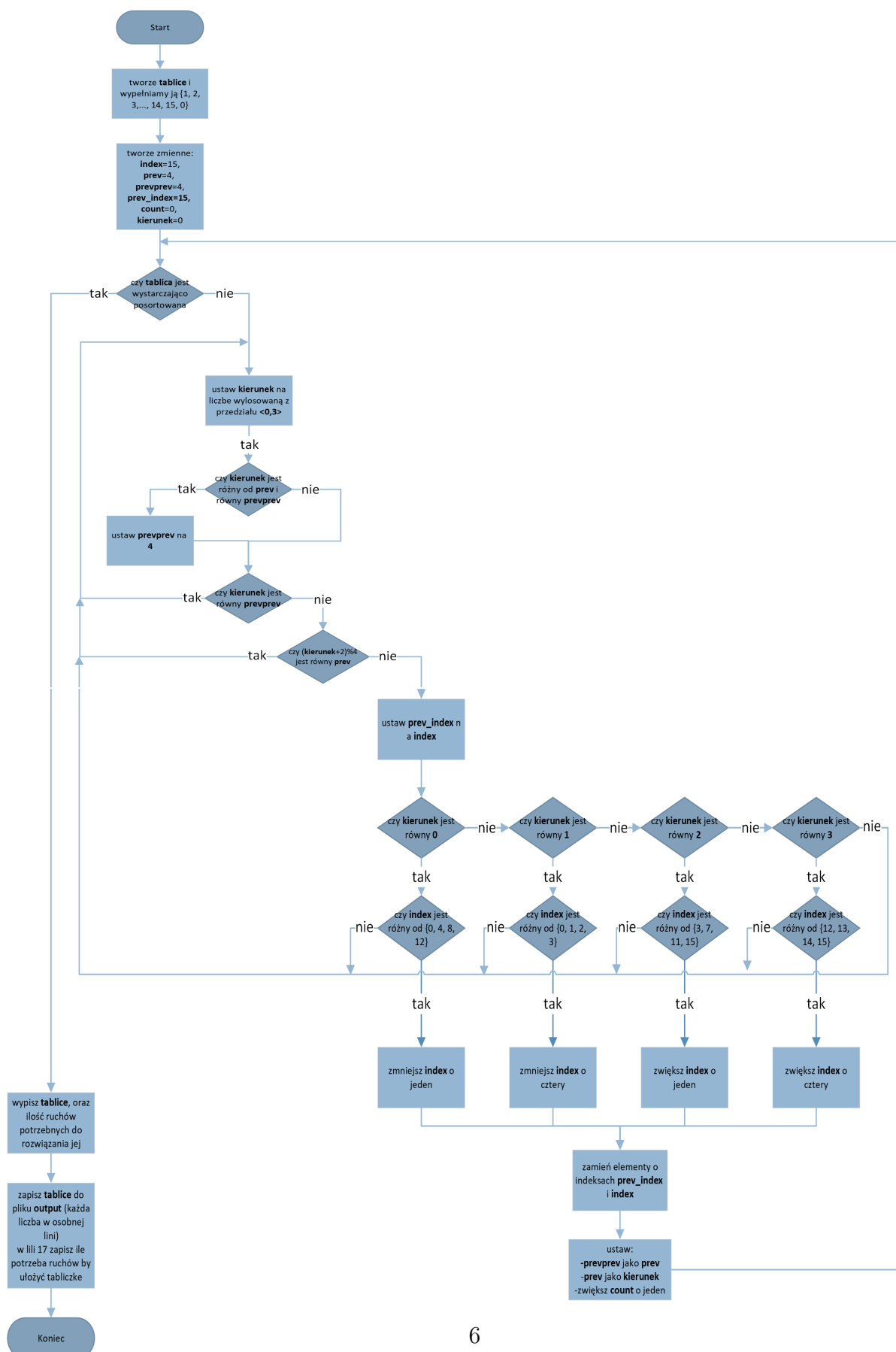
Część II

Opis działania

Skrypt "MENU.bat" po uruchomieniu i wybraniu opcji pierwszej uruchamia skrypt python "main.py", który działa w następujący sposób:

1. Tworzona jest tablica zawierająca pola (1, 2, 3, ... , 14, 15, 0). Będziemy "poruszać" się po tej tablicy przy pomocy pola zawierającego "0",
2. Losujemy, w którą stronę się poruszymy (0- w lewo, 1- w górę, itd.), wylosowany kierunek musi spełnić pewne wymagania:
 - nie można wylosować dwa razy z rzędu tego samego kierunku (np. nie można poruszyć się dwa razy z rzędu w górę)
 - nie można się cofać (np. po przejściu jednego pola w prawo, w następnym ruchu nie możemy wylosować przejścia w lewo)
3. Powtarzamy krok 2 do momentu gdy w całej tablicy nie ma dwóch następnych liczb leżących obok siebie (np. nie dopuszczamy ułożenia 12, 13; musiało by to być np. 12, 5, 13)

Schemat blokowy



Wykorzystywane biblioteki

W projekcie korzystałem tylko z jednej biblioteki:

```
import random
```

Przykład jej użycia:

```
while sprawdz():
    git = False
    while git is False:
        git = True
        r = random.randint(0, 3)

        if r is not prev and r is prevprev:
            prevprev = 4
        if r is prevprev:
            git = False
        if (r+2)%4 is prev:
            git = False
```

Implementacja systemu

1. - zapisywanie wygenerowanej tablicy do pliku .txt

```
1 try:
2     plik = open("output/output.txt", "w")
3 except IOError:
4     print("Bład otwierania pliku")
5     exit()
6 for i in range(16):
7     plik.write(str(tab[i]) + "\n")
8 plik.write(str(count))
9 plik.close()
```

2. - generowanie pliku "raport.html"(w pliku "raport.py")

```
1 table = "<table>"
2 for j in range(4):
3     table += "\n <tr>"
4     for i in range(4):
5         table += "\n      <td>" + str(zawartosc[i+j]) + "      &nbsp;\n" + "
6         </td>\n"
7     table += "    </tr>\n"
8 table += "</table>\n\n"
9 table2 = "<table>" + "\nZeby rozwiaczac tabliczke potrzeba: " + str(
    zawartosc[16]) + " ruchow \n</table>\n "
```

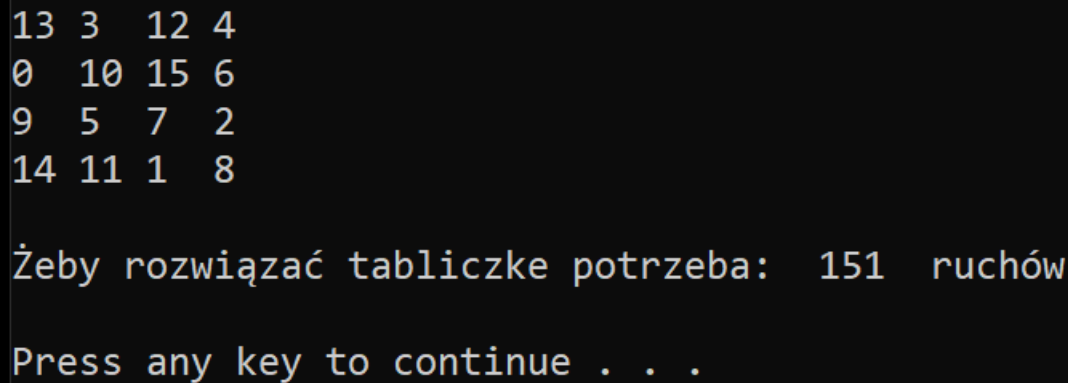
3. - tworzenie backupu (w pliku MENU.bat)

```
1 for /f %%a in ('powershell -Command "Get-Date -format  
    yyyy_MM_dd__HH_mm_ss"') do set datetime=%%a  
2 cd backups  
3 mkdir "%datetime%"  
4 cd %datetime%  
5 mkdir output  
6 cd ../  
7 cd ../  
8 xcopy /s output backups "\"%datetime%"\"output  
9 xcopy raport.html backups "\"%datetime%"  
10 echo Backup wykonany pomyslnie
```

Testy

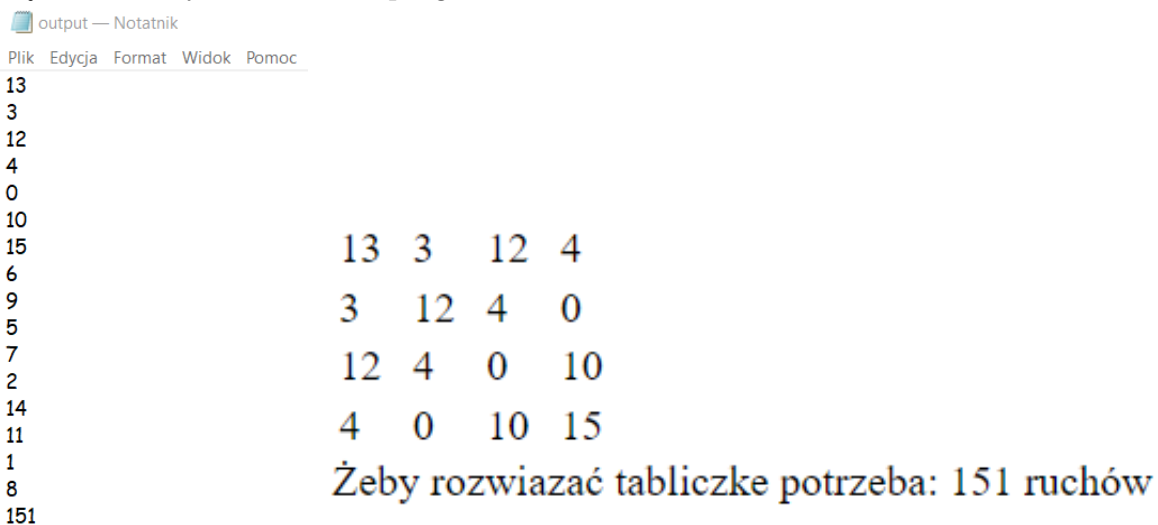
Uruchamiam plik "MENU.bat", następnie:

1. Wybieram pierwszą opcję (uruchom program). Na ekranie wyświetla się wygenerowana tablica i liczba ruchów potrzebanych do rozwiązania jej:



```
13 3 12 4  
0 10 15 6  
9 5 7 2  
14 11 1 8  
  
Żeby rozwiązać tabliczkę potrzeba: 151 ruchów  
  
Press any key to continue . . .
```

Równocześnie z zakończeniem generowania tablicy, w pliku "output.txt" i "raport.html" zapisuje się ten sam wynik działania programu



output — Notatnik

Plik Edycja Format Widok Pomoc

```
13  
3  
12  
4  
0  
10  
15  
6  
9  
5  
7  
2  
14  
11  
1  
8  
151  
  
13 3 12 4  
3 12 4 0  
12 4 0 10  
4 0 10 15  
  
Żeby rozwiązać tabliczkę potrzeba: 151 ruchów
```


2. Wybieram drugą opcję (backup). Program zwraca informacje o skopiowaniu elementów

```
output\output.txt
1 File(s) copied
D:raport.html
1 File(s) copied
Backup wykonany pomyślnie
Press any key to continue . . .
```

W folderze "backups" tworzy się folder o nazwie aktualnej daty i godziny

2023_01_06_16_36_23
2023_01_06_16_09_07

A w tym folderze znajduje się kopia pliku "output.txt" i "raport.html"

output 06.01.2023 16:36
raport 06.01.2023 16:29

output

stępnianie Widok

backups > 2023_01_06_16_36_23 > output

Nazwa	Data modyfikacji
output	06.01.2023 16:29

Zarówno plik "output" jak i "raport" mają tę samą, prawidłową zawartość

```
13 3 12 4
3 12 4 0
12 4 0 10
4 0 10 15
Żeby rozwiązać tabliczkę potrzeba: 151 ruchów
```

output -
Plik Edycja
13
3
12
4
0
10
15
6
9
5
7
2
14
11
1
8
151

3. Wybieram trzecią opcję (informacje o programie). Program wypisuje autora i informacje o programie:

```
autor: Mirosław Michalik, grupa 1/2
```

```
Zadaniem tego programu jest wygenerowanie tabliczki 4 na 4, zawierającej liczby od 1 do 15 i jedno puste pole oznaczane przez 0.
Liczby w tabliczce muszą być ułożone tak, aby dało się ją rozwiązać w następujący sposób: zamieniamy puste pole z jednym z jego sąsiadów, robimy to tak długo, aż dojdziemy do stanu początkowego - tabliczka będzie miała postać (1, 2, 3, ..., 14, 15, 0).
```

```
Press any key to continue . . .
```

Część III

Pełen kod aplikacji

```
1 # Algorytmion 2012 zad 5 - "Tabliczka"
2 import random
3
4 tab = [ "1", "2", "3", "4",
5         "5", "6", "7", "8",
6         "9", "10", "11", "12",
7         "13", "14", "15", "0" ]
8
9 def wypisz():
10     for m in range(16):
11         print(tab[m], end=" ")
12         if int(tab[m]) < 10:
13             print("", end=" ")
14         if (m+1)%4 == 0:
15             print()
16
17 def sprawdz():
18     for i in range(0, 15):
19         if int(tab[i]) + 1 == int(tab[i + 1]):
20             return True
21         if int(tab[i]) - 1 == int(tab[i + 1]):
22             return True
23         if i < 11:
24             if int(tab[i])+1 == int(tab[i + 4]):
25                 return True
26             if int(tab[i]) - 1 == int(tab[i + 4]):
27                 return True
28     return False
29
30
31 def losuj():
32     index = 15 # indeks pustego pola
33     prev = 4 # poprzedni kierunek
34     prevprev = 4 # poprzedni poprzedni kierunek
35     prev_index = 15
36
37     count = 0
38     while sprawdz():
39         git = False
40         while git is False:
41             git = True
42             r = random.randint(0, 3)
43
44             if r is not prev and r is prevprev:
45                 prevprev = 4
46             if r is prevprev:
47                 git = False
48             if (r+2)%4 is prev:
49                 git = False
50
```

```

51         if git is True:
52             prev_index = index
53             match r:
54                 case 0: # w lewo
55                     if index not in {0, 4, 8, 12}:
56                         index -= 1
57                     else:
58                         git = False
59                 case 1: # w gore
60                     if index not in {0, 1, 2, 3}:
61                         index -= 4
62                     else:
63                         git = False
64                 case 2: # w prawo
65                     if index not in {3, 7, 11, 15}:
66                         index += 1
67                     else:
68                         git = False
69                 case 3: # w dol
70                     if index not in {12, 13, 14, 15}:
71                         index += 4
72                     else:
73                         git = False
74             if git is True:
75                 temp = tab[prev_index]
76                 tab[prev_index] = tab[index]
77                 tab[index] = temp
78                 prevprev = prev
79                 prev = r
80                 count += 1
81     wypisz()
82     print("\nZeby rozwiazac tabliczke potrzeba: ", count, " ruchow \n")
83     try:
84         plik = open("output/output.txt", "w")
85     except IOError:
86         print("Blad otwierania pliku")
87         exit()
88     for i in range(16):
89         plik.write(str(tab[i]) + "\n")
90     plik.write(str(count))
91     plik.close()
92
93     losuj()

```
