

# Języki skryptowe

Dokumentacja projektu „Miejskie Widoki”

Dominika Wiśniewska, grupa 4/8

# Część I

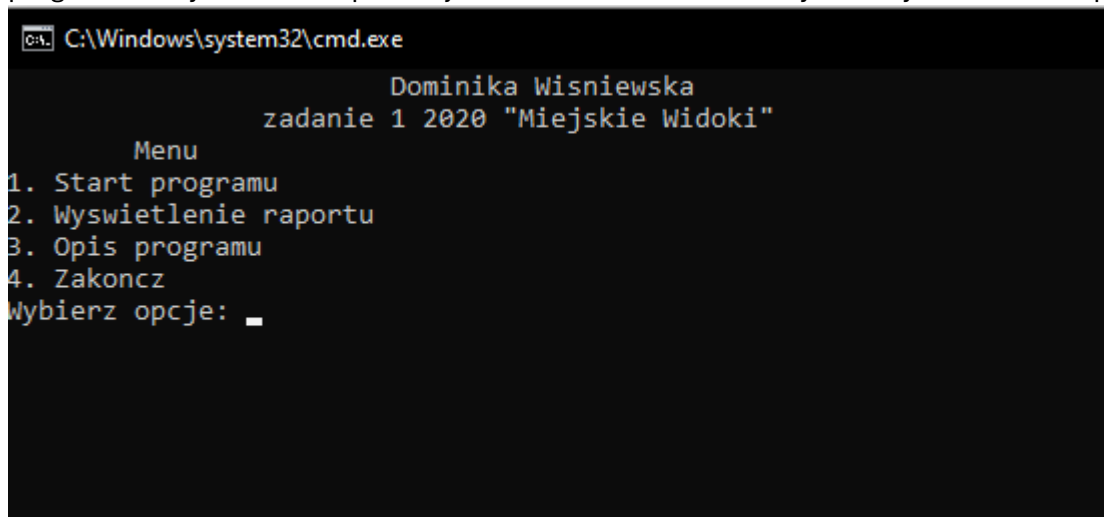
## Opis programu

Zadanie polega na stworzeniu programu, który w tablicy 5x5 rozmieści cyfry 1,2,3,4,5 w taki sposób, że w każdym wierszu i kolumnie cyfry nie będą się powtarzać. Metoda rozmieszczania cyfr ma mieć cechę losowości przynajmniej dla pierwszego wiersza, aby kolejne wywołania programu mogły generować inne ustawienia tych cyfr w tej tablicy. Liczby w tablicy symbolizują liczbę pięter stojącego w tym miejscu budynku. Następnie program dla każdego wiersza i kolumny dopisze liczbę odpowiadającą liczbie widocznych budynków z danego miejsca. Zakładamy, że budynek o większej liczbie pięter zastania budynki o mniejszej liczbie pięter, a za budynkiem o mniejszej liczbie widoczne są budynki o większej liczbie pięter.

Program opisany w oryginalnym zadaniu nie ma żadnych danych wejściowych. Moja wersja tego programu będzie generować tablicę o dowolnym rozmiarze, podanym w pliku wejściowym. Jeśli generowanie tablicy będzie trwało zbyt długo, program się zakończy.

## Instrukcja obsługi

Aby program się wykonał potrzebny będzie co najmniej jeden plik wejściowy w katalogu in. Aby uruchomić program należy uruchomić plik Projekt.bat. Po uruchomieniu wybieramy w menu start programu.



```
C:\Windows\system32\cmd.exe
Dominika Wisniewska
zadanie 1 2020 "Miejskie Widoki"
Menu
1. Start programu
2. Wyświetlenie raportu
3. Opis programu
4. Zakończ
Wybierz opcje: _
```

Program nas poprosi o wpisanie numeru pliku wejściowego. Używając podanego pliku program uruchomi skrypt *main.py*, który wykona opisane zadanie, następnie uruchomi *raport.py*, który generuje plik *raport.html*.

```
C:\Windows\system32\cmd.exe
Dominika Wisniewska
zadanie 1 2020 "Miejskie Widoki"

Menu
1. Start programu
2. Wyświetlenie raportu
3. Opis programu
4. Zakoncz
Wybierz opcje: 1
id pliku powinno odpowiadac liczbie wewnatrz pliku
Podaj id pliku in: 8
sprawdz czy istnieje plik z koncowka 8
0 | 4 5 2 3 2 5 1 3 | 0
-----
3 | 3 2 7 6 4 1 8 5 | 2
2 | 6 3 8 5 2 4 1 7 | 2
4 | 4 1 5 7 8 6 3 2 | 4
2 | 7 4 1 8 6 5 2 3 | 4
4 | 1 6 3 2 5 7 4 8 | 1
2 | 5 8 4 1 3 2 7 6 | 3
1 | 8 5 2 4 7 3 6 1 | 4
3 | 2 7 6 3 1 8 5 4 | 3
-----
0 | 2 2 2 3 3 1 4 3 | 0
Czas wykonania: 0.629697322845459 s.
Raport został utworzony w pliku raport.html.
Press any key to continue . . .
```

## 0.1 Instrukcja wdrożenia

### Dodatkowe informacje

Wymagania: Python3, gwarantowane działanie na 3.12.1

## Część II

### Opis działania

Tablica jest generowana kolumna po kolumnie. Tak długo jak kolumna nie będzie unikalna, będzie ona losowana. Kiedy wygenerowana kolumna będzie unikalna, zostanie wpisana do tablicy. Po wygenerowaniu całej tablicy program przejdzie do sprawdzania widoczności bloków z każdej strony.

### Algorytmy

#### Generacja tablicy

Każda kolumna jest najpierw generowana z liczbami uporządkowanymi rosnąco przy użyciu funkcji *ordered*, a potem liczby w niej są losowo zamieniane miejscami z użyciem funkcji *shuffle* i *swap*. Następnie funkcje *compare2D* i *compare* sprawdzają czy dana kolumna jest różna od wszystkich poprzednich. Funkcje do losowego zamieniania liczb miejscami są używane dopóki dana kolumna nie będzie unikalna.

**Funkcja *ordered(size, start\_value)*:**

```
num = [];  
for i w size do  
  | num.append(i + start_value)  
end  
return num;
```

**Algorithm 1:** Algorytm tworzenia kolumny wypełnionej liczbami rosnąco.

**Funkcja *shuffle(num)*:**

```
rnd = Random();  
for i w len(num) do  
  | swap(num, i, rnd.randint(0, length(num) - 1))  
end
```

**Algorithm 2:** Algorytm losujący, które dwie liczby zostaną zamienione miejscami.

**Funkcja *swap(num, a, b)*:**

```
temp = num[a];  
num[a] = num[b];  
num[b] = temp;
```

**Algorithm 3:** Algorytm zamieniający liczby miejscami.

**Funkcja** `compare2D(num1, num2, start, end):`

```
for i w (start,end) do
    if !compare(num1, num2[i]) then
        return False
    end
end
return True
```

**Algorithm 4:** Algorytm porównujący daną kolumnę ze wszystkimi wygenerowanymi kolumnami tablicy.

**Funkcja** `compare(num1, num2):`

```
if len(num1) != len(num2) then
    return False
end
for i w len(num1) do
    if num1[i] == num2[i] then
        return False
    end
end
return True
```

**Algorithm 5:** Algorytm porównujący dwie kolumny ze sobą.

**Funkcja** `Main():`

```
Pobierz rozmiar z pliku wejściowego;
Wypełnij tablice kolumnami z liczbami posortowanymi rosnąco;
Wymieszaj pierwszą kolumnę;
for x w (1, rozmiar) do
    while !compare2D(board[x], board, 0, x) do
        shuffle(board[x])
    end
end
Zwiększ rozmiar o 2;
Stwórz nową tablicę result;
Przekopiuj zawartość board do result tak, aby pierwsza i ostatnia kolumna i
wiersz były puste;
Oblicz widoczność bloków z każdej strony;
```

**Algorithm 6:** Generowanie tablicy z unikalnymi kolumnami.

### Obliczanie widoczności bloków

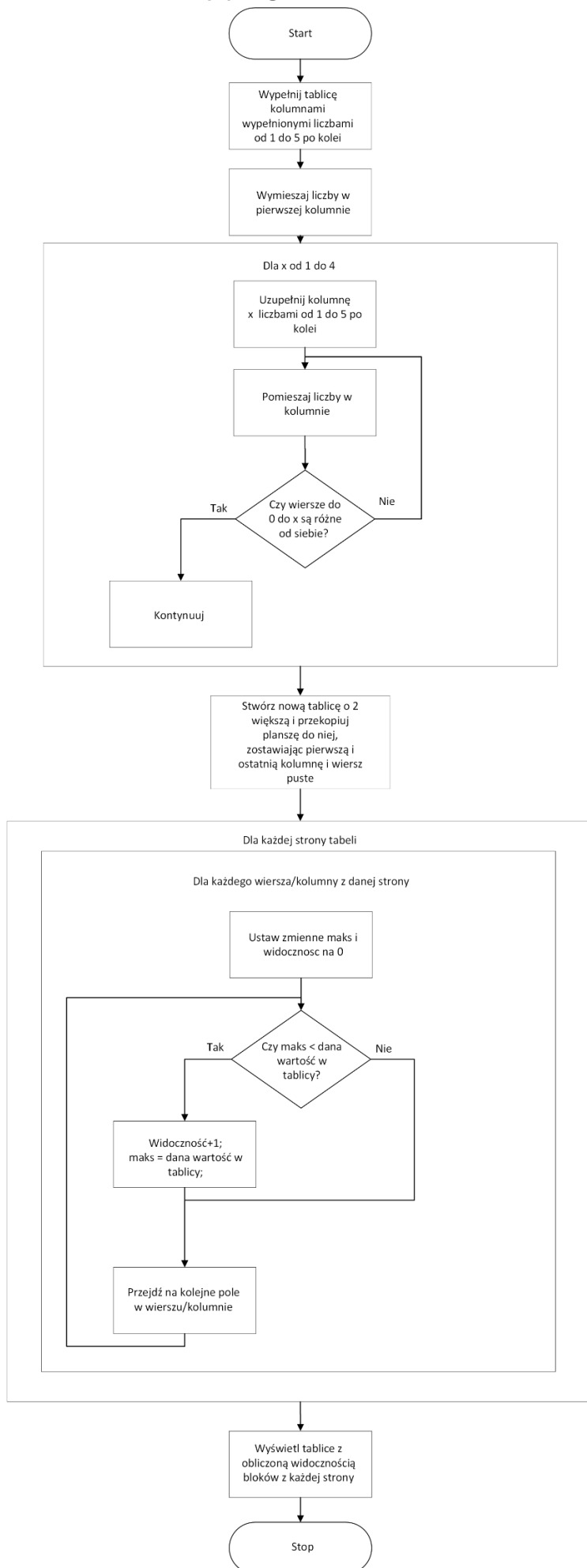
Na początek zerujemy liczbę widocznych budynków i zmienną maks. Sprawdzamy każdą liczbę w wierszu/kolumnie czy jest większa od wartości maks. Jeśli jest, zwiększamy liczbę widocznych budynków o 1, do wartości maks zapisujemy daną liczbę i przechodzimy dalej. Jeśli maks jest większa, przechodzimy dalej bez robienia niczego więcej. Gdy dotrzemy do końca zerujemy wartości zmiennych i przechodzimy do kolejnego wiersza/kolumny. Czynność powtarzamy dla każdej strony planszy.

**Funkcja** *view(tab, strona, rozmiar):*

```
Zmniejsz rozmiar o 2;  
widoczne = 0;  
for kazda kolumna/wiersz do  
  widoczne = 0;  
  maks = 0;  
  for kazda komorka do  
    if dana komorka > maks then  
      maks = dana komorka;  
      widoczne += 1;  
    end  
    Zapisz widoczne do komorki odpowiadającej danej kolumnie/wierszu z  
      danej strony;  
  end  
end
```

**Algorithm 7:** Algorytm obliczający widoczność bloków z danej strony

## Schemat blokowy programu



## Implementacja systemu

Projekt nie używa klas, całość zawiera się w jednym pliku.

## Testy

Wyniki są zgodne z oczekiwanymi wartościami wyjściowymi. W programie uwzględnione są wszystkie wyjątki, jakie mogą wyniknąć podczas wprowadzania danych.

## Eksperymenty

Z podanym algorytmem generowania tablicy program w przeciągu minuty wygeneruje tablicę o rozmiarach co najwyżej 10x10. Ze względu na losowe przemieszczanie liczb w kolumnie wyniki czasowe generowania tablicy mogą się różnić nawet o kilkadziesiąt sekund. W trakcie pisania projektu rozważałam zmianę sposobu generowania tablicy na bardziej optymalny. Ostatecznie zostałam przy napisanym już algorytmie, w przyszłości można zmienić sposób generowania tablicy na wydajniejszy.



## Peten kod aplikacji

```
1. import random
2. import time
3. import sys
4.
5.
6. def shuffle(num):
7.     rnd = random.Random()
8.     for i in range(len(num)):
9.         swap(num, i, rnd.randint(0, len(num) - 1))
10.
11.
12. def swap(num, a, b):
13.     temp = num[a]
14.     num[a] = num[b]
15.     num[b] = temp
16.
17.
18. def ordered(size, start_value):
19.     num = [i + start_value for i in range(size)]
20.     return num
21.
22.
23. def compare2D(num1, num2, start, end):
24.     for i in range(start, end):
25.         if not compare(num1, num2[i]):
26.             return False
27.     return True
28.
29.
30. def compare(num1, num2):
31.     if len(num1) != len(num2):
32.         return False
33.     for i in range(len(num1)):
34.         if num1[i] == num2[i]:
35.             return False
36.     return True
37.
38.
39. def view(tab, strona, rozmiar):
40.     rozmiar -= 2
41.     widoczne = 0
42.     if strona == "prawo":
43.         for i in range(1, rozmiar + 1):
44.             widoczne = 0
45.             maks = 0
46.             for j in range(rozmiar, 0, -1):
47.                 if tab[i][j] > maks:
48.                     maks = tab[i][j]
49.                     widoczne += 1
50.             tab[i][rozmiar + 1] = widoczne
```

```

51. if strona == "lewo":
52.     for i in range(1, rozmiar + 1):
53.         maks = 0
54.         widoczne = 0
55.         for j in range(1, rozmiar + 1):
56.             if tab[i][j] > maks:
57.                 maks = tab[i][j]
58.                 widoczne += 1
59.         tab[i][0] = widoczne
60. if strona == "gora":
61.     for i in range(1, rozmiar + 1):
62.         widoczne = 0
63.         maks = 0
64.         for j in range(1, rozmiar + 1):
65.             if tab[j][i] > maks:
66.                 maks = tab[j][i]
67.                 widoczne += 1
68.         tab[0][i] = widoczne
69. if strona == "dol":
70.     for i in range(1, rozmiar + 1):
71.         widoczne = 0
72.         maks = 0
73.         for j in range(rozmiar, 0, -1):
74.             if tab[j][i] > maks:
75.                 maks = tab[j][i]
76.                 widoczne += 1
77.         tab[rozmiar + 1][i] = widoczne
78. return widoczne
79.
80.
81. def print_table(result, size):
82.     for i in range(size):
83.         if i == 1 or i == size - 1:
84.             print("-" * (2 * size + 4 + size))
85.         for j in range(size):
86.             if j == 1 or j == size - 1:
87.                 print("| ", end="")
88.                 print(str(result[i][j]).ljust(2), end=" ")
89.             print()
90.
91.
92. def save_to_file(result, size, wyjscie):
93.     with open(f"{wyjscie}", "w") as file:
94.         for i in range(size):
95.             if i == 1 or i == size - 1:
96.                 file.write("-" * (2 * size + 4 + size) + "\n")
97.             for j in range(size):
98.                 if j == 1 or j == size - 1:
99.                     file.write("| ")
100.                    file.write(str(result[i][j]).ljust(2) if result[i][j] >= 10 else f"{result[i][j]}")
101.                    file.write(" ")
102.                    file.write("\n")

```

```
103.
104.
105. def main():
106.     try:
107.         wejscie=sys.argv[1]
108.         wyjscie=sys.argv[2]
109.         with open(f"{wejscie}", "r") as file:
110.             rozmiar = int(file.readline().strip())
111.             if rozmiar <= 0:
112.                 raise ValueError("Rozmiar planszy nie moze byc mniejszy lub rowny 0.")
113.     except ValueError:
114.         print("Plik wejsciuowy zawiera cos innego niz pojedyncza liczba calkowita dodatnia.")
115.         return
116.     except FileNotFoundError:
117.         print("Nie ma takiego pliku.")
118.
119.     start_time = time.time()
120.
121.     board = [ordered(rozmiar, 1) for _ in range(rozmiar)]
122.     shuffle(board[0])
123.
124.     for x in range(1, rozmiar):
125.         board[x] = ordered(rozmiar, 1)
126.         while not compare2D(board[x], board, 0, x):
127.             elapsed_time = time.time() - start_time
128.             if elapsed_time > 60:
129.                 print("Timeout - Program przekroczył limit czasu (60s).")
130.                 with open(f"{wyjscie}", "w") as file:
131.                     file.write(str(0))
132.                 return 0
133.             shuffle(board[x])
134.
135.     size = rozmiar + 2
136.     result = [[0] * size for _ in range(size)]
137.     for i in range(rozmiar):
138.         for j in range(rozmiar):
139.             result[i + 1][j + 1] = board[i][j]
140.
141.     view(result, "prawo", size)
142.     view(result, "lewo", size)
143.     view(result, "gora", size)
144.     view(result, "dol", size)
145.
146.     print_table(result, size)
147.     save_to_file(result, size, wyjscie)
148.
149.     elapsed_time = time.time() - start_time
150.     print(f"Czas wykonania: {elapsed_time} s.")
151.
152.
153. if __name__ == "__main__":
154.     main()
```