



Politechnika Śląska

Języki Skryptowe

Dokumentacja projektu PIONEK

Kacper Grabiec, grupa 2C

Część I

Opis programu

PIONEK, XXV OI, etap I

W punkcie $(0, 0)$ nieskończonej kratki stoi pionek. Pionek ma n dozwolonych ruchów. Każdy z nich jest opisany za pomocą wektora o współrzędnych całkowitych. Pionek może każdy z ruchów wykonać co najwyżej raz, w dowolnej kolejności. Wektory opisujące ruchy mogą się powtarzać i wtedy pionek może wykorzystać każdy z nich.

Naszym celem jest dostać się pionkiem do punktu położonego możliwie najdalej od punktu początkowego (w odległości euklidesowej). Jak daleko może on dotrzeć?

Instrukcja obsługi

W celu rozwiązania zadania należy uruchomić plik *bat.bat*. Po jego włączeniu ukaże się nam menu z kilkoma opcjami do wyboru.

```
-----  
-----MENU-----  
-----  
1. Stworz raport  
2. Wyświetl ostatni raport  
3. Usun raporty  
4. Uruchom skrypt py1  
5. Uruchom skrypt py2  
6. Koniec  
-----  
Podaj opcje:
```

Program czeka, aż wybierzemy jedną z dostępnych opcji:

- **Stwórz raport** – program rozwiązuje problem dla zadanych danych wejściowych oraz tworzy raport na podstawie danych wyjściowych;
- **Wyświetl ostatni raport** – program wyświetla nam najnowszy plik *.html* zawierający gotowy raport;
- **Usuń raporty** – program usuwa wszystkie raporty znajdujące się w folderze */raporty*;
- **Uruchom skrypt py1.py** – program uruchamia tylko plik *py1.py*, odpowiedzialny za rozwiązanie zadania;
- **Uruchom skrypt py2.py** – program uruchamia tylko plik *py2.py*, odpowiedzialny za stworzenie raportu;
- **Koniec** – koniec działania programu;

Po wykonaniu każdej opcji menu wyświetla się ponownie.

Każdy plik zawierający dane wejściowe (np. *in1.txt*) znajduje się w folderze */input_data*. Jego zawartość wygląda następująco:

```
1 5
2 2 -2
3 -2 -2
4 0 2
5 3 1
6 -3 1
```

Pierwszy wiersz standardowego wejścia zawiera jedną dodatnią liczbę całkowitą n oznaczającą liczbę możliwych ruchów pionka. Każdy z kolejnych n wierszy zawiera dwie liczby całkowite x, y oddzielone pojedynczym odstępem i oznaczające wektor $[x, y]$ opisujący możliwy ruch pionka.

Każdy plik zawierający dane wyjściowe (n. *out1.txt*) znajduje się w folderze */output_data*. Jego zawartość wygląda następująco:

```
1 26
```

Program wypisuje na standardowe wyjście liczbę całkowitą oznaczającą kwadrat odległości od punktu $(0, 0)$ do najdalszego punktu, do którego może doskoczyć pionek.

Wygenerowany raport wygląda następująco:

Kacper Grabiec Języki Skryptowe: PIONEK		
5 2 -2 -2 -2 0 2 3 1 -3 1		26
10 -3367 407 1495 5978 -1201 1172 -3809 -4844 -1862 -6169 -7967 5501 1643 9629 9635 2644 4478 -167 -5833 -1862		892299938
1 4909 -9148		107784185

Po lewej stronie wyświetlone zostają kolejne pliki zawierające dane wejściowe, po prawej dane wyjściowe.

Część II

Opis działania

Do rozwiązywania zadania program wykorzystuje rekurencję. Po wczytaniu danych wejściowych program sprawdza wszystkie możliwości wywołując tą samą funkcję oraz sprawdzając, czy dany ruch nie został już wykorzystany. Jeśli wszystkie ruchy zostały wykorzystane program tworzy dane wyjściowe, tworzy raport oraz kończy działanie.

Implementacja

```
28 def solve():
29     for i in range(0, n):
30         skip = False
31         vector = [0, 0]
32         global max_distance
33         for j in range(0, len(indexes)):
34             if i == indexes[j]:
35                 skip = True
36                 break
37         if skip:
38             continue
39         else:
40             indexes.append(i)
41             for j in range(0, len(indexes)):
42                 vector = sum_vectors(vector, moves[indexes[j]])
43             results.append(vector)
44             distance = calc_sqr_vec_len(results[-1])
45             if distance > max_distance:
46                 max_distance = distance
47             while len(results) > 0:
48                 results.pop()
49             solve()
50             indexes.pop()
```

Funkcja *solve()* jest najważniejszą częścią w całym programie. Za pomocą tablicy *indexes[]* sprawdza, czy dane ruchy nie zostały już wykorzystane. Następnie wywołuje inne funkcje, które np. obliczają dystans. Na końcu porównuje wynik z największym dystansem *max_distance* i usuwa wykorzystane już dane.

```

4  def load_data(filename: str):
5      file = open('./input_data/' + filename, 'r')
6      data = []
7      for line in file:
8          line = line.strip()
9          data.append([int(value) for value in line.split(' ')])
10     file.close()
11     return data

```

Funkcja *load_data()* służy do wczytania plików z danymi wejściowymi. Funkcja czyta plik i zwraca zawartość w postaci tablicy.

```

14 def print_data(filename: str, data: str):
15     file = open('./output_data/' + filename, 'w')
16     file.write(data)
17     file.close()

```

Funkcja *print_data()* tworzy plik z danymi wyjściowymi.

```

20 def calc_sqr_vec_len(vector):
21     return pow(vector[0], 2) + pow(vector[1], 2)
22
23
24 def sum_vectors(v1, v2):
25     return [v1[0] + v2[0], v1[1] + v2[1]]

```

Funkcja *calc_sqr_vec_len()* oblicza nam długość wektora natomiast *sum_vectors()* zwraca sumę wektorów.

```

34 def print_data():
35     time = datetime.now()
36     time = time.strftime("%H-%M-%S")
37     today = date.today()
38     today = today.strftime("%d-%m-%Y")
39     file = open(f'./raporty/{today}_{time}_raport.html', 'w')
40     file.write(f'''
41     <!DOCTYPE HTML>
42     <html lang="pl-PL">
43         <head>
44             <meta charset="utf-8">
45             <title>{today}_{time}_Raport</title>
46             <meta name="viewport" content="width=device-width, initial-scale=1">
47
48             <link rel="stylesheet" href="../style.css">
49         </head>
50
51         <body>
52             <h1>Kacper Grabiec</h1>
53             <h3>Języki Skryptowe PIONEK</h3>
54             {generate_template()}
55         </body>
56     </html>
57     ''')
58     file.close()

```

Funkcja *print_data()* pobiera obecny czas oraz datę, a następnie tworzy raport. Wypisuje do pliku podstawowy wzorzec składni *HTML*. W sekcji *<body>* wywołuje funkcję *generate_template()* odpowiedzialną za odpowiednie umiejscowienie danych wejściowych i wyjściowych w raporcie.

```

16 def generate_template():
17     template = ''
18     for i in range(0, len(input_files)):
19         template += '''
20         <div class="line">
21             <div class="data">
22                 '''
23         for j in input_files[i]:
24             template += f'''
25             <div class="input">{j}</div>
26             '''
27         template += f'''
28         </div>
29         <div class="data">{output_files[i][0]}</div>
30         </div>'''
31     return template

```

Funkcja *generate_template()* tworzy osobne elementy *HTML* dla każdego danych wejściowych oraz wyjściowych.

Pełen kod aplikacji

Plik *bat.bat*

```
1  @echo off
2
3  :menu
4  echo -----
5  echo -----MENU-----
6  echo -----
7  echo 1. Stworz raport
8  echo 2. Wyszwietl ostatni raport
9  echo 3. Usun raporty
10 echo 4. Uruchom skrypt py1
11 echo 5. Uruchom skrypt py2
12 echo 6. Koniec
13 echo -----
14 set /p "wybor=Podaj opcje: "
15
16 if %wybor%==1 (
17     cls
18
19     call py py1.py
20     call py py2.py
21     echo Raport gotowy!
22
23     goto :menu
24 )
25
26 if %wybor%==2 (
27     cls
28     if exist .\raporty\*.html (
29         for /f %%F in ('dir .\raporty\ /b /o-d') do (
30             call .\raporty\%%F
31             goto :menu
32         )
33     ) else (
34         echo Brak raportow!
35     )
36
37     goto :menu
38 )
```

```
40  if %wybor%==3 (
41      cls
42      if exist .\raporty\*.html (
43          del .\raporty\*.html
44          echo Raporty usuniete!
45      ) else (
46          echo Raporty usuniete!
47      )
48
49      goto :menu
50  )
51
52  if %wybor%==4 (
53      cls
54      call py py1.py
55      echo Plik uruchomiony!
56
57      goto :menu
58  )
59
60  if %wybor%==5 (
61      cls
62      call py py2.py
63      echo Plik uruchomiony!
64
65      goto :menu
66  )
67
68  if %wybor%==6 (
69      goto :eof
70  )
71
72  if %wybor% GTR 6 (
73      cls
74      goto :menu
75  )
76
77  pause
```


Plik *py1.py*

```
1      import os
2
3
4      def load_data(filename: str):
5          file = open('./input_data/' + filename, 'r')
6          data = []
7          for line in file:
8              line = line.strip()
9              data.append([int(value) for value in line.split(' ')])
10         file.close()
11         return data
12
13
14     def print_data(filename: str, data: str):
15         file = open('./output_data/' + filename, 'w')
16         file.write(data)
17         file.close()
18
19
20     def calc_sqr_vec_len(vector):
21         return pow(vector[0], 2) + pow(vector[1], 2)
22
23
24     def sum_vectors(v1, v2):
25         return [v1[0] + v2[0], v1[1] + v2[1]]
26
27
```

```

28 def solve():
29     for i in range(0, n):
30         skip = False
31         vector = [0, 0]
32         global max_distance
33         for j in range(0, len(indexes)):
34             if i == indexes[j]:
35                 skip = True
36                 break
37         if skip:
38             continue
39         else:
40             indexes.append(i)
41             for j in range(0, len(indexes)):
42                 vector = sum_vectors(vector, moves[indexes[j]])
43             results.append(vector)
44             distance = calc_sqr_vec_len(results[-1])
45             if distance > max_distance:
46                 max_distance = distance
47             while len(results) > 0:
48                 results.pop()
49             solve()
50             indexes.pop()
51
52
53 files = os.listdir('./input_data/')
54 file_number = len(files)
55 for f in range(1, file_number + 1):
56     data = load_data('in' + str(f) + '.txt')
57     n = data[0][0]
58     moves = data[1:]
59     max_distance = -1
60     results = []
61     indexes = []
62     solve()
63     print_data('out' + str(f) + '.txt', str(max_distance))

```

Plik *py2.py*

```
1  import os
2      from datetime import datetime
3  from datetime import date
4
5
6  def load_data(directory: str):
7      data = []
8      path = os.listdir(directory)
9      for file in path:
10         f = open(str(directory + file))
11         data.append(f.readlines())
12         f.close()
13     return data
14
15
16  def generate_template():
17      template = ''
18      for i in range(0, len(input_files)):
19         template += ''
20         <div class="line">
21             <div class="data">
22                 ''
23             for j in input_files[i]:
24                 template += f''
25                 <div class="input">{j}</div>
26                 ''
27             template += f''
28             </div>
29             <div class="data">{output_files[i][0]}</div>
30             </div>''
31     return template
```

```

34 def print_data():
35     time = datetime.now()
36     time = time.strftime("%H-%M-%S")
37     today = date.today()
38     today = today.strftime("%d-%m-%Y")
39     file = open(f'./raporty/{today}_{time}_raport.html', 'w')
40     file.write(f'''
41     <!DOCTYPE HTML>
42     <html lang="pl-PL">
43         <head>
44             <meta charset="utf-8">
45             <title>{today}_{time}_Raport</title>
46             <meta name="viewport" content="width=device-width, initial-scale=1">
47
48             <link rel="stylesheet" href="../style.css">
49         </head>
50
51         <body>
52             <h1>Kacper Grabiec</h1>
53             <h3>Języki Skryptowe PIONEK</h3>
54             {generate_template()}
55         </body>
56     </html>
57     ''')
58     file.close()
59
60
61 input_files = load_data('./input_data/')
62 output_files = load_data('./output_data/')
63
64 print_data()

```