

# 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.

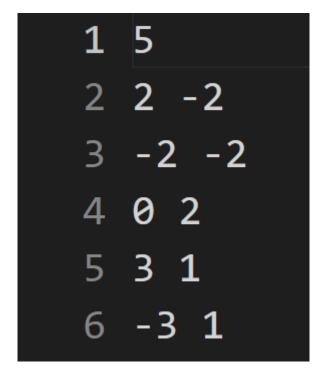
```
1. Stworz raport
2. Wyswietl ostatni raport
3. Usun raporty
4. Uruchom skrypt py1
5. Uruchom skrypt py2
6. Koniec
```

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:

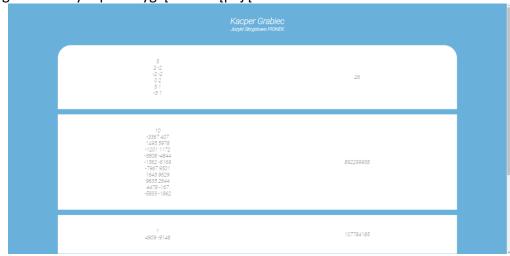


Pierwszy wiersz standardowego wejścia zawiera jedną dodatnią liczbę całkowitą n oznaczającą liczbę możliwych ruchów pionka. Każdy z kolejnych n wierszy zawiera dwie liczby całkowite x, y oddzielone pojedynczym odstępem i oznaczające wektor [x, y] opisujący możliwy ruch pionka.

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:



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

## Część II

### Opis działania

Do rozwiązania 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**

```
def solve():
    for i in range(0, n):
        skip = False
        vector = [0, 0]
        global max_distance
        for j in range(0, len(indexes)):
            if i == indexes[j]:
                skip = True
                break
        if skip:
            continue
        else:
            indexes.append(i)
            for j in range(0, len(indexes)):
                vector = sum_vectors(vector, moves[indexes[j]])
            results.append(vector)
            distance = calc_sqr_vec_len(results[-1])
            if distance > max_distance:
                max_distance = distance
            while len(results) > 0:
                results.pop()
            solve()
            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.

```
def load_data(filename: str):
    file = open('./input_data/' + filename, 'r')
    data = []
    for line in file:
       line = line.strip()
       data.append([int(value) for value in line.split(' ')])
    file.close()
    return data
```

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

Funkcja print\_data() tworzy plik z danymi wyjściowymi.

Funkcja *calc\_sqr\_vec\_len()* oblicza nam długość wektora natomiast *sum\_vectors()* zwraca sumę wektorów.

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.

Funkcja *generate\_template()* tworzy osobne elementy HTML dla każdych danych wejściowych oraz wyjściowych.

### Pełen kod aplikacji

Plik bat.bat

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

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

#### Plik *py1.py*

```
def load_data(filename: str):
     file = open('./input_data/' + filename, 'r')
     data = []
     for line in file:
         line = line.strip()
         data.append([int(value) for value in line.split(' ')])
     file.close()
    return data
 def print_data(filename: str, data: str):
     file = open('./output_data/' + filename, 'w')
     file.write(data)
    file.close()
def calc_sqr_vec_len(vector):
     return pow(vector[0], 2) + pow(vector[1], 2)
def sum_vectors(v1, v2):
    return [v1[0] + v2[0], v1[1] + v2[1]]
```

```
def solve():
    for i in range(0, n):
        skip = False
        vector = [0, 0]
        global max_distance
        for j in range(0, len(indexes)):
            if i == indexes[j]:
                skip = True
                break
        if skip:
            continue
        else:
            indexes.append(i)
            for j in range(0, len(indexes)):
                 vector = sum_vectors(vector, moves[indexes[j]])
            results.append(vector)
            distance = calc_sqr_vec_len(results[-1])
            if distance > max_distance:
                max_distance = distance
            while len(results) > 0:
                results.pop()
            solve()
            indexes.pop()
files = os.listdir('./input_data/')
file_number = len(files)
for f in range(1, file_number + 1):
    data = load_data('in' + str(f) + '.txt')
    n = data[0][0]
    moves = data[1:]
    max_distance = -1
    results = []
    indexes = []
    solve()
    print_data('out' + str(f) + '.txt', str(max_distance))
```

```
import os
      from datetime import datetime
      from datetime import date
      def load_data(directory: str):
          data = []
          path = os.listdir(directory)
          for file in path:
              f = open(str(directory + file))
              data.append(f.readlines())
              f.close()
          return data
      def generate_template():
          template = ''
          for i in range(0, len(input_files)):
              template += '''
                   <div class="data">
23
               for j in input_files[i]:
                   template += f'''
                       <div class="input">{j}</div>
               template += f'''
                   </div>
                   <div class="data">{output_files[i][0]}</div>
               </div>'''
           return template
```

```
def print_data():
    time = datetime.now()
    time = time.strftime("%H-%M-%S")
    today = date.today()
    file.write(f'''
            {generate_template()}
    file.close()
input_files = load_data('./input_data/')
output_files = load_data('./output_data/')
print_data()
```