



POLITECHNIKA ŚLĄSKA  
WYDZIAŁ MATEMATYKI STOSOWANEJ  
KIERUNEK INFORMATYKA

## **Języki Skryptowe**

dokumentacja projektu

Ireneusz Kosek

Gliwice, Styczeń 2023

---

# Część I

## Opis programu

Napisz program, który dla zadanej liczby naturalnej  $n \geq 1$ , zwracał będzie wartość sumy:

$$\sum_{i=1}^n \frac{1}{\sqrt{i} + \sqrt{i+1}}. \quad (1)$$

Za wynik numeryczny przyznawany jest jeden punkt, a za wynik w postaci numerycznej i symbolicznej (sposób zwracania wyniku symbolicznego pozostawiamy rozwiązującemu) komplet punktów.

## Instrukcja obsługi

Program uruchamiamy za pomocą pliku `run.sh`

wyбираamy jedną z interesujących nas opcji:

- Exec - uruchom program `projekt.py`.
- Backup - wykonaj kopie zapasową w folderze `./backup/$date`.
- Author - wyświetl informacje na temat autora.
- Exit - zakończ działanie programu.

program przyjmuje dane wejściowe w pliku `./input` i przechowuje dane wyjściowe w pliku `./output`. Przy każdym uruchomieniu program generuje raport w pliku `./output.html`

---

## Część II

### Opis działania

By ułatwić obliczenia program stosuje przekształconą postać wzoru z polecenia:

$$\sum_{i=1}^n \frac{1}{\sqrt{i} + \sqrt{i+1}} = \sqrt{n+1}. \quad (2)$$

dowód:

$$\begin{aligned} \sum_{i=1}^n \frac{1}{\sqrt{i} + \sqrt{i+1}} &= \sum_{i=1}^n \frac{1}{\sqrt{i+1} + \sqrt{i}} \cdot \frac{\sqrt{i+1} - \sqrt{i}}{\sqrt{i+1} - \sqrt{i}} \\ &= \sum_{i=1}^n \frac{\sqrt{i+1} - \sqrt{i}}{i+1-i} \\ &= \sum_{i=1}^n \sqrt{i+1} - \sqrt{i} \\ &= \sqrt{0+1} - \sqrt{0} + \sqrt{1+1} - \sqrt{1} + \dots + \sqrt{n-1+1} - \sqrt{n-1} + \sqrt{n+1} - \sqrt{n} \\ &= \underline{\sqrt{1}} + \sqrt{2} - \underline{\sqrt{1}} + \dots + \underline{\sqrt{n}} - \sqrt{n-1} + \sqrt{n+1} - \underline{\sqrt{n}} \\ &= \sqrt{n+1}. \end{aligned} \quad (3)$$

Każdy poprzedni wyraz postaci  $\sqrt{i-1+1}$  jest redukowany przez następny postaci  $-\sqrt{i}$  pozostawiając jedynie ostatni wyraz  $\sqrt{i+1} = \sqrt{n+1}$

---

## Algorytmy

Schemat przejścia programu wygląda następująco:

**Data:** Dane wejściowe: liczba  $n$

**Result:** Suma ciągu w postaci numerycznej i symbolicznej

**if**  $n < 1$  **then**

  | zakończ działanie

**end**

$suma := \sqrt{n+1};$

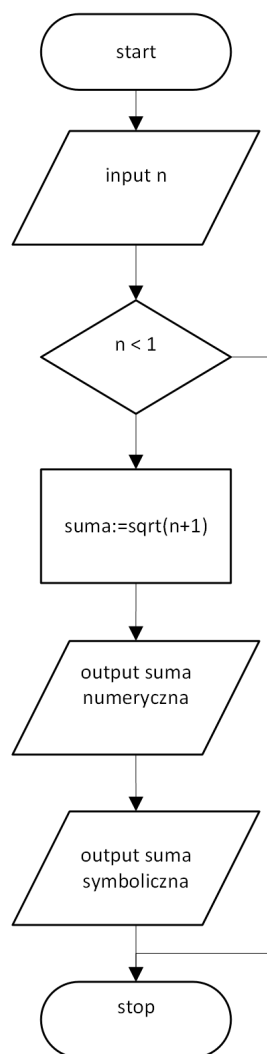
Wydrukuj sumę w postaci numerycznej;

Wydrukuj sumę w postaci symbolicznej;

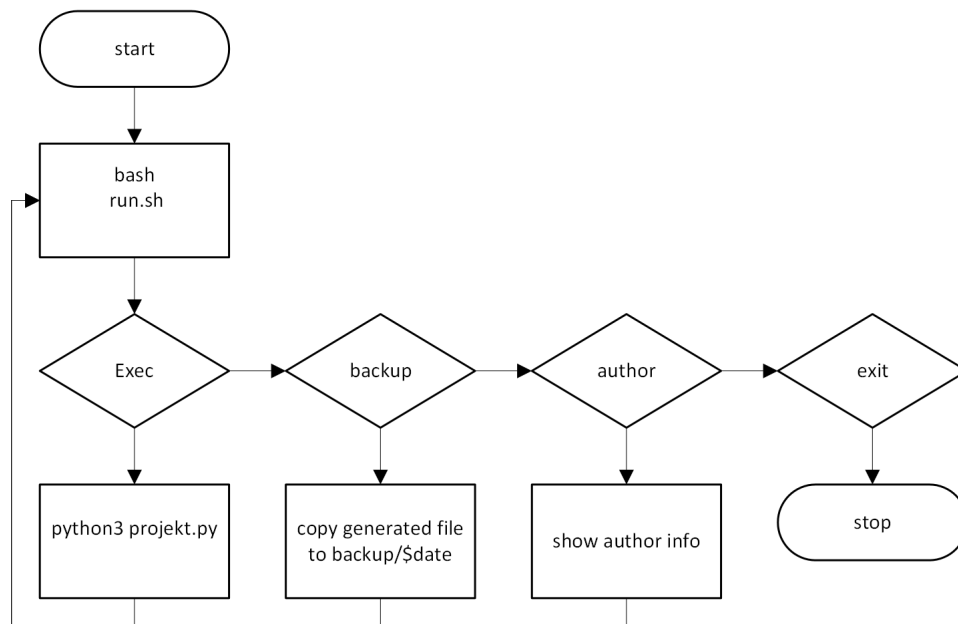
**Algorithm 1:** Uproszczona forma działania programu.

---

## Schemat blokowy



Rysunek 1: Schemat działania algorytmu



Rysunek 2: Schemat działania programu

## Implementacja systemu

Opis, zasada i działanie programu ze względu na podział na pliki, następnie funkcje programu wraz ze szczegółowym opisem działania (np.: formie pseudokodu, czy odniesienia do równania) Program składa się z pliku.sh zawierającego skrypt generujący menu programu:

```
1 #!/usr/bin/env bash
2 while true; do
3     exec 3>&1
4     selection=$(dialog \
5         --backtitle "Projekt" \
6         --title "Projekt" \
7         --cancel-label "Exit" \
8         --ok-label "OK" \
9         --hline "" \
10        --default-item "${selection:-}" \
11        --menu "" 16 0 16 \
12        "Exec" "Execute the algorithm" \
13        "Backup" "Backup the algorithm's output" \
```

---

```

14         "Authors" "See info about Author(s)" \
15         "Exit" "Exit the program" \
16         2>&1 1>&3)
17     exit_status=$?
18     case $exit_status in
19         $DIALOG_CANCEL)
20         clear
21         exit 0
22         ;;
23         $DIALOG_ESC)
24         clear
25         exit
26         ;;
27     esac
28     case $selection in
29         0)
30         clear
31         exit 0
32         ;;
33         Exec)
34             if [[ "0" != "$exit_status" ]]; then
35                 continue
36             fi
37             clear
38             python3 projekt.py
39             ;;
40         Backup)
41             mkdir -p backup/${date +%Y-%m-%d_%H-%M-%S}/
42             set -x
43             cp -r output backup/${date +%Y-%m-%d_%H-%M-%S}/output 3>>
                logs 2>>logs
44             cp -r input backup/${date +%Y-%m-%d_%H-%M-%S}/input 3>>logs
                2>>logs

```

---

```
45         mv *.html "backup/$(date +%Y-%m-%d_%H-%M-%S)/" 3>>>logs 2>>>
           logs
46         set +x
47         ;;
48     Authors)
49         dialog \
50             --backtitle "Projekt" \
51             --title "Authors" \
52             --msgbox "Author: Ireneusz Kosek" 0 0
53         ;;
54     Exit)
55         clear
56         exit 0
57         ;;
58     esac
59     exec 3>&-
60 done
```

---

oraz z pliku projekt.py w którym znajduje się klasa odpowiedzialna za rozwiązanie zadanego problemu:

```
1 class Algorithm():
2     def solution(self, n):
3         sum = math.sqrt(n + 1)
4         output=""
5         output+=f"numeric sum: {sum}\n"
6         output+=f"symbolic sum: sqrt({n+1})\n"
7         return output
```

---

jak również część programu odpowiedzialna za pobranie danych, wywołanie obliczeń i wygenerowanie wartości wyjściowych

```
1 if __name__ == "__main__":
2     exitcode = 0
3     htmlendl = "<br>\n"
```



---

```
4     now = datetime.now()
5     fulldate = now.strftime("%d-%m-%Y %H:%M:%S")
6
7     file_input = open("input", "r")
8     file_output = open("output", "w")
9     file_html = open("output.html", "w")
10
11     file_html.write("<html><body>")
12     file_html.write(f"<h1>{fulldate}</h1>\n")
13
14     calculate = Algorithm()
15
16     for line in file_input:
17         try:
18             file_html.write(f"Enter n: {line}"+htmlendl)
19             n = int(line)
20
21             if n < 1:
22                 raise Exception("n must be greater than 0")
23
24             output=calculate.solution(n)
25             output_html=output.replace("\n", htmlendl)
26
27             file_output.write(output)
28             file_html.write(output_html)
29         except ValueError:
30             file_html.write("n must be a number"+htmlendl)
31             exitcode = 1
32         except Exception as e:
33             file_html.write(str(e)+htmlendl)
34             exitcode = 1
35     file_html.write("</body></html>")
36
```

---

```
37     file_input.close()
38     file_output.close()
39     file_html.close()
40
41     sys.exit(exitcode)
```

---

## Testy

Dla zadanych danych testowych program zwraca następujące wartości:

```
1 Enter n: 1
2 numeric sum: 1.4142135623730951
3 symbolic sum: sqrt(2)
```

---

```
1 Enter n: 2
2 numeric sum: 1.7320508075688772
3 symbolic sum: sqrt(3)
```

---

```
1 Enter n: 0
2 n must be greater than 0
```

---

```
1 Enter n: -1
2 n must be greater than 0
```

---

```
1 Enter n: foo
2 n must be a number
```

---

```
1 Enter n: 2147483647
2 numeric sum: 46340.95001184158
3 symbolic sum: sqrt(2147483648)
```

---

```
1 Enter n: -2147483647
2 n must be greater than 0
```

---

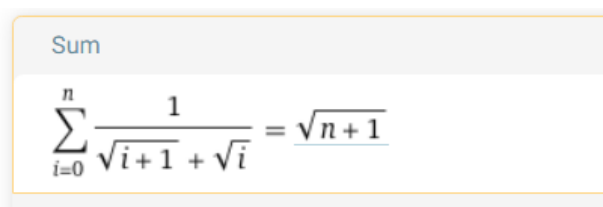
---

## Eksperymenty oraz historia rozwoju

Pierwszą myślą przy podejściu do realizacji projektu była obawa dotycząca problemu, jaki mogą sprawić obliczenia symboliczne. Pojawił się więc pomysł, by skorzystać z zewnętrznego API (np. wolfram alpha, Mathematica), aby zmarginalizować zagadnienie obliczeń symbolicznych do minimum. W tym celu niezbędnym okazała się lektura dokumentacji oraz wygenerowanie Tokenu do komunikowania się z RESTful API WolframAlpha.



Jednakże z tyłu głowy nadal istniała myśl, że coś może pójść nie tak, więc planem zapasowym było użycie bibliotek do generowania obliczeń symbolicznych. Całe podejście do problemu zmieniło się jednak po wpisaniu wyrażenia z treści zadania do programu liczącego WolframAlpha:

The image shows a screenshot of a WolframAlpha result. The title 'Sum' is in blue. Below it is the mathematical expression 
$$\sum_{i=0}^n \frac{1}{\sqrt{i+1} + \sqrt{i}} = \sqrt{n+1}$$
 rendered in a serif font.

Z nabytą wiedzą wyprowadzenie dowodu na powyższe równanie było już tylko formalnością. Jako wniosek można wyciągnąć to, że najprostsze rozwiązania to czasem nie te najbardziej sprytne, tylko najbardziej oczywiste.

---

# Pełen kod aplikacji

run.sh:

```
1  #!/usr/bin/env bash
2  while true; do
3      exec 3>&1
4      selection=$(dialog \
5          --backtitle "Projekt" \
6          --title "Projekt" \
7          --cancel-label "Exit" \
8          --ok-label "OK" \
9          --hline "" \
10         --default-item "${selection:-}" \
11         --menu "" 16 0 16 \
12         "Exec" "Execute the algorithm" \
13         "Backup" "Backup the algorithm's output" \
14         "Authors" "See info about Author(s)" \
15         "Exit" "Exit the program" \
16         2>&1 1>&3)
17     exit_status=$?
18     case $exit_status in
19         $DIALOG_CANCEL)
20         clear
21         exit 0
22         ;;
23         $DIALOG_ESC)
24         clear
25         exit
26         ;;
27     esac
28     case $selection in
29         0)
30         clear
```

---

```

31         exit 0
32     ;;
33     Exec)
34         if [[ "0" != "$exit_status" ]]; then
35             continue
36         fi
37         clear
38         python3 projekt.py
39         ;;
40     Backup)
41         mkdir -p backup/$(date +%Y-%m-%d_%H-%M-%S)/
42         set -x
43         cp -r output backup/$(date +%Y-%m-%d_%H-%M-%S)/output 3>>
            logs 2>>logs
44         cp -r input backup/$(date +%Y-%m-%d_%H-%M-%S)/input 3>>logs
            2>>logs
45         mv *.html "backup/$(date +%Y-%m-%d_%H-%M-%S)/" 3>>logs 2>>
            logs
46         set +x
47         ;;
48     Authors)
49         dialog \
50             --backtitle "Projekt" \
51             --title "Authors" \
52             --msgbox "Author: Ireneusz Kosek" 0 0
53         ;;
54     Exit)
55         clear
56         exit 0
57         ;;
58     esac
59     exec 3>&-
60 done

```

---

---

projekt.py:

```
1 import math
2 import sys
3 from datetime import datetime
4
5 class Algorithm():
6     def solution(self, n):
7         sum = math.sqrt(n + 1)
8         output=""
9         output+=f"numeric sum: {sum}\n"
10        output+=f"symbolic sum: sqrt({n+1})\n"
11        return output
12
13 if __name__ == "__main__":
14     exitcode = 0
15     endl = "<br>\n"
16     now = datetime.now()
17     fulldate = now.strftime("%d-%m-%Y %H:%M:%S")
18
19     file_input = open("input", "r")
20     file_output = open("output", "w")
21     file_html = open("output.html", "w")
22
23     file_html.write("<html><body>")
24     file_html.write(f"<h1>{fulldate}</h1>\n")
25
26     calculate = Algorithm()
27
28     for line in file_input:
29         try:
30             file_html.write(f"Enter n: {line}" + endl)
31             n = int(line)
32
```

---

```
33         if n < 1:
34             raise Exception("n must be greater than 0")
35
36         output=calculate.solution(n)
37         output_html=output.replace("\n", htmlendl)
38
39         file_output.write(output)
40         file_html.write(output_html)
41     except ValueError:
42         file_html.write("n must be a number"+htmlendl)
43         exitcode = 1
44     except Exception as e:
45         file_html.write(str(e)+htmlendl)
46         exitcode = 1
47     file_html.write("</body></html>")
48
49     file_input.close()
50     file_output.close()
51     file_html.close()
52
53     sys.exit(exitcode)
```

---