****

**Algorytmy i struktury danych**

*Laboratorium sprawozdania*

Prowadzący: Autor:

mgr inż. Ewa Żesławska Kacper Kulig

w69199

Kierunek: IID-P/2022, grupa SL05

Rzeszów r.a. 2022/2023

Spis treści

[**Laboratorium 1** 3](#_Toc136292788)

[**Laboratorium 2** 8](#_Toc136292789)

[**Laboratorium 3** 10](#_Toc136292790)

[**Laboratorium 4** 17](#_Toc136292791)

[**Laboratorium 5** 19](#_Toc136292792)

[**Laboratorium 6** 22](#_Toc136292793)

[**Laboratorium 7** 23](#_Toc136292794)

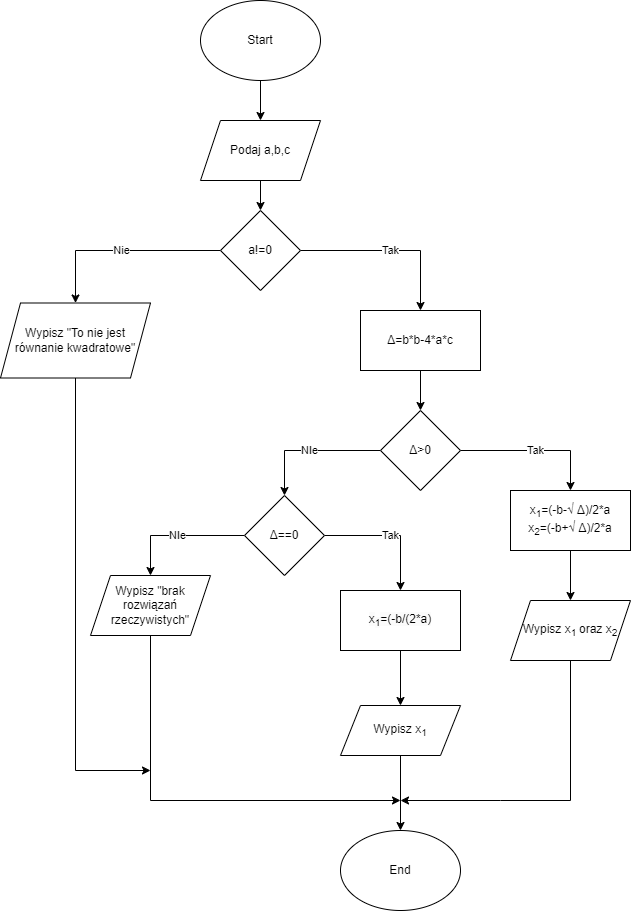
[**Laboratorium 8** 28](#_Toc136292795)

[**Laboratorium 9** 28](#_Toc136292796)

**Laboratorium 1**

**Zadanie 1.**

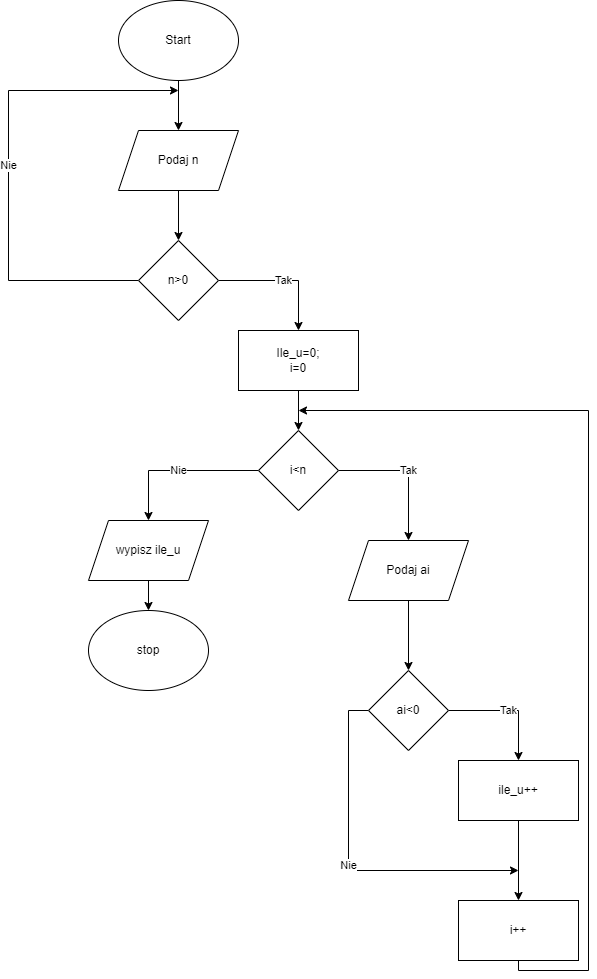
Na Rysunku 1 przedstawiono algorytm wyznaczania pierwiastków równania kwadratowego. Implementacja algorytmu w języku Python umieszczona jest w repozytorium pod adresem <https://github.com/Kacper20001/Algorytmy-i-struktura-danych-laboratorium-2022-2023.git> [[1]](#footnote-1).



Rysunek . Schemat blokowy algorytmu wyznaczenia pierwiastków równania kwadratowego.

**Zadanie 2.**

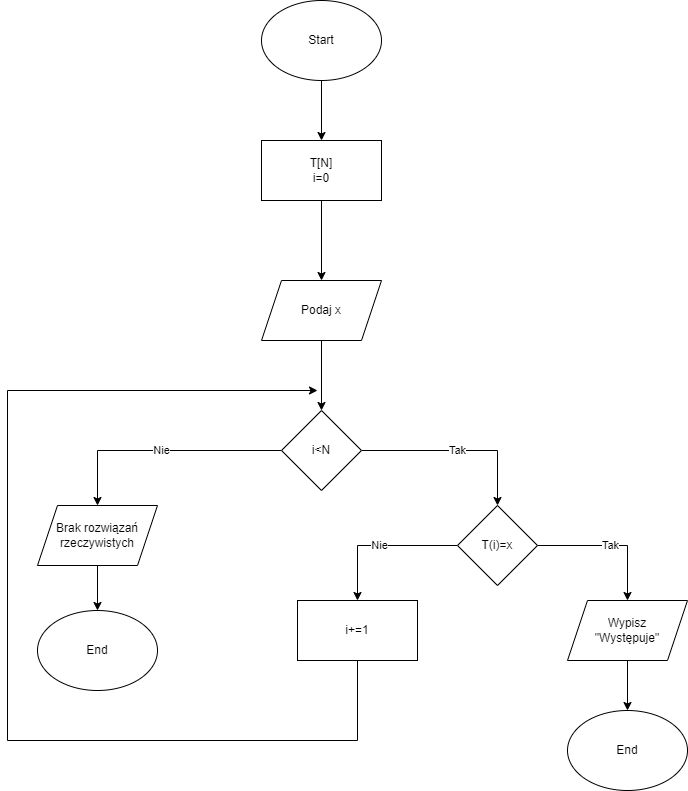
Na Rysunku 2 przedstawiono algorytm wczytywania ciągu n liczb całkowitych (N>0) i wyznaczania ilości liczb ujemnych w tym ciągu.



Rysunek . Schemat blokowy algorytmu wczytywania ciągu n liczb całkowitych (N>0) i wyznaczania ilości liczb ujemnych w tym ciągu.

**Zadanie 3.**

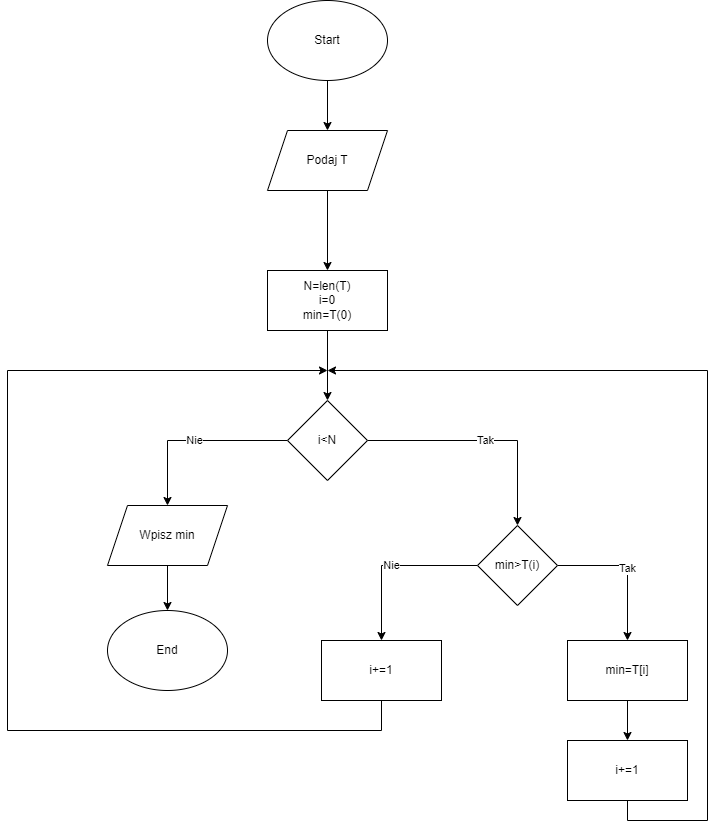
Na Rysunku 3 przedstawiono algorytm, który sprawdza czy podana przez użytkownika wartość występuje w tablicy jednowymiarowej.



Rysunek . Schemat blokowy algorytmu, który sprawdza czy podana przez użytkownika wartość występuje w tablicy jednowymiarowej.

**Zadanie 4.**

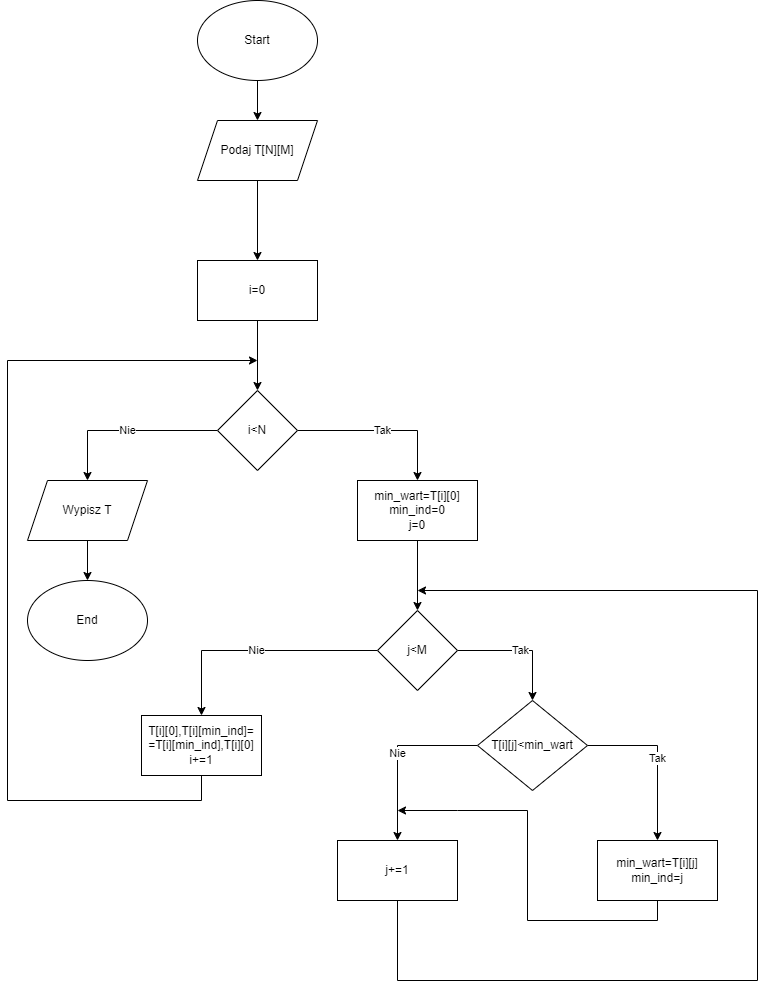
Na Rysunku 4 przedstawiono algorytm wyszukiwania w tablicy jednowymiarowej minimalnej wartości. Implementacja algorytmu w języku Python umieszczona jest w repozytorium.



Rysunek . Schemat blokowy algorytmu wyszukiwania w tablicy jednowymiarowej minimalnej wartości.

**Zadanie 5.**

Na Rysunku 5 przedstawiono algorytm wyszukiwania w tablicy dwuwymiarowej minimalnej wartości w każdym wierszu. Po znalezieniu minimalnej wartości wstawiana jest na początek danego wiersza (poprzez zamianę miejsc).



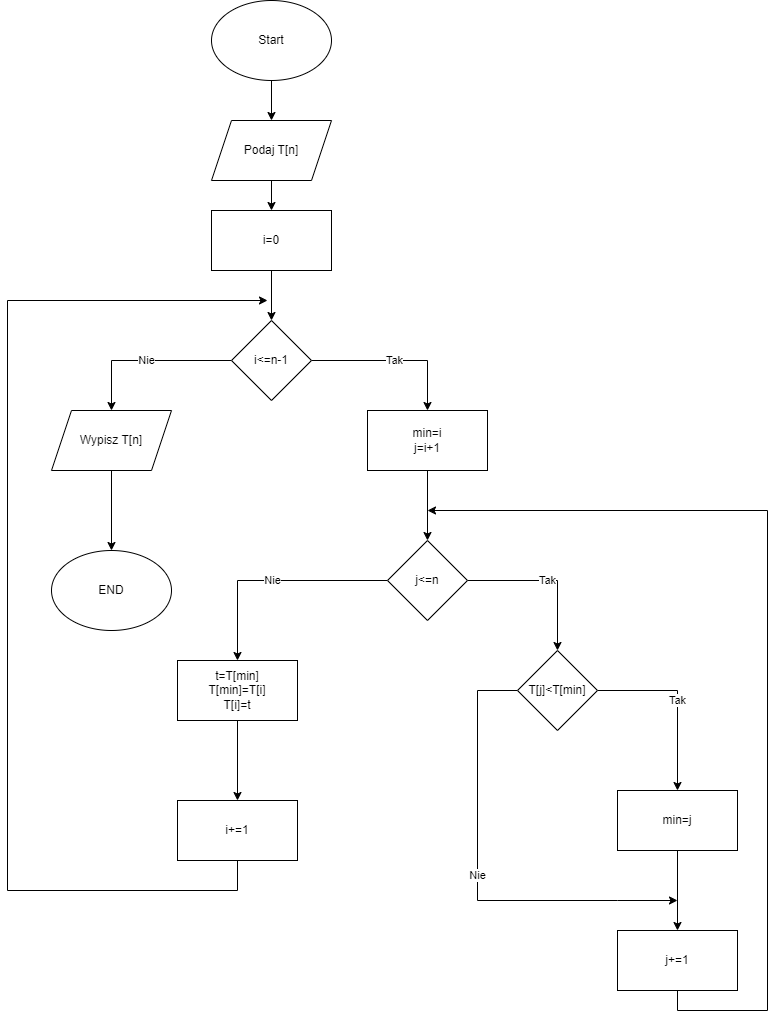
Rysunek . Schemat blokowy algorytmu wyszukiwania w tablicy dwuwymiarowej minimalnej wartości w każdym wierszu. Po znalezieniu minimalnej wartości wstawiana jest na początek danego wiersza (poprzez zamianę miejsc).

**Laboratorium 2**

Rozwiązanie zadań z laboratorium 2 znajduje się w pliku xls dołączonym do niniejszego sprawozdania.

**Zadanie 1.**

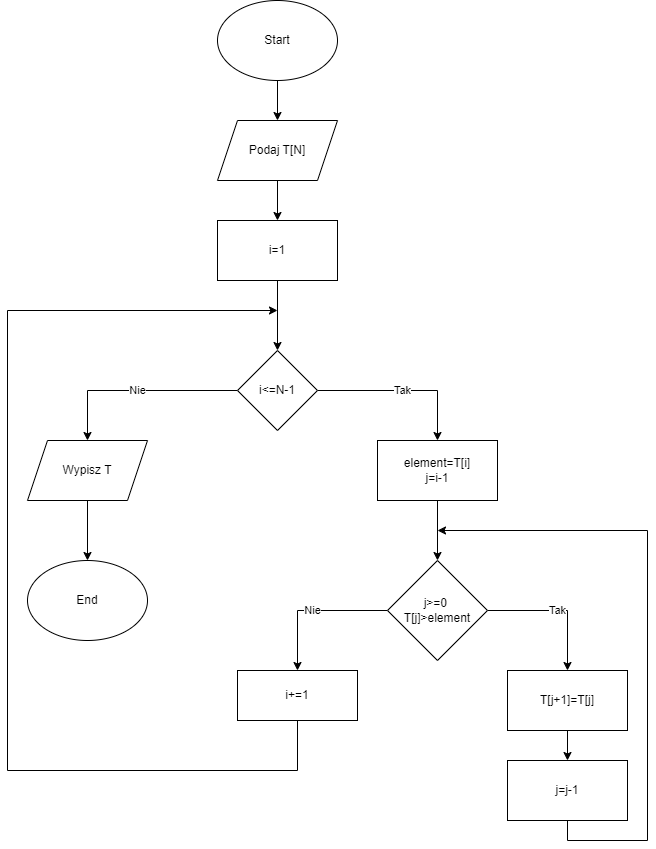
Na Rysunku 6 przedstawiono algorytm sortowania przez wybór.



Rysunek . Schemat blokowy algorytmu sortowania przez wybór.

**Zadanie 2.**

Na Rysunku 7 przedstawiono algorytm sortowania przez wstawianie.



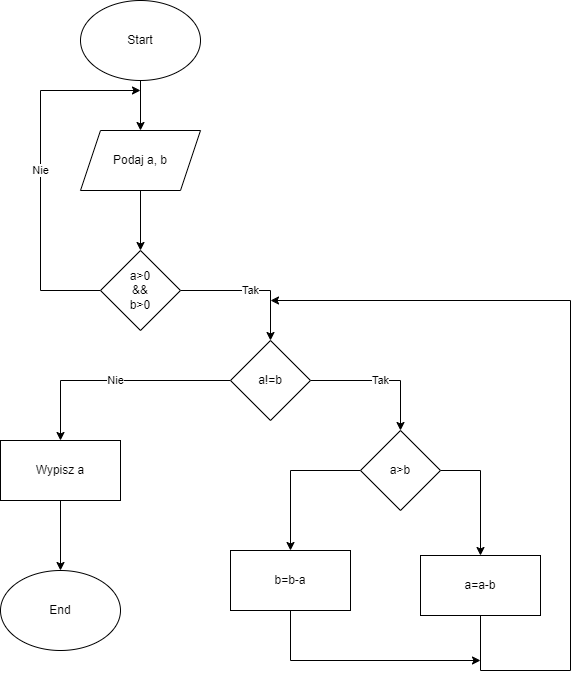
Rysunek . Schemat blokowy algorytmu sortowania przez wstawianie.

**Laboratorium 3**

**Zadanie 1.**

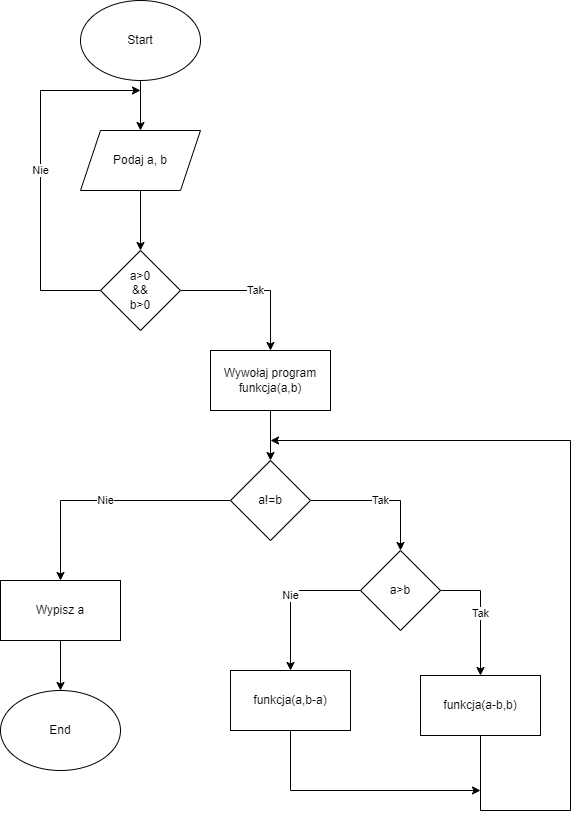
Na rysunkach przedstawiono algorytm NWD w dwóch wersjach oraz schemat dla nich w formie iteracyjnej oraz rekurencyjnej.

Wersja I – nieoptymalna postać algorytmu NWD w formie iteracyjnej.



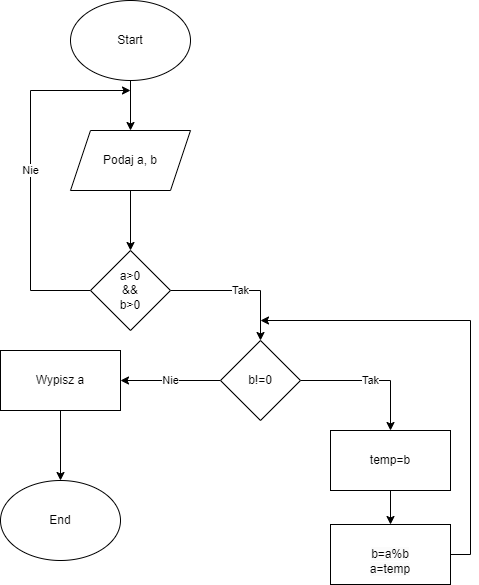
Rysunek . Schemat nieoptymalnej postaci algorytmu NWD w postaci iteracyjnej.

Wersja I – nieoptymalna postać algorytmu NWD w formie rekurencyjnej,



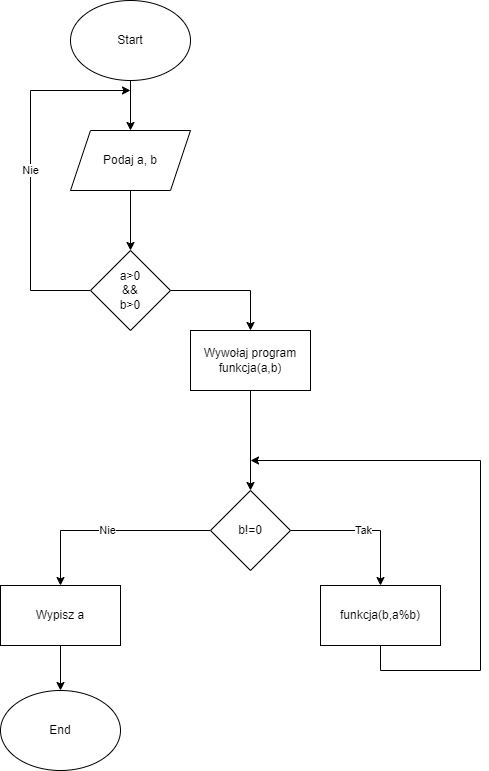
Rysunek . Schemat nieoptymalnej postaci algorytmu NWD w postaci rekurencyjnej.

Wersja II – zoptymalizowana postać algorytmu NWD w formie iteracyjnej.



Rysunek . Schemat zoptymalizowanej postaci algorytmu NWD w postaci iteracyjnej.

Wersja II – zoptymalizowana postać algorytmu NWD w formie rekurencyjnej.



Rysunek . Schemat zoptymalizowanej postaci algorytmu NWD w postaci rekurencyjnej.

**Zadanie 2.**

Rozwiązanie zadania 2 znajduje się w poniżej przedstawionej tabeli.

|  |  |  |
| --- | --- | --- |
| i | wynik(i) | Ilość wywołań funkcji bez wywołania głównego |
| 2 | 1 | 0 |
| 3 | 1 | 1 |
| 4 | 3 | 3 |
| 5 | 3 | 4 |
| 6 | 5 | 7 |
| 7 | 5 | 8 |
| 8 | 9 | 14 |

Tabela . Rozwiązanie zadania 2.

**Zadanie 3.**

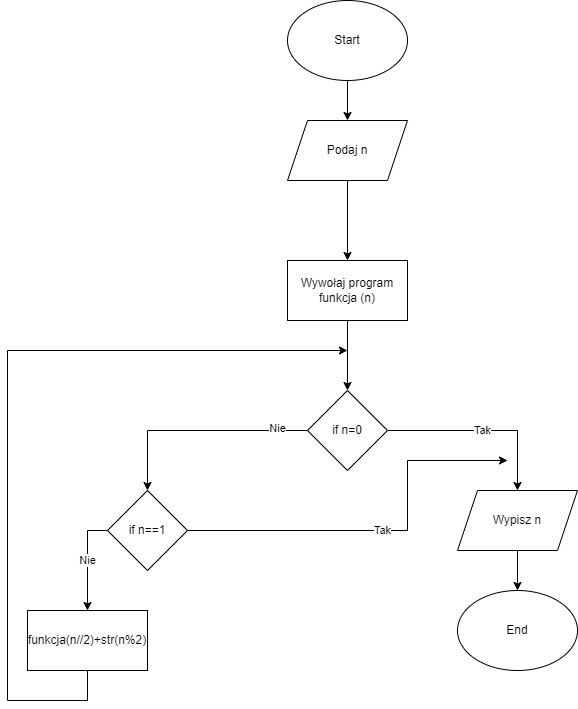
Rozwiązanie zadania 3 znajduje się w poniżej przedstawionej tabeli.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 |  | P | **F** |
| 2 |  | **P** | F |
| 3 |  | **P** | F |
| 4 |  | P | **F** |

Tabela . Rozwiązanie zadania 3.

**Zadanie 4.**

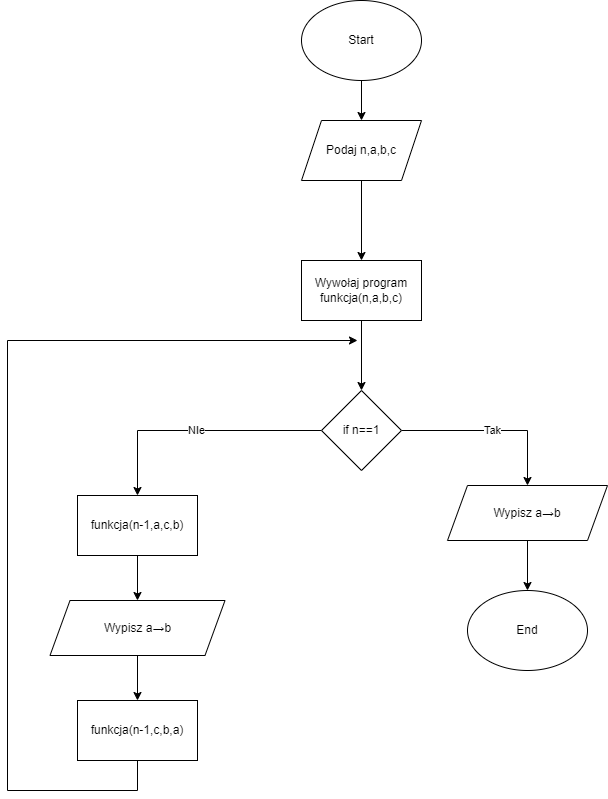
Na Rysunku 12 przedstawiono rekurencyjny algorytm zamiany liczby dziesiętnej na binarną.



Rysunek . Algorytm rekurencyjny zamiany liczby dziesiętnej na binarną.

**Zadanie 5.**

Na Rysunku 13 przedstawiono rekurencyjny algorytm rekurencyjny wieży Hanoi.



Rysunek . Rekurencyjny algorytm wieży Hanoi.

Lista kroków:

1. Jeśli n=1 przenieś krążek z palika źródłowego na docelowy palik i zakończ.

2. W przeciwnym przypadku przenieś (n-1) krążków z palika źródłowego na palik pomocniczy przy użyciu docelowego palika jako pomocniczego.

3. Przenieś największy krążek z palika źródłowego na docelowy palik.

4. Przenieś (n-1) krążków z palika pomocniczego na docelowy palik przy użyciu palika źródłowego jako pomocnika,

5. Powtarzaj kroki 2-4, aż wszystkie krążki zostaną przeniesione z palika źródłowego na docelowy palik.

# **Laboratorium 4**

**Zadanie 1.**

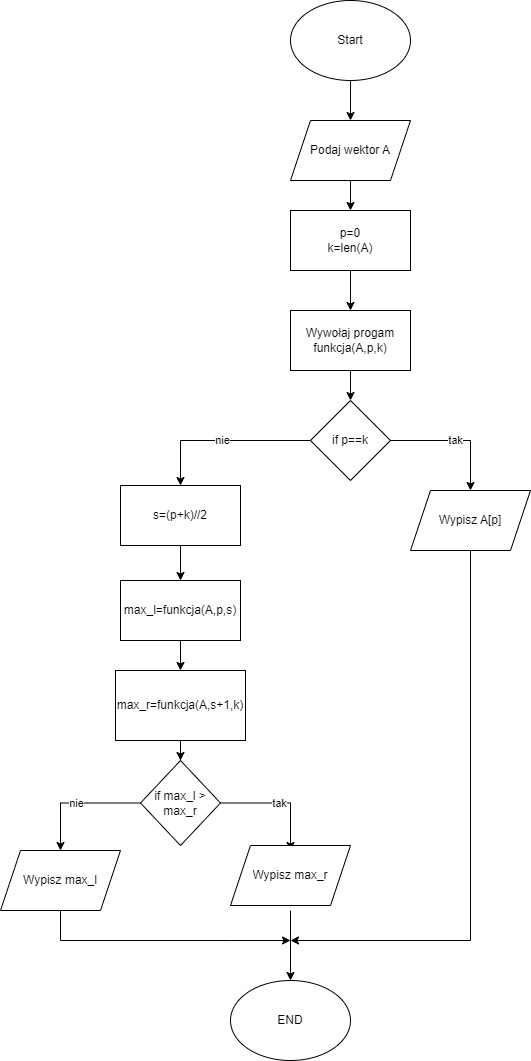
Analiza działania algorytmu sortowania szybkiego znajduje się w pliku xls dołączonym do niniejszego sprawozdania.

**Zadanie 2.**

Analiza działania algorytmu sortowania przez scalanie znajduje się w pliku xls dołączonym do niniejszego sprawozdania.

**Zadanie 3.**

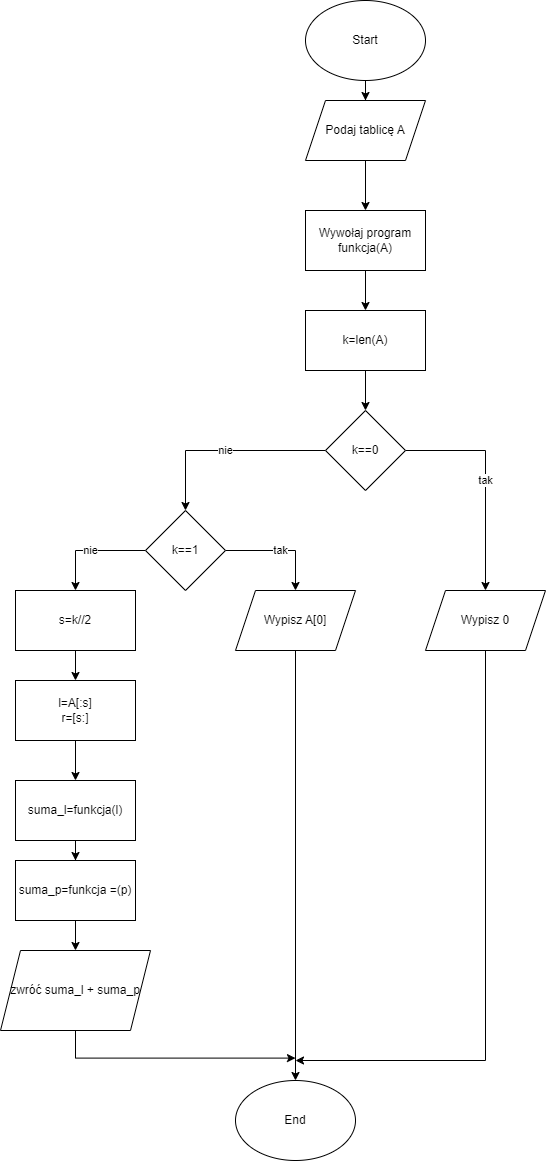
Na rysunku 14 przedstawiono schemat algorytmu wyznaczania największego elementy wektora z zastosowaniem metody „dziel i zwyciężaj”.



Rysunek . Schemat algorytmu wyznaczania największego elementu wektora z wykorzystaniem metody ,,dziel i zwyciężaj".

**Zadanie 4.**

Na rysunku 15 przedstawiono schemat algorytmu liczącego sumę elementów w tablicy z zastosowaniem metody „dziel i zwyciężaj”.



Rysunek . Schemat algorytmu liczącego sumę elementów w tablicy z zastosowaniem metody ,,dziel i zwyciężaj".

**Laboratorium 5**

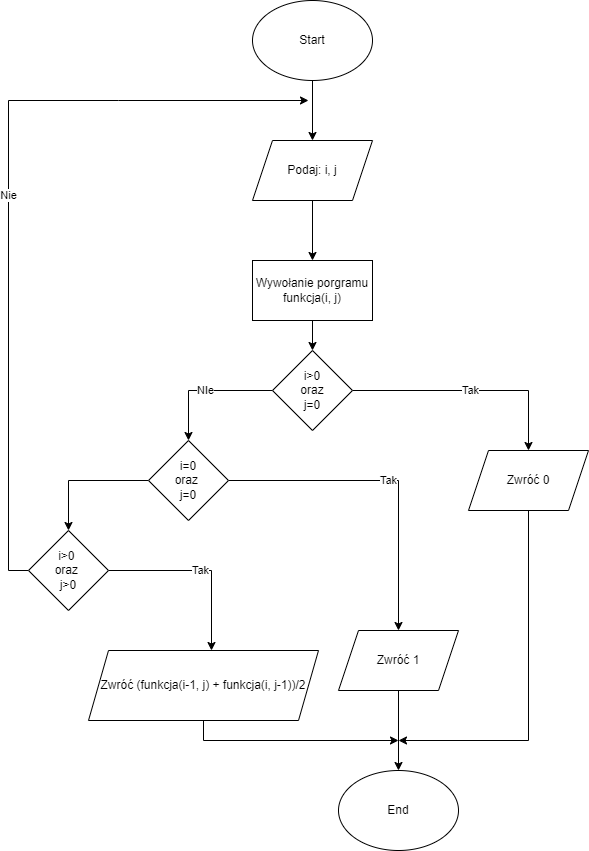
**Zadanie 1.**

Korzystając z techniki programowania dynamicznego napisano program obliczania elementów ciągu Fibonacciego.

**Zadanie 2.**

Korzystając z techniki programowania dynamicznego wyznaczono wartość wyrażania:

Rozwiązanie zadania znajduję się w pliku .xls dołączonym do niniejszego sprawozdania. Na rysunku 16 przedstawiono schamat algorytmu realizującego powyższe zadanie.

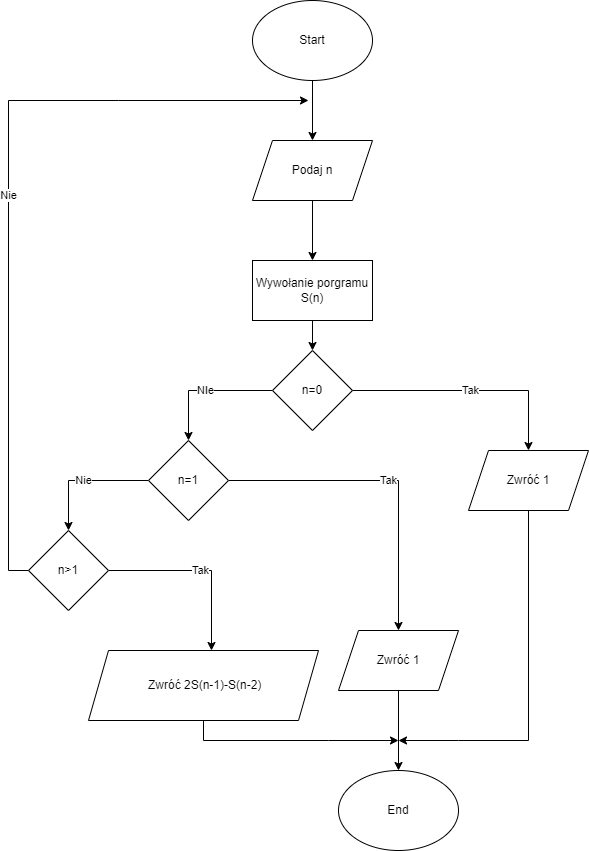


Rysunek . Algorytm realizujący równanie z zadania 2 z wykorzystaniem programowania dynamicznego.

**Zadanie 3.**

Na rysunku 17 przedstawiono schemat blokowy algorytmu obliczania n-tego wyrazu ciągu przedstawionego równaniem:

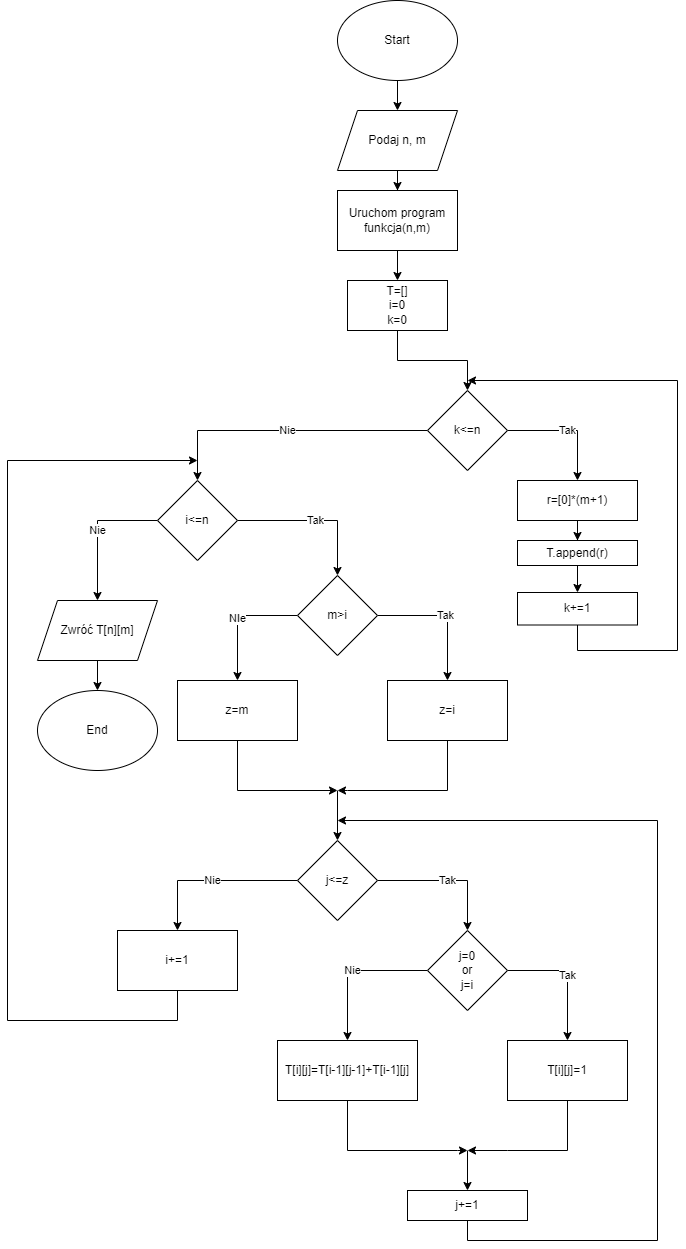
metodą programowania dynamicznego.



Rysunek . Algorytm obliczania n-tego wyrazu ciągu równania z zadania 3 wykorzystując metodę programowania dynamicznego

**Zadanie 4.**

Na rysunku 18 przestawiono schemat blokowy algorytmu wyznaczania współczynnika dwumianowego korzystając z techniki programowania dynamicznego.

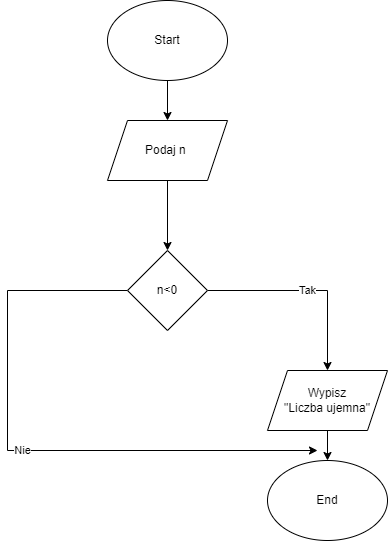


Rysunek . Schemat algorytmu wyznaczania współczynnika dwumianowego.

**Laboratorium 6**

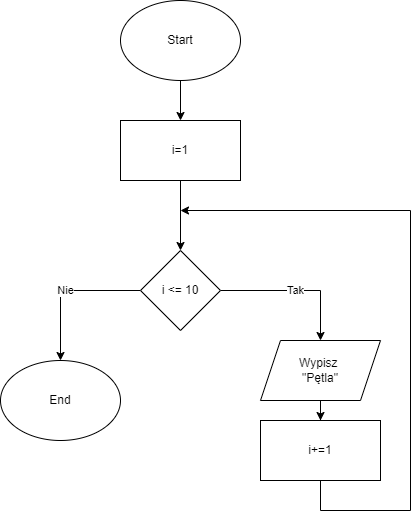
**Zadanie 1.**

Na rysunku 19 przestawiono schemat blokowy algorytmu pobierającego liczbę od użytkownika oraz w przypadku liczby ujemnej wypisze „Liczba ujemna”.



Rysunek . Schemat algorytmu pobierającego liczbę od użytkownika oraz sprawdzenie czy jest ona ujemna.

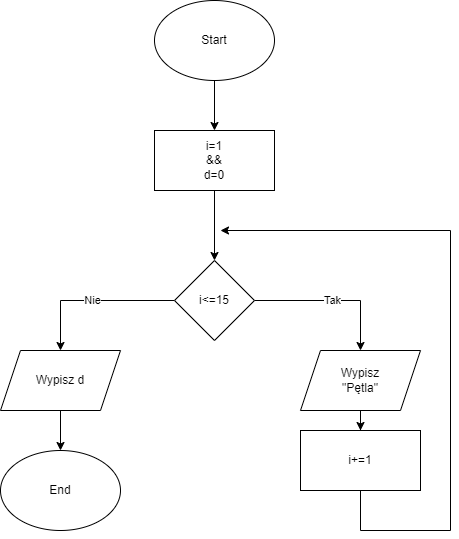
Na rysunku 20 przestawiono schemat blokowy algorytmu wypisującego 10 razy napis „Pętla”.



Rysunek . Schemat algorytmu wypisującego 10 razy napis "Pętla".

**Zadanie 2.**

Na rysunku 21 przestawiono schemat blokowy algorytmu pobierającego od użytkownika 15 liczb oraz liczącego ilość liczb dodatnich.



Rysunek . Schemat blokowy algorytmu pobierającego od użytkownika 15 liczb oraz zliczającego ilość liczb dodatnich.

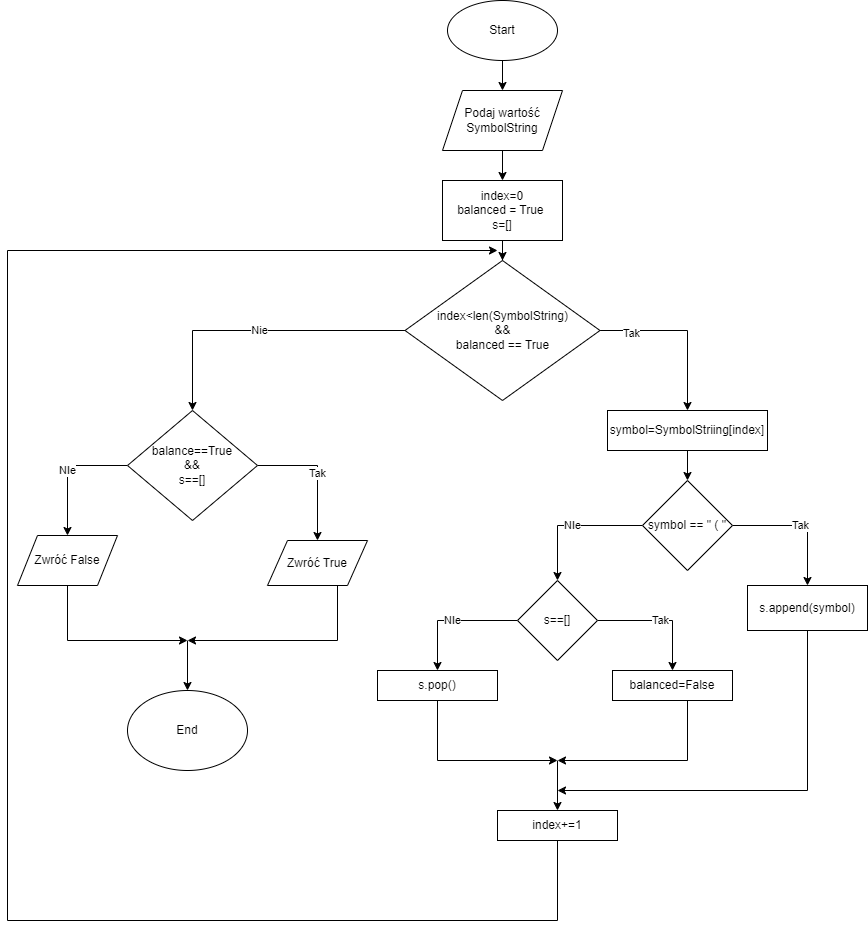
**Laboratorium 7**

**Zadanie 1.**

Zaimplementowanie struktury stosu w Pythonie.

**Zadanie 2.**

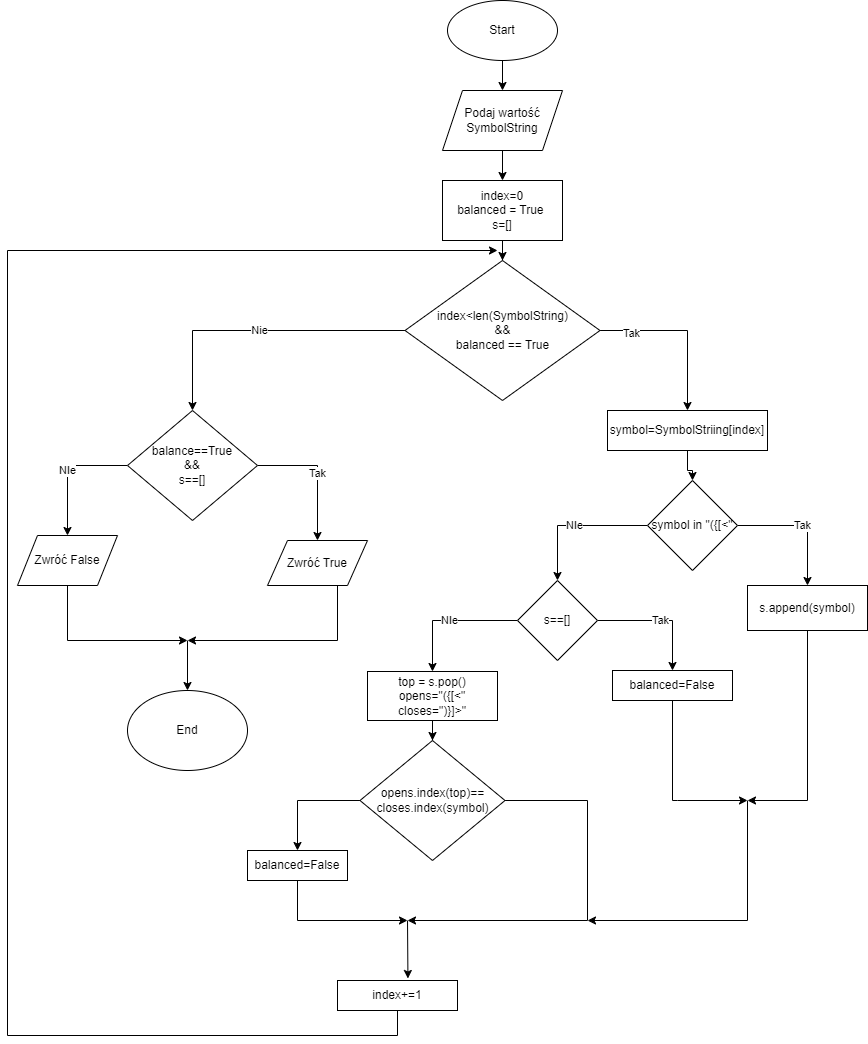
Na rysunku 22 przestawiono schemat blokowy algorytmu sprawdzania poprawności nawiasów.



Rysunek . Schemat blokowy algorytmu sprawdzającego poprawność nawiasów.

**Zadanie 3.**

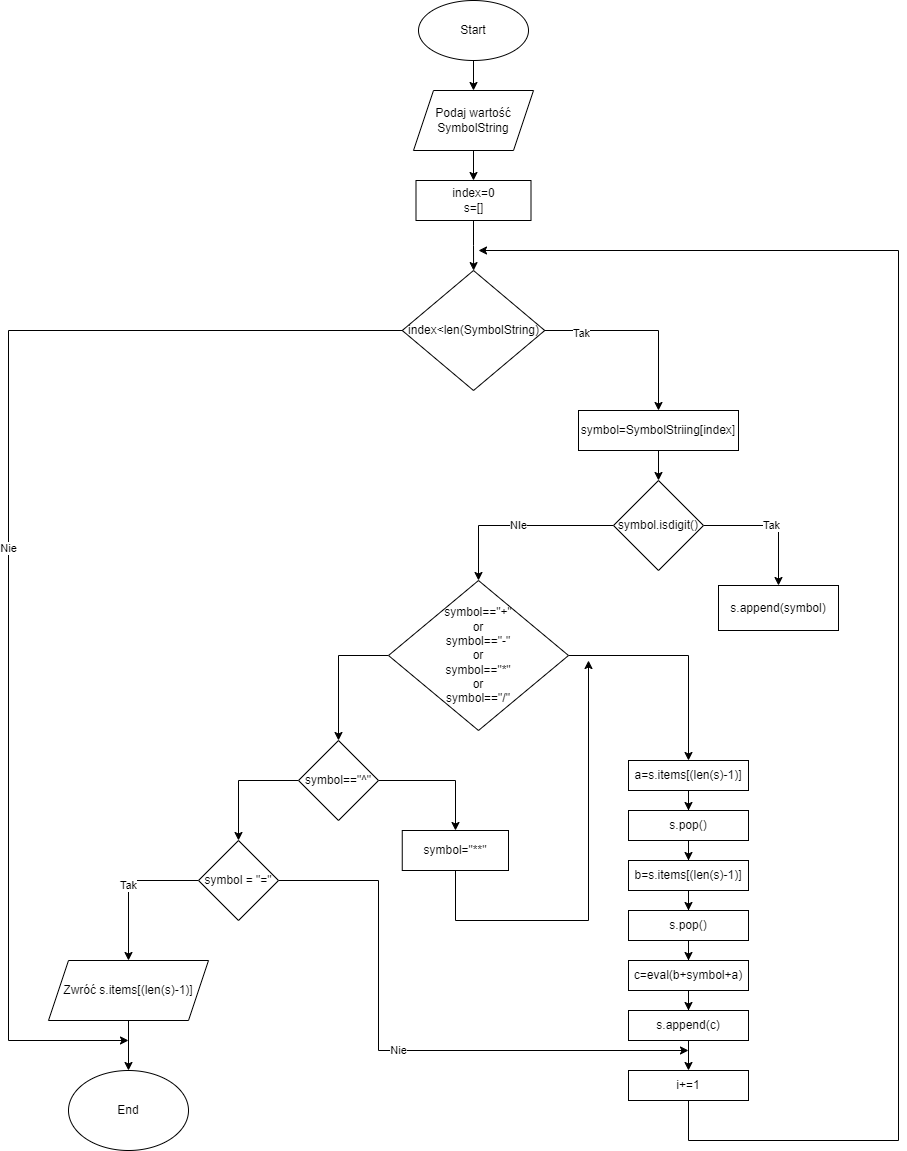
Na rysunku 23 przestawiono schemat blokowy algorytmu sprawdzania poprawności nawiasów.



Rysunek . Schemat blokowy algorytmu sprawdzającego poprawność różnych symboli.

**Zadanie 4.**

Na rysunku 24 przestawiono schemat blokowy algorytmu czytającego wyrażenie arytmetyczne w notacji postfiksowej oraz obliczającego jego wartość.



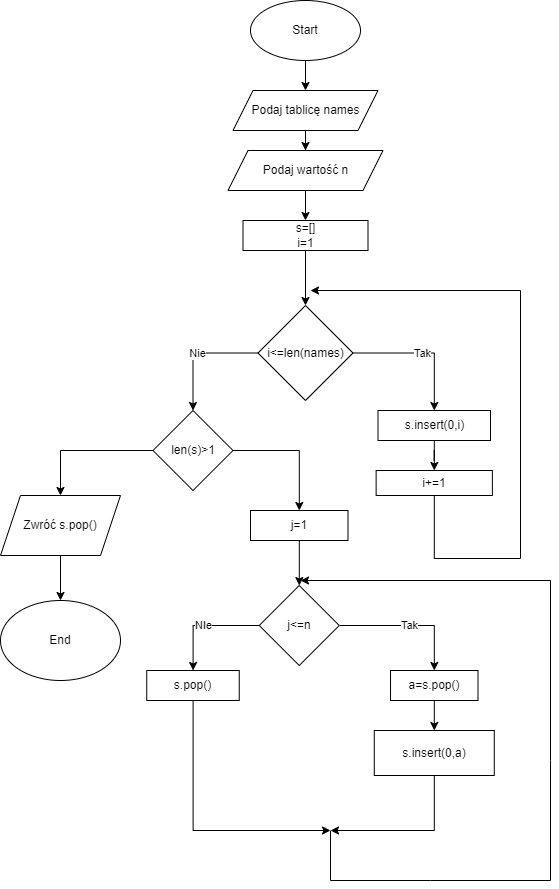
Rysunek . Schemat blokowy algorytmu czytającego wyrażenie arytmetyczne w notacji postfiksowej oraz obliczającego jego wartość.

**Zadanie 5.**

Zaimplementowanie struktury kolejki w Pythonie.

**Zadanie 6.**

Na rysunku 25 przestawiono schemat blokowy algorytmu gry w ,,gorącego ziemniaka”.



Rysunek . Schemat blokowy algorytmu gry w ,,gorącego ziemniaka”.

**Laboratorium 8**

**Zadanie 1.**

Napisanie funkcji wyszukującej na liście podaną wartość.

**Zadanie 2.**

Napisanie funkcji usuwającej i-ty węzeł z listy.

**Zadanie 3.**

Napisanie funkcji scalającej (merge) dwie uporządkowane listy w jedną uporządkowana listę.

**Zadanie 4.**

Napisanie funkcji zapisującej do listy liczby w taki sposób, aby w każdym momencie działania programu lista była posortowana.

**Zadanie 5.**

Napisanie implementacji algorytmu wyszukiwania binarnego.

**Laboratorium 9**

**Zadanie 1.**

Wykonanie analizy działania algorytmu Dijkstry. Rozwiązanie zadania znajduje się w pliku .xls dołączonym do niniejszego sprawozdania.

**Zadanie 2.**

Wykonanie analizy działania algorytmu Dijkstry. Rozwiązanie zadania znajduje się w pliku .xls dołączonym do niniejszego sprawozdania.

**Zadanie 3.**

Implementacja programu, który pozwoli użytkownikowi na zdefiniowanie grafu według założeń:  
- program po uruchomieniu pyta użytkownika jaki graf chce zbudować (skierowany, nieskierowany, ważony, inny możliwy),

- użytkownik może podać ilość wierzchołków oraz połączeń pomiędzy nimi,

- z otrzymanych informacji program wyświetla macierz sąsiedztwa oraz listę sąsiedztwa oraz wyświetla interpretację graficzną grafu.

**Zadanie 4.**

Implementacja algorytmu wyznaczania najkrótszej drogi z wykorzystaniem algorytmu Dijkstry.

1. W repozytorium znajdują się wszystkie implementacje projektowych algorytmów w ramach zajęć z AISD, które będą dostępne publicznie do 31.12.2023r. [↑](#footnote-ref-1)