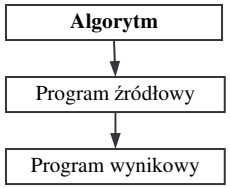
Wprowadzenie do algorytmów.

1. Podstawowe określenia.

Algorytmika – dział informatyki, zajmujący się różnymi aspektami tworzenia i analizowania algorytmów.



Rys. A. Każdy program działa według określonego algorytmu.

Algorytm – dokładny („krok po kroku") opis rozwiązania postawionego problemu lub sposobu osiągnięcia jakiegoś celu.

Specyfikacja problemu (specyfikacja algorytmu) – opisanie problemu przez podanie **danych**, z których korzysta algorytm oraz określenie **wyniku**, który ma być efektem działania algorytmu. W specyfikacji podajemy również:
- <u>warunki</u>, jakie powinny spełniać dane i wyniki,
- ewentualne zależności między danymi i wynikami.

2. Metody opisu algorytmów.

a) Opis słowny

Kolejne kroki algorytmu przedstawiamy za pomocą słów języka naturalnego.

b) Pseudo-kod

W opisie kolejnych kroków algorytmu łączymy język naturalny (opis słowny) ze znanymi nam elementami składni wybranego języka programowania.

Przykład 1.
Algorytm obliczania pierwiastków równania kwadratowego.

<u>Krok 1</u>: Na podstawie współczynników równania kwadratowego oblicz wartość delty:
`Δ:=b*b-4*a*c`.

<u>Krok 2</u>: Jeśli `Δ>0` to oblicz dwa pierwiastki:
`x1:=(-b-pierwiastek(Δ))/(2*a)`, `x2:=(-b+pierwiastek(Δ))/(2*a)`;
wypisz ich wartość na ekran i zakończ działanie algorytmu.

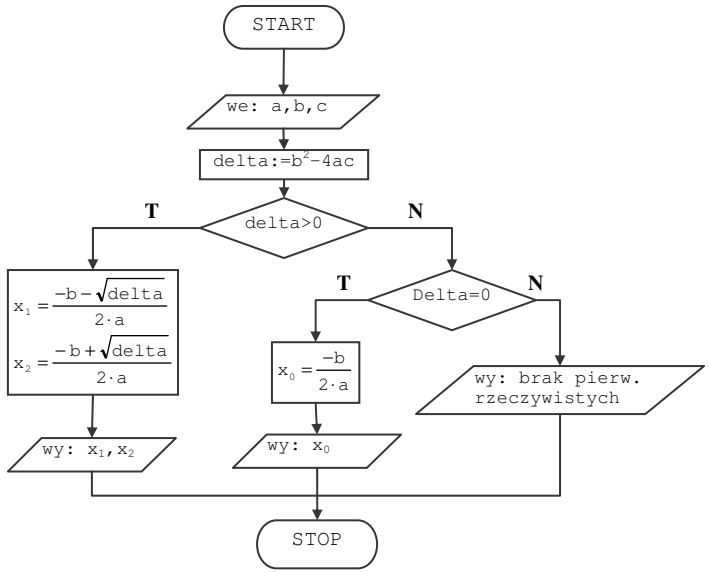
<u>Krok 3</u>: W przeciwnym wypadku, jeśli `Δ=0` to oblicz jeden pierwiastek:
`x0:=-b/(2*a)`; wypisz jego wartość na ekran i zakończ działanie algorytmu.

<u>Krok 4</u>: W przeciwnym wypadku, (jeśli `Δ<0`) wypisz na ekran komunikat: „brak pierwiastków rzeczywistych" i zakończ działanie algorytmu.

c) Składnia języka programowania

3. Graficzna reprezentacja algorytmów.

a) Schemat blokowy

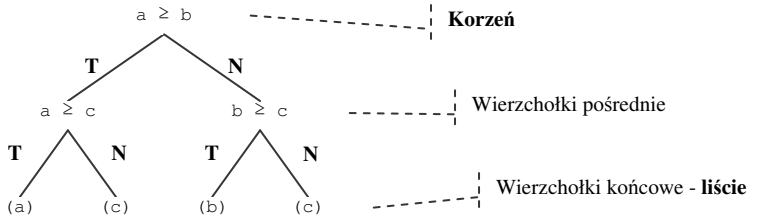


Rys. B. Schemat blokowy algorytmu obliczania pierwiastków równania kwadratowego.

Elementy schematu – bloki:

START STOP	początek i koniec algorytmu
	blok wprowadzania i wyprowadzania danych
	blok przeprowadzania obliczeń
	blok podejmowania decyzji

b) Drzewo

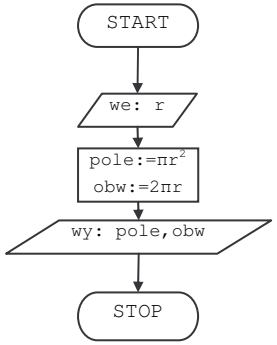


Rys. C. Drzewo algorytmu wyznaczania największej spośród trzech liczb.

5. Podział algorytmów.

a) Algorytm liniowy

Algorytm, w którym nie występują bloki decyzyjne.



Rys. D. Algorytm liniowy – brak elementów decyzyjnych.

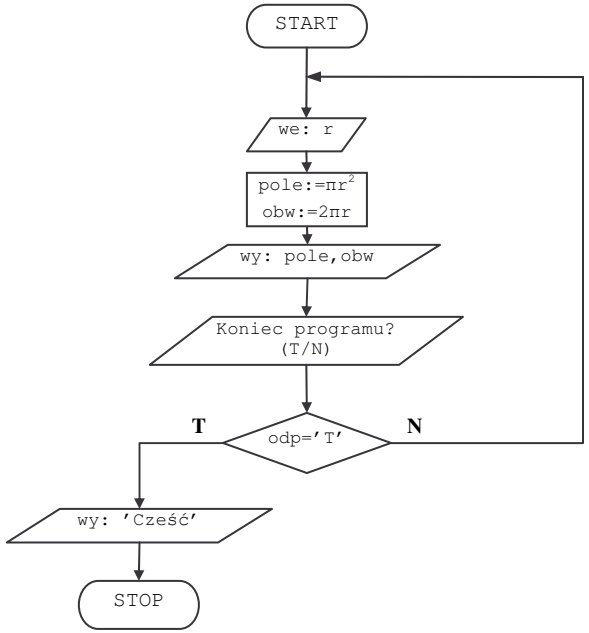
b) Algorytm nieliniowy (warunkowy)

Algorytm, w którym występuje, co najmniej jeden blok decyzyjny, tzw. **algorytm z rozgałęzieniami** (przykłady na rys. B i rys. C).

Algorytmy iteracyjne.

Iteracja – wielokrotne powtarzanie tej samej czynności (operacji).

1. Wielokrotne wykonywanie całego programu.



Rys. E. Algorytm iteracyjny – powtarzanie całego programu po decyzji użytkownika.

2. Schemat Hornera - obliczanie wartości wielomianu.

a) Operacje dodawania i mnożenia w wielomianach.

	„+”	„*”
$w(x) = \mathbf{ax^2 + bx + c}$	2	**3**
$w(x) = \mathbf{ax^3 + bx^2 + cx + d}$	3	**5**
$w(x) = \mathbf{ax^4 + bx^3 + cx^2 + dx + e}$	4	**7**

Po odpowiednich przekształceniach zapisu wielomianu ilość operacji mnożenia zmniejsza się.

	„+”	„*”
$w(x) = ax^2 + bx + c = \mathbf{(ax + b)x + c}$	2	**2**
$w(x) = ax^3 + bx^2 + cx + d = (ax^2 + bx + c)x + d = \mathbf{((ax +b)x + c)x + d}$	3	**3**
$w(x) = ax^4 + bx^3 + cx^2 + dx + e = (ax^3 + bx^2 + cx +d)x + e = ((ax^2 + bx + c)x + d)x + e =$ $= \mathbf{(((ax +b)x + c)x + d)x + e}$	4	**4**

b) Opracowanie algorytmu obliczania wartości wielomianu.

Dla wielomianu 3 stopnia:

$\mathbf{w_3(x) = a_0x^3 + a_1x^2 + a_2x + a_3 = (a_0x^2 + a_1x + a_2)x + a_3 = ((a_0x + a_1)x + a_2)x + a_3}$

wprowadzamy pomocniczą zmienną **y** i wykonujemy kolejne **operacje przypisania** (**:=**):

```
y := a0
y := yx + a1
y := yx + a2          } y := yx + ai      dla i=1..3
y := yx + a3
```

Dla wielomianu n-tego stopnia:

$\mathbf{w_n(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n}$

```
y := a0
y := yx + a1
y := yx + a2
...                   } y := yx + ai      dla i=1..n
y := yx + an-1
y := yx + an
```

Specyfikacja algorytmu:

Dane:

n – stopień wielomianu (nieujemna liczba całkowita),

$\mathbf{a_0}$, …, $\mathbf{a_n}$ – współczynniki wielomianu (ilość współczynników: n-1),

x – wartość argumentu.

Wyniki:

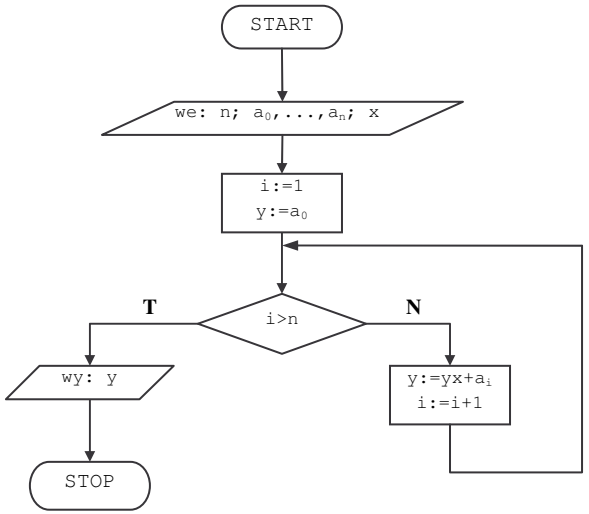
Wartość wielomianu **w(x)** **n**-tego stopnia dla danej wartości argumentu **x**.

Słowny opis algorytmu:

Krok 1: Przyjmij współczynnik a_0 wielomianu (stojący przy argumencie x o najwyższej potędze) za wartość początkową i przypisz do pomocniczej zmiennej y.
/ $y:=a_0$ /

Krok 2: Oblicz n razy wartość wyrażenia $y:=yx+a_i$ dla i=1, 2, …, n.

Schemat blokowy algorytmu:



Rys. F. Schemat blokowy algorytmu obliczania wartości wielomianu (schemat Hornera).

3. Wyszukiwanie największego (najmniejszego) elementu w zbiorze.

a) Wstępne informacje.

Odszukanie określonego elementu w zbiorze wymaga przejrzenia wszystkich jego elementów. W związku z tym musimy określić ilość wszystkich elementów, tzw. **moc** zbioru. Wykonujemy to zwykle dwoma metodami:
- poprzedzamy elementy zbioru liczbą, która określa ilość wszystkich elementów, lub
- przyjmujemy określony element za „wartownika", który nie należy do zbioru i wskazuje jego koniec.

Dodatkowo zakładamy też, że zbiór nie może być nieskończony oraz że elementy zbioru nie są uporządkowane.

b) Algorytm wyszukiwania największego elementu.

Specyfikacja algorytmu:

Dane:

n – liczba naturalna określająca ilość liczb w zbiorze,

$\mathbf{x_1}$, $\mathbf{x_2}$, …, $\mathbf{x_n}$ – ciąg liczb zbioru.

Wynik:

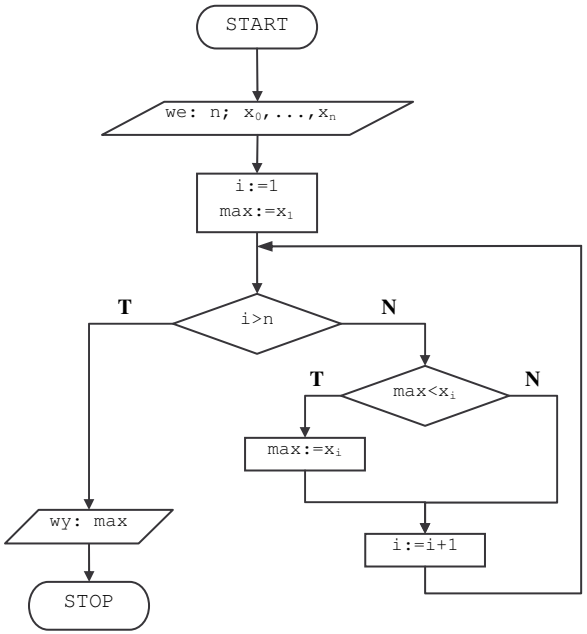
max – największa spośród liczb: x_1, x_2, …, x_n.

Słowny opis algorytmu:

Krok 1: Przyjmij pierwszy element zbioru (x_1) jako max [$max:=x_1$].

Krok 2: Dla kolejnych elementów x_i (i = 1, 2, …, n) sprawdzaj czy max jest mniejsze od x_i [$max<x_1$] i jeśli tak to przyjmij za max – x_i [$max:=x_1$].

Schemat blokowy algorytmu:



Rys. G. Schemat blokowy algorytmu wyszukiwania największego elementu.

c) Złożoność algorytmów wyszukiwania największego lub najmniejszego elementu.

Zgodnie z widocznym na rys. F schematem algorytmu odszukanie największego bądź najmniejszego elementu w zbiorze wymaga wykonania **n-1** porównań, gdzie **n** jest ilością wszystkich elementów. Ponieważ nie jest możliwe wykonanie mniejszej liczby porównań (dla poprawnego działania algorytmu), dlatego możemy powiedzieć, że powyższy algorytm jest **optymalny** pod względem **złożoności obliczeniowej**. Optymalny, a więc najszybszy.

ĆWICZENIA:

1. Co należy zmienić w algorytmie umieszczonym na rys. G, aby odszukać element najmniejszy?
2. Uzupełnij schemat algorytmu z rys. G o możliwość odczytania pozycji, na której znajdował się poszukiwany element.
3. Narysuj schemat blokowy algorytmu jednoczesnego wyszukiwania elementu największego i najmniejszego.

Algorytmy wyszukiwania danych.

1. Wyszukiwanie w zbiorze nieuporządkowanym (liniowe).

Specyfikacja algorytmu:

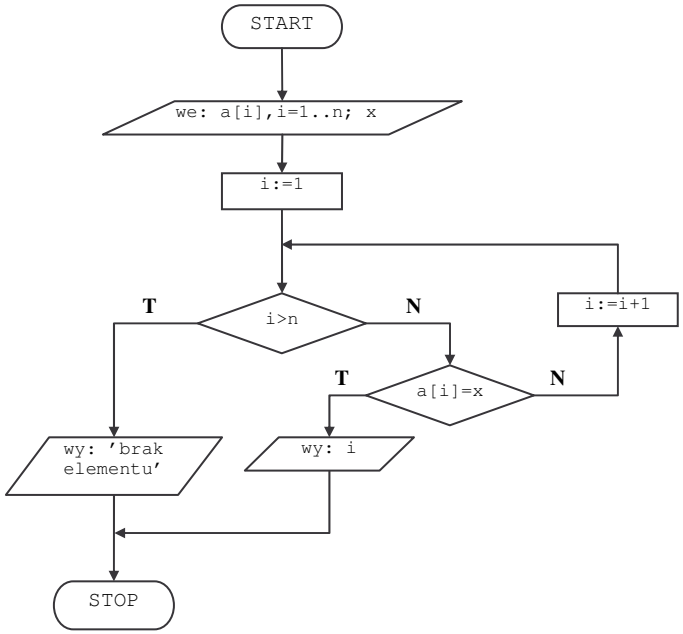
Dane:

a[i] (i=1..n) – nieuporządkowana n-elementowa, jednowymiarowa tablica (**a**),
x – poszukiwany element w tablicy **a**.

Wynik:

Położenie (indeks – **i**) elementu **x** w tablicy **a** lub informacja o braku elementu.

Schemat blokowy algorytmu:



Rys. H. Schemat blokowy algorytmu wyszukiwania „liniowego".

ĆWICZENIA:

1. Uzupełnij schemat algorytmu z rys. H o możliwość wypisania wszystkich pozycji, na których znajdował się poszukiwany element.

2. Wyszukiwanie w zbiorze uporządkowanym (binarne).

Specyfikacja algorytmu:

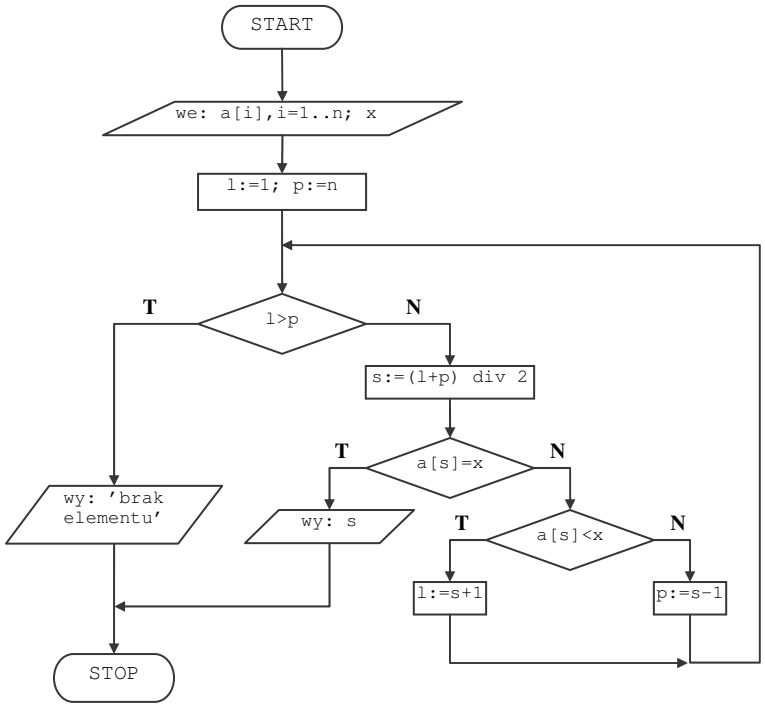
Dane:

a[i] (i=1..n) – uporządkowana n-elementowa, jednowymiarowa tablica (**a**),
x – poszukiwany element w tablicy **a**.

Wynik:
Położenie (indeks – **i**) elementu **x** w tablicy **a** lub informacja o braku elementu.

Pomocnicze zmienne:
l, p – tymczasowy lewy i prawy koniec zakresu,
s – tymczasowy środek zakresu.

Schemat blokowy algorytmu:



Rys. I. Schemat blokowy algorytmu wyszukiwania „binarnego”.

Tworzenie algorytmów – przykłady.

1. Suma i średnia arytmetyczna.

Przykładowy algorytm korzysta z elementu zwanego **wartownikiem**. Wartość tego elementu jest znana użytkownikowi. Jej wprowadzenie kończy działanie algorytmu.

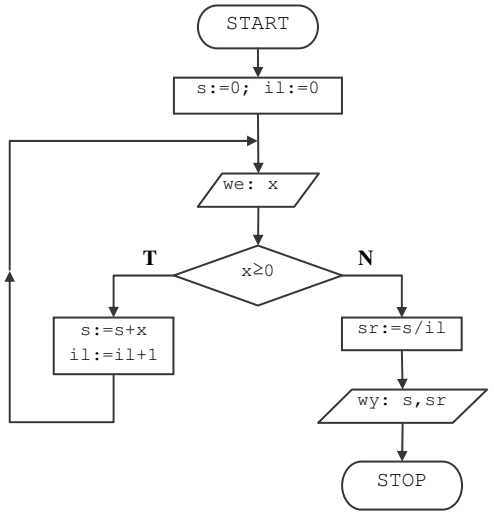
Specyfikacja algorytmu:

Dane:
x – kolejno wprowadzane liczby (wartownikiem jest dowolna całkowita liczba ujemna).

Wynik:
Suma **s** i średnia arytmetyczna **sr** wprowadzonych liczb.

Pomocnicze zmienne:
il – ilość wprowadzonych liczb.

Schemat blokowy algorytmu:



Rys. J. Schemat blokowy algorytmu obliczania sumy i średniej arytmetycznej liczb.

2. Silnia.

Zapis matematyczny:

$$n! = \begin{cases} 1 & \text{dla } n = 0 \quad (0! = 1) \\ 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n & \text{dla } n \geq 1 \end{cases}$$

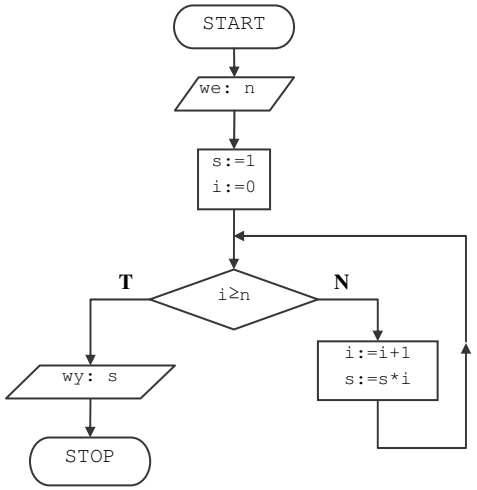
Specyfikacja algorytmu:

Dane:
n – liczba naturalna, z której obliczamy silnię.

Wynik:
s – wartość silni (n!).

Pomocnicze zmienne:
i – przyjmując kolejne wartości (od 0 do n) zapewnia iterację w algorytmie.

Schemat blokowy algorytmu:



Rys. K. Schemat blokowy algorytmu obliczania silni (n!).

3. Algorytm Euklidesa.

Algorytm ten odnajduje największy wspólny dzielnik – **NWD(m,n)** – dla dwóch liczb naturalnych **m** i **n**.

Np.: NWD(160,96)

```
160 = 1·96 + 64
 96 = 1·64 + 32
 64 = 2·32 + 0
```

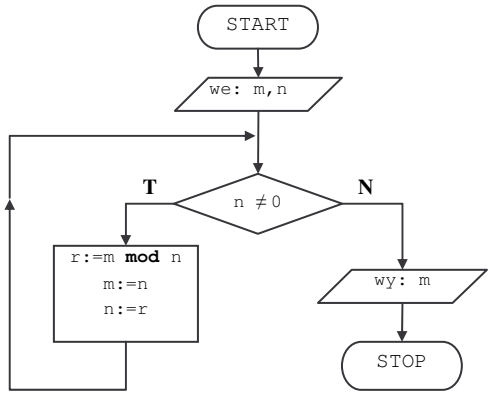
NWD(**m**,**n**) **m** ≥ **n**

m = q·n + r **q** – iloraz, czyli „ile razy" mieści się **n** w **m**
r – reszta (0 ≤ **r** < **n**)

Jeśli dana liczba jest wspólnym dzielnikiem **m** i **n** to jest również dzielnikiem **r**.
Możemy więc zastąpić szukanie wspólnego dzielnika liczb **m** i **n**, szukaniem dzielnika liczb **n** i **r**. Operację tę powtarzamy do uzyskania reszty równej zero. Ostatnia niezerowa reszta z dzielenia jest szukanym największym wspólnym dzielnikiem liczb **m** i **n**.

Specyfikacja algorytmu:
Dane:
n, **m** – liczby naturalne (**m** ≥ **n**);
Wynik:
NWD(**m**,**n**) – największy wspólny dzielnik **m** i **n**.

Schemat blokowy algorytmu:



Rys. L. Schemat blokowy algorytmu obliczania największego wspólnego dzielnika – NWD(m,n).

ZADANIA:

1. Odszukaj lub przypomnij sobie pojęcia: **liczba pierwsza**, **liczba złożona**, **sito Eratostenesa**. Spróbuj narysować schemat blokowy algorytmu znajdującego liczby pierwsze przy wykorzystaniu sita Eratostenesa.
2. W jaki sposób „powstają" kolejne **liczby Fibonacciego**? Spróbuj przedstawić algorytm powstawania tych liczb w postaci schematu blokowego.

Algorytmy porządkowania danych.

W kolejnych trzech sposobach porządkowania danych specyfikacja algorytmu przedstawia się następująco:

Specyfikacja algorytmu:

Dane:

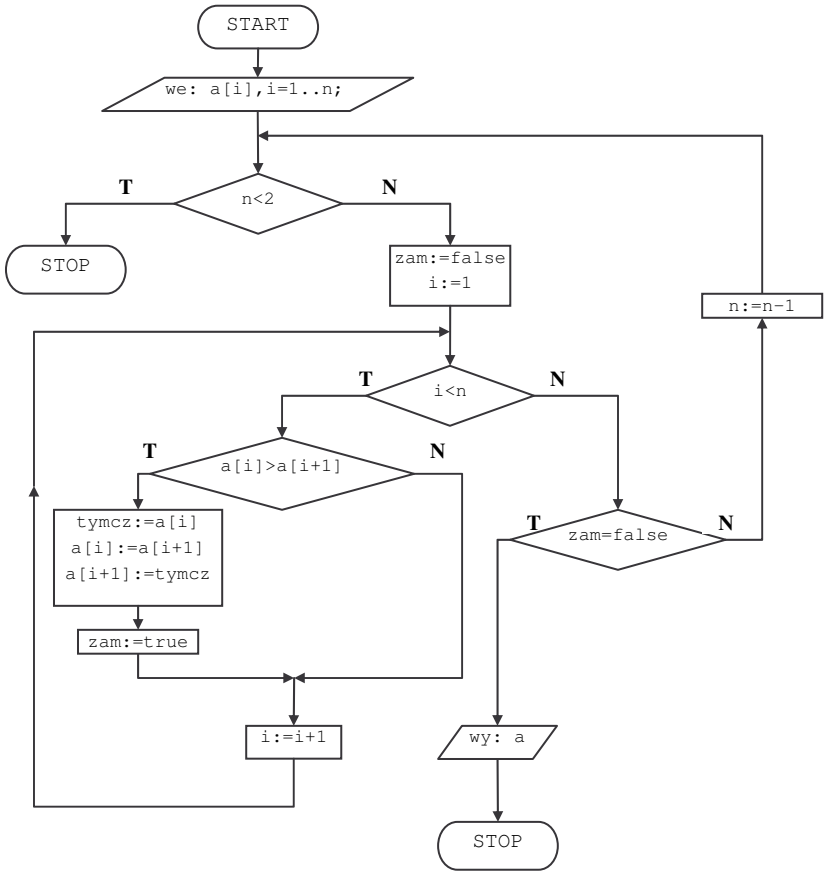
a[i] (i=1..n) – nieuporządkowana n-elementowa, jednowymiarowa tablica (**a**),

Wynik:

a[i] – uporządkowana tablica (**a**),

1. Metoda bąbelkowa (*ang. bubble sort*).

Schemat blokowy algorytmu:



Rys. M. Schemat blokowy algorytmu sortowania przez wybór (ang. selection sort).

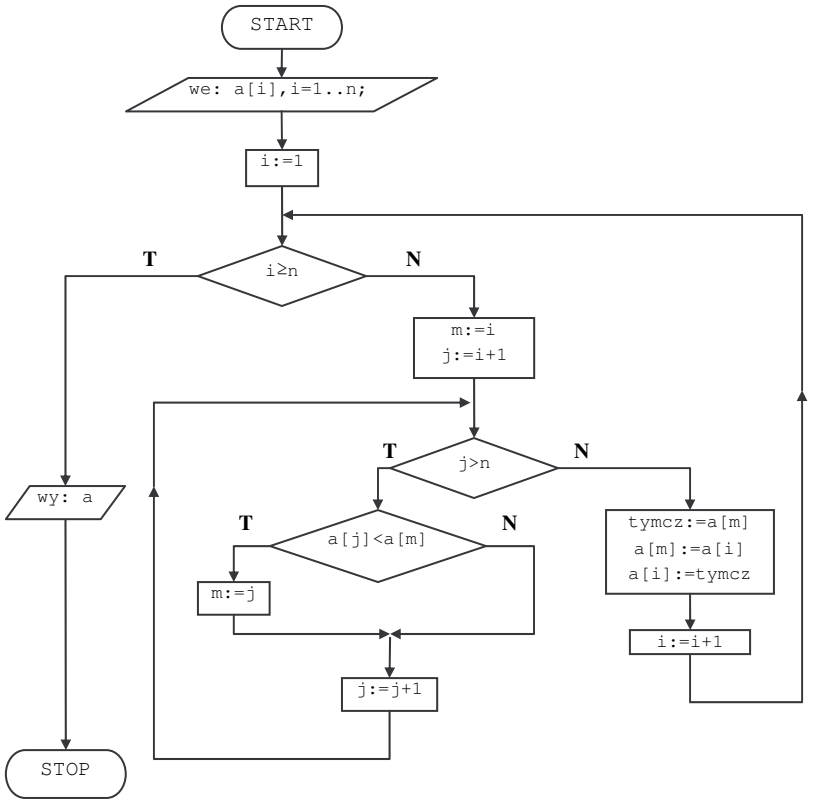
Pomocnicze zmienne:

zam – zmienna „logiczna” przechowująca informację o zamianie elementów (true/false).

tymcz – tymczasowo przechowywana wartość wybranego elementu.

3. Sortowanie przez wybór (*ang. selection sort*).

Schemat blokowy algorytmu:



Rys. O. Schemat blokowy algorytmu sortowania przez wybór (ang. selection sort).

Pomocnicze zmienne:

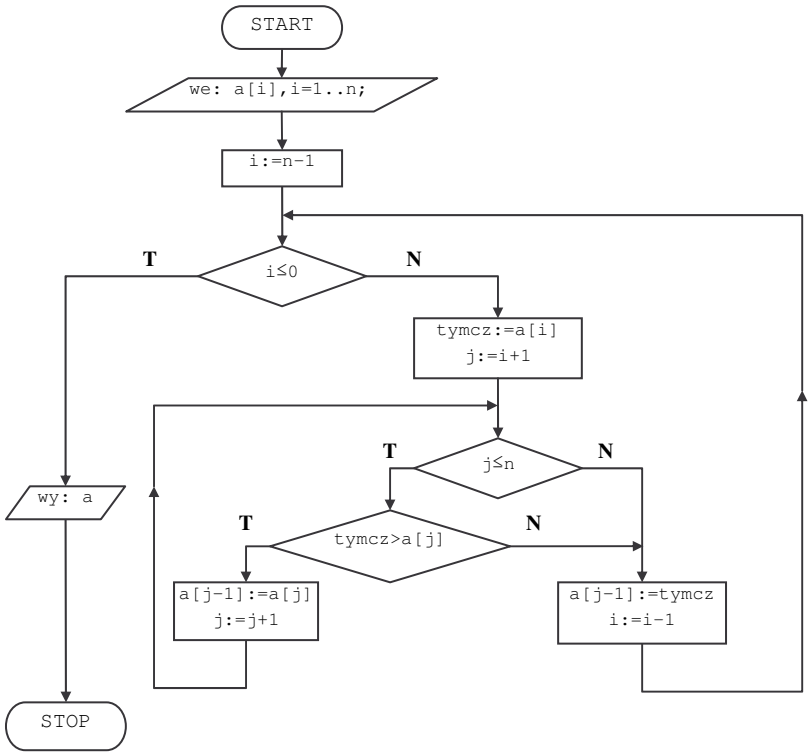
m – indeks elementu najmniejszego.

j – indeks pętli wewnętrznej (przy wyszukiwania najmniejszego elementu)

tymcz – tymczasowo przechowywana wartość wybranego elementu.

2. Sortowanie przez wstawianie (*ang. insertion sort*).

Schemat blokowy algorytmu:



Rys. N. Schemat blokowy algorytmu sortowania przez wstawianie (ang. insertion sort).

Pomocnicze zmienne:
`j` – indeks elementów uporządkowanych
`tymcz` – tymczasowo przechowywana wartość wybranego elementu.

ZADANIA:

1. Przedstawione powyżej dość „popularne” metody sortowania nie należą do najszybszych. Poszukaj informacji o innych sposobach porządkowania danych, np.: **sortowanie przez scalanie**, **sortowanie szybkie**.
2. Na czym polega metoda porządkowania **kubełkowego** (**koszykowego**) i **pozycyjnego**. Jakiego rodzaju dane najczęściej sortujemy tymi metodami?