Kodowanie danych i algorytmy - wykład w ramach przedmiotu „Informatyka I” (EE-DI)

Dawid Warchoł

Informatyka

Czym <u>nie</u> jest informatyka?

- *Informatyka nie jest nauką o komputerach*
- *(chociaż komputer jest najważniejszym narzędziem pracy informatyka).*

Informatyka

Czym jest informatyka?

- *Informatyka jest nauką o (automatycznym) przetwarzaniu informacji.*

Czym więc jest informacja?

- *Informacja to uporządkowane dane.*

Czym więc są dane?

- *Dane są zbiorami liczb i tekstów (liter, cyfr, symboli).*

Pozycyjne systemy liczbowe

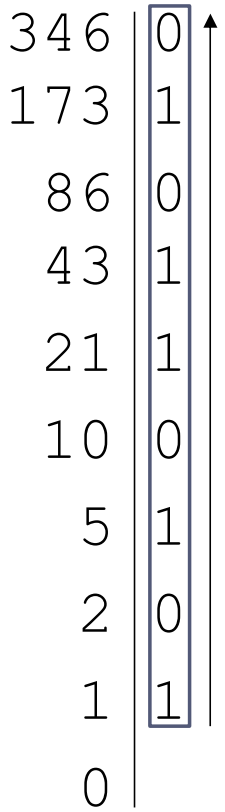
- System dziesiątkowy
  - Składa się z dziesięciu cyfr: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
  - Podstawowy system, używany na co dzień.
- System dwójkowy (binarny)
  - Składa się z dwóch cyfr (bitów): 0, 1.
  - Używany do zapisu liczb w komputerze.
- System szesnastkowy (heksadecymalny)
  - Składa się z szesnastu cyfr: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
  - Używany po to, aby przedstawić liczby dwójkowe w sposób bardziej przyjazny dla człowieka.
  - Liczby szesnastkowe są czytelniejsze od dwójkowych i łatwo z nich przejść do systemu dwójkowego.

Pozycyjne systemy liczbowe

- Inne systemy liczbowe
  - Możliwy jest zapis liczb w dowolnym systemie, np. piątkowym, dwudziesto-czwórkowym.
  - Nie mają one jednak szerszego zastosowania.
  - Wyjątkiem jest system ósemkowy, który czasem jest używany zamiast systemu szesnastkowego do czytelniejszego przedstawienia liczb binarnych.

Zapis liczb dziesiątkowych w systemie dwójkowym

Liczby z lewej strony dzielimy przez 2 i zapisujemy resztę po prawej stronie:



Cyfry z prawej strony kreski zapisujemy w kolejności **od dołu do góry**:

$346_{(10)} = 101011010_{(2)}$

Zapis liczb dwójkowych w systemie dziesiątkowym

$$\mathbf{101011010_{(2)}} =$$

$$= 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6 + 0 \cdot 2^7 + 1 \cdot 2^8 =$$

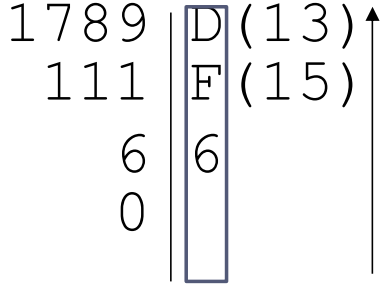
$$= 2^1 + 2^3 + 2^4 + 2^6 + 2^8 =$$

$$= 2 + 8 + 16 + 64 + 256 = \mathbf{346_{(10)}}$$

Należy pamiętać, że najmniej znaczące cyfry są zawsze z prawej strony, a więc im przypisujemy mniejsze potęgi.

Zapis liczb dziesiątkowych w systemie szesnastkowym

Liczby z lewej strony dzielimy przez 16 i zapisujemy resztę po prawej stronie:



Cyfry z prawej strony kreski zapisujemy w kolejności **od dołu do góry**:

$1789_{(10)} = 6FD_{(16)}$

Jak obliczyć resztę z dzielenia przez 16

- Przykład:

```
1789/16 = 111.8125
0.8125*16 = 13 (D)
```

13 jest resztą z dzielenia 1789 przez 16.
13 ma symbol D w systemie szesnastkowym.

Zapis liczb szesnastkowych w systemie dziesiątkowym

$$\boldsymbol{6FD_{(16)}} =$$

$$= 13 \cdot 16^0 + 15 \cdot 16^1 + 6 \cdot 16^2 =$$

$$= 13 + 240 + 1536 = \boldsymbol{1789_{(10)}}$$

Należy pamiętać, że najmniej znaczące cyfry są zawsze z prawej strony, a więc im przypisujemy mniejsze potęgi.

Zapis liczb dwójkowych w systemie szesnastkowym

Zamiana liczby 101011010(2) na liczbę szesnastkową

- Krok 1 – grupowanie bitów w czwórki:

  `1 0101 1010`

- Krok 2 – uzupełnienie zerami bitów z lewej strony (jeśli trzeba):

  `0001 0101 1010`

- Krok 3 – zamiana każdej czwórki na jednocyfrową liczbę szesnastkową:

  `0001 0101 1010`

  `1 5 A`

`101011010(2) = 15A(16)`

Zapis liczb szesnastkowych w systemie dwójkowym

Zamiana liczby $15A_{(16)}$ na liczbę dwójkową

- Krok 1 – zamiana każdej z cyfr na liczbę dwójkową:

```
1     5     A
1     0101  1010
```

- Krok 2 – złączenie wszystkich bitów:

```
101011010
```

$15A_{(16)} = 101011010_{(2)}$

Zapis liczb ze znakiem

- Liczby bez znaku są zawsze dodatnie.
- Liczby ze znakiem, to liczby, w których określa się, czy są dodatnie, czy ujemne.
- Do kodowania liczb całkowitych ze znakiem kiedyś używano prostych systemów kodowych: ZM (znak-moduł, ang. sign-magnitude) i U1 (uzupełnień do jedności, ang. one's complement).
- W nowoczesnych komputerach, poza nielicznymi wyjątkami, stosuje się zapis kodowy U2 (uzupełnień do dwóch, ang. two's complement).

Kodowanie ZM, U1, U2

- Systemy ZM, U1 i U2 różnią się kodowaniem liczb niedodatnich. Liczby dodatnie koduje się w nich tak samo.
- Kodując liczbę w systemach ZM, U1 lub U2 musimy z góry określić liczbę bitów, przy użyciu których będą zapisywane liczby.
- Jeśli zamieniona liczba ma mniej bitów niż ustalono, należy uzupełnić ją z lewej strony:
  - zerami, w przypadku liczby dodatniej,
  - jedynkami, w przypadku liczby ujemnej.
- Tabelki przedstawione na kolejnych slajdach zawierają wszystkie możliwe liczby, które możemy zapisać przy użyciu czterech bitów.

Kodowanie ZM (znak-moduł)

Kod ZM	Wartość	Przeliczenie
0000	0	0
0001	1	2^0
0010	2	2^1
0011	3	$2^1 + 2^0$
0100	4	2^2
0101	5	$2^2 + 2^0$
0110	6	$2^2 + 2^1$
0111	7	$2^2 + 2^1 + 2^0$
1000	0	$(-1)^1 \cdot 0$
1001	(-1)	$(-1)^1 \cdot (2^0)$
1010	(-2)	$(-1)^1 \cdot (2^1)$
1011	(-3)	$(-1)^1 \cdot (2^1 + 2^0)$
1100	(-4)	$(-1)^1 \cdot (2^2)$
1101	(-5)	$(-1)^1 \cdot (2^2 + 2^0)$
1110	(-6)	$(-1)^1 \cdot (2^2 + 2^1)$
1111	(-7)	$(-1)^1 \cdot (2^2 + 2^1 + 2^0)$

Kodowanie U1 (uzupełnień do jedności)

Kod U1	Wartość	Przeliczenie
0000	0	0
0001	1	2^0
0010	2	2^1
0011	3	$2^1 + 2^0$
0100	4	2^2
0101	5	$2^2 + 2^0$
0110	6	$2^2 + 2^1$
0111	7	$2^2 + 2^1 + 2^0$
1000	(-7)	$(-2^3 + 1)$
1001	(-6)	$(-2^3 + 1) + 2^0$
1010	(-5)	$(-2^3 + 1) + 2^1$
1011	(-4)	$(-2^3 + 1) + 2^1 + 2^0$
1100	(-3)	$(-2^3 + 1) + 2^2$
1101	(-2)	$(-2^3 + 1) + 2^2 + 2^0$
1110	(-1)	$(-2^3 + 1) + 2^2 + 2^1$
1111	0	$(-2^3 + 1) + 2^2 + 2^1 + 2^0$

Kodowanie U2 (uzupełnień do dwóch)

Kod U2	Wartość	Przeliczenie
0000	0	0
0001	1	2^0
0010	2	2^1
0011	3	$2^1 + 2^0$
0100	4	2^2
0101	5	$2^2 + 2^0$
0110	6	$2^2 + 2^1$
0111	7	$2^2 + 2^1 + 2^0$
1000	(-8)	(-2^3)
1001	(-7)	$(-2^3) + 2^0$
1010	(-6)	$(-2^3) + 2^1$
1011	(-5)	$(-2^3) + 2^1 + 2^0$
1100	(-4)	$(-2^3) + 2^2$
1101	(-3)	$(-2^3) + 2^2 + 2^0$
1110	(-2)	$(-2^3) + 2^2 + 2^1$
1111	(-1)	$(-2^3) + 2^2 + 2^1 + 2^0$

Kodowanie U2 – przykład 1

Zapisz liczbę -59 w 8-bitowym systemie kodowania U2.

- Krok 1 - zapisz moduł liczby -59 (czyli 59) w systemie dwójkowym:
  `111011`
- Krok 2 - dopisz zera z lewej strony, tak aby łącznie było osiem bitów:
  `00111011`
- Krok 3 - Wykonaj operację negacji na wszystkich bitach liczby (zamiana 0 na 1 i 1 na 0):
  `11000100`
- Krok 4 - Do liczby dodaj 1 (zgodnie z arytmetyką liczb binarnych):
  `11000101`

$-59_{(10)} = 11000101_{(U2)}$

Pomoc do kroku 4 – tabliczka dodawania liczb binarnych

0 + 0 =	0
0 + 1 =	1
1 + 0 =	1
1 + 1 =	10

Kodowanie U2 – przykład 2

Dana jest liczba 10011001 w 8-bitowym systemie kodowania U2. Zapisz ją w systemie dziesiątkowym.

$$\mathbf{10011001_{(U2)}} =$$

$$= 1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 + 0 \cdot 2^6 - 1 \cdot 2^7 =$$

$$= 2^0 + 2^3 + 2^4 - 2^7 =$$

$$= 1 + 8 + 16 - 128 = \mathbf{-103_{(10)}}$$

Kodowanie liczb zmiennoprzecinkowych

- Do zapisu liczb ułamkowych w pamięci komputera stosuje się tzw. liczby zmiennoprzecinkowe (zmiennopozycyjne).

- System kodowania liczb zmiennoprzecinkowych we współczesnych komputerach określa standard IEEE 754.

Kodowanie zgodne ze standardem IEEE 754

- Liczbę zmiennoprzecinkową w standardzie IEEE 754 zapisuje się w postaci:

$$-1^{Z} \cdot M \cdot 2^{W}$$

gdzie:

  - **Z – bit określający znak liczby**
  - **M – bity określające mantysę liczby** – liczba ułamkowa z przedziału [1-2)
  - **W – bity określające wykładnik liczby** (inaczej cecha, pozycja przecinka) – liczba całkowita (może być ujemna)

- Pierwszy bit liczby dotyczy znaku, następne bity określają wykładnik, ostatnie bity tworzą mantysę.
- W liczbach 32-bitowych (pojedynczej precyzji) mantysa ma 24 bity, a wykładnik 8 bitów.
- W liczbach 64-bitowych (podwójnej precyzji) mantysa ma 52 bity, a wykładnik 11 bitów.

Kodowanie zgodne ze standardem IEEE 754 - przykład

- Kodowanie przykładowych liczb w standardzie IEEE 754 najlepiej sprawdzić na stronie:

  https://calculla.pl/liczby_zmiennoprzecinkowe

- Dobre przykłady znajdują się również na stronie:

  https://eduinf.waw.pl/inf/alg/006_bin/0022.php

Kodowanie tekstów

- Teksty (inaczej napisy, łańcuchy znakowe, ang. strings) są kodowane w określonych standardach kodowania (np. ASCII, Unikod).
- Polega to na tym, że każdy znak tekstu (litery, cyfry, znaki interpunkcyjne, białe i inne) ma przypisany inny kod liczbowy.
- W języku Python obowiązuje system kodowania znaków Unikod (ang. Unicode), a najczęściej jego wariant UTF-8.
- Tablica znaków (i odpowiadających im kodów) systemu UTF-8 jest dostępna pod adresem: https://www.utf8-chartable.de

Czym jest algorytm?

- Algorytm jest instrukcją, ciągiem czynności opisującym wykonanie pewnego zadania.
- W informatyce algorytmy opisują zadania, które mogą być wykonywane przez program komputerowy.
- Należy w tym celu zapisać algorytm w ustalonym języku programowania tak, aby mógł go wykonać komputer. Nazywa się to implementacją algorytmu.
- Algorytmy najczęściej zapisujemy w formie:
  - pseudokodu (pseudojęzyka),
  - schematu blokowego.

Pseudokod (pseudojęzyk)

- Pseudokod jest formą zapisu algorytmu przypominającą język programowania.
- Taki język nie ma ściśle ustalonej składni.
- Powinien być mało szczegółowy (ogólny) i zrozumiały dla człowieka.

Przykład algorytmu w pseudokodzie

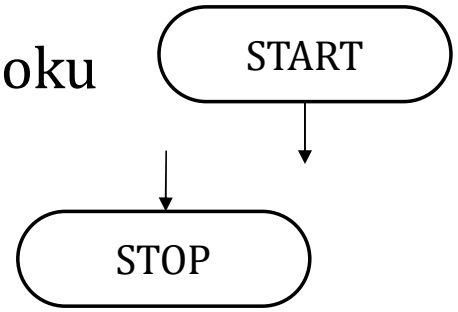
Algorytm sortowania bąbelkowego:

```
K01: Dla j = 1,2,...,n - 1:
          Wykonuj krok K02
K02:      Dla i = 1,2,...,n - 1:
               Jeśli d[i] > d[i + 1],
               to d[i] ↔ d[i + 1]
K03: Zakończ
```

Źródło: https://eduinf.waw.pl/inf/alg/003_sort/0004.php

Schematy blokowe

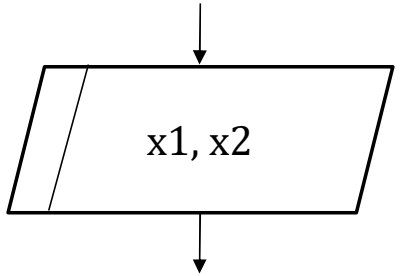
- Schemat blokowy jest formą zapisu algorytmu za pomocą bloków połączonych jednokierunkowymi strzałkami.
- Algorytm rozpoczyna się od bloku
- Algorytm kończy się na bloku



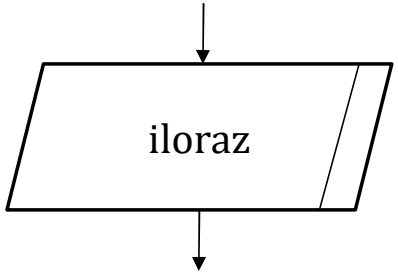
- Kolejne operacje wykonywane są zgodnie z kierunkiem strzałek.

Schematy blokowe

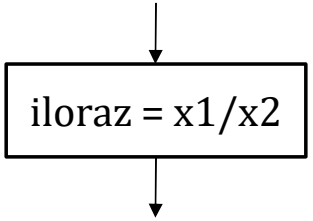
- Blok wejściowy (wczytywania, pobierania danych):



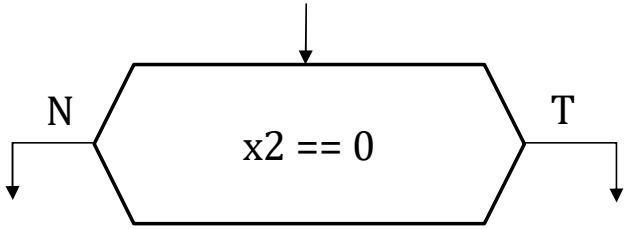
- Blok wyjściowy (wyświetlania, zapisywania, wysyłania danych):

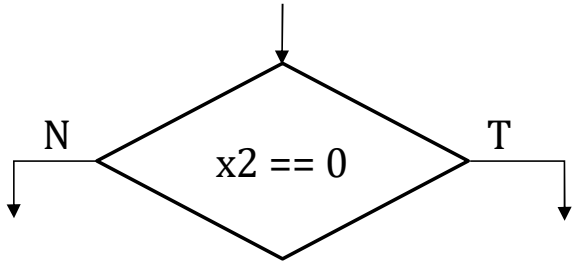


- Blok typu polecenie/instrukcja:



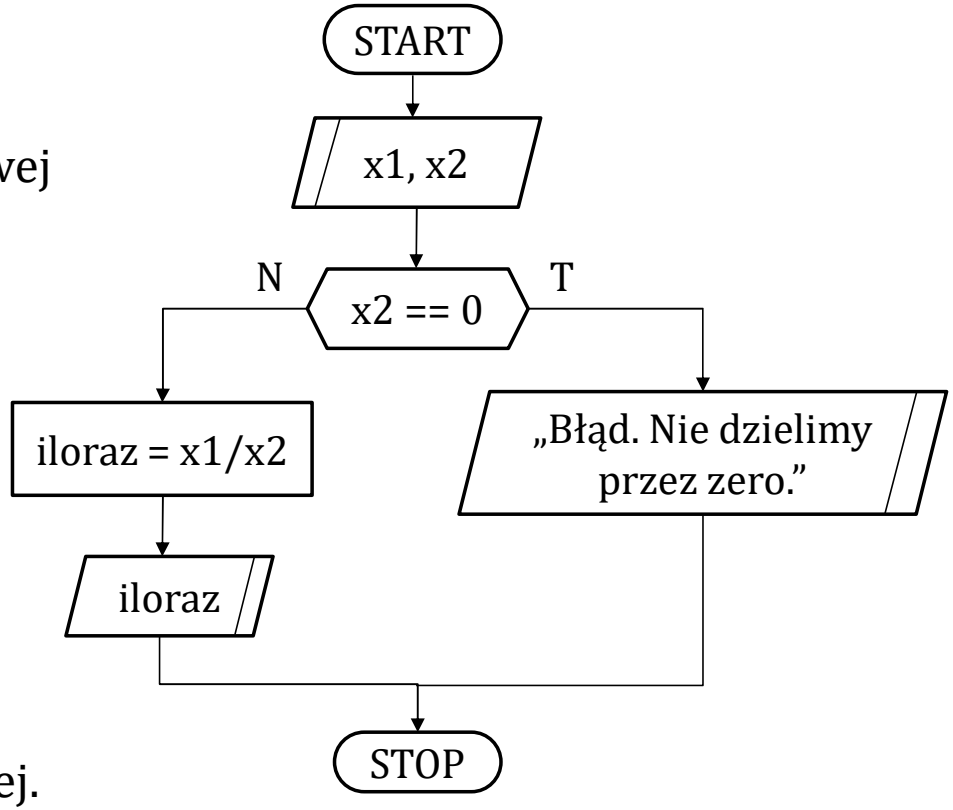
- Blok warunkowy/decyzyjny:



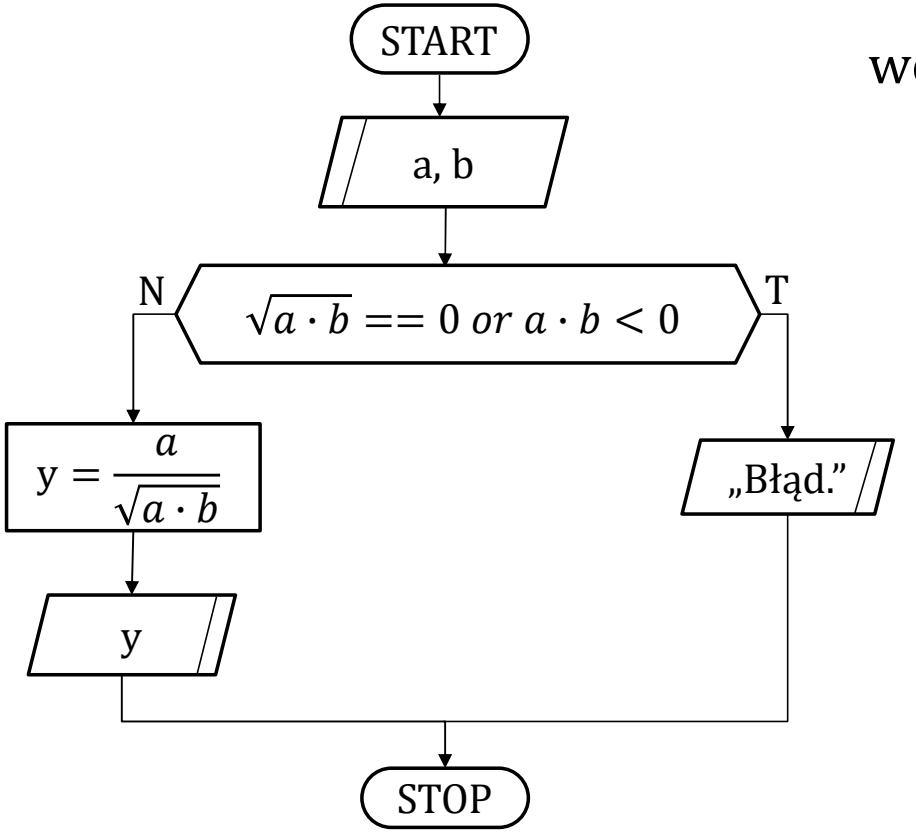


Schematy blokowe

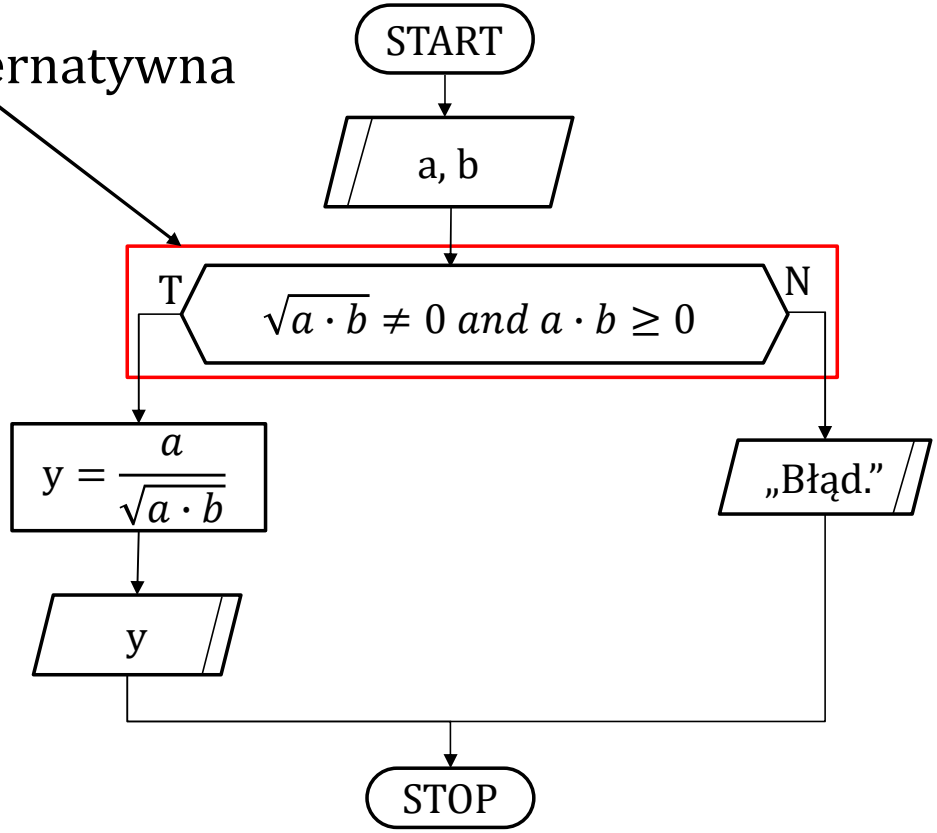
- **Jak zinterpretować ten algorytm?**
- Symbol = jest operatorem przypisującym wartość z prawej strony do zmiennej z lewej strony.
- Symbol == jest operatorem sprawdzającym, czy jedna wartość jest równa drugiej.
- Słowa i symbole bez cudzysłowu oznaczają nazwy zmiennych.
- Teksty w cudzysłowie oznaczają dane w formie tekstowej.



Zad. 1. Algorytm obliczający wartość wyrażenia $y = \frac{a}{\sqrt{a \cdot b}}$.



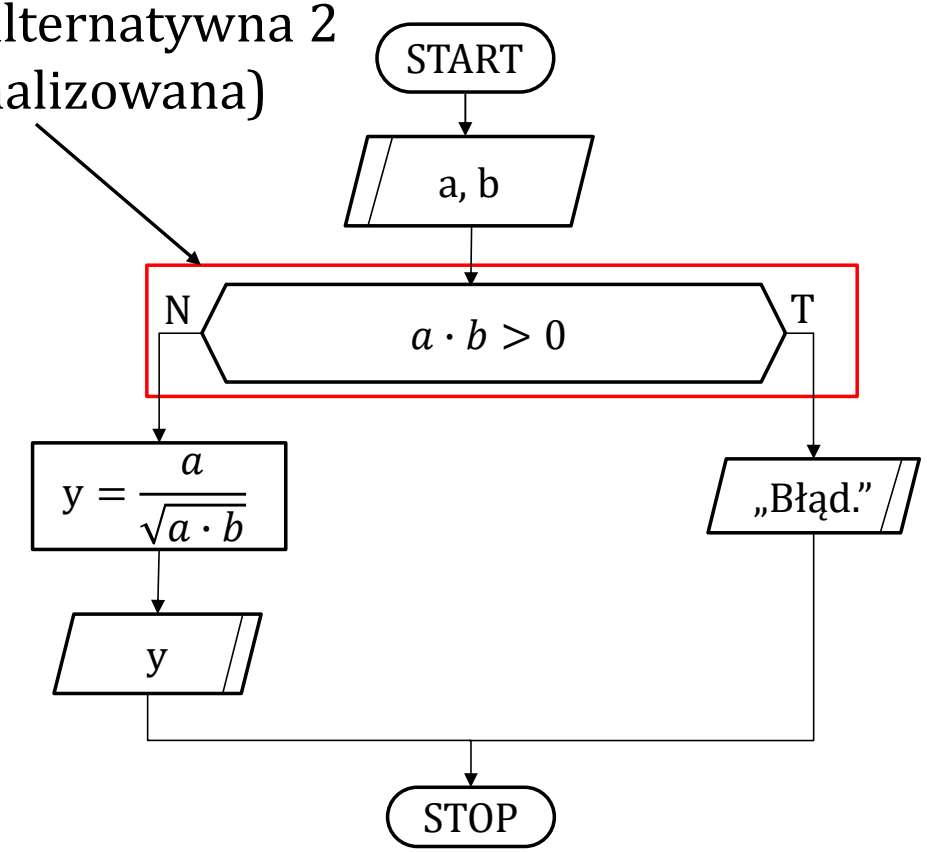
wersja alternatywna



Zad. 1. Algorytm obliczający wartość wyrażenia $y = \frac{a}{\sqrt{a \cdot b}}$.

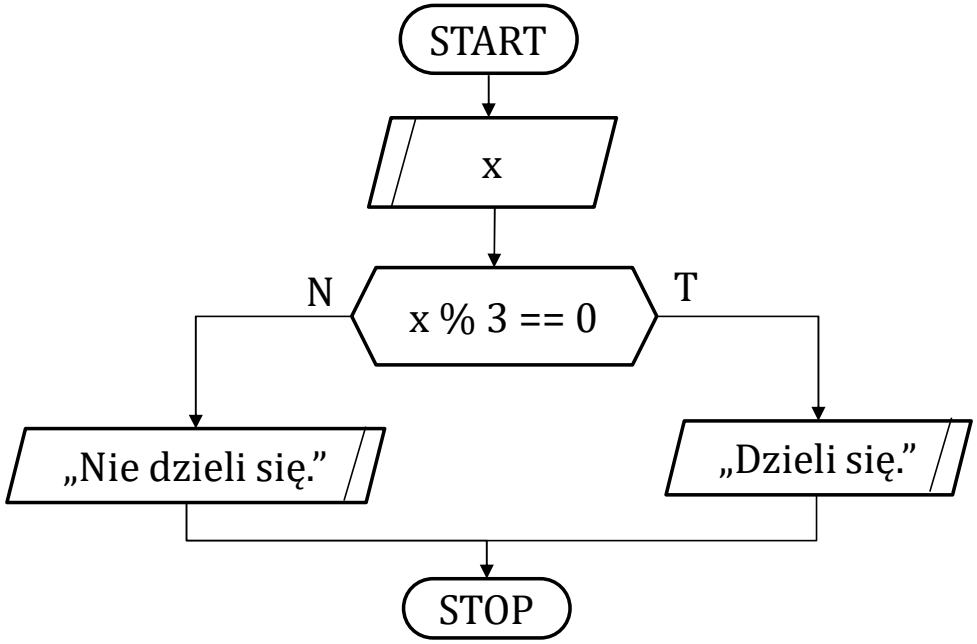
- Można zauważyć, że wyrażenie: $\sqrt{\boldsymbol{a \cdot b}} \neq \boldsymbol{0}\ \boldsymbol{and}\ \boldsymbol{a \cdot b} \geq \boldsymbol{0}$ jest równoważne wyrażeniu $\boldsymbol{a \cdot b > 0}$.
- Można więc zapisać schemat blokowy jeszcze prościej. Taki program miałby mniej operacji do wykonania.

wersja alternatywna 2 (zoptymalizowana)



Zad. 2. Algorytm sprawdzający czy dana liczba dzieli się przez 3.

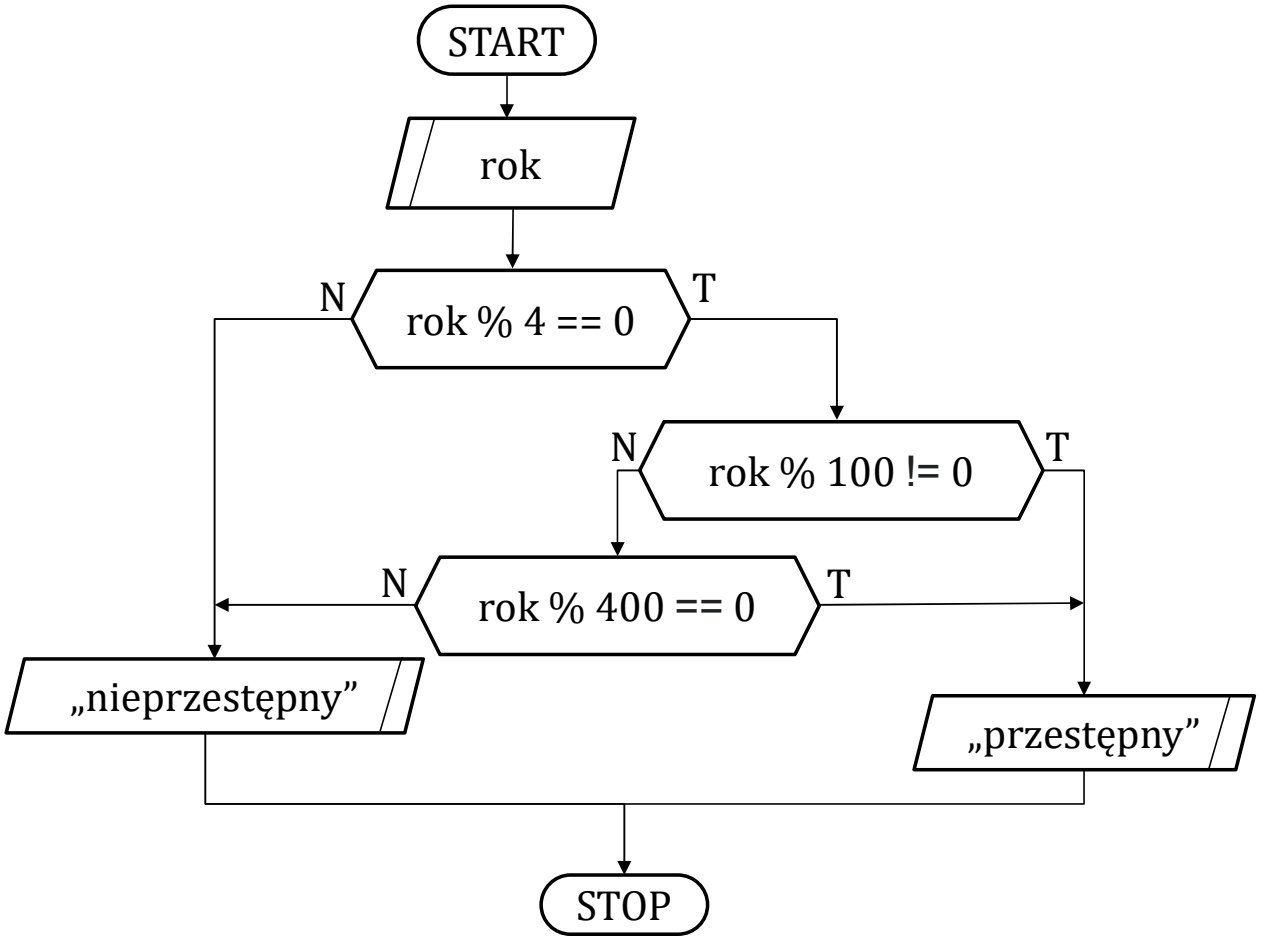
- Symbol % jest operatorem obliczającym resztę z dzielenia pierwszej liczby przez drugą.
- Przykład 1: wyrażenie 4 % 2 zwróci wynik 0.
- Przykład 2: wyrażenie 8 % 3 zwróci wynik 2.



- Czasami resztę z dzielenia na algorytmach zapisuje się za pomocą operatora słownego MOD, np. 4 MOD 2 oblicza resztę z dzielenia 4 przez 2. MOD jest skrótem słowa modulo.

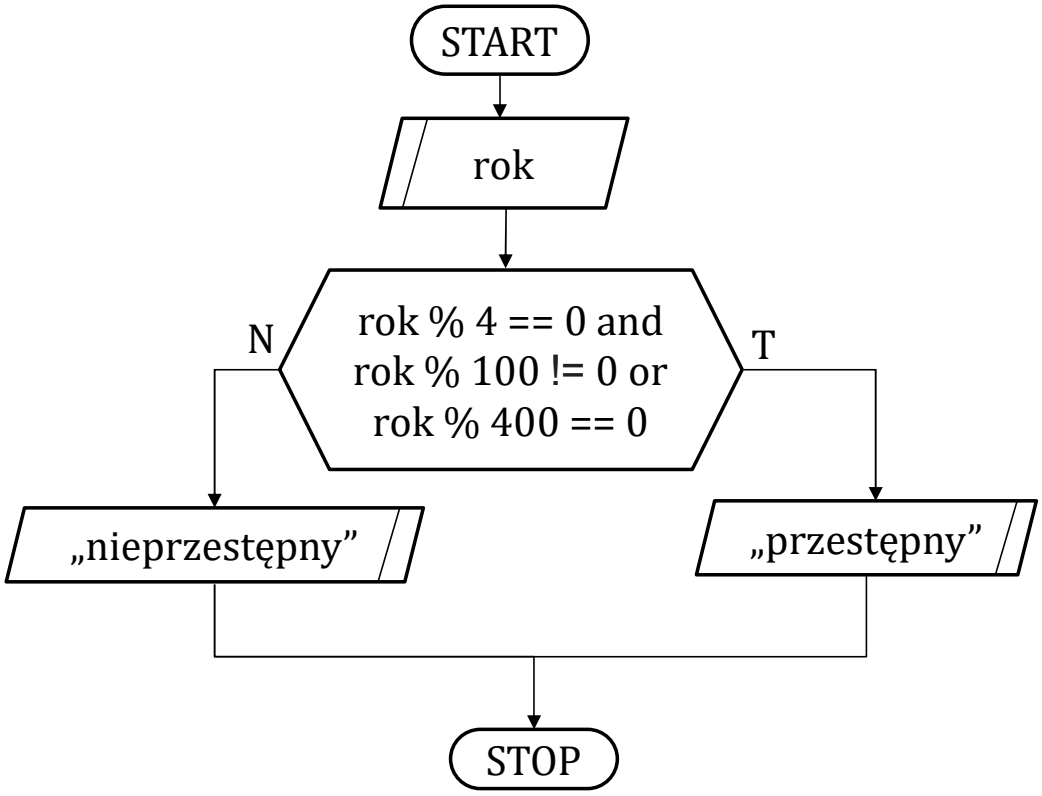
Zad. 3. Algorytm sprawdzający czy dana liczba oznacza rok przestępny.

- Kiedy rok jest uznawany za przestępny?
- Kiedy jest podzielny przez 4 oraz niepodzielny przez 100 lub podzielny przez 400.
- Symbol != jest operatorem sprawdzający, czy pierwsza liczba jest różna od drugiej. Odpowiada on matematycznemu symbolowi ≠.



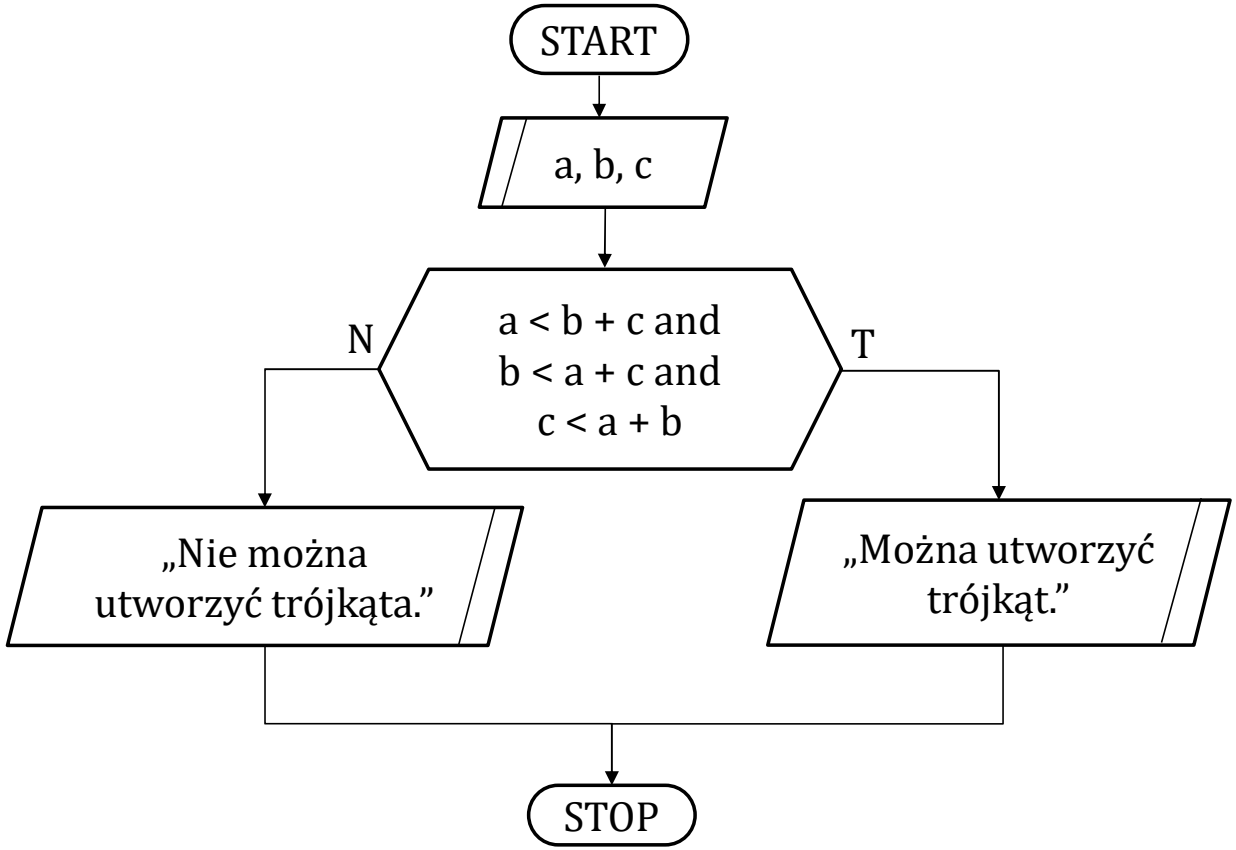
Zad. 3. Algorytm sprawdzający czy dana liczba oznacza rok przestępny – wersja alternatywna.

- Kiedy rok jest uznawany za przestępny?
- Kiedy jest podzielny przez 4 oraz niepodzielny przez 100 lub podzielny przez 400.



Zad. 4. Dane są liczby a, b, c. Opracuj algorytm sprawdzający czy mogą one być bokami trójkąta.

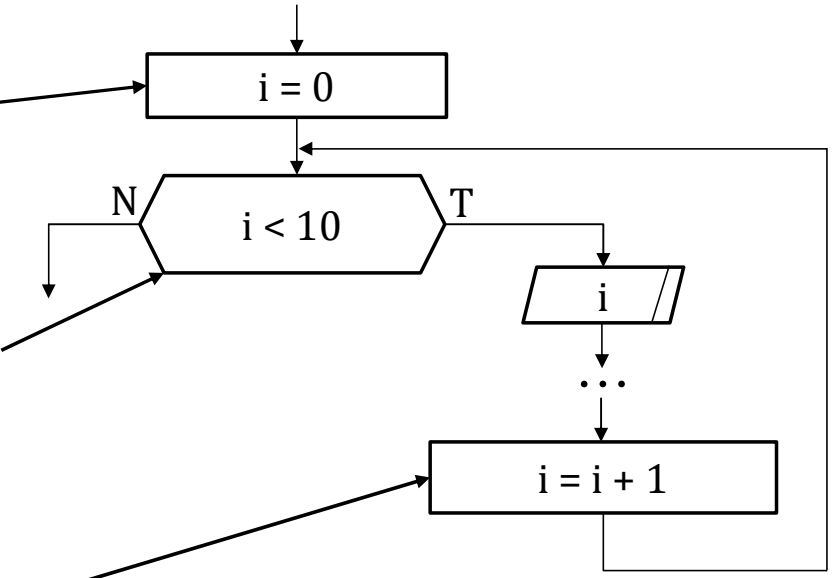
- Warunek budowy trójkąta - **każdy bok trójkąta musi być mniejszy od sumy dwóch pozostałych boków**.



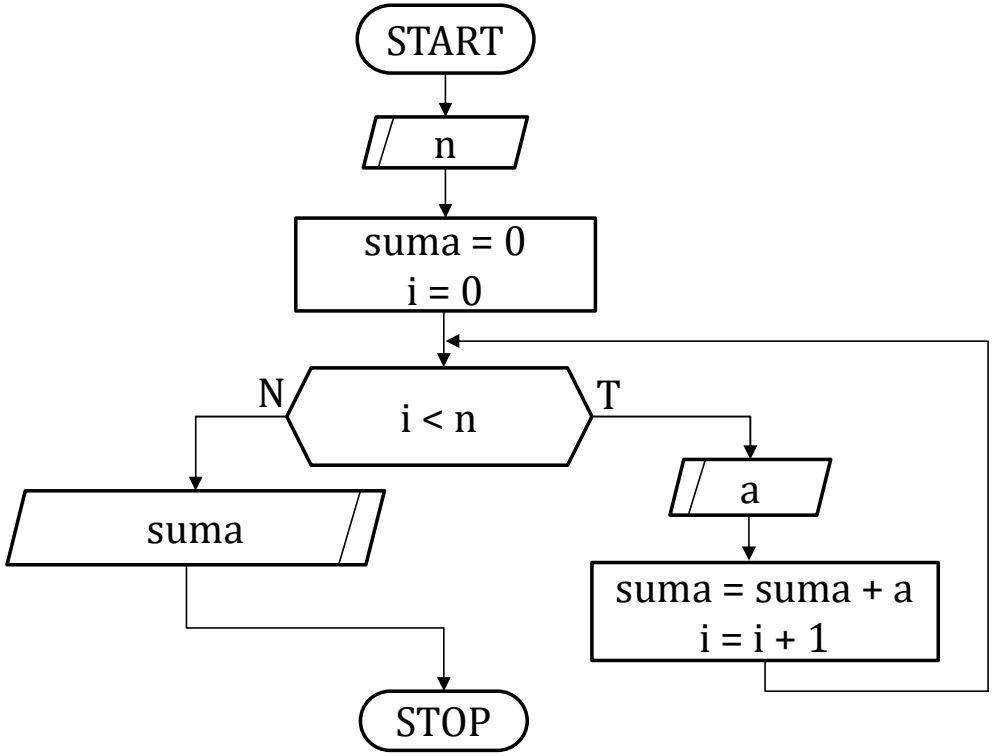
Pętle

- Pętle służą do wielokrotnego powtarzania operacji.
- Standardowo pętla powinna mieć licznik, który przechowuje wartość aktualnego obiegu (iteracji). Licznik najczęściej będziemy nazywać *i*.
- Pętla powinna mieć tzw. warunek kończący. Sprawdza się w nim, czy wykonywać kolejny obieg, czy zakończyć pętlę.
- Należy pamiętać aby w pętli (najlepiej na końcu) zamieścić operację inkrementacji licznika, czyli zwiększenia jego wartości o 1. W przeciwnym razie pętla nigdy się nie zakończy.

Ogólny schemat pętli



Zad. 5. Algorytm wczytujący *n*-elementowy ciąg liczb i obliczający sumę jego elementów.



Zad. 5. Algorytm wczytujący *n*-elementowy ciąg liczb i obliczający sumę jego elementów.

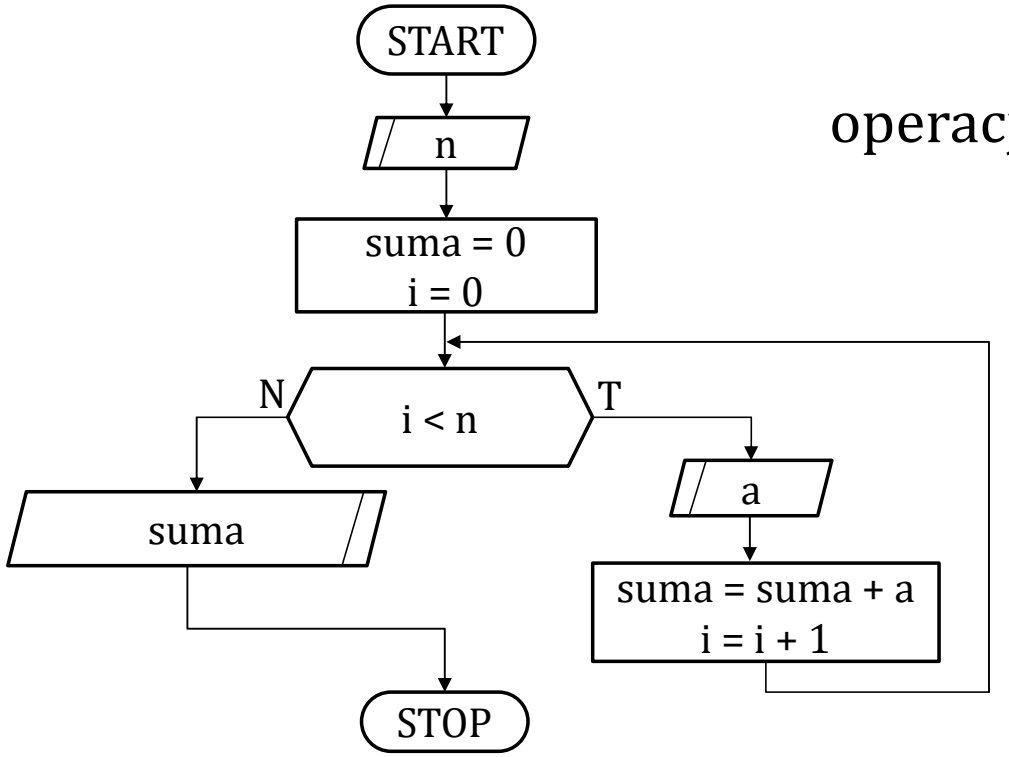


Tabela pamięci dla danych wejściowych: [4, 5, 3, 0, -2]

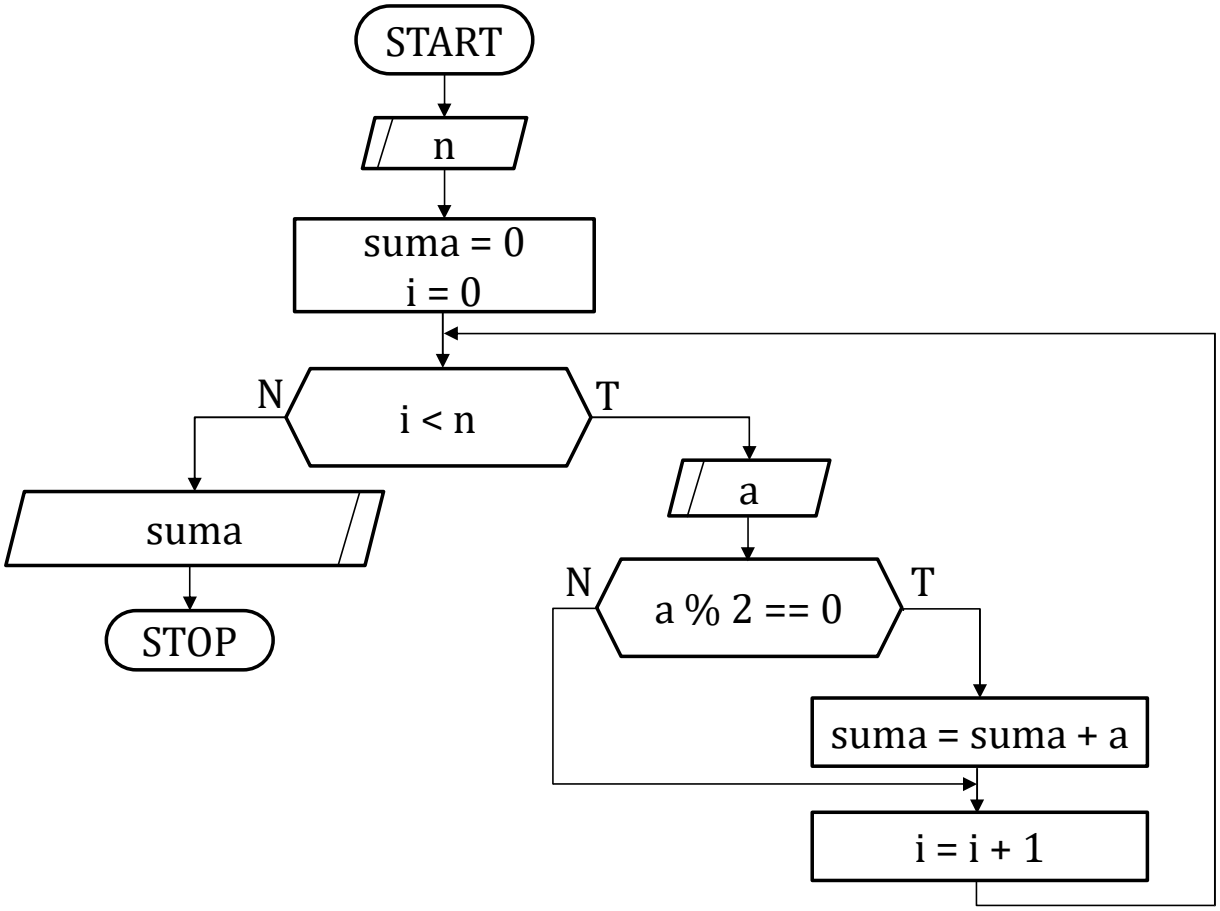
operacje przed pętlą

kolejne iteracje pętli

operacje po pętli

n	suma	i	a	wyjście
4	0	0		
	5	1	5	
	8	2	3	
	8	3	0	
	6	4	-2	
				6

Zad. 6. Algorytm wczytujący *n*-elementowy ciąg liczb i obliczający sumę wszystkich liczb parzystych ciągu.



Zad. 6. Algorytm wczytujący *n*-elementowy ciąg liczb i obliczający sumę wszystkich liczb parzystych ciągu.

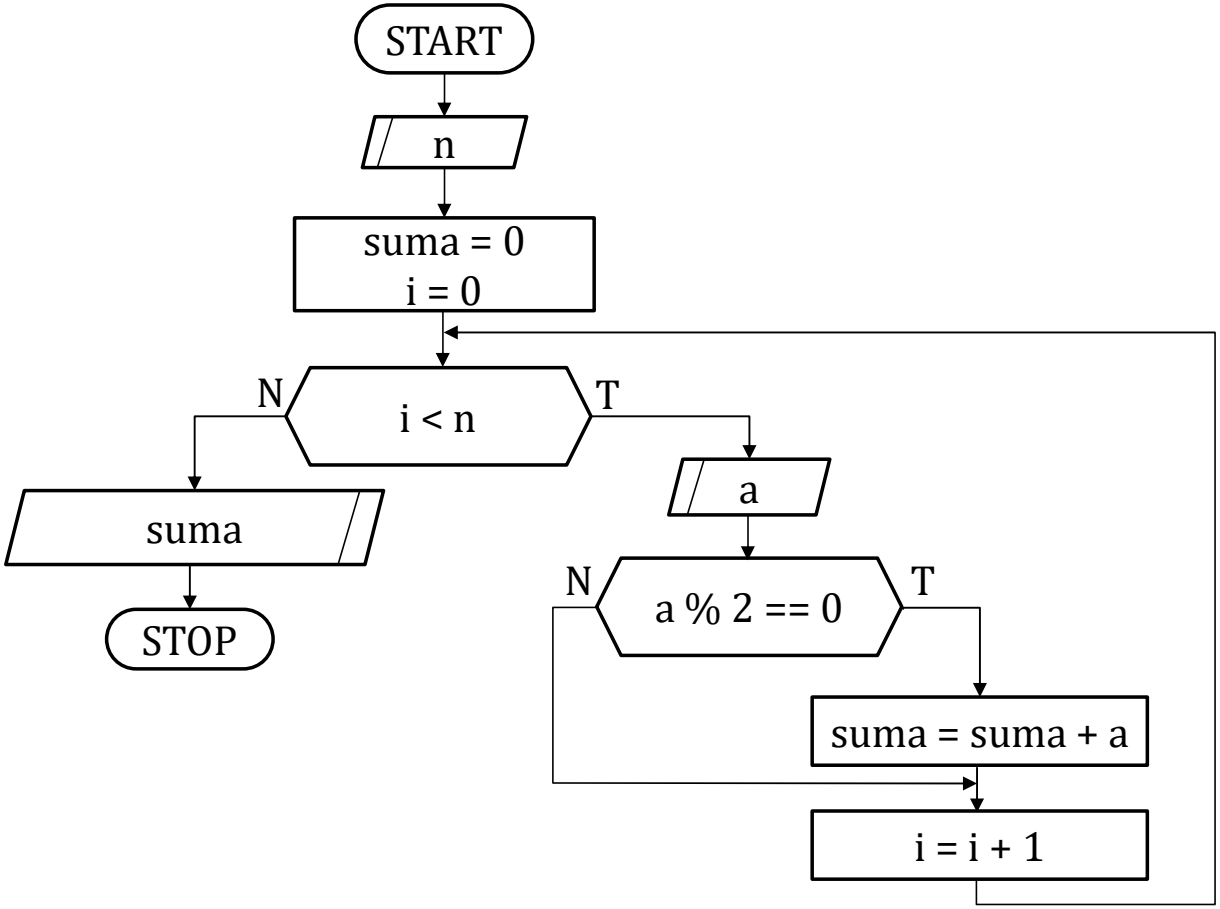
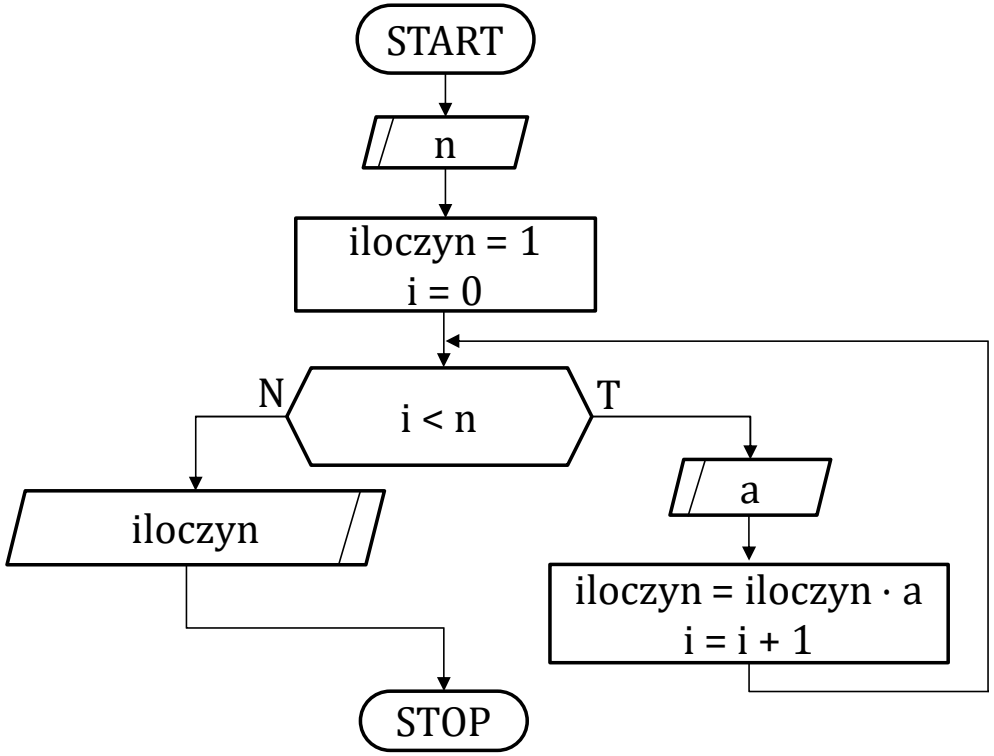


Tabela pamięci dla danych wejściowych: [5, 6, 3, 0, 4, 7]

n	suma	i	a	wyjście
5	0	0		
	6	1	6	
		2	3	
	6	3	0	
	10	4	4	
		5	7	
				10

Zad. 7. Algorytm wczytujący *n*-elementowy ciąg liczb i obliczający iloczyn jego elementów.



Zad. 7. Algorytm wczytujący *n*-elementowy ciąg liczb i obliczający iloczyn jego elementów.

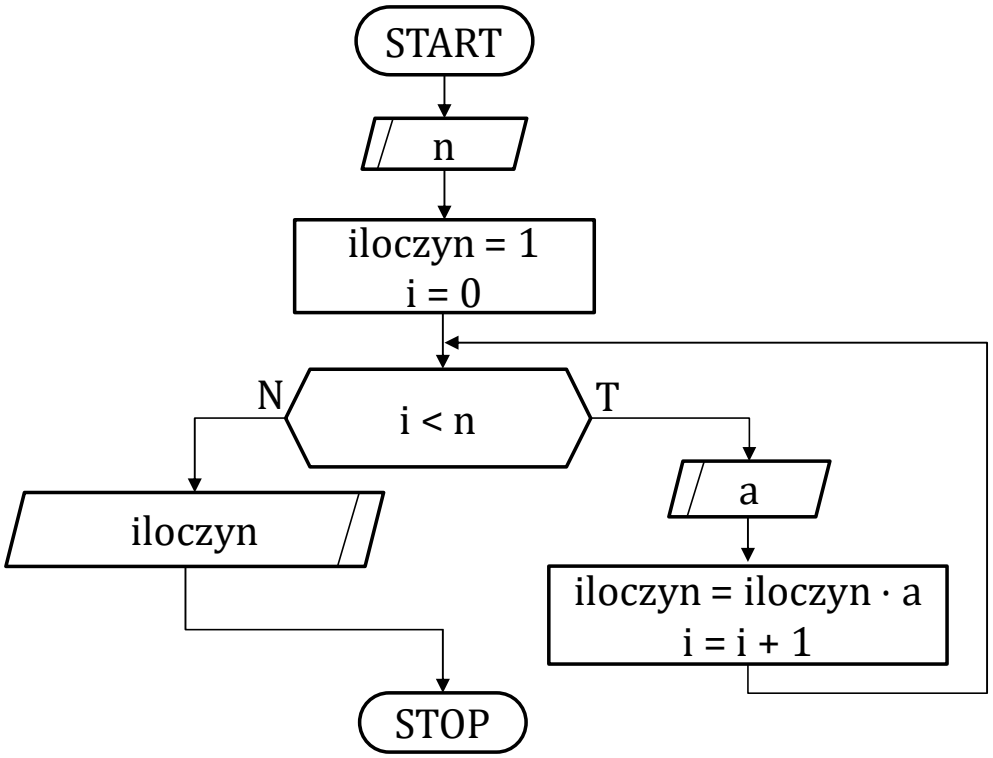
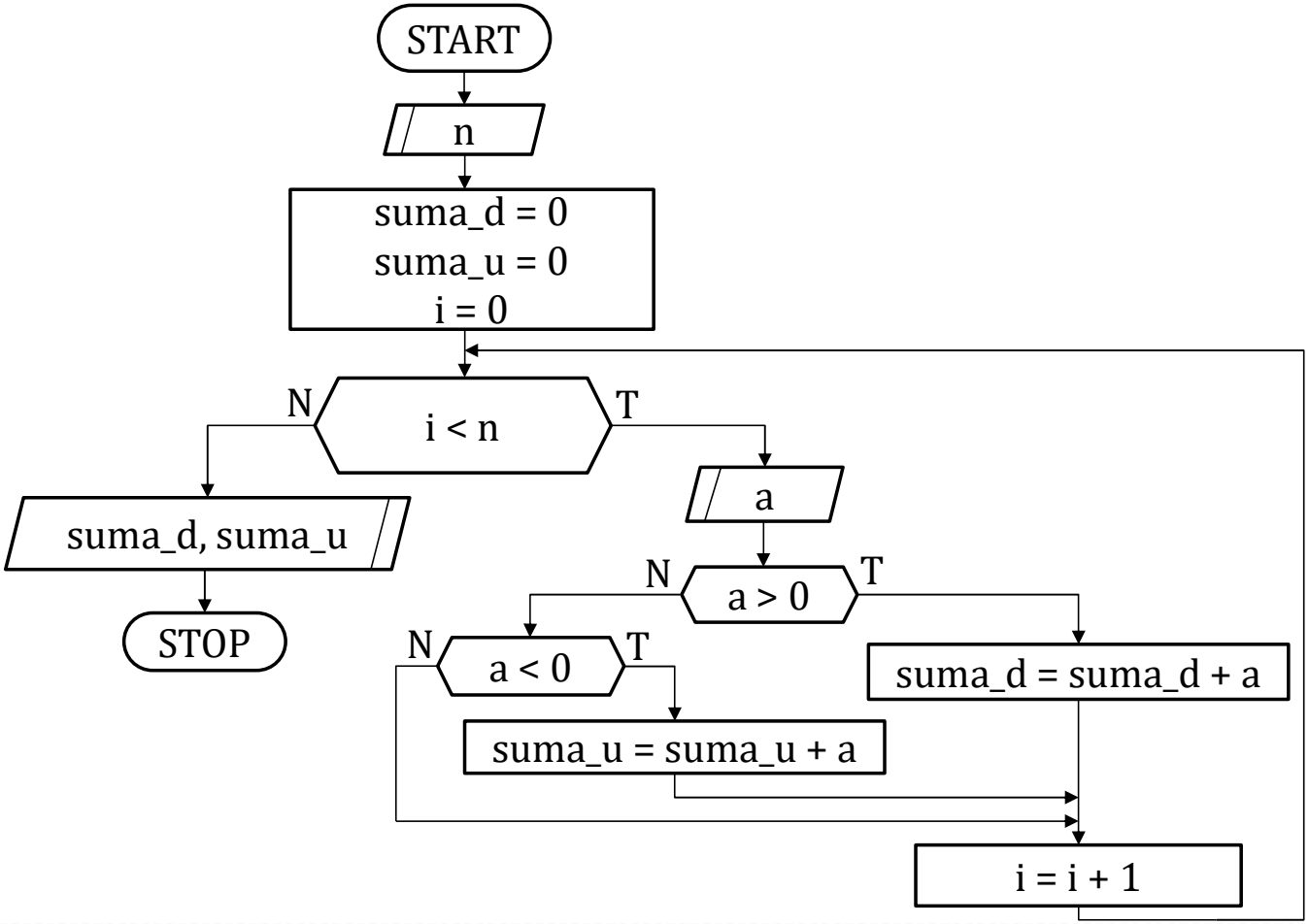


Tabela pamięci dla danych wejściowych: [3, 2, 4, 3]

n	iloczyn	i	a	wyjście
3	1	0		
	2	1	2	
	8	2	4	
	24	3	3	
				24

Zad. 8. Algorytm wczytujący *n*-elementowy ciąg liczb i obliczający sumę liczb dodatnich i sumę liczb ujemnych.



Zad. 8. Algorytm wczytujący *n*-elementowy ciąg i obliczający sumę liczb dodatnich i sumę liczb ujemnych.

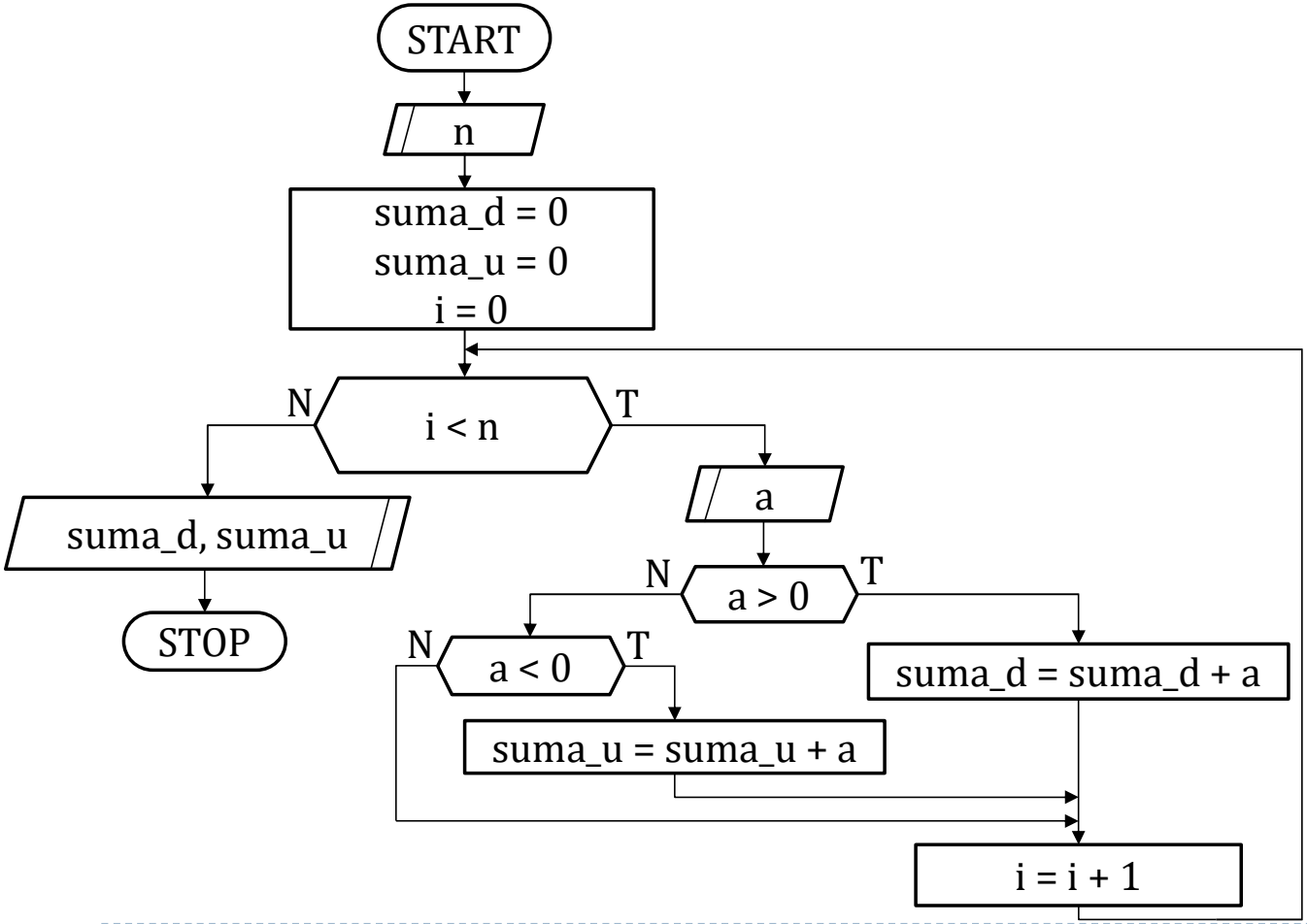


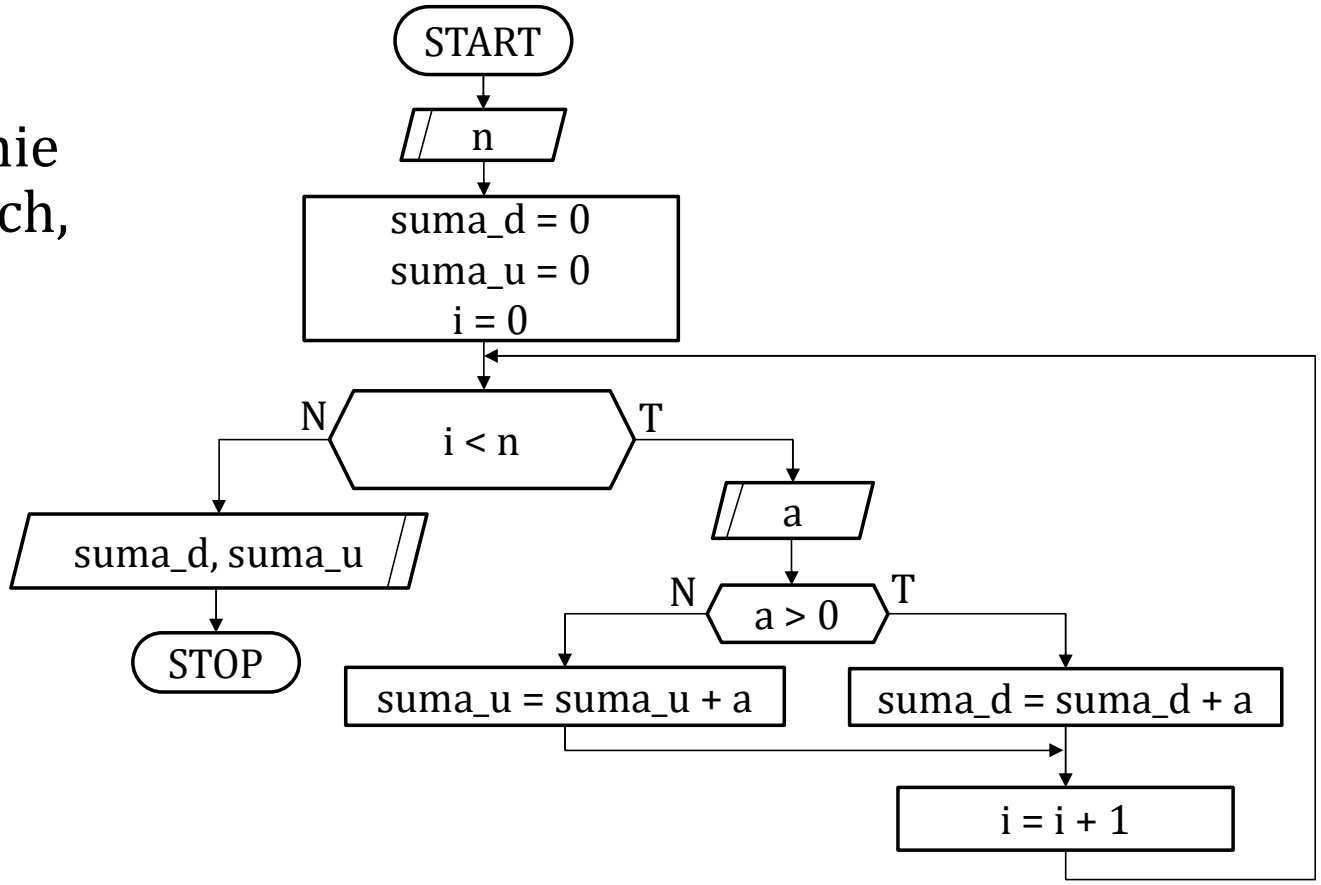
Tabela pamięci dla danych wejściowych: [5, -6, 3, 0, -4, 7]

n	suma_d	suma_u	i	a	wyjście
5	0		0		
		-6	1	-6	
	3		2	3	
			3	0	
		-10	4	-4	
	10		5	7	
					10 -10

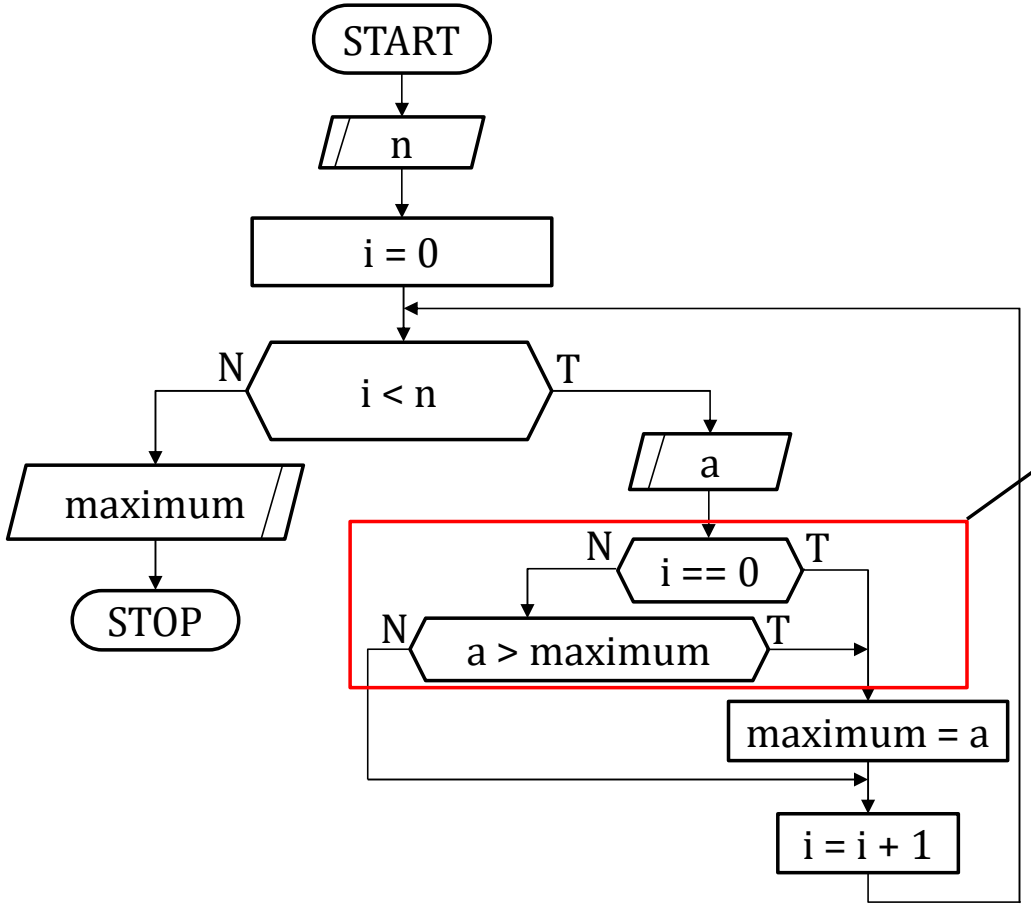
Zad. 8. Algorytm wczytujący *n*-elementowy ciąg i obliczający sumę liczb dodatnich i sumę liczb ujemnych - wersja alternatywna.

- Zauważmy, że liczba zero nie zmienia wyniku algorytmu, niezależnie od tego, czy zostanie dodana do sumy liczb dodatnich, czy sumy liczb ujemnych (x + 0 = x).

- W związku z tym algorytm można uprościć.

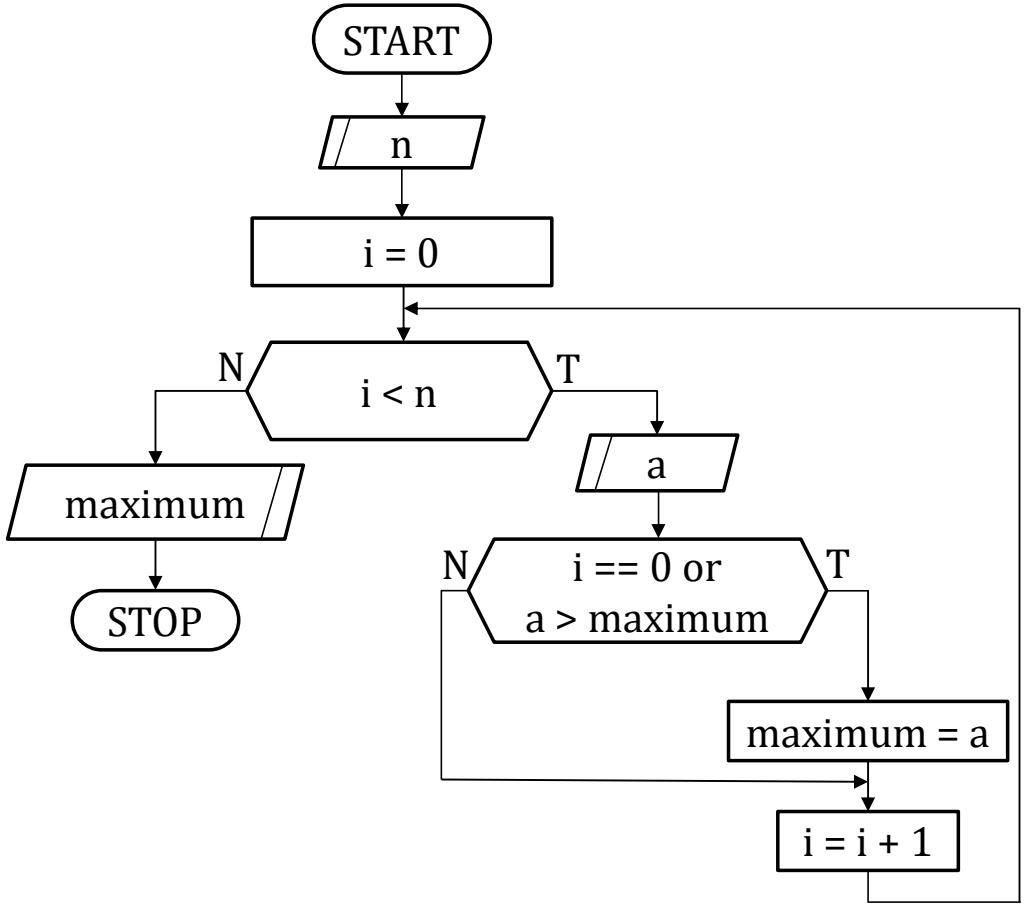


Zad. 9. Algorytm wczytujący *n*-elementowy ciąg i wyznaczający największy wyraz ciągu.

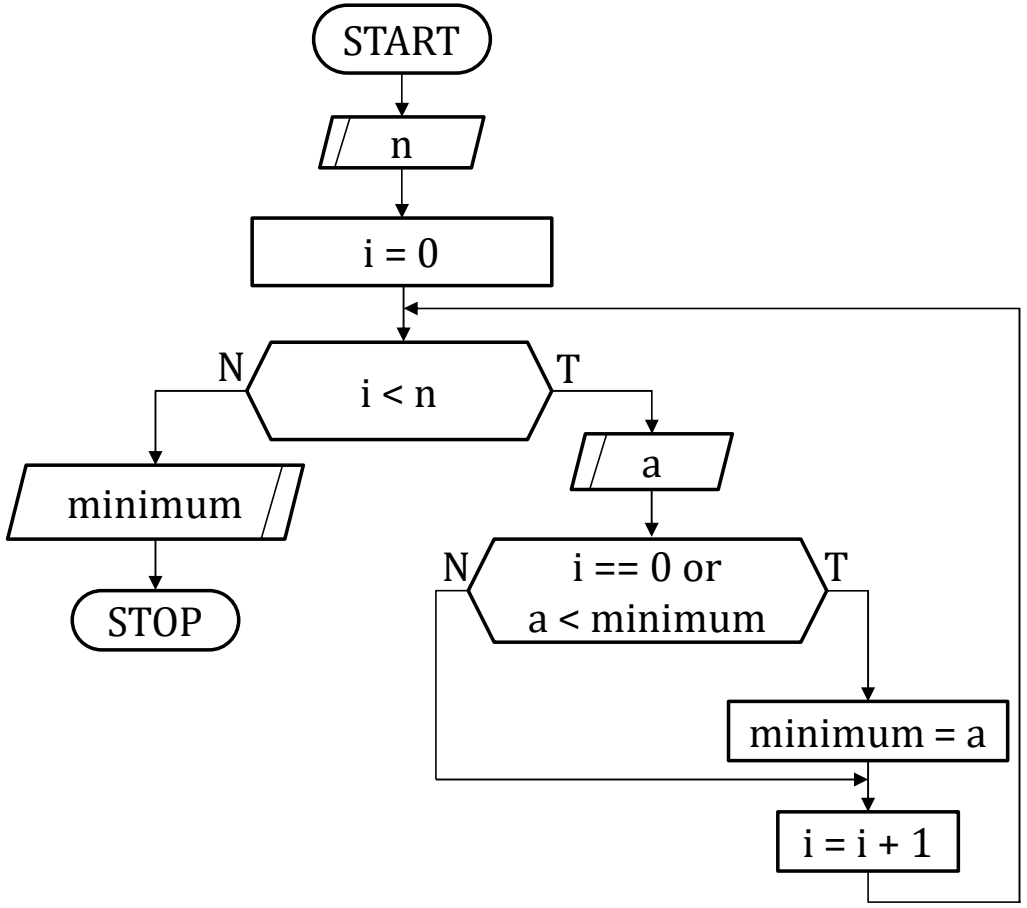


Te dwa bloki można złączyć w jeden operatorem logicznym. Jakim?

Zad. 9. Algorytm wczytujący *n*-elementowy ciąg i wyznaczający największy wyraz ciągu – wersja alternatywna.



Zad. 10. Algorytm wczytujący *n*-elementowy ciąg i wyznaczający najmniejszy wyraz ciągu.



Zad. 10. Algorytm wczytujący *n*-elementowy ciąg i wyznaczający najmniejszy wyraz ciągu.

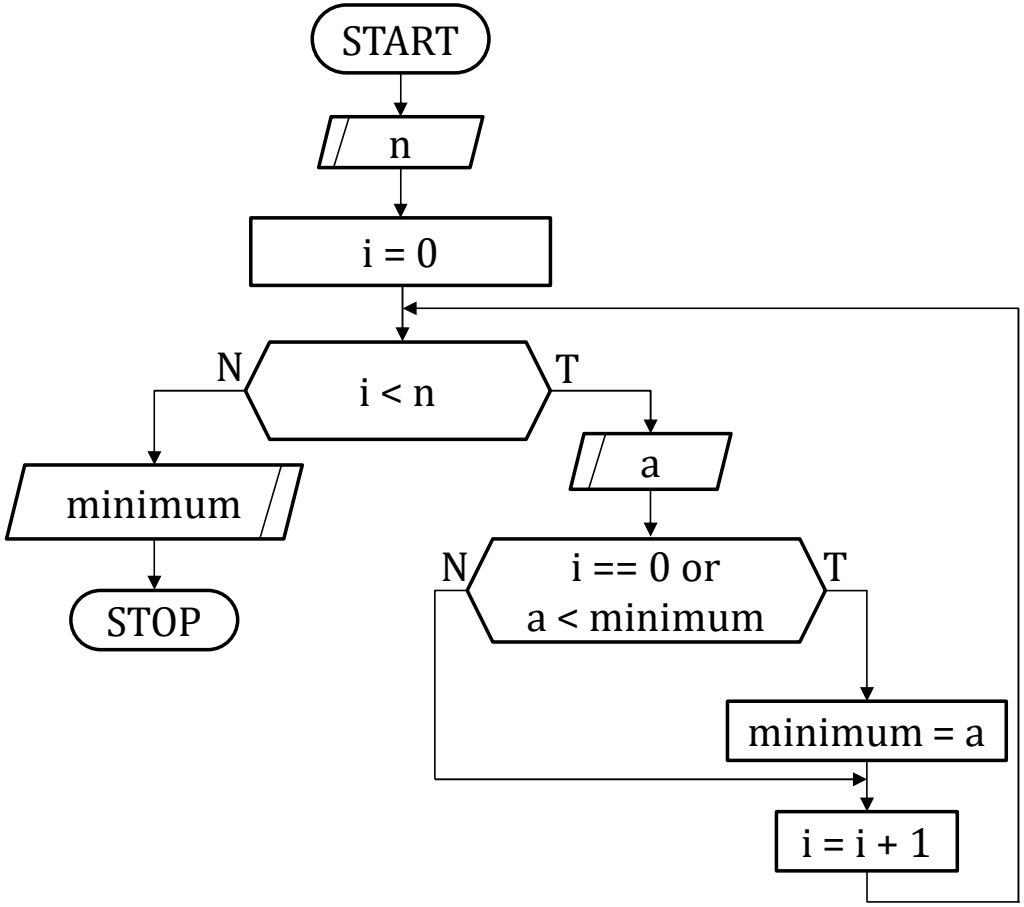


Tabela pamięci dla danych wejściowych: [4, 3, -3, 0, -5]

n	minimum	i	a	wyjście
4		0		
	3	1	3	
	-3	2	-3	
		3	0	
	-5	4	-5	
				-5

Zad. 11. Algorytm wczytujący *n*-elementowy ciąg i obliczający średnią harmoniczną wszystkich jego elementów.

Wzór na średnią harmoniczną:

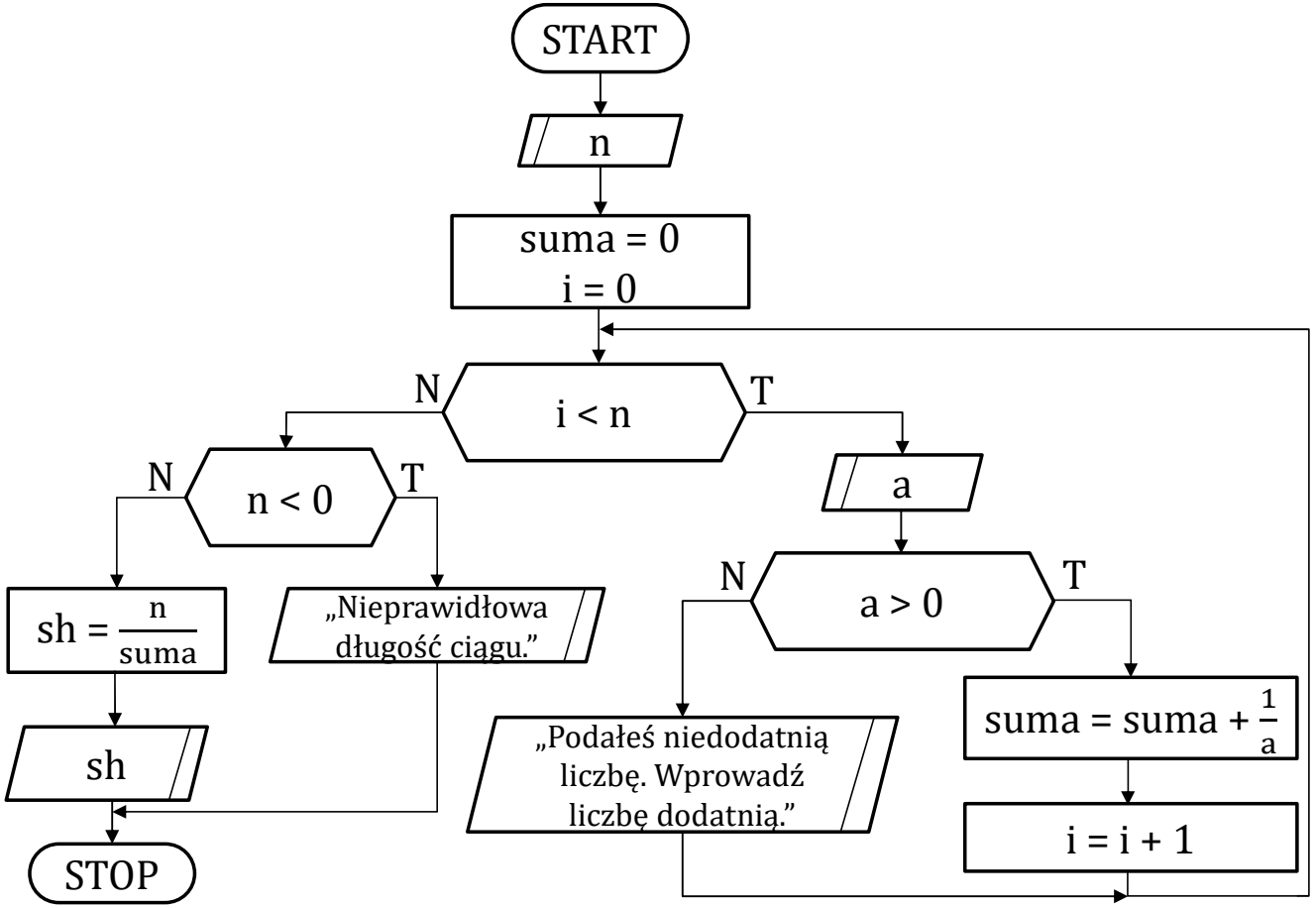
$$sh = \frac{n}{\frac{1}{a1}+\frac{1}{a2}+\cdots+\frac{1}{an}},$$

gdzie:

n - długość ciągu.

a1, *a2*, …, *an* – kolejne wyrazy ciągu.

Uwaga: średnią harmoniczną można obliczyć tylko dla liczb dodatnich.



Zad. 11. Algorytm wczytujący *n*-elementowy ciąg i obliczający średnią harmoniczną wszystkich jego elementów.

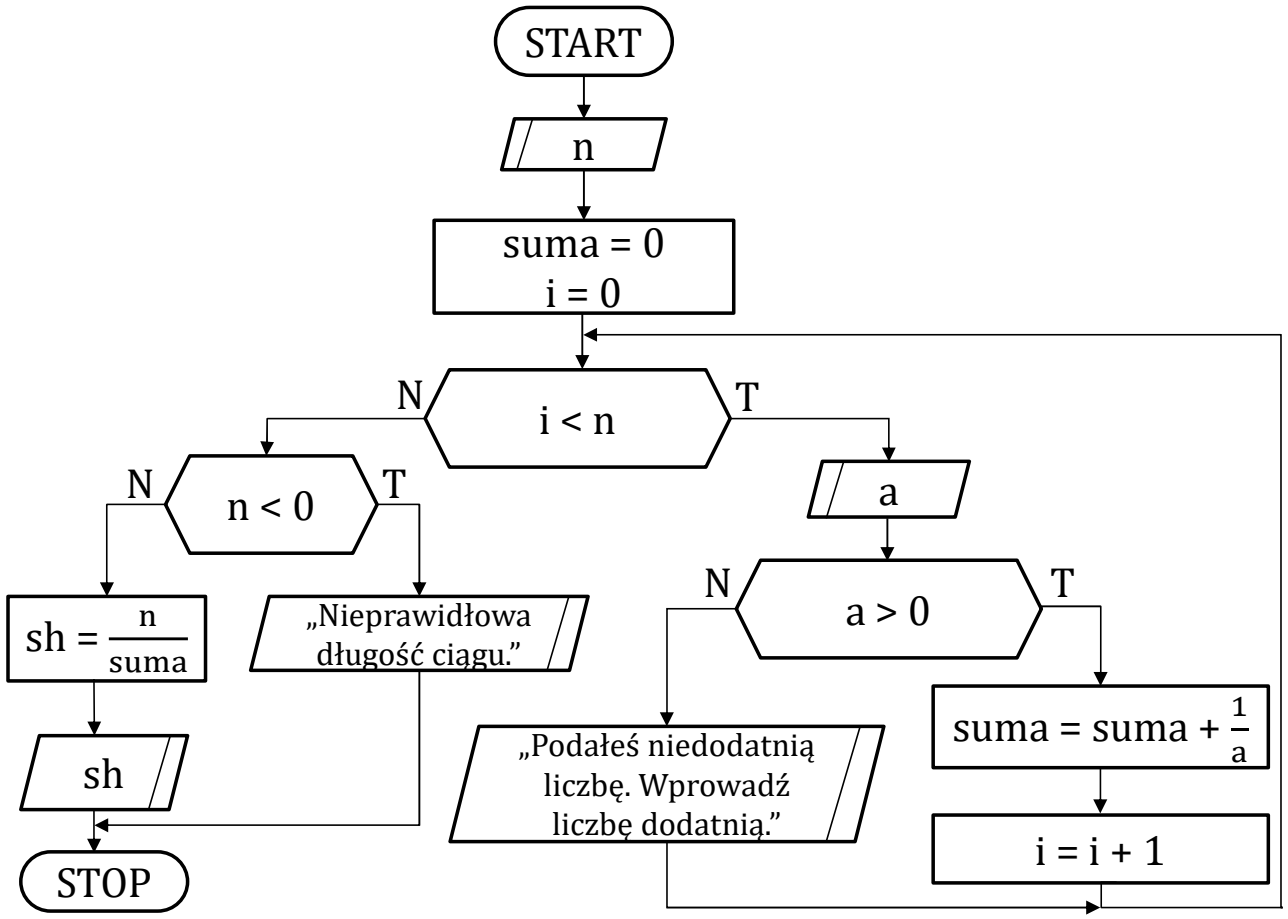


Tabela pamięci dla danych wejściowych: [4, 1, 3, 15, 0, 5]

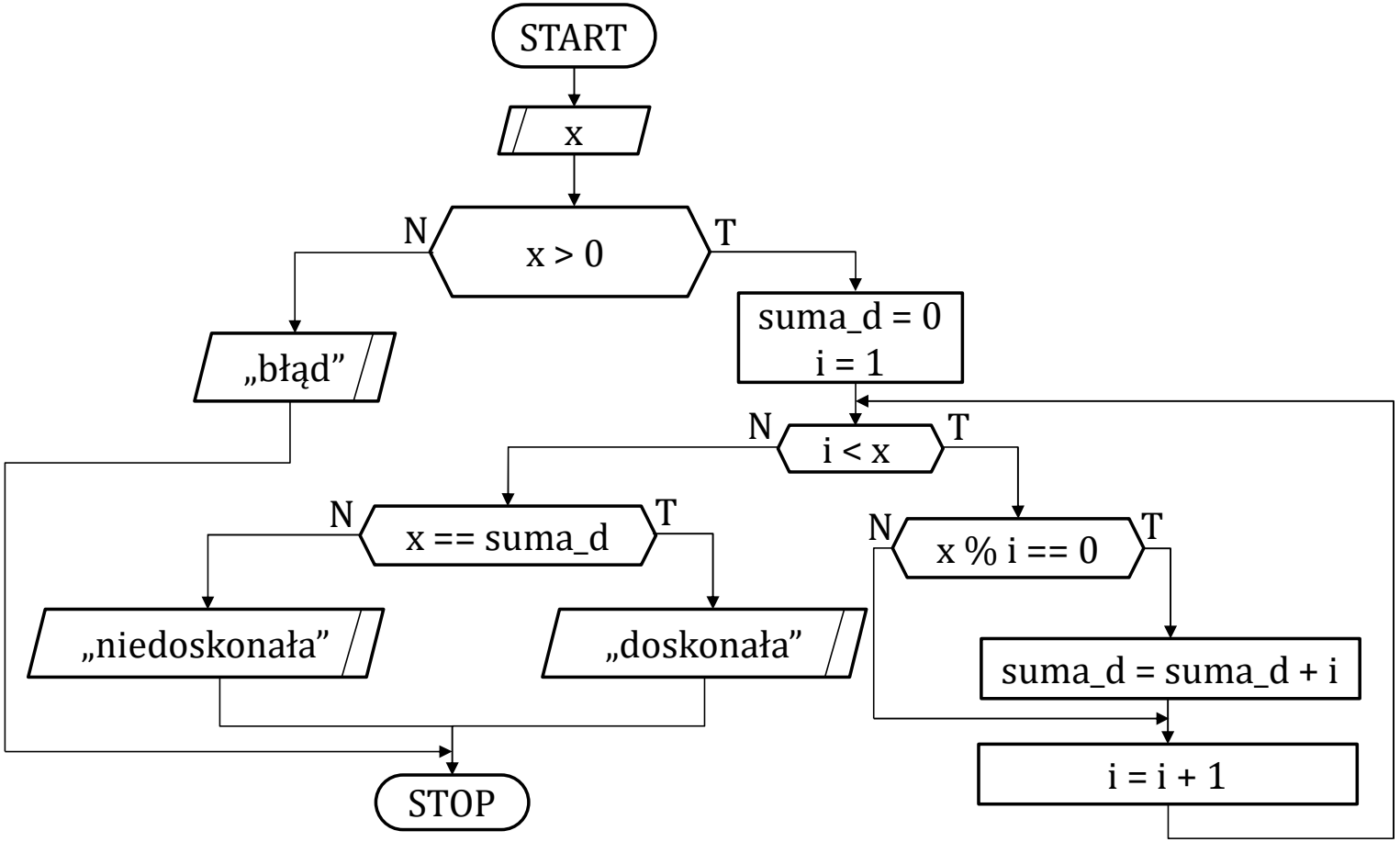
n	suma	i	a	sh	wyjście
4	0	0			
	1	1	1		
	$\frac{4}{3}$	2	3		
	$\frac{21}{15} = \frac{7}{5}$	3	15		
			0		Podałeś niedodatnią liczbę…
	$\frac{8}{5}$	4	5		
				$\frac{20}{8} = \frac{5}{2}$	$\frac{5}{2}$

Zad. 12. Algorytm wczytujący liczbę naturalną i sprawdzający czy jest to liczba doskonała.

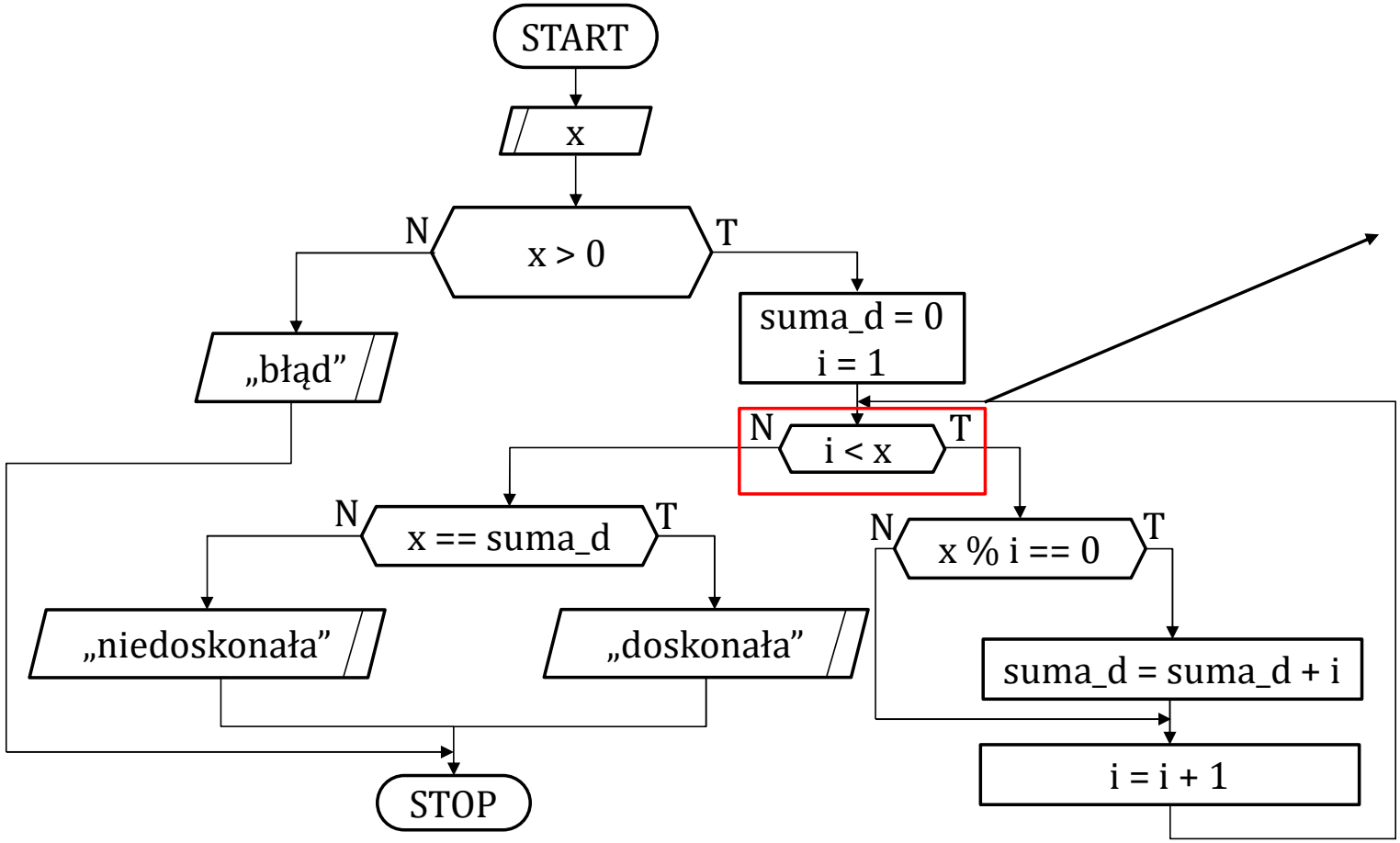
- Liczba doskonała – liczba naturalna, która jest sumą wszystkich swych dzielników właściwych.

- Dzielnik właściwy – liczba, która dzieli daną liczbę bez reszty i jest od niej mniejsza.

Zad. 12. Algorytm wczytujący liczbę naturalną i sprawdzający czy jest to liczba doskonała.



Zad. 12. Algorytm wczytujący liczbę naturalną i sprawdzający czy jest to liczba doskonała.



Zauważmy, że liczby większe niż $\frac{x}{2}$ nie mogą być dzielnikami właściwymi liczby x.

Algorytm można więc zoptymalizować zapisując ten warunek inaczej.

Zad. 12. Algorytm wczytujący liczbę naturalną i sprawdzający czy jest to liczba doskonała – wersja zoptymalizowana.

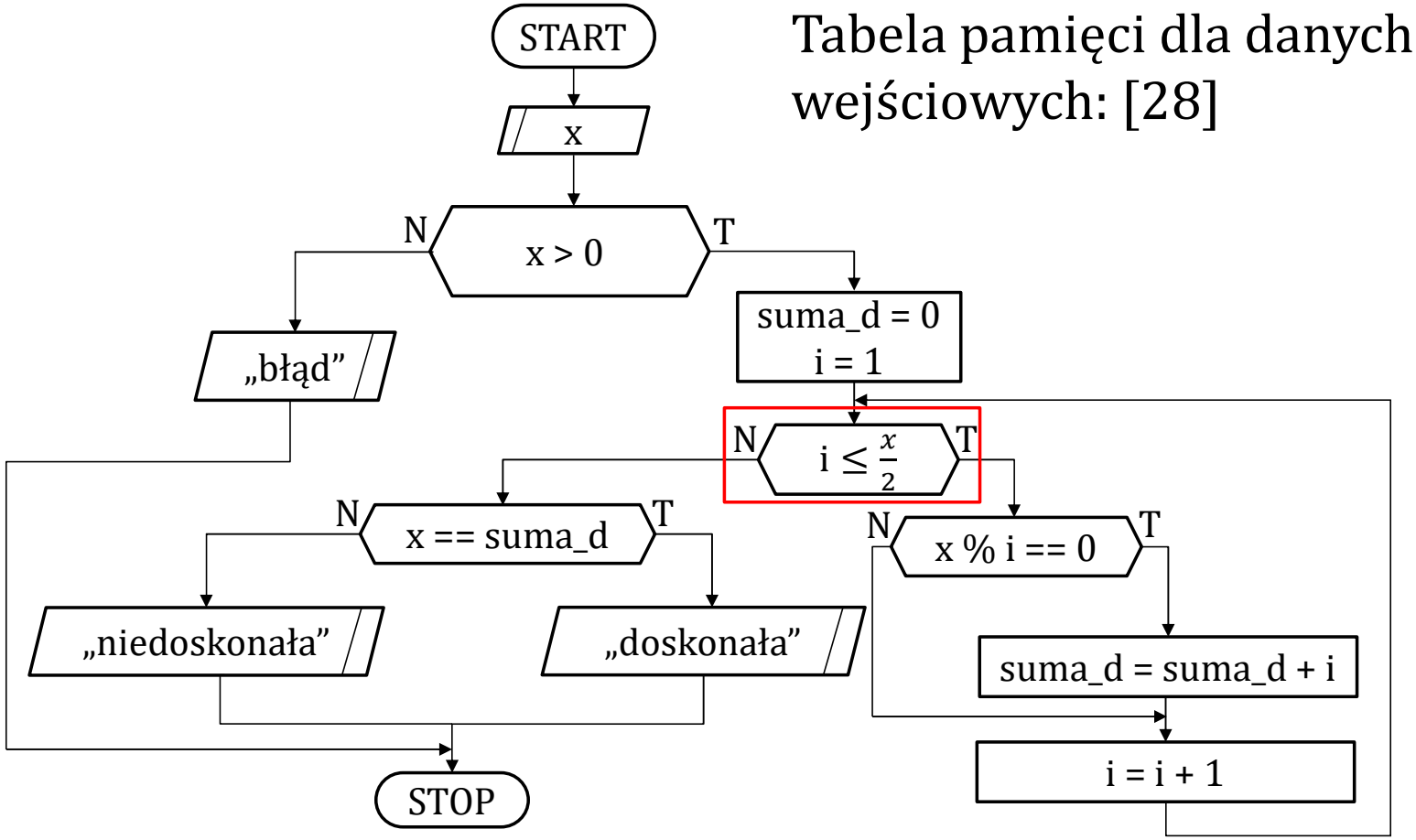
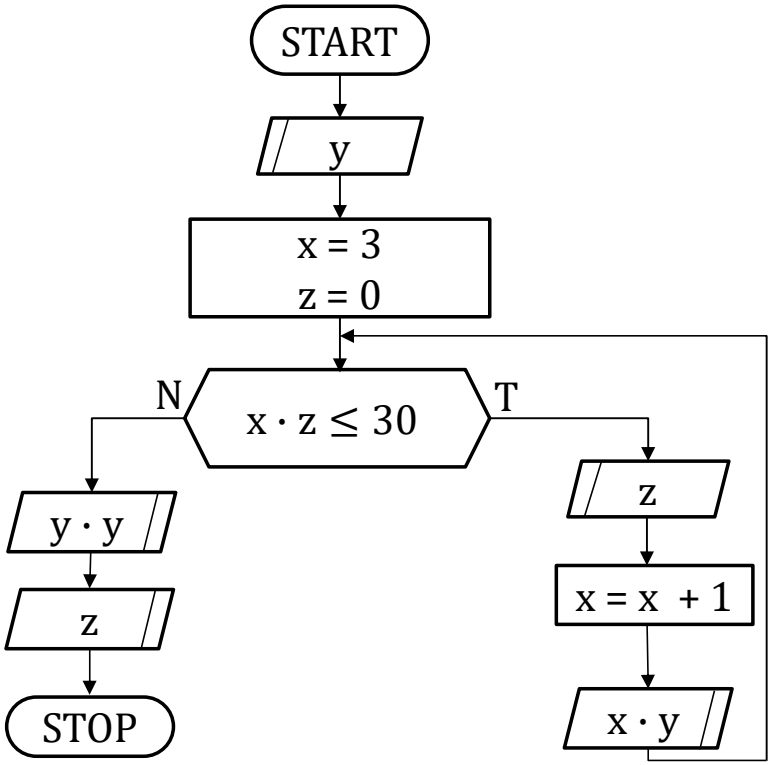


Tabela pamięci dla danych wejściowych: [28]

x	suma_d	i	wyjście
28	0	1	
	1	2	
	3	3	
		4	
	7	5	
		6	
		7	
	14	8	
...	...	...	...
	28	15	
			doskonała

Zad. 13. Co wypisze algorytm, jeśli na wejście podane zostaną liczby [-1, 1, 3, 5, 5]? Narysuj tabelę pamięci.

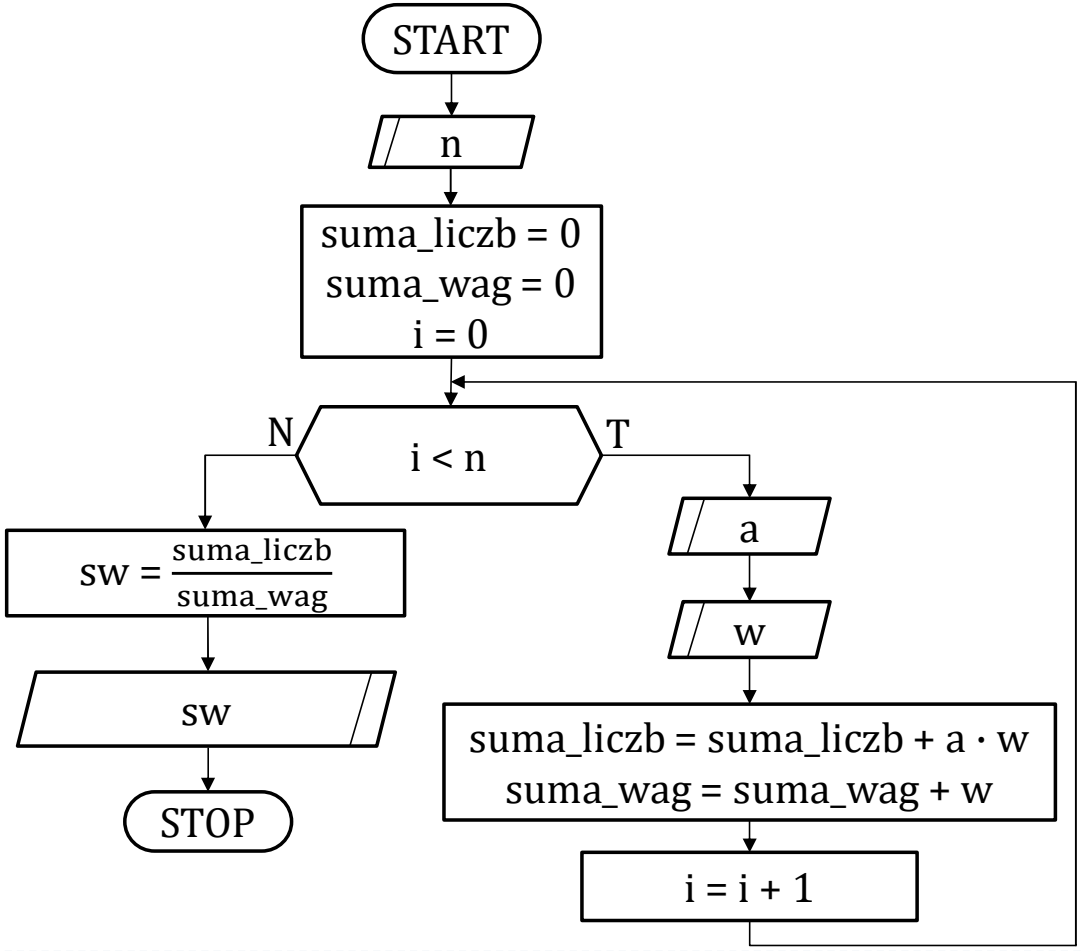


x	y	z	wyjście
3	-1	0	
4		1	-4
5		3	-5
6		5	-6
7		5	-7
			1 5

Odpowiedź:

Algorytm wypisze: -4 -5 -6 -7 1 5

Zad. 14. Algorytm wczytujący *n*-elementowy ciąg liczb oraz wag i obliczający ważoną średnią arytmetyczną podanych liczb.



Zad. 14. Algorytm wczytujący *n*-elementowy ciąg liczb oraz wag i obliczający ważoną średnią arytmetyczną podanych liczb.

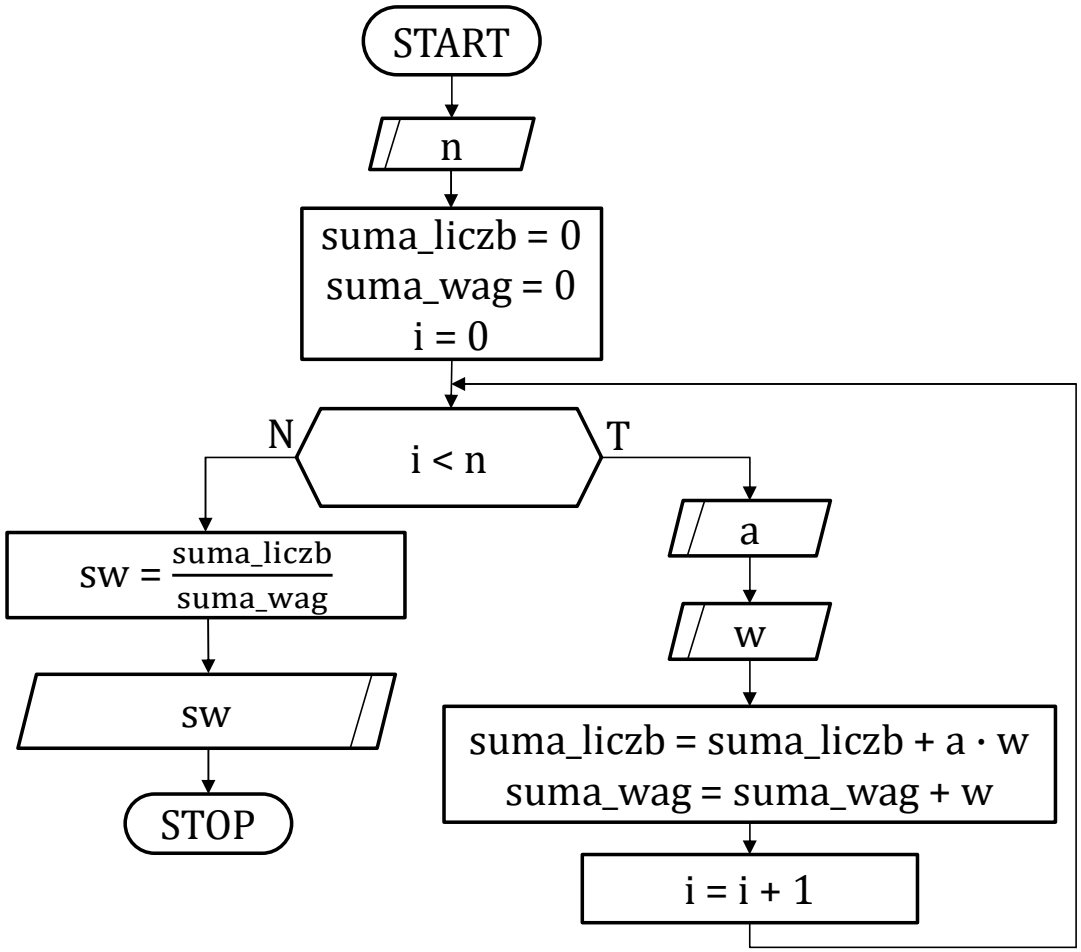


Tabela pamięci dla danych wejściowych: [3, 4.5, 3, 4, 5, 3.5, 2]

n	suma_liczb	suma_wag	i	a	w	sw	wyjście
3	0	0	0				
	13.5	3	1	4.5	3		
	33.5	8	2	4	5		
	40.5	10	3	3.5	2		
						4.05	4.05