MAŁA BRANIBORSKA 2026 - DYWIZJA A (algorytmiczno – programistyczna)

ELIMINACJE SZKOLNE (26 stycznia 2026)

Niniejsza notatka zawiera wybrane zagadnienie dla **dywizji algorytmiczno-programistycznej**. Materiał został opracowany pod kątem 10 pytań zamkniętych oraz 2 zadań otwartych, które pojawią się na arkuszu etapu szkolnego. Dzięki zamieszczonemu materiałowi uczeń powinien poznać cechy poprawnego algorytmu, rozpoznawać i budować schematy blokowe, a także zrozumieć operacje dzielenia całkowitego oraz wyznaczania reszty. Najtrudniejszą umiejętnością do opanowania jest samodzielne śledzenie zmian wartości zmiennych w pętlach i warunkach pseudokodu, co pozwala na precyzyjne odtworzenie przebiegu algorytmu w tabeli dla konkretnych danych wejściowych. Zakładamy, że uczeń zna elementarne pojęcia zmiennej czy iteracji.

Zachęcamy do jej przeczytania i wykonania zamieszczonych w treści ćwiczeń bez podglądania rozwiązań. Odpowiedzi do nich znajdują się na końcu opracowania. Powodzenia!

Pojęcie algorytmu i wybrane sposoby przedstawiania algorytmów

Algorytm to przepis na rozwiązanie określonego zadania w skończonej liczbie kroków.

Wybrane własności algorytmu:

- Jest **skończony**, jeśli dla dowolnych danych wejściowych spełniających warunki specyfikacji generuje wyniki w skończonej liczbie kroków.
- Jest **poprawny**, jeśli dla poprawnych danych wejściowych daje poprawne dane wyjściowe. Zarówno dane, jak i wyniki powinny spełniać warunki określone w specyfikacji.
- Jest **dobrze określony**, jeśli wszystkie polecenia i działania wykonywane przez algorytm są czytelne oraz możliwe do wykonania, a ich kolejność jest dokładnie określona, co umożliwia zapisanie go w języku programowania wysokiego poziomu.
- Jest **uniwersalny**, co oznacza, że algorytm nie może służyć do rozwiązania tylko jednego, konkretnego zadania (np. "dodaj 2 + 5"). Musi on być schematem postępowania dla całej klasy problemów tego samego typu. Oznacza to, że powinien poprawnie przetwarzać dowolne dane wejściowe, o ile mieszczą się one w granicach określonych w specyfikacji (np. "dodaj dwie dowolne liczby całkowite a i b").

Algorytmy możemy przedstawiać na wiele sposobów. Wybraliśmy kilka z nich:

1. **Opis słowny**

   To przedstawienie kroków algorytmu w języku naturalnym (potocznym). Jest to forma najmniej precyzyjna.

   Przykład:
   Algorytm parzenia herbaty (opis słowny): *Nalej wodę do czajnika i zagotuj ją. Włóż torebkę herbaty do kubka. Zalej torebkę wrzątkiem i odczekaj 5 minut. Wyjmij torebkę i gotowe.*

2. Lista kroków

Algorytm zapisany jest w postaci ponumerowanych punktów. Każdy punkt opisuje jedną konkretną operację. Jest to forma bardziej uporządkowana niż opis słowny, pozwalająca na łatwe odwoływanie się do konkretnych etapów (np. „wróć do kroku 2").

Przykład:
Przeanalizujmy algorytm, którego zadaniem jest ***wypisanie kolejnych liczb dwucyfrowych***

1. *Zmiennej n przypisz wartość 10.*
2. *Wypisz n.*
3. *Zwiększ n o 1.*
4. *Jeśli n jest większe od 5, to wróć do kroku 2.*
5. *Zakończ.*

Gdy zaczniemy wypisywać liczby zgodnie z działaniem tej wersji algorytmu, szybko dojdziemy do wniosku, że warunek określony w czwartym kroku jest zawsze spełniony, a zatem w nieskończoność wracalibyśmy do kroku drugiego. Do kroku piątego, w którym następuje zakończenie algorytmu, nie doszlibyśmy nigdy. Taki algorytm nie jest poprawny, gdyż nigdy nie kończy swojego działania.

Dokonajmy innej modyfikacji wyjściowego algorytmu:

1. *Zmiennej n przypisz wartość 10.*
2. *Wypisz n.*
3. *Zwiększ n o 2.*
4. *Jeśli n jest mniejsze od 100, to wróć do kroku 2.*
5. *Zakończ.*

Zobaczmy, co zostanie wypisane zgodnie z działaniem tego algorytmu: 10, 12, 14, ..., 98. Czy o to chodziło w treści zadania? Nie, zatem ten algorytm również nie jest poprawny, ponieważ nie wykonuje określonego w specyfikacji zadania: nie wypisuje wszystkich liczb dwucyfrowych, tylko te, które są parzyste.

Ćwiczenie 1. Spróbuj zapisać poprawnie algorytm wypisujący kolejne liczby dwucyfrowe w postaci listy kroków.

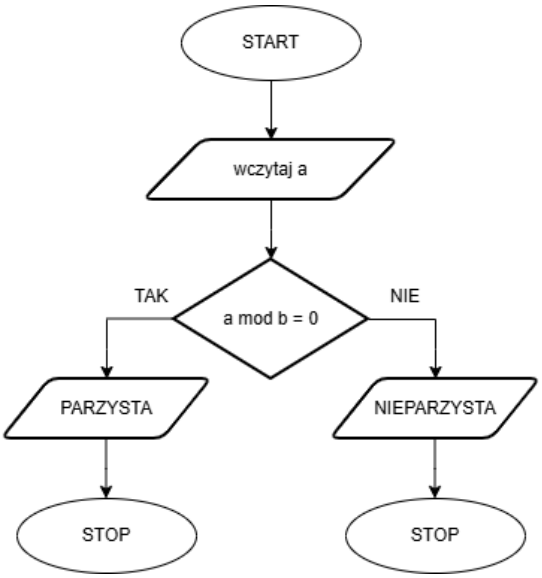
3. Schemat blokowy

Najbardziej popularna wizualna metoda przedstawiania algorytmów. Wykorzystuje standaryzowane figury geometryczne połączone strzałkami, co pozwala szybko zrozumieć strukturę logiczną, zwłaszcza instrukcje warunkowe i pętle.

W schematach blokowych każda skrzynka ma ściśle określone znaczenie:

- Owal (Start/Koniec): Rozpoczyna i kończy algorytm. Wewnątrz wpisujemy np. „Start", „Koniec" lub „Stop".
- Równoległobok (Wejście/Wyjście): Służy do wprowadzania danych (np. Pobierz x) lub wypisywania wyników (np. Wypisz suma).
- Prostokąt (Operacyjny): Tu wpisujemy obliczenia i przypisania wartości (np. suma := a + b).
- Romb (Decyzyjny): Zawiera pytanie/warunek (np. a > 0). Ma zawsze dwa wyjścia: TAK i NIE.

Przykład:
Sprawdzanie parzystości liczby *to jeden z najprostszych algorytmów decyzyjnych. Wykorzystuje operację dzielenia modulo (reszta z dzielenia).*

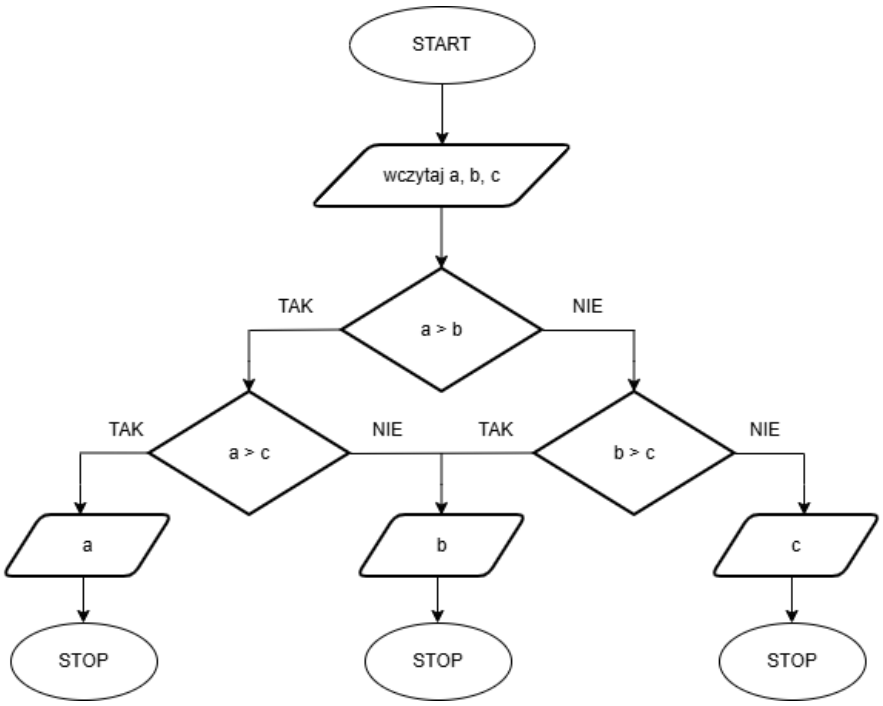


Logika: Jeśli reszta z dzielenia liczby przez 2 wynosi 0, to liczba jest parzysta. W przeciwnym razie jest nieparzysta.
Zwróć uwagę na operator dzielenia modulo (mod) wykorzystany w przedstawionym schemacie.

Często spotkasz się tutaj z pascalowymi operatorami:

- **:=** – przypisz wartość (np. x := 10).
- **<>** – różne od (np. x <> 0).
- **div** – dzielenie bez reszty (np. 13 div 4 = 3).
- **mod** – reszta z dzielenia (np. 13 mod 4 = 1).

Inny przykład: ***Szukamy największej wśród trzech liczb****:*



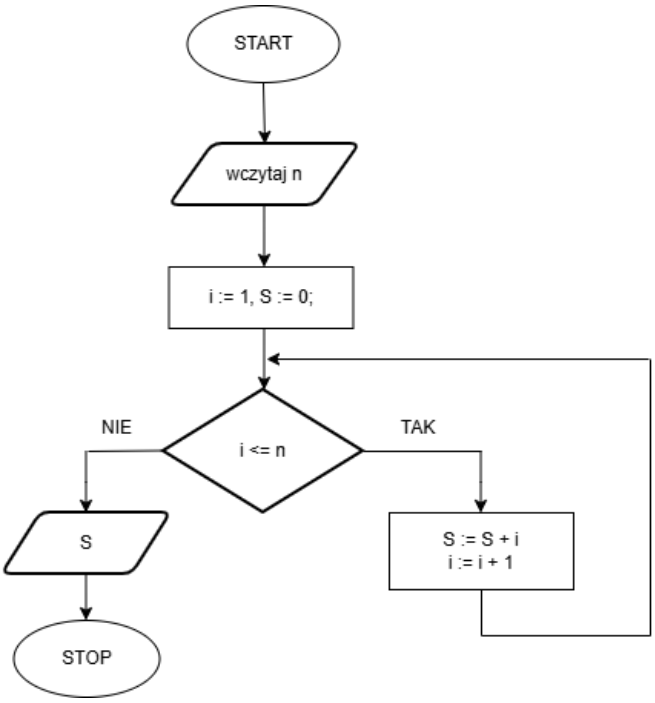
Można go przeanalizować za pomocą następującej liczby kroków:

1. START – rozpoczęcie algorytmu.
2. POBIERZ wartości zmiennych a, b, c.
3. SPRAWDŹ WARUNEK: Czy a > b?
    - Jeśli TAK: Sprawdź, czy a > c?
        - Jeśli TAK: Wypisz wynik a.
        - Jeśli NIE: Wypisz wynik c.
    - Jeśli NIE: Sprawdź, czy b > c?
        - Jeśli TAK: Wypisz wynik b.
        - Jeśli NIE: Wypisz wynik c.
4. STOP – zakończenie algorytmu.

Ćwiczenie 2. Spróbuj narysować schemat blokowy algorytmu, który wczytuje trzy liczby reprezentujące długości odcinków i odpowiada na pytanie, czy można z nich zbudować trójkąt.

Jeszcze inny przykład: ***sumowanie liczb od 1 do n****:*

Ostatni przykład schematu blokowego przedstawia algorytm sumowania liczb od 1 do n, gdzie liczba n jest podana przez użytkownika.



Algorytm rozpoczyna się od pobrania od użytkownika liczby n, która określa, ile kolejnych wartości należy dodać. Następnie przygotowujemy zmienną sumy S z wartością początkową 0 (zwróć uwagę, że 0 jest elementem neutralnym dodawania i nie może to być 1) oraz licznik i ustawiony na 1. W kolejnym etapie sprawdzamy, czy licznik nie przekroczył podanego n. Dopóki ten warunek jest spełniony, do aktualnej sumy dodajemy wartość licznika, a sam licznik zwiększamy o jeden i wracamy do sprawdzenia warunku. Gdy licznik stanie się większy od n, działanie pętli zostaje przerwane, a algorytm wyświetla końcowy wynik sumowania i kończy pracę.

To samo rozumowanie w postaci listy kroków:

1. START
2. Pobierz n
3. Ustaw S := 0 oraz i := 1
4. Warunek: Czy i <= n?
    - TAK:
        - S := S + i
        - i := i + 1
        - Wróć do sprawdzenia warunku.
    - NIE:
        - Wypisz S
5. STOP

Ćwiczenie 3. Spróbuj zbudować schemat blokowy algorytmu sumującego n liczb, tym razem podanych przez użytkownika.

4. Pseudokod

Zapis przypominający język programowania, ale pozbawiony rygorystycznej składni. Wykorzystuje słowa kluczowe (np. JEŻELI, DOPÓKI, WYKONUJ, ZWRÓĆ) w języku ojczystym lub angielskim. Jest to etap pośredni, który łatwo można przetłumaczyć na konkretny kod źródłowy. Przypomina też trochę budowane w przykładach do poprzedniego punktu listy kroków. Bardzo istotne są tutaj wcięcia, które budują bloki (podobnie jest np. w języku Python). Ponadto, w przeciwieństwie do schematu blokowego operatorem przypisania jest tutaj najczęściej strzałeczka skierowana w do zmiennej (<-). Ten zapis algorytmu zilustrujemy dwoma przykładami.

Przykład 1.: funkcja P realizująca potęgowanie.
Funkcja wyznacza y^z. Jest to klasyczna metoda potęgowania przez wielokrotne mnożenie podstawy przez siebie.
Algorytm ustawia wynik początkowy x na 1. Następnie w pętli mnoży ten wynik przez podstawę y dokładnie tyle razy, ile wynosi wartość z. Po każdym mnożeniu zmniejsza licznik z o jeden. Gdy licznik osiągnie zero, algorytm kończy pracę i zwraca ostateczny wynik, który jest w zmiennej x.

```
P(y, z):
  x ← 1
  dopóki z > 0 wykonuj:
    x ← x * y
    z ← z - 1
  zwróć x
```

Przeanalizujmy ten algorytm za pomocą tabelki dla konkretnych danych. Załóżmy, że y = 3 a z = 4.

Obieg pętli	Warunek: z > 0	x (wynik)	z (licznik)
Start	—	1	4
Iteracja 1	TAK (4 > 0)	**3** (1 * 3)	3
Iteracja 2	TAK (3 > 0)	**9** (3 * 3)	2
Iteracja 3	TAK (2 > 0)	**27** (9 * 3)	1
Iteracja 4	TAK (1 > 0)	**81** (27 * 3)	0
Koniec	NIE (0 > 0)	**Wynik: 81**	0

Przykład 2.: funkcja M realizująca mnożenie metodą binarną.
Funkcja M ma dwa argumenty: y oraz z, które są które są liczbami naturalnymi.
Jest to algorytm mnożenia metodą binarną (znany również jako mnożenie rosyjskich chłopów). Algorytm ten rozkłada jedną z liczb na sumę potęg dwójki i sumuje odpowiednie wielokrotności drugiej liczby:

```
M(y, z):
   x ← 0
   dopóki z > 0 wykonuj:
      jeżeli (z mod 2 = 1) to:
         x ← x + y
      y ← 2 * y
      z ← z div 2
   zwróć x
```

Algorytm rozpoczyna się od ustawienia zmiennej wynikowej x na zero. Następnie wchodzi w pętlę, która wykonuje się tak długo, jak liczba z jest większa od zera. Wewnątrz pętli sprawdzana jest parzystość z – jeśli liczba jest nieparzysta, do aktualnego wyniku x dodawana jest wartość y. Niezależnie od tego warunku, w każdym obiegu pętli wartość y jest podwajana, a wartość z dzielona całkowicie przez dwa. Gdy z osiągnie zero, pętla kończy działanie, a algorytm zwraca zgromadzony wynik x, który jest iloczynem liczb wejściowych.

Prześledźmy działanie algorytmu w tabeli dla konkretnych danych wejściowych (dla y = 5, z = 6)

Obieg pętli	**Warunek: z > 0**	**Czy z mod 2 = 1?**	**x (wynik)**	**y (mnożenie)**	**z (dzielenie)**
Start	—	—	0	5	6
Iteracja 1	TAK (6 > 0)	NIE	0	10	3
Iteracja 2	TAK (3 > 0)	**TAK**	**10** (0+10)	20	1
Iteracja 3	TAK (1 > 0)	**TAK**	**30** (10+20)	40	0
Koniec	NIE (0 > 0)	—	**Wynik: 30**	—	—

Co dzieje się w tej tabeli? Kolumna x to suma częściowa. Dodajemy do niej y tylko wtedy, gdy z jest nieparzyste. Kolumna y w każdym kroku rośnie dwukrotnie. Kolumna z w każdym kroku maleje o połowę a gdy spadnie do 0, pętla się kończy a wynik jest w kolumnie x. Algorytm mnoży liczby, sumując tylko wybrane potęgi (podwojenia) liczby y.

Uwaga: Są też inne sposoby przedstawiania algorytmów, chociażby drzewo decyzyjne (graf) lub zapis w konkretnym języku programowania. Pomijamy je i nie obowiązują w etapie szkolnym.

Ćwiczenie 4. Algorytm przyjmuje liczbę n i zeruje zmienną sumy. W pętli, która trwa dopóki n jest większe od zera, wykonuje dwie operacje. Jakie? Gdy liczba n zostanie zredukowana do zera, algorytm zwraca końcowy wynik. Przeanalizuj działanie funkcji opisanej pseudokodem funkcji S dla dowolnych danych wejściowych (np. 123) i odpowiedz na pytanie, co ta funkcja realizuje?

```
S(n):
   S ← 0
   dopóki n > 0 wykonuj:
      S ← S + (n mod 10)
      n ← n div 10
   zwróć S
```

Ćwiczenie 5.

Przeanalizuj poniższy pseudokod i odpowiedz na pytanie: Jaką wartość zwróci funkcja dla n = 3?

```
Kwadraty(n):
1.  r ← 0
2.  for i ← 1 to n
3.      do for j ← 1 to i
4.          do r ← r + j
5.  return r
```

Rozwiązanie Ćwiczenia 1: Lista kroków (Liczby dwucyfrowe)

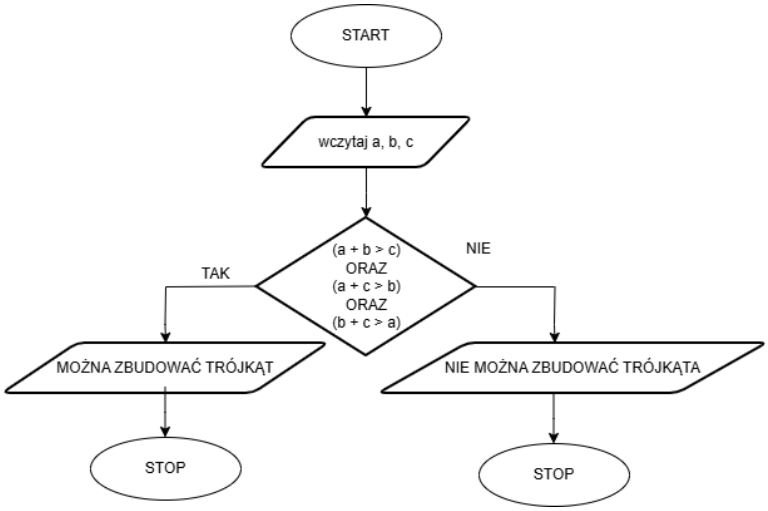
Należało zapisać poprawnie algorytm wypisujący wszystkie liczby dwucyfrowe (10, 11, ..., 99).

1. Zmiennej **n** przypisz wartość **10**.
2. Wypisz wartość zmiennej **n**.
3. Zwiększ wartość **n** o 1 (**n := n + 1**).
4. Jeśli **n < 100**, wróć do kroku 2.
5. Zakończ algorytm.

Przykładowe rozwiązanie Ćwiczenia 2: Schemat blokowy (Warunek trójkąta)

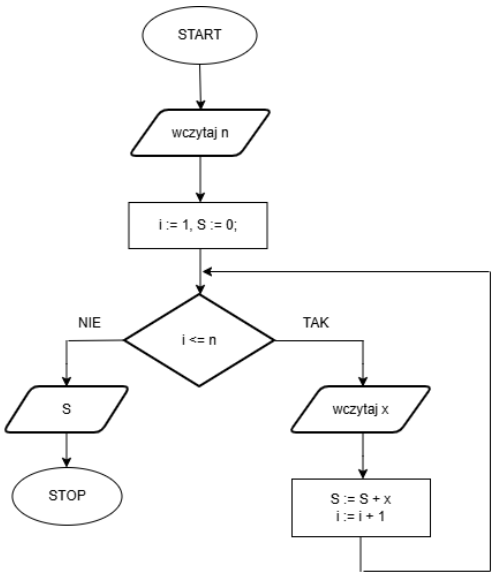
Należało sprawdzić, czy z odcinków a, b, c można zbudować trójkąt. Aby było to możliwe, s*uma każdych dwóch boków musi być większa od trzeciego boku (warunek trójkąta).*

W poniższym schemacie użyto pojedynczej skrzynki warunkowej ale można też użyć trzech skrzynek z pojedynczym warunkiem:



Rozwiązanie Ćwiczenia 3: Schemat blokowy (Suma n liczb podanych przez użytkownika)

Użytkownik najpierw podaje n, a potem n razy podaje konkretne liczby do zsumowania.



Rozwiązanie Ćwiczenia 4: Analiza funkcji S(n)

Co realizuje funkcja S(n) dla danych np. n = 123?

Iteracja	n	n mod 10 (reszta)	S (suma)	n div 10 (nowe n)
Start	123	—	0	123
1	123	**3**	**3** (0+3)	12
2	12	**2**	**5** (3+2)	1
3	1	**1**	**6** (5+1)	0
Koniec	0	—	**Zwróć 6**	—

Funkcja S(n) realizuje **obliczanie sumy cyfr podanej liczby**. Dla liczby 123 wynik wynosi 6.

Rozwiązanie Ćwiczenia 5: Zliczanie w zagnieżdżonej pętli

Aby poprawnie rozwiązać to zadanie, należy zauważyć, że pętla wewnętrzna (po zmiennej sterującej j) zależy od aktualnej wartości pętli zewnętrznej (po i).

Obieg (i)	Obieg (j)	Operacja	r (wynik)
Start	—	—	0
i = 1	j = 1	0 + 1	**1**
i = 2	j = 1	1 + 1	2
	j = 2	2 + 2	**4**
i = 3	j = 1	4 + 1	5
	j = 2	5 + 2	7
	j = 3	7 + 3	**10**

Wynik dla n = 3 wynosi 10.