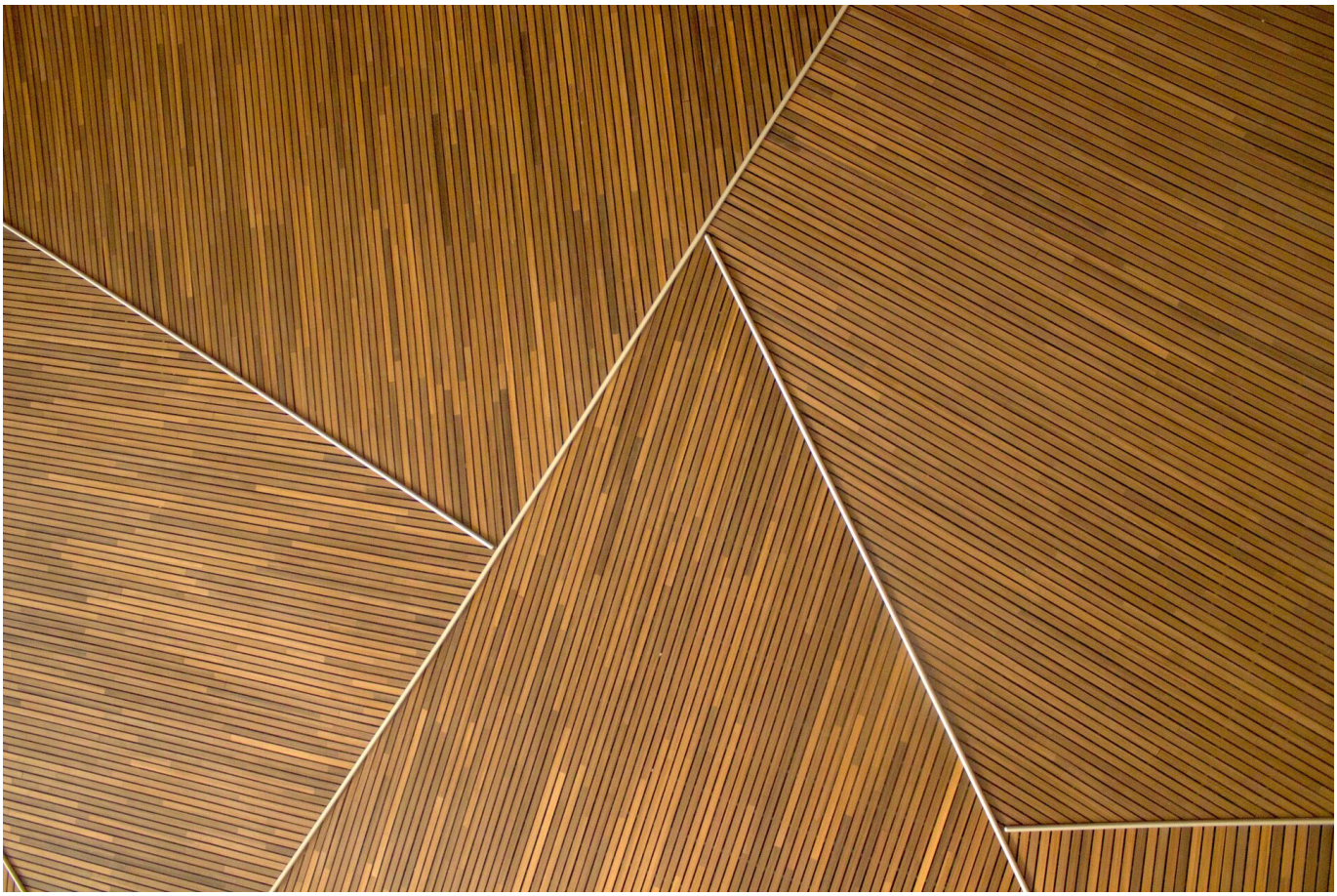
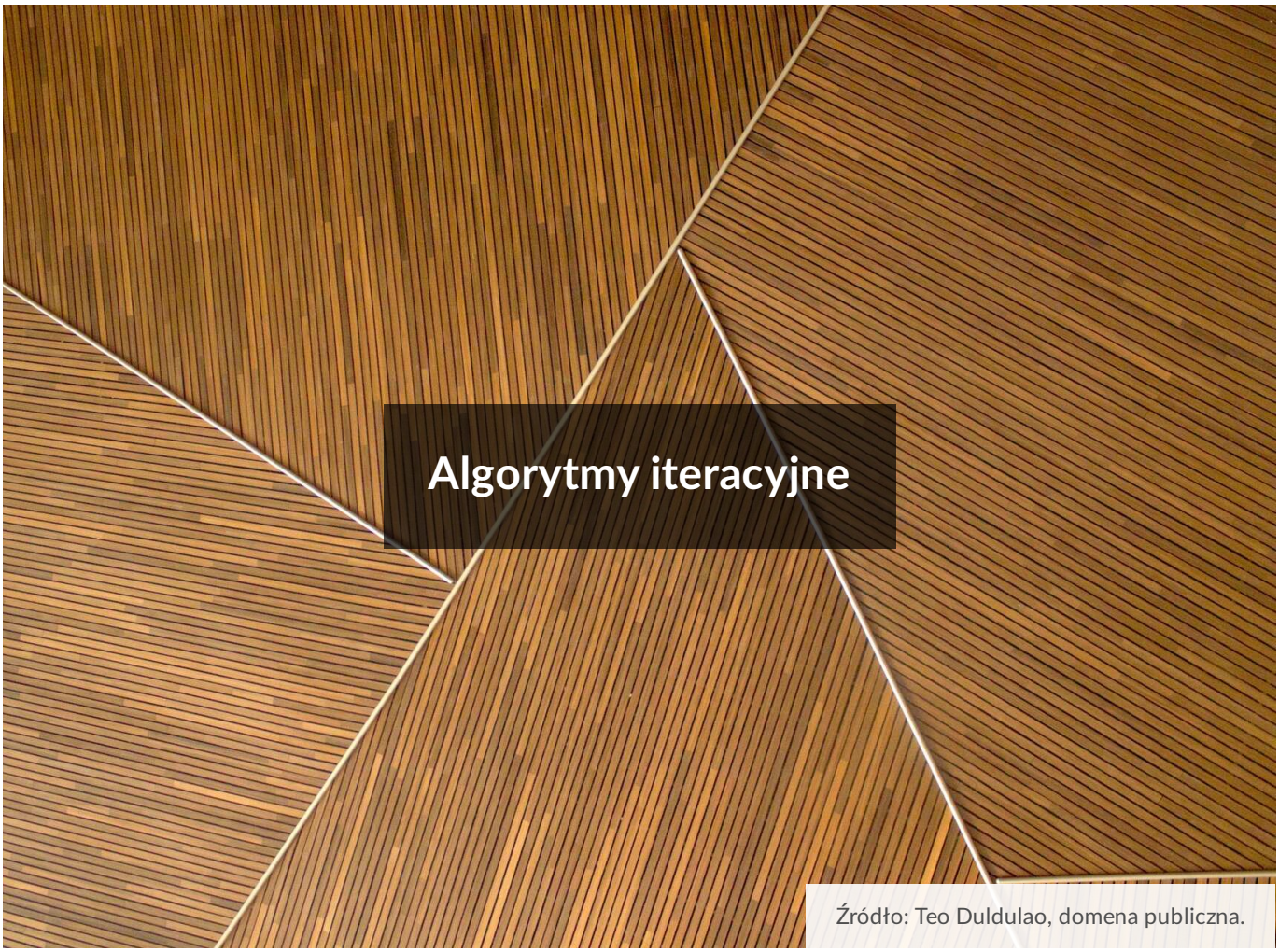


# Algorytmy iteracyjne




Źródło: Teo Duldulao, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Iteracja polega na wielokrotnym wykonywaniu tych samych czynności – np. powtarzaniu zapisanych w programie instrukcji – do momentu osiągnięcia określonego celu. Można opisać ją słowami: „dopóki spełniony jest pewien warunek, wykonuj instrukcję” lub: „na każdym elemencie zbioru wykonaj pewną operację”. Np. „każdemu produktowi w koszyku nalicz 10% zniżki”, „każdemu uczniowi w klasie III wyślij wiadomość”, „oblicz sumę wszystkich liczb całkowitych z przedziału $\langle 1, 100 \rangle$”.

Mechanizm iteracji to podstawowa operacja wykorzystywana w programowaniu. Odpowiednio użyty, pozwala skrócić kod programu i uczynić go bardziej czytelnym.

Implementacje algorytmów iteracyjnych znajdziesz w e-materiałach:

- Algorytmy iteracyjne w języku C++,
- Algorytmy iteracyjne w języku Java,
- Algorytmy iteracyjne w języku Python.

Więcej zadań? Sięgnij do Algorytmy iteracyjne – zadania maturalne.

> **Twoje cele**

- Wyjaśnisz, kiedy należy zastosować mechanizm iteracji.
- Rozwiążesz problemy obliczeniowe, wykorzystując mechanizm iteracji.
- Przeanalizujesz schematy blokowe algorytmów iteracyjnych.

# Przeczytaj

## Iteracja i pętle

Mamy za zadanie obliczyć sumę liczb parzystych z przedziału $\langle a,\ b \rangle$, gdzie `a` i `b` są liczbami całkowitymi. Zgodnie z poleceniem w tym przedziale będziemy rozważać tylko liczby całkowite. Jest to zadanie wymagające zastosowania iteracji, czyli wielokrotnego wykonania tych samych czynności.

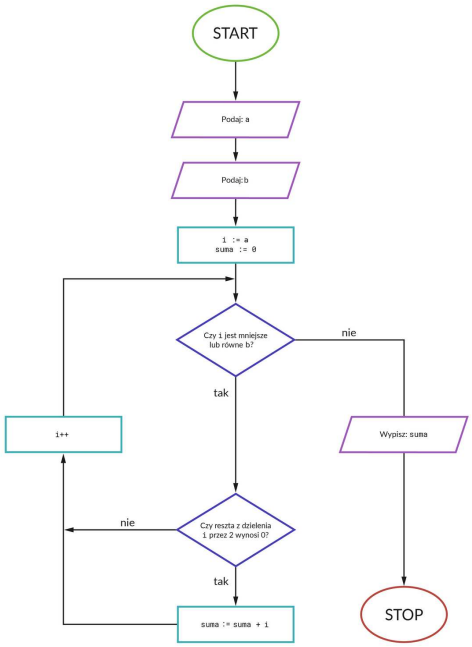
> **Problem 1**
>
> Przygotuj i omów algorytm, który pobierze od użytkownika dwie liczby całkowite `a` oraz `b`, oznaczające początek i koniec przedziału $\langle a,\ b \rangle$, a następnie obliczy sumę liczb parzystych z zadanego zakresu.
>
> **Specyfikacja problemu:**
>
> *Dane:*
>
> - `a`, `b` – liczby całkowite; kolejno: lewy i prawy kraniec przedziału, $a \leqslant b$
>
> *Wynik:*
>
> Program wypisuje sumę liczb parzystych z przedziału $\langle a,\ b \rangle$.

Aby rozwiązać przedstawiony problem, należy dla każdej liczby z przedziału $\langle a,\ b \rangle$ sprawdzić, czy jest ona parzysta, a jeżeli tak, dodać ją do wartości zmiennej `suma` przechowującej sumę kolejnych liczb parzystych. Te czynności trzeba wykonać wielokrotnie, w zależności od tego, ile liczb znajduje się w przedziale $\langle a,\ b \rangle$. Algorytm przedstawimy w postaci schematu blokowego.

W schemacie zastosujemy pętlę, czyli będziemy wielokrotnie wykonywać te same zestawy instrukcji. W naszym przypadku pętla będzie wykonywać się, dopóki `i` będzie mniejsze lub równe `b`.

Zmiennej `i` przypisujemy wartość `a`, czyli wartość początku sprawdzanego przedziału. Przy każdym wykonaniu pętli do zmiennej `i` dodajemy liczbę `1`, aż do momentu, kiedy zmienna `i` będzie większa od `b` (czyli liczby będącej końcem przedziału).

Oto schemat blokowy algorytmu obliczającego i wypisującego sumę liczb parzystych z przedziału ⟨`a, b`⟩.



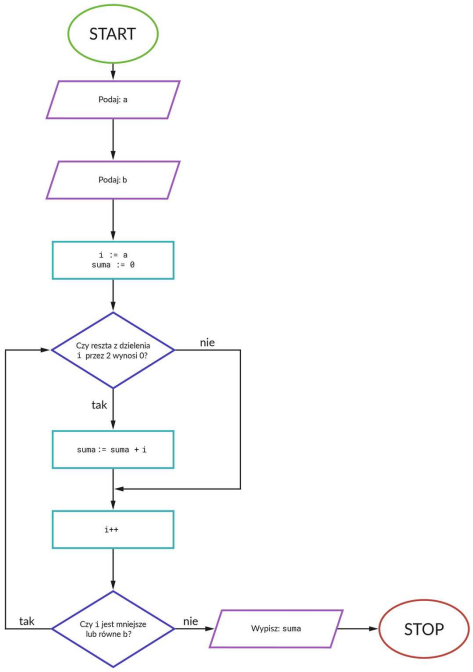

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Omówmy kolejne etapy algorytmu.

- Rozpoczynamy od wczytania dwóch wartości: `a` oraz `b`. Są to odpowiednio lewy i prawy koniec przedziału.
- Następnie inicjalizujemy zmienne `i` oraz `suma`. Pierwszej przypisujemy wartość `a`, drugiej zero. Zmienna `i` jest tzw. zmienną sterującą. Wykorzystujemy ją, aby określić, ile razy powinny być wykonane instrukcje zapisane wewnątrz pętli. Po wykonaniu całego bloku poleceń zwiększamy wartość `i` o 1. W rezultacie będziemy mogli zakończyć działanie pętli po dokładnie tylu cyklach, ile jest wymaganych do poprawnego rozwiązania problemu.
- Kolejnym etapem jest skonstruowanie warunku pętli. Instrukcje wewnątrz pętli będą wykonywać się, dopóki zmienna `i` będzie mniejsza lub równa `b`. Jeżeli okaże się, że ten warunek nie jest spełniony, możemy wypisać obliczoną sumę na ekranie i zakończyć program. Gdy zmienna `i` jest jednak mniejsza lub równa `b`, trzeba ponownie wykonać instrukcje wewnątrz pętli.
- Sprawdzamy parzystość liczby, czyli obliczamy resztę z dzielenia zmiennej `i` przez 2. Jeżeli reszta wynosi 0 (czyli w sytuacji, gdy liczba jest parzysta), dodajemy wartość zmiennej `i` do wartości zmiennej `suma`. Następnie dokonujemy inkrementacji zmiennej `i`.
- Wracamy na początek pętli i ponownie sprawdzamy warunek. Przedstawione operacje będą wykonywane `b - a + 1` razy. W momencie, gdy zmienna `i` osiągnie wartość

`b + 1`, warunek wykonania pętli nie będzie już spełniony. Na ekranie zostanie wyświetlona suma liczb parzystych; jest to ostania czynność opisana w algorytmie.

Spróbujmy zmodyfikować schemat blokowy. Warunek decydujący o wykonaniu kolejnego cyklu pętli lub jej zakończeniu umieścimy na końcu sekwencji poleceń:




Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Zwróćmy uwagę, że niezależnie od wartości zmiennej sterującej `i` oraz budowy warunku, operacje zapisane wewnątrz pętli zostaną wykonane co najmniej jeden raz. Wynika to ze zmiany miejsca pojawienia się warunku pętli („czy zmienna `i` jest mniejsza lub równa `b`?"). Wartość logiczna takiego wyrażenia jest obecnie sprawdzana dopiero po wykonaniu całego bloku instrukcji.

Gdybyśmy wykorzystali zmodyfikowany algorytm do rozwiązania zadania przedstawionego na wstępie, okazałby się on równie skuteczny jak poprzednia wersja. Zestaw instrukcji zapisanych wewnątrz pętli zostałby wykonany tyle samo razy – gdy zmienna `i` przyjęłaby wartość większą od `b`, na ekranie pojawiłaby się suma liczb parzystych, a pętla zostałaby przerwana. Poprawność obu sposobów rozwiązania wynika z założenia, że $a \leqslant b$, czyli że w przedziale znajduje się co najmniej jedna liczba.

Należy jednak zaznaczyć, że w pewnych przypadkach lepiej jest zastosować algorytm zgodny z pierwszym schematem blokowym, zaś w innych – użyć wariantu drugiego.

## Zoptymalizowana wersja algorytmu

Przeanalizujmy algorytm ponownie i spróbujmy rozwiązać go innym sposobem.

**Problem 2**

Przygotuj i omów algorytm, który pobierze od użytkownika dwie liczby całkowite a oraz b, oznaczające początek i koniec przedziału $\langle a, b \rangle$, a następnie obliczy sumę liczb parzystych z zadanego zakresu. Zmodyfikuj przedstawiony wcześniej algorytm tak, aby wyeliminować konieczność każdorazowego sprawdzania parzystości liczby.

**Specyfikacja problemu:**

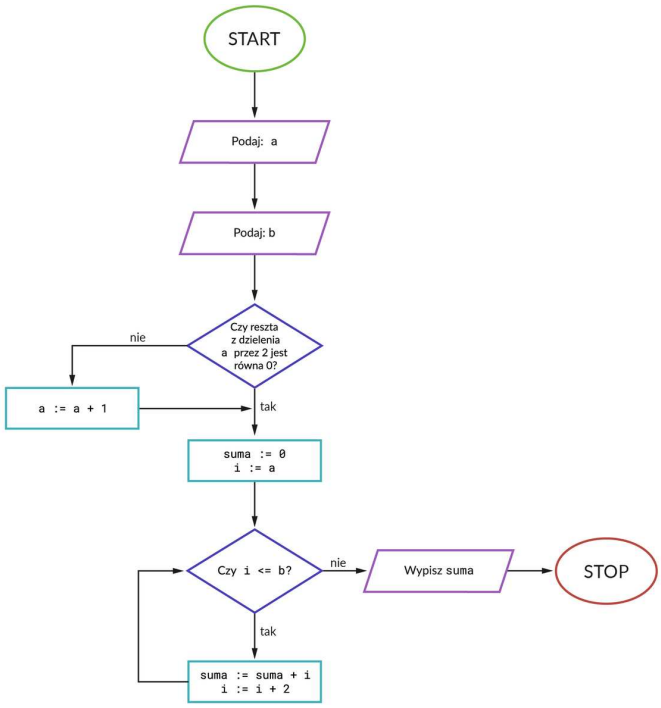
*Dane:*

- a, b – liczby całkowite; kolejno: lewy i prawy kraniec przedziału, $a \leqslant b$

*Wynik:*

Program wypisuje sumę liczb parzystych z przedziału $\langle a, b \rangle$.

Zauważmy, że liczby parzyste zawsze różnią się o 2, więc możemy wyeliminować konieczność sprawdzania podzielności – wystarczy, że iterator będzie zmieniał się o 2. Tak zmodyfikowany algorytm prezentuje się następująco:



Zwróćmy uwagę na konieczność upewnienia się, że początek sprawdzanego przedziału jest liczbą parzystą – jeżeli tak nie jest, dodajemy do niego jeden. Możemy to zrobić, ponieważ nieparzysta liczba i tak nie zostałaby dodana do sumy.

## Suma liczb parzystych jako suma ciągu arytmetycznego

> **Problem 3**
>
> Przygotuj i omów algorytm, który pobierze od użytkownika dwie liczby całkowite a oraz b, oznaczające początek i koniec przedziału $\langle \mathsf{a}, \mathsf{b} \rangle$, a następnie obliczy sumę liczb parzystych z zadanego zakresu. W swoim rozwiązaniu skorzystaj ze wzoru na sumę ciągu arytmetycznego.
>
> **Specyfikacja problemu:**
>
> *Dane:*
>
> - a, b – liczby całkowite; kolejno: lewy i prawy kraniec przedziału, $\mathsf{a} \leqslant \mathsf{b}$
>
> *Wynik:*
>
> Program wypisuje sumę liczb parzystych z przedziału $\langle \mathsf{a}, \mathsf{b} \rangle$.

Ponieważ kolejne liczby parzyste różnią się od siebie o 2, jeszcze skuteczniejszym rozwiązaniem byłoby skorzystanie ze wzoru na sumę elementów ciągu arytmetycznego. Spójrzmy na jego postać ogólną:

$$S_n = \frac{a_1 + a_n}{2} \cdot n,$$

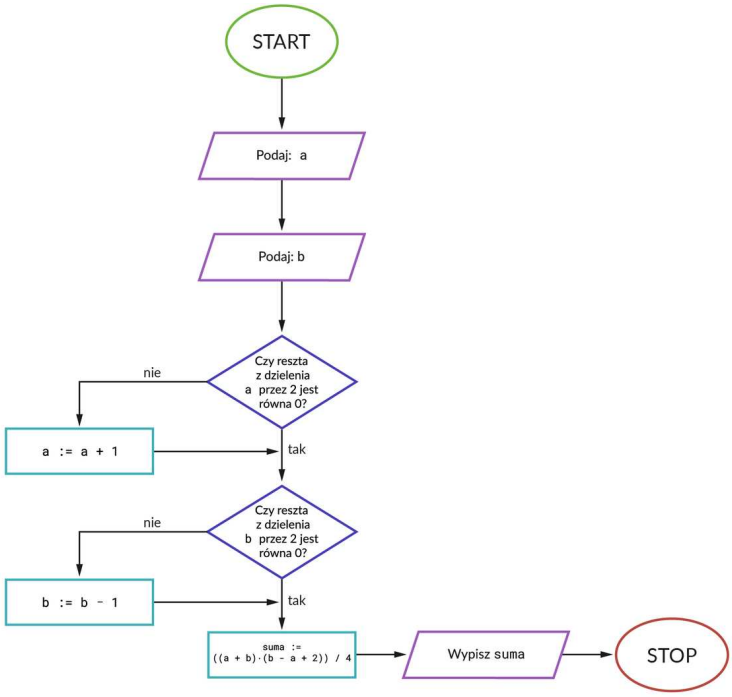
gdzie $n$ to liczba elementów, które chcemy zsumować, zaś $a_1, a_n$ to pierwszy oraz ostatni element tego ciągu w zadanym przedziale.

W jaki sposób wyznaczyć $n$? Po raz kolejny skorzystamy z faktu, że sumujemy tylko liczby parzyste. Możemy zmodyfikować wartość lewego i prawego krańca przedziału (czyli a i b) – w przypadku nieparzystego a zwiększyć o 1, a w przypadku nieparzystego b zmniejszyć o 1. Liczbę wszystkich liczb całkowitych w przedziale możemy zapisać jako `b - a + 1`. Jeżeli ta wartość byłaby parzysta, liczby parzyste stanowiłyby dokładnie połowę jej elementów – jednak zawsze będzie ona nieparzysta, gdyż jest to wynik odejmowania dwóch liczb parzystych zwiększony o 1. Ponieważ oba krańce przedziału to liczby parzyste, liczb parzystych w przedziale będzie więcej niż nieparzystych. Oznacza to, że liczb parzystych w przedziale będzie $\frac{b-a+2}{2}$.

To wszystko daje nam następujący wzór pozwalający obliczyć sumę liczb parzystych w zadanym przedziale:

$$S_a^b = \frac{(a+b)(b-a+2)}{4}.$$

Algorytm wykorzystujący ten wzór prezentuje się następująco:




Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

## Wyszukiwanie minimum w zbiorze liczb

Przeanalizujmy teraz nieco trudniejszy przykład. Załóżmy, że w zbiorze liczb podanych przez użytkownika chcemy wskazać liczbę najmniejszą.

**Problem 4**

Przygotuj i omów algorytm, który pobierze od użytkownika n liczb i wyłoni spośród nich najmniejszą.

**Specyfikacja problemu:**

*Dane:*

- n – liczba naturalna; liczba elementów wprowadzanych przez użytkownika; $n \geqslant 1$
- $x_1$, $x_2$, ..., $x_n$ – liczby całkowite podawane przez użytkownika

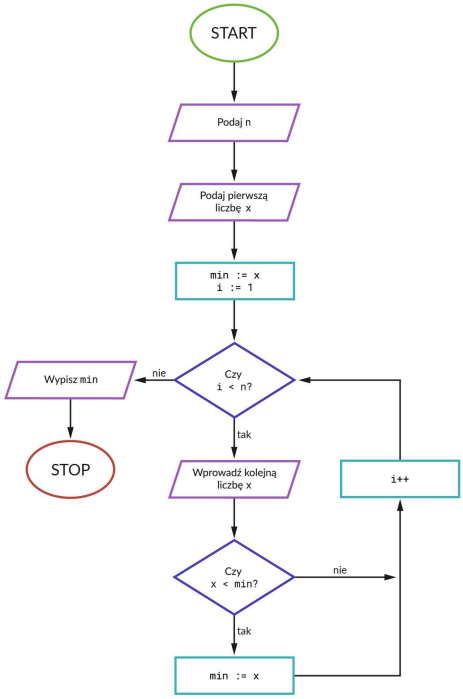
*Wynik:*

Program wypisuje najmniejszą spośród liczb podanych przez użytkownika.

Algorytm znajdowania minimum (lub maksimum) ma zastosowanie nie tylko w problemach matematycznych, ale i w codziennym życiu – w ten sposób szukamy, np. sklepu z najniższą ceną interesującego nas produktu.

Aby rozwiązać problem, trzeba zacząć od wskazania dowolnego elementu jako najmniejszego. Następnie porównujemy go z kolejnym. Jeśli drugi element okaże się mniejszy, to on zaczyna być traktowany jako minimum. Kolejnym krokiem jest porównanie elementu aktualnie najmniejszego z trzecim elementem – i tak aż do końca badanego ciągu.

W algorytmie użyjemy zmiennych `n`, `min` oraz `x`. Zmienna `n` będzie liczbą danych, a zmienna `x` będzie przechowywać kolejne wprowadzone przez użytkownika wartości. Zmienna `min` posłuży do przechowywania liczby, która w danym momencie jest najmniejsza.




# Słownik

**dekrementacja**

zmniejszanie wartości zmiennej o jeden

**inkrementacja**

zwiększenie wartości zmiennej o jeden

**zmienna sterująca pętlą (iterator)**

zmienna tworzona i wykorzystywana do sterowania wykonaniem instrukcji iteracyjnej; zazwyczaj to zmienna typu porządkowego, np. liczba naturalna, całkowita lub wartość logiczna, aczkolwiek w niektórych językach, np. C++ i Java, typ zmiennej sterującej nie jest ograniczony

# Animacja

> **Polecenie 1**
>
> Zapoznaj się z animacją, a następnie wykonaj polecenie 2.


Film dostępny pod adresem /preview/resource/ROG3dMWobOBmp

Film nawiązujący do treści materiału: Schemat blokowy algorytmu iteracyjnego.

**Polecenie 2**

Korzystając z pseudokodu bądź wybranego języka programowania, utwórz odwrócony trójkąt składający się z symboli *. Pobierz od użytkownika wysokość h generowanego trójkąta.

**Specyfikacja problemu:**

*Dane:*

- h – wysokość generowanego trójkąta; liczba naturalna

*Wynik:*

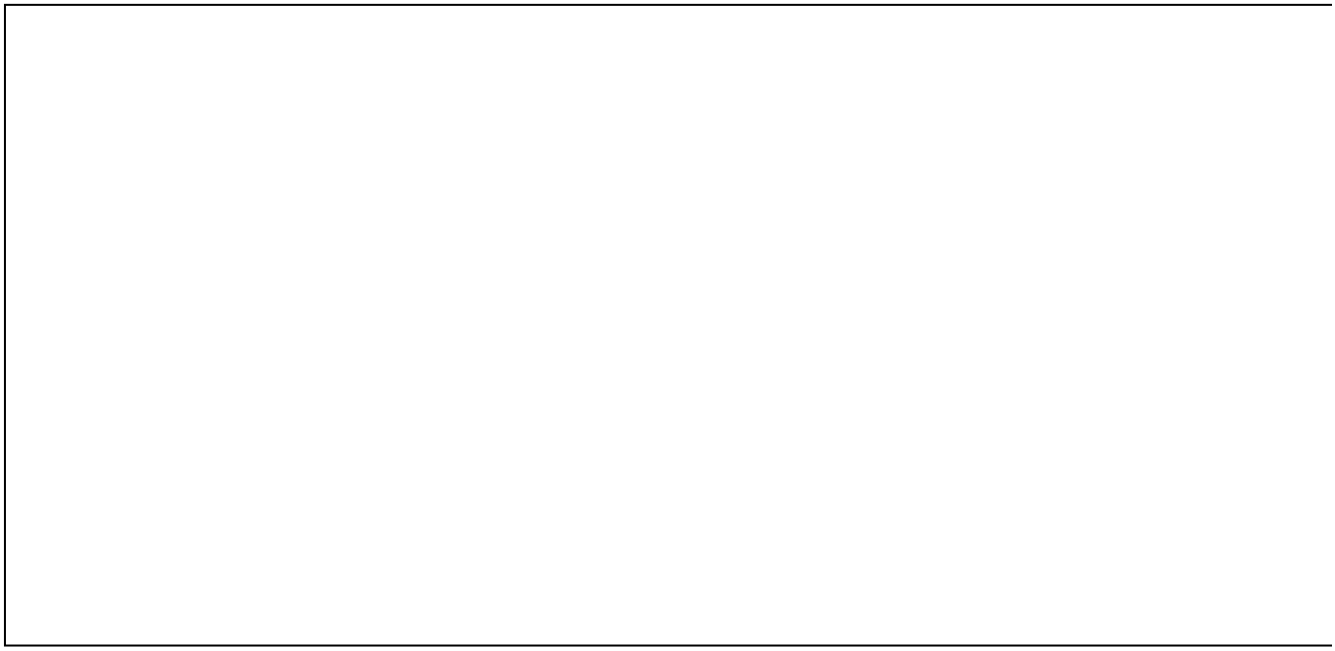
Algorytm generuje odwrócony trójkąt o zadanej wysokości h.

*Przykład:*

Rozwiązanie zadania dla h = 5.

```
*********
 *******
  *****
   ***
    *
```

```
1
```



**Polecenie 3**

Opracuj algorytm rozwiązujący następujący problem. Użytkownik podaje trzy liczby: $a$, $b$ oraz $r$.

Następnie z przedziału $\langle a, b)$ wybieramy liczby należące do ciągu arytmetycznego o różnicy $r$ i pierwszym wyrazie $a$ – czyli liczby $a$, $a + r$, $a + 2r$ itd. Obliczamy kwadraty kolejnych wyrazów ciągu i wypisujemy ich wartości tylko wtedy, gdy są one większe od zadanego $x$. Jednocześnie obliczamy sumę wypisanych elementów. Ją również wypisujemy. Algorytm zapisz za pomocą schematu blokowego lub wybranego języka programowania.

Rozwiązując problem, pamiętaj, by najpierw wypisać kwadraty liczb, a następnie podać ich sumę.

**Specyfikacja problemu:**

*Dane:*

- $a$, $b$ – kolejno: lewy i prawy kraniec przedziału, $a < b$; liczby całkowite
- $x$ – wartość porównywana z kwadratami kolejnych wyrazów ciągu; liczba naturalna
- $r$ – różnica ciągu; liczba całkowita

*Wynik:*

Algorytm wypisuje kwadraty liczb, które są większe od $x$, a następnie sumę tych kwadratów.

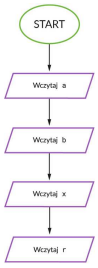
**Polecenie 4**

Porównaj swoje rozwiązanie z przedstawionym w prezentacji.

W ramach opracowywania algorytmu przygotujemy schemat blokowy. Rozpoczniemy

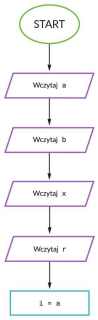
od realizacji operacji wypisywania kwadratów liczb spełniających wyróżnione w treści zadania warunki, a następnie uzupełnimy schemat o proces obliczania i drukowania sumy tych kwadratów.

2



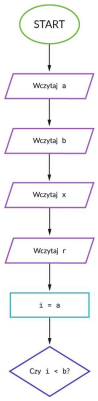

Budowanie schematu zaczynamy od narysowania bloku START. Dodamy też cztery bloki, które wczytają kolejno wartości: `a`, `b`, `x` oraz `r`.

3



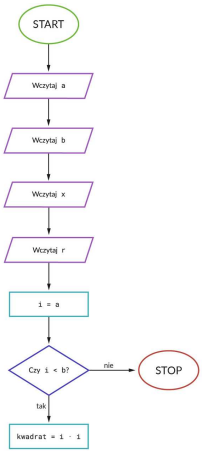

Następnie zmiennej sterującej `i` przypisujemy wartość `a`. Jest to pierwsza liczba należąca do badanego zbioru. Jeżeli jej kwadrat będzie większy niż `x`, wyświetlimy go na ekranie.

4



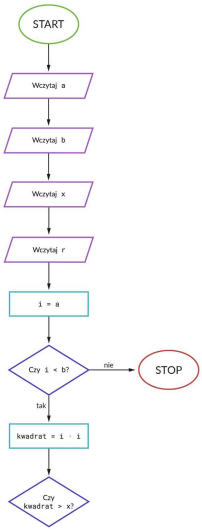

Ponieważ mamy podnosić do kwadratu liczby mniejsze niż b, dodajemy do schematu blok warunkowy, w którym sprawdzamy, czy zmienna sterująca jest mniejsza od b.

5



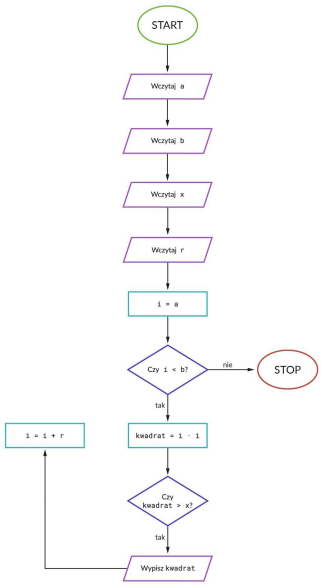

Gdyby okazało się, że warunek nie został spełniony (czyli liczba jest równa co najmniej b), kończymy algorytm. Natomiast w sytuacji, gdy warunek jest spełniony, obliczamy kwadrat liczby.

6



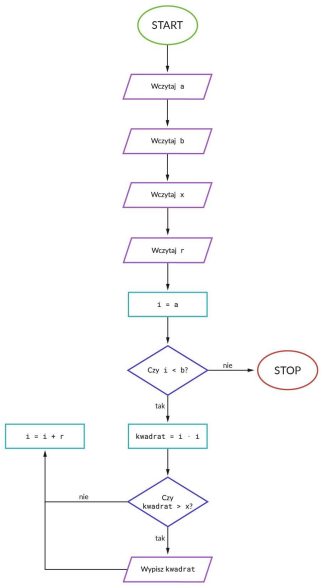

Musimy sprawdzić, czy obliczona wartość kwadratu jest większa niż x. Umieszczamy w schemacie blok warunkowy zawierający odpowiednie wyrażenie logiczne.

7



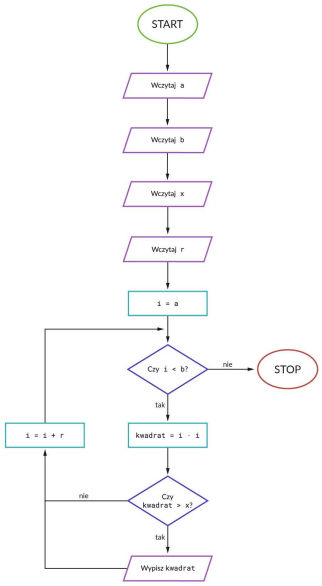

Jeżeli kwadrat liczby jest większy niż x, wypisujemy go na ekranie. Wartość zmiennej sterującej zwiększamy o r.

8




Gdy kwadrat liczby nie jest większy od `x`, również zwiększamy wartość zmiennej sterującej o `r`.

9



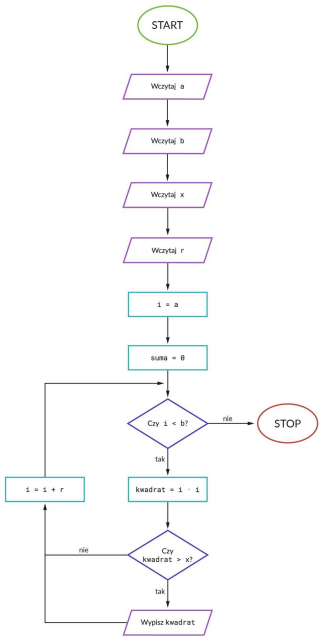

Wartość zmiennej `i` zwiększyła się o `r`. Ponownie sprawdzamy warunek pętli – upewniamy się, czy `i` jest mniejsze od `b`. Jeżeli warunek pętli okaże się prawdziwy, wówczas powtarzamy wszystkie operacje umieszczone w jej wnętrzu: wyliczenie nowej wartości zmiennej `kwadrat`, zbadanie, czy `kwadrat`

jest większy od `x` i ewentualne wypisanie wartości zmiennej `kwadrat`.

10

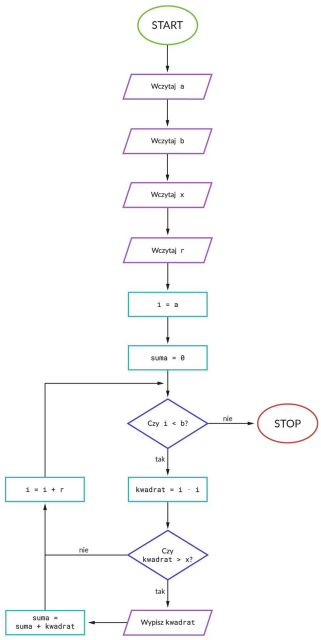
Schemat blokowy uzupełnimy o instrukcje, za pomocą których obliczymy wartość sumy wypisanych liczb.

11



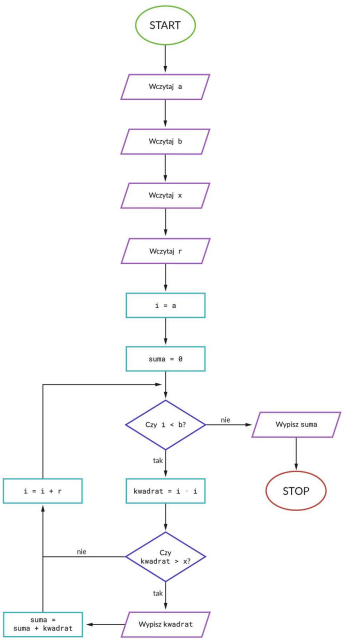

W schemacie za inicjalizacją zmiennej `i` wprowadzamy zmienną `suma`, której przypisujemy wartość 0.

12




Zgodnie ze specyfikacją chcemy obliczyć sumę wyłącznie tych kwadratów liczb, które są większe od x. Za blokiem wejścia/wyjścia wypisującym wartość zmiennej `kwadrat` dodajemy blok operacyjny, w którym zwiększymy wartość zmiennej `suma` o `kwadrat`.

13



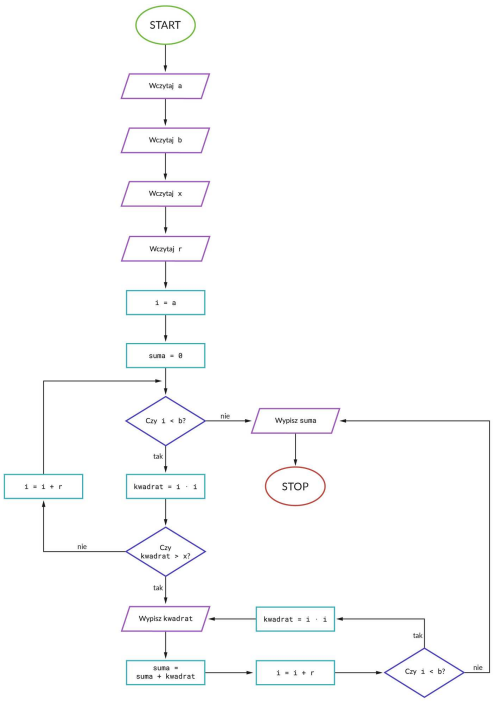

Ostatnią operacją przed zakończeniem działania algorytmu powinno być wypisanie wartości

zmiennej `suma`. Po opuszczeniu pętli dodajemy blok wejścia/wyjścia, w którym wypisujemy wartość zmiennej `suma`.

14

Możesz zauważyć, że po znalezieniu pierwszego kwadratu liczby, kwadrat każdej kolejnej liczby również będzie większy od `x`. Wówczas nie ma potrzeby sprawdzania warunku `Czy kwadrat > x?`. Wykorzystaj tę informację i spróbuj zmodyfikować schemat blokowy samodzielnie, w taki sposób, by nie wykonywał niepotrzebnych sprawdzeń.

15




Rozwiązanie zadania.

# Sprawdź się

Pokaż ćwiczenia:

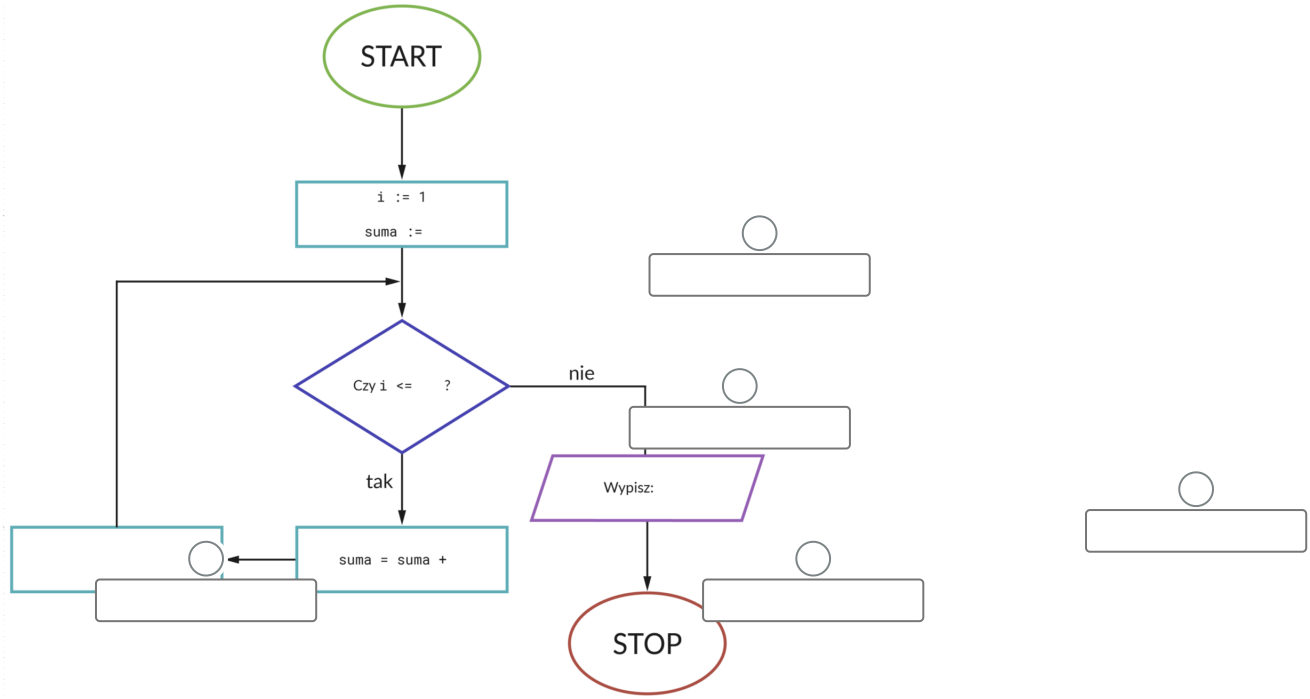
**Ćwiczenie 1**

Zaznacz wszystkie poprawne odpowiedzi. Wskaż, które zdania dotyczące iteracji są prawdziwe.

- [ ] Każdy problem można rozwiązać za pomocą algorytmu iteracyjnego.
- [ ] W iteracji można użyć zmiennej, która odpowiada za liczbę cykli wykonanych przez pętlę.
- [ ] Iteracja polega na wielokrotnym wykonywaniu ciągu instrukcji.
- [ ] Iteracja realizowana jest z użyciem pętli.

## Ćwiczenie 2

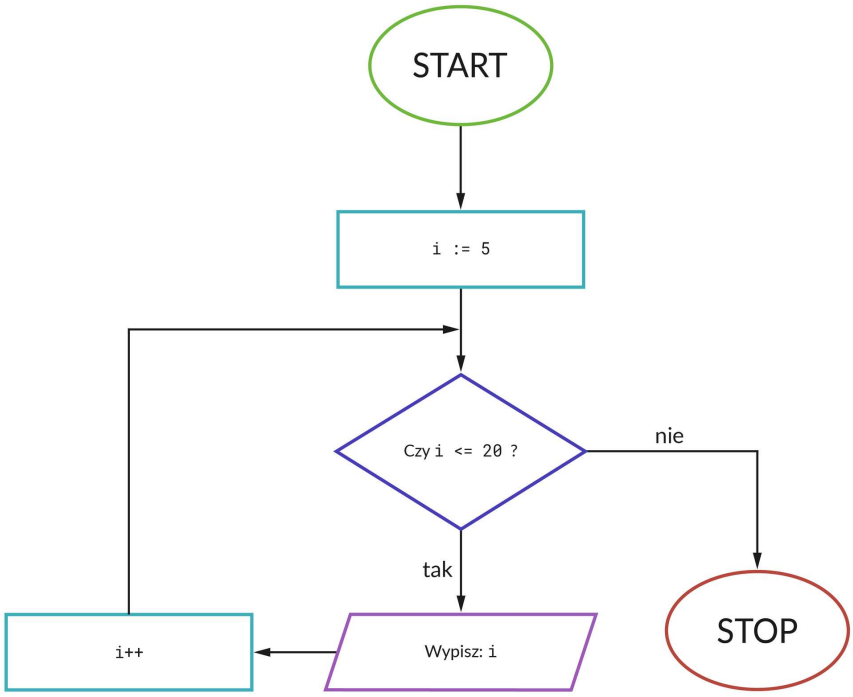
Uzupełnij schemat. Rysunek przedstawia niepełny schemat blokowy algorytmu obliczania sumy liczb całkowitych z zakresu od 1 do 25. Uzupełnij go odpowiednimi wartościami lub nazwami zmiennych.




`25` `i` `suma` `0` `i++`

## Ćwiczenie 3

Zapoznaj się ze schematem blokowym i wykonaj ćwiczenie.

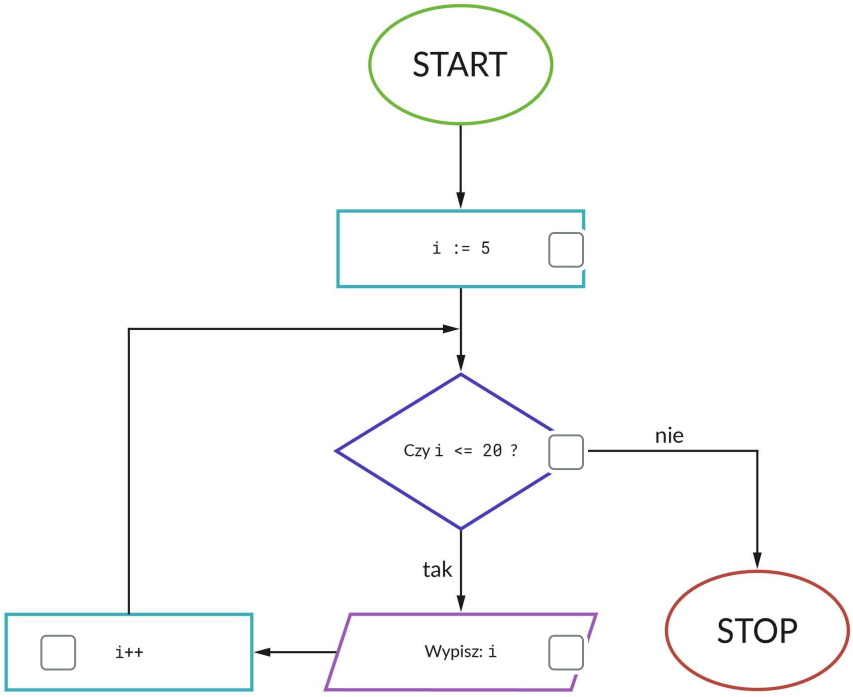



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Wyjaśnij, jaki będzie rezultat zastosowania algorytmu przedstawionego na schemacie blokowym.

## Ćwiczenie 4

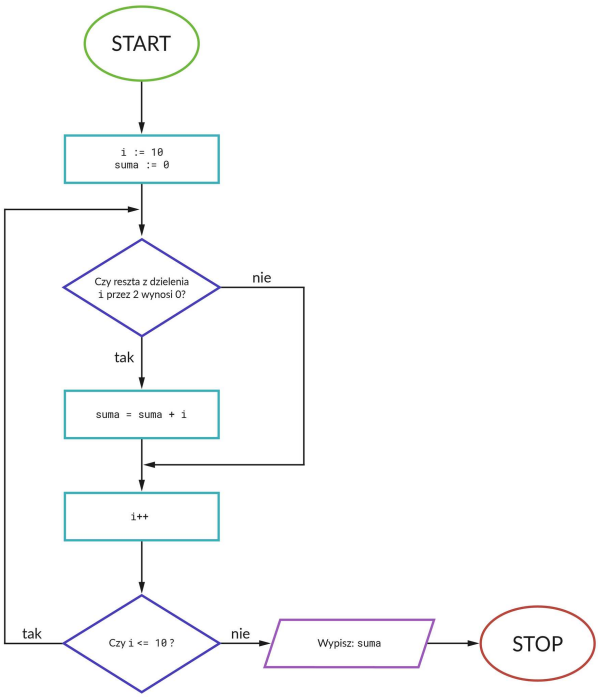
Wskaż na rysunku blok odpowiedzialny za inkrementację zmiennej sterującej.




Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

## Ćwiczenie 5

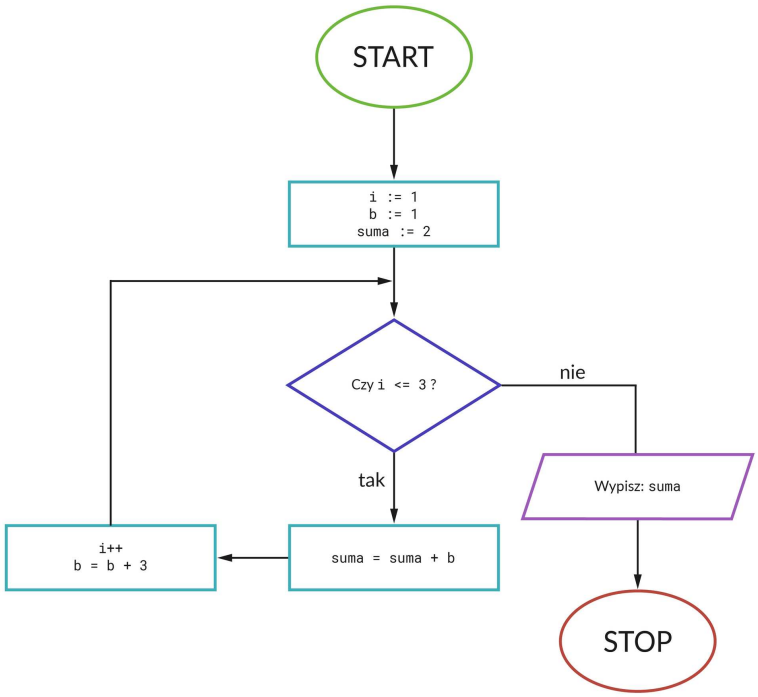
Wskaż, jaka wartość zmiennej `suma` zostanie wyświetlona na ekranie.




- ◯ 10
- ◯ 11
- ◯ 55
- ◯ 0
- ◯ 21

## Ćwiczenie 6

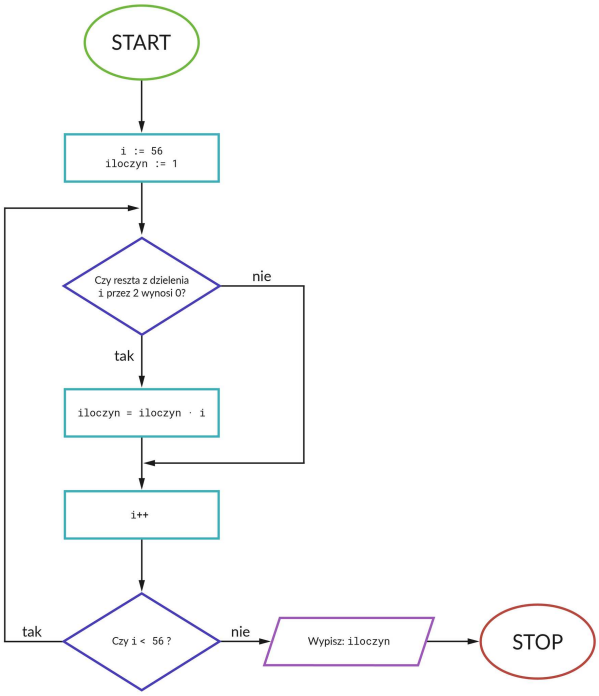
Zaznacz, jaka wartość zmiennej `suma` zostanie wyświetlona na ekranie.




- ◯ 8
- ◯ 11
- ◯ 7
- ◯ 14

## Ćwiczenie 7

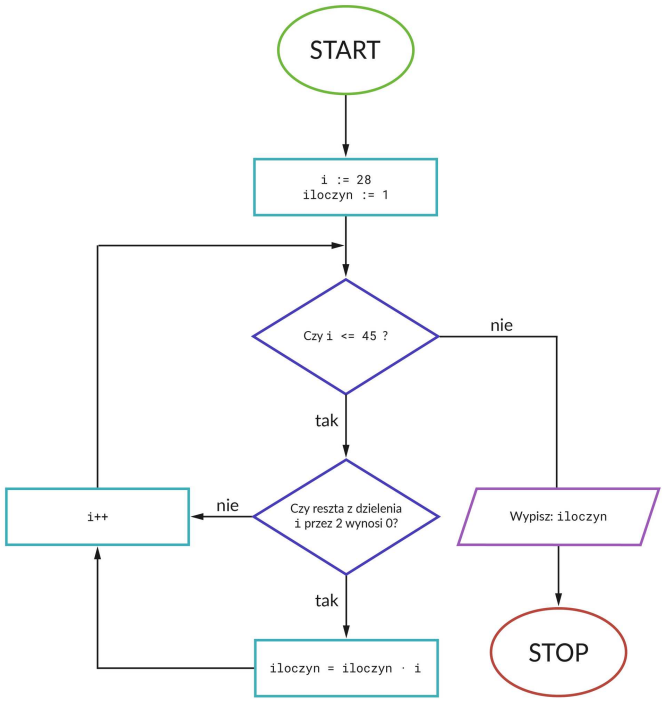
Wskaż, w jaki sposób zmodyfikować przedstawiony schemat blokowy, aby ciąg instrukcji w pętli nie wykonał się ani razu.




- ◯ przenieść wypisywanie na ekran zmiennej na początek ciągu instrukcji
- ◯ przenieść warunek pętli na koniec ciągu instrukcji
- ◯ przenieść inkrementację zmiennej sterującej na początek ciągu instrukcji
- ◯ przenieść warunek pętli na początek ciągu instrukcji

## Ćwiczenie 8

Wskaż, ile razy zostanie wykonana instrukcja `iloczyn = iloczyn · i`.




- ◯ 18
- ◯ 17
- ◯ 8
- ◯ 9

**Ćwiczenie 9**

Pewien ciąg opisany jest wzorem:

$$a_0 = 0$$

$$a_n = 1 + 2 \cdot (a_{n-1})$$

Za pomocą schematu blokowego zapisz iteracyjny algorytm obliczania n-tego wyrazu ciągu.

Przetestuj działanie programu dla $n = 7$.

**Specyfikacja problemu:**

*Dane:*

- n – numer wyrazu ciągu; liczba naturalna

*Wynik:*

Program wypisuje wartość n-tego wyrazu ciągu.

# Dla nauczyciela

**Autor:** Maurycy Gast
**Przedmiot:** Informatyka

**Temat: Algorytmy iteracyjne**

**Grupa docelowa:**

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy

**Podstawa programowa:**

Cele kształcenia – wymagania ogólne

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

e) obliczania wartości elementów ciągu metodą iteracyjną i rekurencyjną, w tym wartości elementów ciągu Fibonacciego.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

**Kształtowane kompetencje kluczowe:**

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

**Cele operacyjne (językiem ucznia):**

- Wyjaśnisz, kiedy należy zastosować mechanizm iteracji.
- Rozwiążesz problemy obliczeniowe, wykorzystując mechanizm iteracji.
- Przeanalizujesz schematy blokowe algorytmów iteracyjnych.

**Strategie nauczania:**

- konstruktywizm;
- konektywizm.

**Metody i techniki nauczania:**

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

**Formy pracy:**

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

**Środki dydaktyczne:**

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio;
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji).

**Przebieg lekcji**

**Przed lekcją:**

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Algorytmy iteracyjne". Uczniowie mają zapoznać się z treściami w sekcji „Przeczytaj".
2. Chętny lub wybrany uczeń przygotowuje rozwiązanie polecenia 2 z sekcji „Animacja". Będzie pełnił rolę eksperta podczas zajęć.

**Faza wstępna:**

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

**Faza realizacyjna:**

1. **Praca z multimedium.** Uczniowie zapoznają się z treścią animacji. Następnie w parach rozwiązują polecenie 2. W kolejnym kroku na forum klasy omawiają swoje rozwiązania. Uczniowie zapoznają się indywidualnie z poleceniem 3 w sekcji „Animacja". Wykonują schemat blokowy, a następnie porównują go z rozwiązaniem przedstawionym w prezentacji multimedialnej. W trakcie pracy zapisują problemy i pytania. Po czym następuje dyskusja, podczas której uczeń-ekspert wyjaśnia niezrozumiałe kroki procedury.
2. **Ćwiczenie umiejętności.** Uczniowie wykonują ćwiczenia nr 1-8 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

**Faza podsumowująca:**

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".

**Praca domowa:**

1. Uczniowie wykonują ćwiczenie 9 z sekcji „Sprawdź się".

**Wskazówki metodyczne:**

- Uczniowie znający podstawy programowania mogą zapisywać rozwiązania problemów w języku programowania.