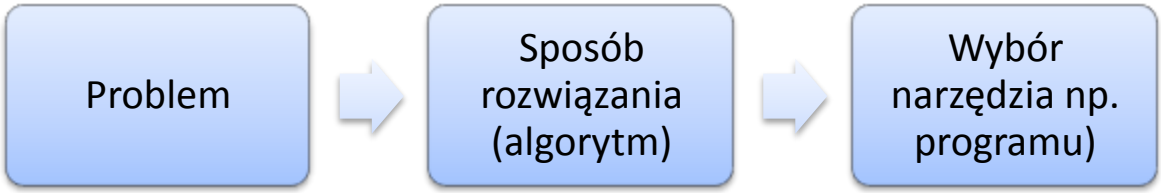
ALGORYTMY

Aby rozwiązać dowolny problem (zadanie), nie tylko z informatyki, trzeba go wcześniej poprawnie sformułować oraz ustalić dane i określić cel, czyli wyniki. Następnie należy zastanowić się nad sposobem rozwiązania, czyli algorytmem, a także wyborem odpowiedniego narzędzia, np. programu komputerowego, który to umożliwi.



Algorytm – to przepis, który podaje wszelkie czynności, jakie należy wykonać, by osiągnąć rozwiązanie określonego problemu w skończonej liczbie kroków. Niezależnie od sposobu dojścia do rozwiązania, opis można uznać za algorytm tylko wtedy, gdy przy dobrze określonych danych wejściowych jest jednoznaczny, skończony i posiada precyzyjnie określony punkt końcowy.
lub
Algorytm - to skończony, uporządkowany zbiór jasno zdefiniowanych czynności, koniecznych do wykonania pewnego zadania, w ograniczonej liczbie kroków. Algorytm ma za zadanie przekształcić zbiór danych *wejściowych* (dane podawane na wejściu algorytmu) na zbiór danych *wyjściowych* (wynik działania algorytmu).

Algorytmika - to dział informatyki zajmujący się poszukiwaniem, konstruowaniem i badaniem własności algorytmów, w kontekście ich przydatności do rozwiązywana problemów za pomocą komputerów

Skąd wywodzi się słowo algorytm?
Słowo algorytm wywodzi się od przydomka perskiego matematyka ABU JAFARA MUHAMMADAIBN MUSA AL-CHWARIZIMIEGO (żył w IX w.). Przydomek al-Chwarizmi w łacińskich dziełach tłumaczono jako ALGORITMI

Etapy budowania algorytmu (rozwiązywania problemu)
Konstruowania algorytmu opiera się na pewnym **schemacie postępowania**:

1. **Przedstawienie zadania** – należy przedstawić problem, jaki mamy rozwiązać.
2. **Określenie rodzaju wprowadzanych danych wejściowych** – podanie typu danych (liczby, znaki).
3. **Zdefiniowanie celu**, inaczej wyniku oraz sposobu jego prezentacji.
4. **Dobór metody wykonania zadania** – dobór odpowiedniej metody, najbardziej optymalnej.
5. **Zapisanie algorytmu za pomocą wybranej metody** – opisu słownego, listy kroków, schematu blokowego, języka programowania.
6. **Analiza bezbłędności rozwiązania**.
7. **Badanie rozwiązania** – algorytm powinien być uniwersalny i powinien służyć podczas rozwiązywania zadań dla różnych zestawów danych wejściowych.

8. **Opinia na temat wybranej metody** – sprawdzenie skuteczności działania algorytmu.

Cechy poprawnego algorytmu

W celu poprawnego skonstruowania algorytmu, na podstawie którego można stworzyć efektywnie działający program, należy przestrzegać określonych kryteriów. Są to:

- **efektywność** – dane wyjściowe algorytmu (wynik) powinien zamknąć się w skończonej liczbie kroków;
- **poprawność** - algorytm daje dobre wyniki, odzwierciedlające rzeczywistość;
- **jednoznaczność** - brak rozbieżności wyników przy takich samych danych, jednoznaczne opisanie każdego kroku;
- **uniwersalność** - algorytm powinien obejmować pewien zakres problemów, a nie wyłącznie jeden szczególny przypadek;
- **sprawność** - czasowa, czyli szybkość działania, i pamięciowa;
- **dyskretność** - jasno określone kroki w pojedynczym algorytmie, elementy bardziej złożone powinny być przedstawione w oddzielnym algorytmie.

Sposoby opisu algorytmów

Każdy algorytm, którym chcemy się podzielić, musi być podany w formie zrozumiałej dla innych ludzi. Najczęściej stosuje się:

- opis słowny,
- lista kroków,
- schemat blokowy,
- język programowania.

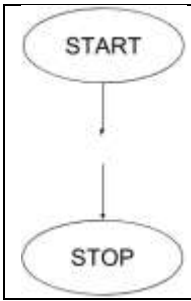
Opis słowny

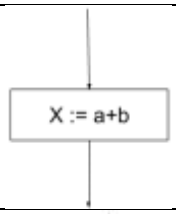
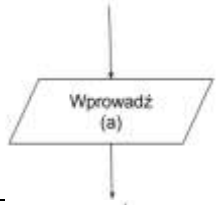
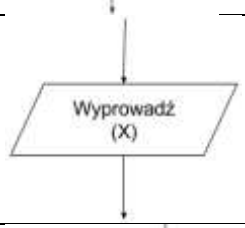
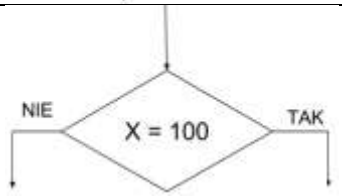
Opis słowny ma za zadanie przedstawić w sposób logiczny i zrozumiały przez użytkownika ciąg czynności, aby uzyskać zamierzony cel, np. recepta wykonania leku itp. Charakterystycznym elementem dla opisu (listy kroków) jest numeracja wierszy, w których opisuje się realizację poszczególnych czynności w kolejnych krokach.

Schemat blokowy - graficzna prezentacja algorytmu

Schemat blokowy - przedstawia algorytm w formie symboli graficznych, opisując szczegółowo wszystkie operacje arytmetyczne, logiczne, przesyłania, sterujące i pomocnicze wraz z kolejnością ich wykonywania.

Tego typu algorytm stanowi podstawę do napisania programu. Do zapisu schematu blokowego stosuje się następującą grupę symboli (bloków):

	**Bloki graniczne** - START (początek algorytmu), STOP (koniec algorytmu). Z bloku START wychodzi tylko jedno połączenie i żadne do niego nie wchodzi; w jednym schemacie musi być dokładnie jeden blok START. Blok z napisem STOP kończy algorytm. Nie ma połączenia wychodzącego. Każdy algorytm musi mieć co najmniej jeden blok STOP

	**Blok operacyjny (instrukcji)** - blok w którym wykonywane są różne operacje (pojedyncza bądź grupa operacji), m.in. obliczenia. W jednym bloku można wpisać więcej niż jedno wyrażenie. Do bloku operacyjnego wchodzi jedno połączenie i wychodzi także jedno połączenie.
	**Blok wejścia** - odpowiada za wprowadzanie danych. Ma jedno połączenie wchodzące i jedno wychodzące.
	**Blok wyjścia** - odpowiada za wyprowadzenie wyników lub komunikatów. Ma jedno połączeni wchodzące i jedno wychodzące.
	**Blok decyzyjny (warunkowy)** - blok podejmowania decyzji, wyboru konkretnej możliwości działania. Ma jedno połączenie wchodzące i dwa wychodzące: • z napisem TAK, gdy warunek jest spełniony; • z napisem NIE, gdy warunek nie jest spełniony.
	Połączenie - służy do łączenia bloków; określa kolejność wykonywanych działań lub kierunek przepływu danych
	**Blok kolekcyjny** - służy do łączenia różnych dróg algorytmu.

Uwaga!

W skrzynce warunkowej wpisujemy warunek logiczny, stosując następujące znaki:

=	równy, np. (a = 5)
< >	różny, np. (a < > 5)
<	mniejszy, np. (a < 5)
>	większy, np. (a > 5)
< =	mniejszy lub równy, np. (a < = 5)
> =	większy lub równy, np. (a > = 5)

Zasady przedstawiania algorytmów w postaci schematu blokowego

1. Operacje algorytmu należy umieszczać w odpowiednich blokach.
2. Każdy schemat blokowy ma jeden blok start, natomiast bloków zakończenia może być kilka.
3. Wszystkie bloki muszą być ze sobą połączone (nie może być przerwy w schemacie).
4. Każde połączenie jest zaczepione do danego bloku i dochodzi do następnego bloku lub innego połączenia.
5. Kolejność wykonywania operacji wyznaczają połączenia między blokami.
6. Do każdego bloku wchodzi jedno połączenie (oprócz bloku początku algorytmu) i jedno połączenie z niego wychodzi (oprócz bloku warunku, z którego wychodzą dwa połączenia, oraz bloku zakończenia algorytmu, z którego nie wychodzi żadne połączenie).

Języki programowania

Opis w języku programowania (program) stanowi realizację projektu w konkretnym języku programowania, powinien być poprzedzony opisem słownym i schematem blokowym.

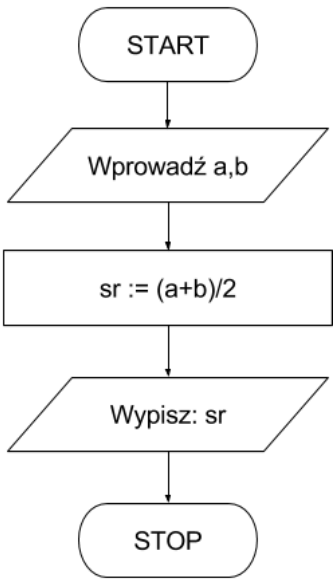
ANALIZA SCHEMATÓW BLOKOWYCH
Specyfikacja problemu algorytmicznego

Specyfikacja problemu algorytmicznego - polega na dokładnym opisie problemu, który ma zostać rozwiązany, oraz na podaniu danych wejściowych i wyjściowych wraz z ich typami

Zmienna - obiekt występujący w algorytmie, określony przez nazwę. Zmienna służy również do zapamiętywania pewnych nazw.

Zmienna pomocnicza - jest zmienną wprowadzoną do zapisu algorytmu w celu umożliwienia jego realizacji. Zmienne pomocnicze służą do pamiętania danych przejściowych, czyli danych potrzebnych do działania algorytmu, ale niebędących danymi wejściowymi ani wynikami.

Przykład:
Specyfikacja problemu algorytmicznego (schemat blokowy)
Problem algorytmiczny: obliczanie średniej arytmetycznej dwóch liczb rzeczywistych.
Dane wejściowe: a, b ϵ R
Dane wyjściowe: sr ϵ R - średnia liczb a, b



Specyfikacja problemu algorytmicznego (lista kroków)
Problem algorytmiczny: obliczanie średniej arytmetycznej dwóch liczb rzeczywistych.
Dane wejściowe: a, b ϵ R
Dane wyjściowe: sr ϵ R - średnia liczb a, b
Krok 1: Zacznij algorytm
Krok 2: Podaj wartość liczb rzeczywistych a i b
Krok 3: Oblicz średnią arytmetyczną dwóch liczb rzeczywistych sr := (a+b)/2
Krok 4: Wypisz wynik sr
Krok 5: Zakończ algorytm

RODZAJE ALGORYTMÓW

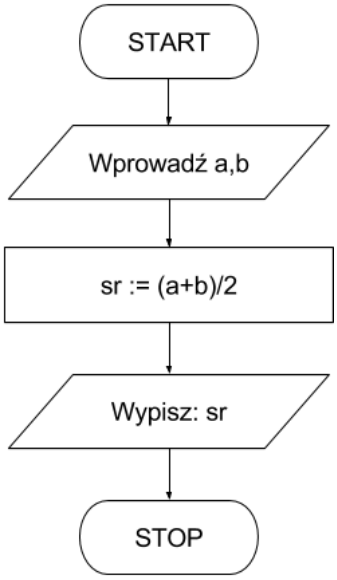
Rodzaje algorytmów pod względem budowy algorytmy dzielimy na:

- algorytmy liniowe;
- algorytmy rozgałęzione;
- algorytmy iteracyjne;
- algorytmy rekurencyjne.

algorytmy liniowe;

Algorytm liniowy (sekwencyjny) - ciąg kolejno wykonywanych instrukcji (działań), w taki sposób, w jaki zostały zapisane (sekwencyjnie).

Przykład: obliczanie średniej arytmetycznej dwóch liczb rzeczywistych.
Dane wejściowe: a, b ϵ R
Dane wyjściowe: sr ϵ R - średnia liczb a, b



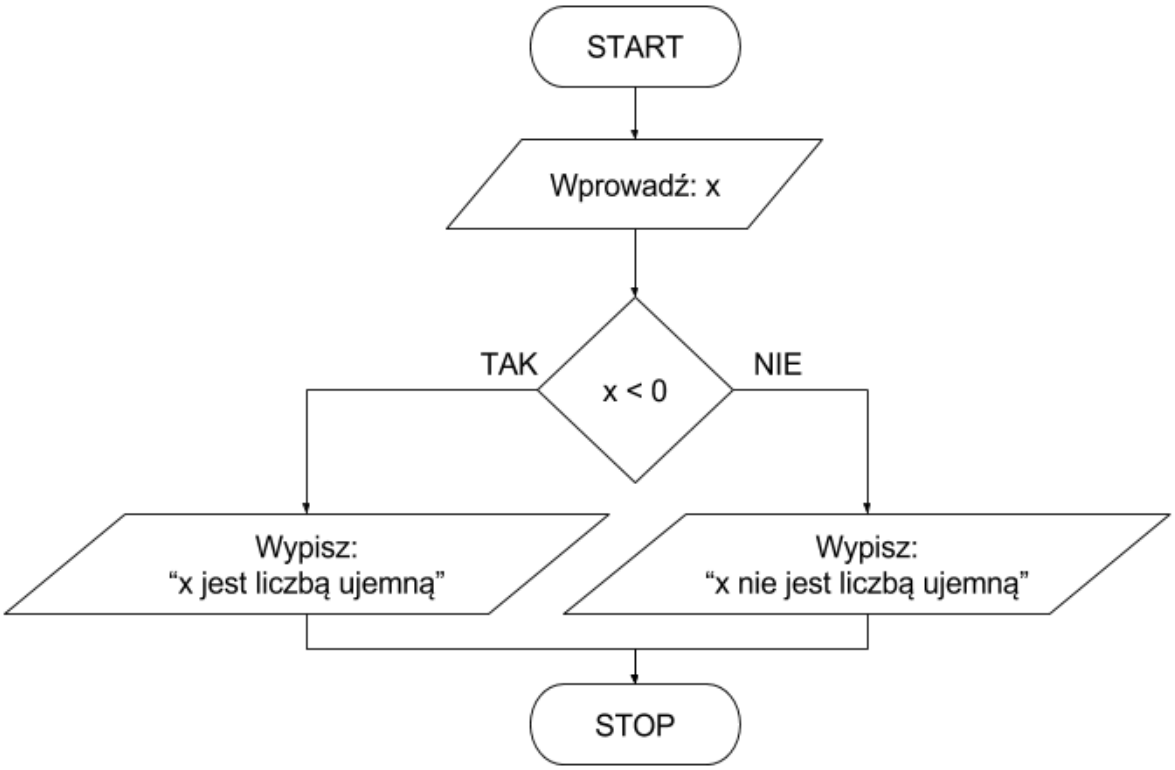
algorytmy rozgałęzione;

Algorytm rozgałęziony (warunkowy) - algorytm, w którym może wystąpić kilka alternatywnych ciągów działań; algorytm rozgałęziony sprawdza warunek i od jego spełnienia zależą dalsze czynności.

Instrukcja warunkowa - zapisywana jest najczęściej w postaci np. *jeżeli spełniony jest warunek **k**, to zrealizuj instrukcję **a**; lub jeżeli spełniony jest warunek **k**, to wykonaj instrukcję **a**, w przeciwnym wypadku wykonaj instrukcję **b**.*

Przykład: **Algorytm sprawdzenia, czy wpisana wartość jest liczbą ujemną.**
(lista kroków)
Dane wejściowe: dowolna liczba rzeczywista
Dane wyjściowe: wartość x
Krok 1: Rozpocznij algorytm
Krok 2: Wprowadź wartość liczby x
Krok 3: Sprawdź, czy x<0. Jeśli tak, to wypisz "x jest liczbą ujemną", w przeciwnym wypadku (gdy x>=0) wypisz "x nie jest liczbą ujemną"
Krok 4: Zakończ algorytm

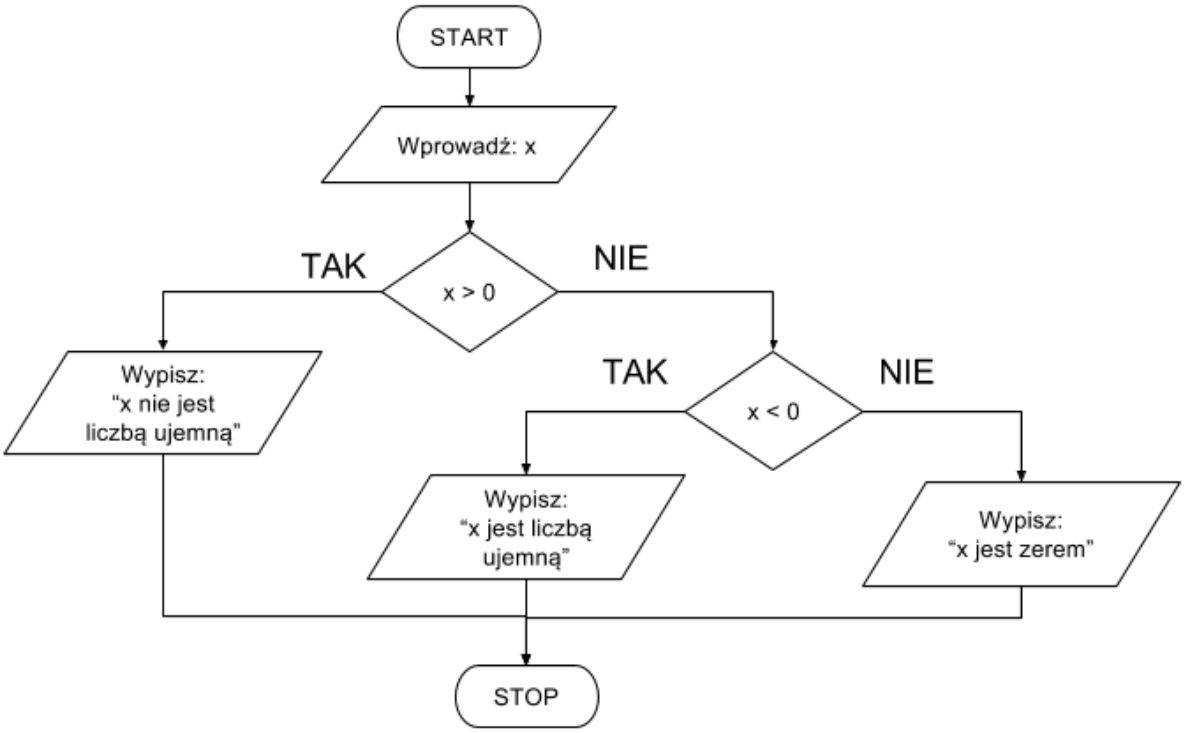
(schemat blokowy)
Dane wejściowe: dowolna liczba rzeczywista
Dane wyjściowe: wartość x



Algorytmy z warunkami zagnieżdżonymi - składają się z kilku warunków (po sprawdzeniu jednego warunku trzeba sprawdzić kolejne). W schemacie blokowym po wyjściu z jednego bloku warunku, np. drogą TAK, umieszczony jest kolejny blok warunkowy.

Przykład: **Algorytm sprawdzenia, czy wpisana wartość jest liczbą ujemną, dodatnią lub przyjmuje wartość zerową.**
(lista kroków)
Dane wejściowe: dowolna liczba rzeczywista
Dane wyjściowe: wartość x
Krok 1: Rozpocznij algorytm
Krok 2: Wprowadź wartość liczby x
Krok 3: Sprawdź, czy x<0. Jeśli tak, to wypisz "x jest liczbą ujemną", w przeciwnym wypadku sprawdź, czy x>0. Jeśli tak, to wypisz "x jest liczbą dodatnią", w przeciwnym wypadku wypisz "x jest zerem"
Krok 4: Zakończ algorytm

(schemat blokowy)
Dane wejściowe: dowolna liczba rzeczywista
Dane wyjściowe: wartość x (informacja, czy liczba jest ujemna, czy nie lub czy przyjmuje wartość 0)



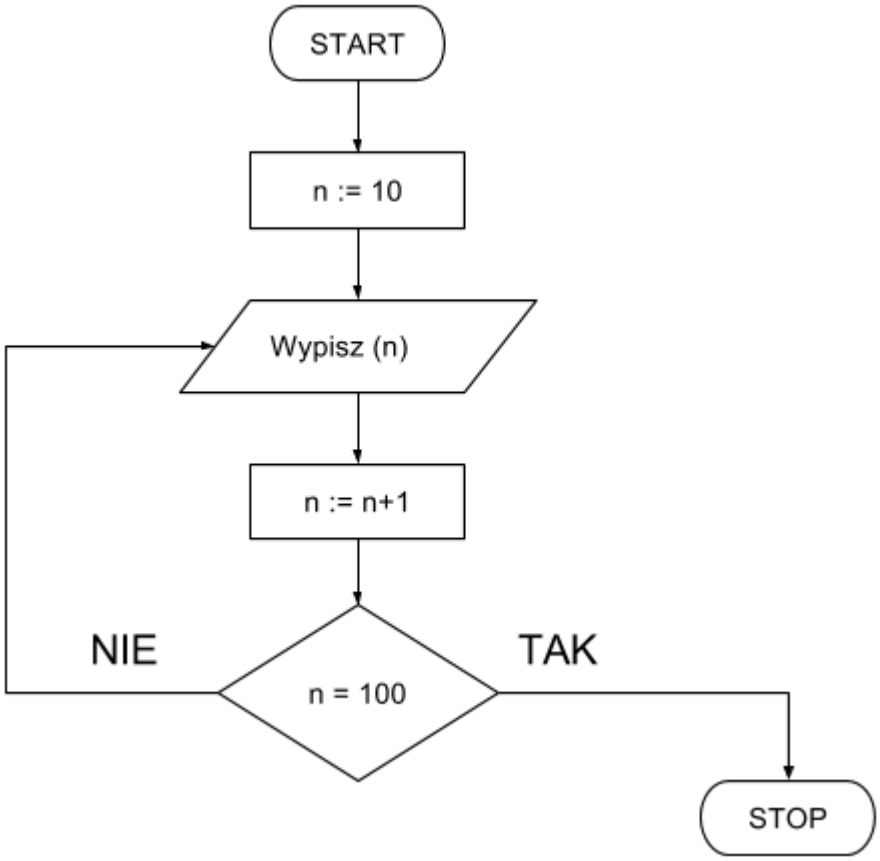
algorytmy iteracyjne;

Iteracja (pętla) - wielokrotne powtarzanie tego samego ciągu operacji.

Algorytm iteracyjny - algorytm, który uzyskuje wynik przez powtarzanie danej operacji określoną ilość razy. Operacje wielokrotnego wywołania instrukcji nazywamy pętlą. Liczba kroków iteracji może być podana z góry lub zależeć od warunku. Iterację (pętle) stosujemy w celu zapobiegania zapisywaniu powtarzających się bloków operacji.

Przykład: **Algorytm wypisujący wszystkie liczby dwucyfrowe**
(lista kroków)
Dane wejściowe: n - pierwsza liczba dwucyfrowa
Dane wyjściowe: wszystkie liczby dwucyfrowe
Krok 1: Rozpocznij algorytm
Krok 2: Zmiennej n przypisz wartość 10 (pierwsza liczba dwucyfrowa)
Krok 3: Wypisz n
Krok 4: Zwiększ n o jeden: n:=n+1
Krok 5: Jeśli n<100, to wróć do kroku numer 3
Krok 6: Zakończ algorytm

(schemat blokowy)



algorytmy rekurencyjne.

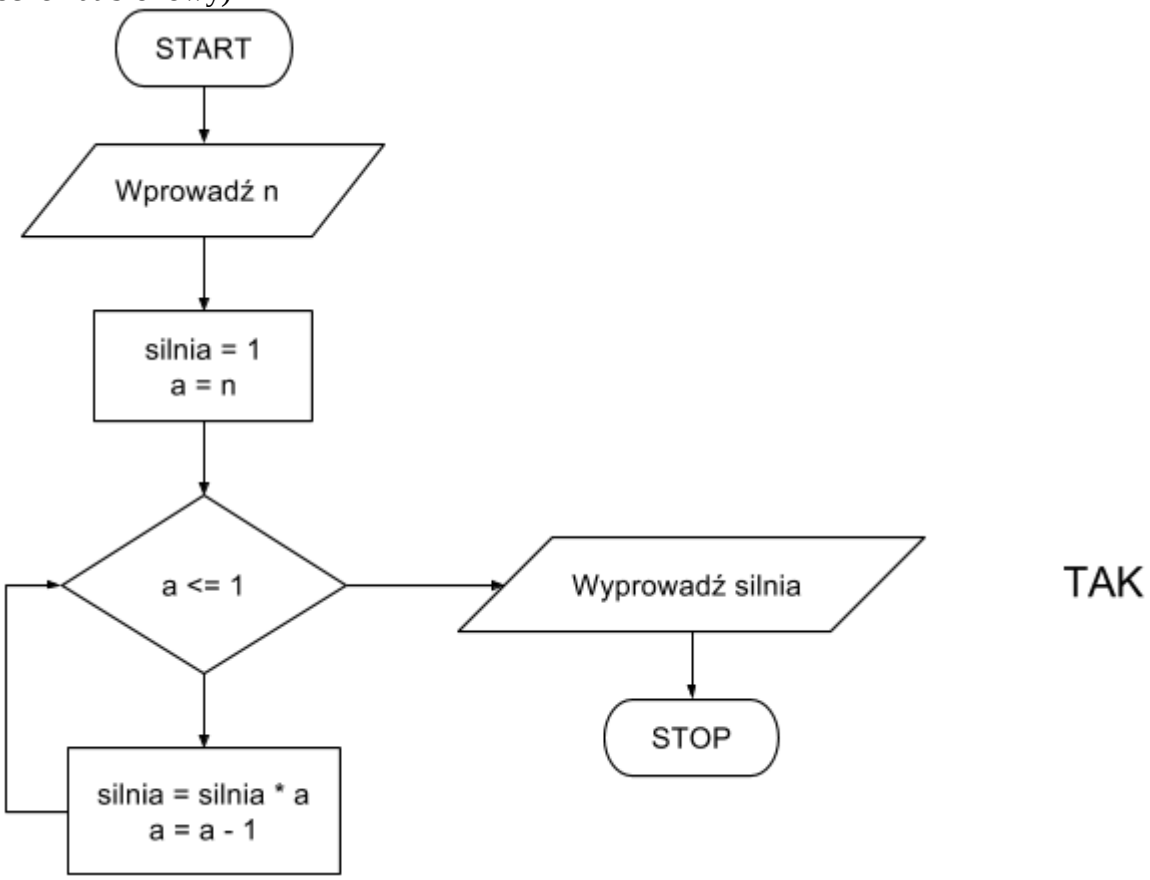
Rekurencja - sposób odwoływania się funkcji do siebie samej. Przy tworzeniu algorytmów rekurencyjnych należy zdefiniować metodę zakończenia wywołania rekurencji.

Technikę rekurencji można spotkać w codziennym życiu. Mając ustawione naprzeciw siebie dwa lustra, zauważmy, że obraz jednego lustra odbija się w drugim. Oznacza to, że obraz lustra stanowi część siebie samego.

Przykład. Algorytm obliczania silni.
Silnia liczby naturalnej n (oznaczenie - n!) jest iloczynem:
n! = 1*2*3*4*5*...*(n-1)*n dla n>0
n! = 1 dla n=0

(lista kroków)
Dane wejściowe: liczba naturalna n
Dane wyjściowe: wartość silni: silnia
Dane pomocnicze: licznik a
Krok 1: Rozpocznij algorytm
Krok 2: Wprowadź liczbę n
Krok 3: Przypisz kolejno wartości silnia=1 i a=n
Krok 4: Dokonaj sprawdzenia, czy wartość a<=1. Jeżeli tak, to wypisz wartość silnia i zakończ działanie algorytmu.
Krok 5: Przypisz wartość silnia=silnia*a
Krok 6: Przypisz wartość a=a-1 i powróć do kroku 4
Krok 7: Zakończ algorytm

(schemat blokowy)



Przykład 2. Wieże Hanoi - rozwiązanie rekurencyjne

Wieże Hanoi to klasyczny problem rekurencyjny z zastosowaniem zagadki (łamigłówki) opartej na liczbie n krążków.

Łamigłówka oparta jest na zasadzie przeniesienia wszystkich krążków z pierwszego palika na ostatni palik, z uwzględnieniem reguły, że większy krążek nie może być nałożony na krążek mniejszy. Pamiętać należy, że z każdego palika można przełożyć jednorazowo wyłącznie jeden krążek.

Dane wejściowe: n - liczba krążków
Dane wyjściowe: lista ruchów, jakich trzeba użyć, aby przenieść krążki do punktu początkowego.

Warto zapamiętać

- Algorytm podaje krok po kroku sposób rozwiązania problemu.
- Rozwiązując dowolny problem, postępujemy według podobnego schematu. Określamy: dane wejściowe, wyniki oraz sposób rozwiązania.
- Sposoby zapisywania algorytmów: opis słowny, lista kroków, schemat blokowy, język programowania.
- Zapis algorytmu w postaci listy kroków polega na przedstawieniu algorytmu w kolejnych punktach (krokach). Każdy punkt takiej listy zawiera opis wykonywanej czynności.
- W schemacie blokowym, czyli graficznej prezentacji algorytmu, układ bloków (klocków) i połączeń między nimi określa, w jakiej kolejności i w jaki sposób będą wykonywane poszczególne kroki algorytmu.
- Programowanie polega na przedstawieniu algorytmu w postaci instrukcji języka programowania, w kolejności wyznaczonej przez ten algorytm.
- W algorytmie z warunkami występują sytuacje warunkowe – wynik lub dalsze działanie algorytmu zależą od spełnienia określonego warunku.
- W schemacie blokowym sytuacje warunkowe przedstawiamy za pomocą bloku warunkowego, w którym wpisujemy warunek logiczny. Z bloku warunkowego wychodzą dwa połączenia: TAK i NIE.
- Technikę iteracji stosuje się do czynności wielokrotnie powtarzanych. Dzięki zastosowaniu pętli możemy wrócić do jednego z wcześniejszych kroków algorytmu.
- Zakończenie pętli może być uzależnione od zadanej liczby powtórzeń.

Zadania

1. Przedstaw w postaci listy kroków algorytm przejścia przez ruchliwe skrzyżowanie.
2. Przedstaw w postaci listy kroków algorytm gotowania budyniu.
3. Przedstaw w postaci listy kroków algorytm obliczania objętości prostopadłościanu.
4. Przedstaw w postaci schematu blokowego algorytm z zadania 3.
5. Przedstaw w postaci schematu blokowego algorytm obliczania objętości ostrosłupa o wysokości *h*. Podstawą ostrosłupa jest trójkąt prostokątny o przyprostokątnych *a* i *b*.
6. Przedstaw w postaci schematu blokowego algorytm wyznaczania wartości bezwzględnej dowolnej liczby rzeczywistej *x*
   Wskazówki:
   - Wartością bezwzględną liczby nieujemnej jest ta sama liczba, a wartością bezwzględną liczby ujemnej jest liczba do niej przeciwna.
   - W bloku warunkowym wpisz x>=0.
   - W bloku wykonania wpisz odpowiednio: w:=x, w:= -x.
   - Wyprowadź wynik w.
7. Przedstaw w postaci schematu blokowego następujący algorytm: wprowadzane są dwie liczby a i b (zakładamy, że są to zawsze liczby dodatnie i różne). Dla liczby większej oblicz objętość sześcianu o krawędzi długości równej tej liczbie. Dla liczby mniejszej oblicz pole kwadratu o boku długości równej tej liczbie.
8. Przedstaw w postaci schematu blokowego algorytm obliczania pola powierzchni sześcianu o krawędzi a. Uwzględnij w schemacie warunek: dla a dodatniego ma nastąpić obliczenie pola i wyprowadzenie wyniku, w przeciwnym przypadku algorytm powinien się od razu zakończyć odpowiednim komunikatem.
9. Przedstaw w postaci schematu blokowego algorytm obliczania objętości prostopadłościanu o krawędziach a, b, c. Uwzględnij w schemacie warunek: dla a, b, c dodatnich ma nastąpić obliczenie objętości i wyprowadzenie wyniku, w przeciwnym przypadku algorytm powinien się od razu zakończyć odpowiednim komunikatem.