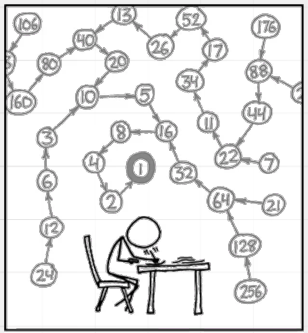


THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

Credit: Randall Munroe, CC BY-NC 2.5

Zapis algorytmów (Algorytmy, Część II)

wer. 14 z drobnymi modyfikacjami!

Wojciech Myszka

2021-12-03 12:05:20 +0100

Politechnika Wrocławska

Część I

Zapis algorytmów

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.
2. **Schematy blokowe.** *O tym za chwilę...*

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.
2. **Schematy blokowe.** *O tym za chwilę...*
3. **Tablice decyzyjne.** *O tym później...*

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.
2. **Schematy blokowe.** *O tym za chwilę...*
3. **Tablice decyzyjne.** *O tym później...*
4. **Pseudojęzyk** — rodzaj formalnego zapisu podobny do...

Zapis algorytmu

1. **Słowami.** Należy używać prostych zdań (raczej równoważników zdań) w trybie rozkazującym.
2. **Schematy blokowe.** *O tym za chwilę...*
3. **Tablice decyzyjne.** *O tym później...*
4. **Pseudojęzyk** — rodzaj formalnego zapisu podobny do...
5. **Język programowania.** *Następny semestr! Wcale???*

Algorytm

Algorytm

skończony **ciąg jasno zdefiniowanych czynności** koniecznych do wykonania pewnego rodzaju zadań, sposób postępowania prowadzący do rozwiązania problemu

Zapis algorytmu

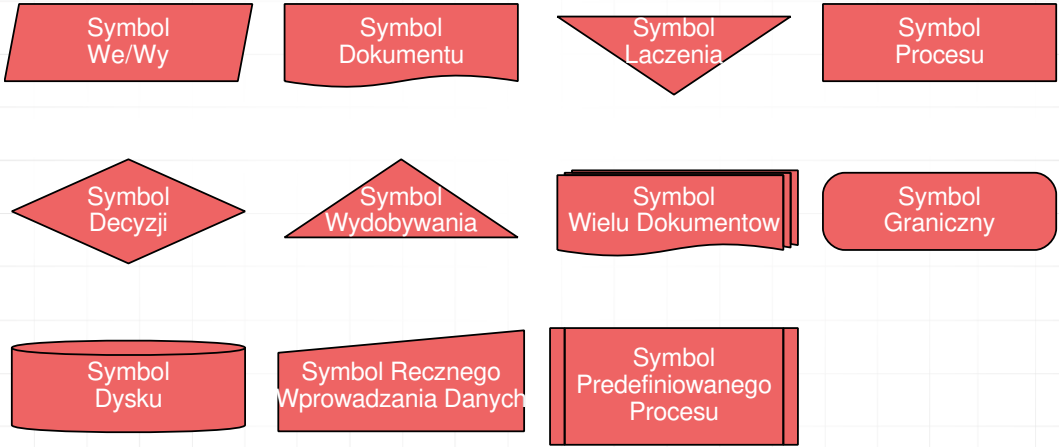
Schemat blokowy

Schemat blokowy (ang. *block diagram, flowchart*) — diagram, na którym procedura, system albo program komputerowy, są reprezentowane przez opisane figury geometryczne połączone liniami zgodnie z kolejnością wykonywania czynności wynikających z przyjętego algorytmu rozwiązania zadania.

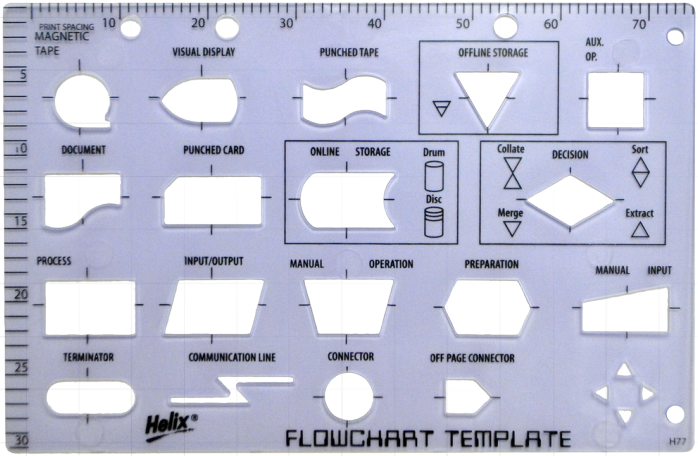
Schemat blokowy pozwala dostrzec istotne etapy algorytm i logiczne zależności między nimi.

Zależnie od przedstawianego zagadnienia stosowane są różne zestawy figur geometrycznych zwanych blokami, których kształty reprezentują umownie rodzaje elementów składowych.

Wszystkie


Symbol We/Wy
Symbol Dokumentu
Symbol Laczenia
Symbol Procesu
Symbol Decyzji
Symbol Wydobywania
Symbol Wielu Dokumentow
Symbol Graniczny
Symbol Dysku
Symbol Recznego Wprowadzania Danych
Symbol Predefiniowanego Procesu
Symbol Wyswietlania
Symbol Recznej Operacji
Preparation Symbol

Szablon


PRINT SPACING
MAGNETIC TAPE
VISUAL DISPLAY
PUNCHED TAPE
OFFLINE STORAGE
AUX. OP.
DOCUMENT
PUNCHED CARD
ONLINE STORAGE
Drum
Disc
Collate
DECISION
Sort
Merge
Extract
PROCESS
INPUT/OUTPUT
MANUAL OPERATION
PREPARATION
MANUAL INPUT
TERMINATOR
COMMUNICATION LINE
CONNECTOR
OFF PAGE CONNECTOR
Helix®
FLOWCHART TEMPLATE
H77

Schemat blokowy

Blok graniczny

Blok graniczny oznacza on początek, koniec, przerwanie lub wstrzymanie wykonywania działania, np. blok startu programu.

Schemat blokowy

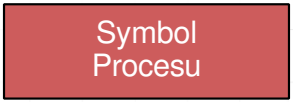
Blok wejścia-wyjścia

Blok wejścia-wyjścia przedstawia czynność wprowadzania danych do programu i przyporządkowania ich zmiennym dla późniejszego wykorzystania jak i wyprowadzenia wyników obliczeń, np. **czytaj z**, **pisz z+10**.

Schemat blokowy

Blok obliczeniowy

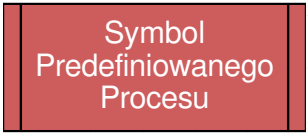
Blok obliczeniowy oznacza wykonanie operacji w efekcie, której zmienią się wartości, postać lub miejsce zapisu danych, np. $z = z + 1$.



Schemat blokowy

Blok fragmentu/podprogramu

Blok fragmentu przedstawia część programu zdefiniowanego odrębnie, np. sortowanie.



Schemat blokowy

Blok decyzyjny

Blok decyzyjny przedstawia wybór jednego z dwóch wariantów wykonywania programu na podstawie sprawdzenia warunku wpisanego w ów blok, np. $\mathbf{a = b}$.

Schemat blokowy

Łącznik zewnętrzny

Łącznik zewnętrzny służy do łączenia odrębnych części schematu znajdujących się na odrębnych stronach, powinien być opisany jak łącznik wewnętrzny, poza tym powinien zawierać numer strony, do której się odwołuje, np. **4.3**, **2,B2**.

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.
4. Koniec algorytmu

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.
4. Koniec algorytmu
5. Weź kolejną osobę z sali.

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.
4. Koniec algorytmu
5. Weź kolejną osobę z sali.
6. Porównaj ją z tą stojącą przy drzwiach — czy jest większa?

Wybór osoby najwyższej na sali

Wersja słowna

1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.
4. Koniec algorytmu
5. Weź kolejną osobę z sali.
6. Porównaj ją z tą stojącą przy drzwiach — czy jest większa?
7. Jeżeli nie — przejdź do 2

Wybór osoby najwyższej na sali

Wersja słowna

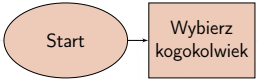
1. Wybierz dowolną osobę z sali, traktuj ją jako największą (i postaw przy drzwiach).
2. Czy zostały jakieś osoby na sali? jeżeli tak — przejdź do punktu 5.
3. Jeżeli nie — największą osobą jest ta stojąca przy drzwiach.
4. Koniec algorytmu
5. Weź kolejną osobę z sali.
6. Porównaj ją z tą stojącą przy drzwiach — czy jest większa?
7. Jeżeli nie — przejdź do 2
8. Jeżeli tak — zamienia osobę stojącą przy drzwiach; przejdź do 2

Wybór osoby najwyższej na sali

Schemat blokowy

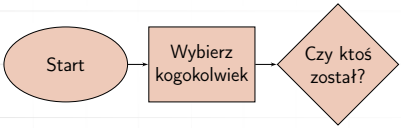
Wybór osoby najwyższej na sali

Schemat blokowy



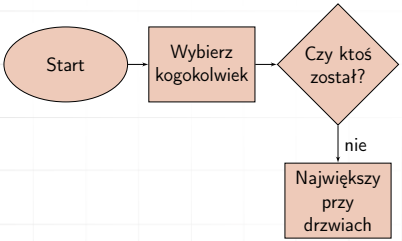
Wybór osoby najwyższej na sali

Schemat blokowy



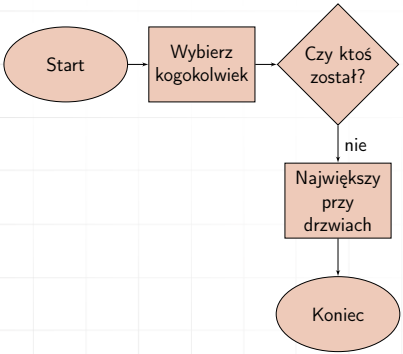
Wybór osoby najwyższej na sali

Schemat blokowy



Wybór osoby najwyższej na sali

Schemat blokowy

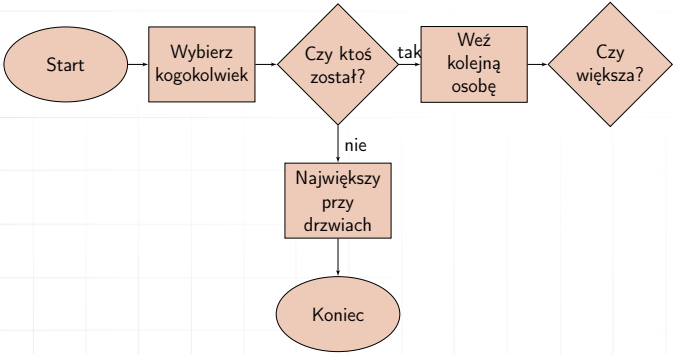


Wybór osoby najwyższej na sali

Schemat blokowy

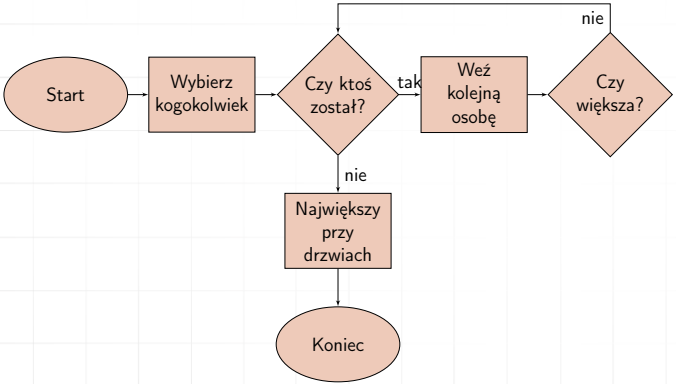
Wybór osoby najwyższej na sali

Schemat blokowy



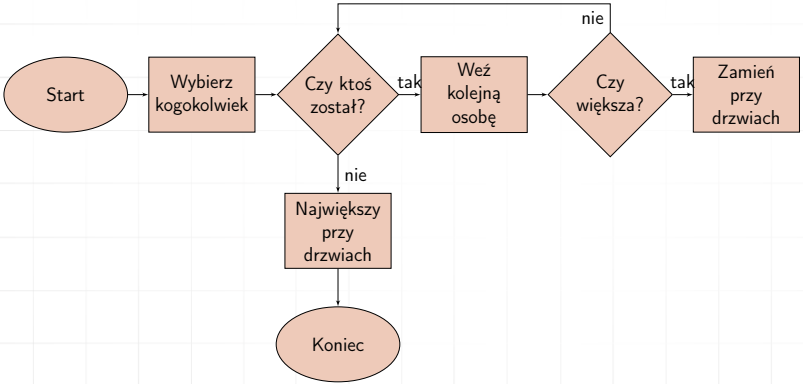
Wybór osoby najwyższej na sali

Schemat blokowy



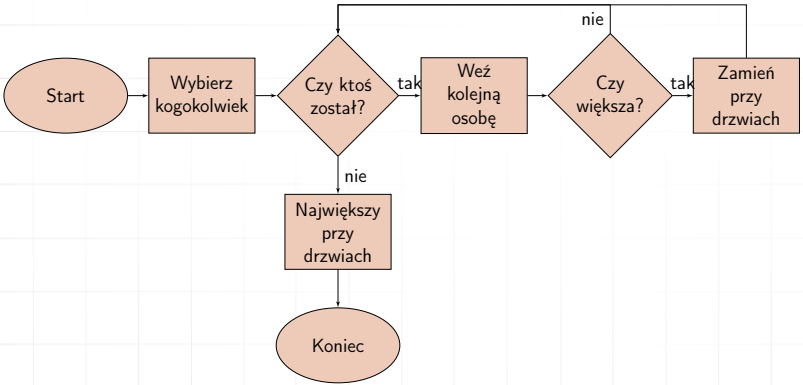
Wybór osoby najwyższej na sali

Schemat blokowy



Wybór osoby najwyższej na sali

Schemat blokowy



Algorytm Euklidesa

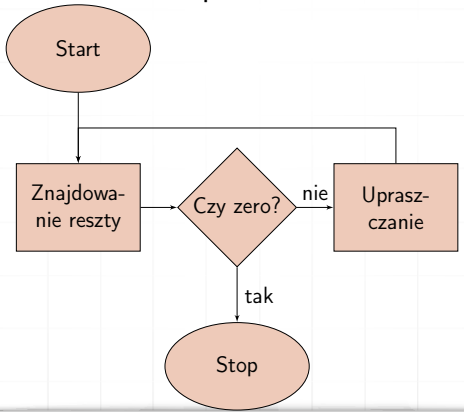
Słownie

1. [Znajdowanie reszty] Podziel m przez n i niech r oznacza resztę z tego dzielenia. (Mamy $0 \leq r < n$.)
2. [Czy wyszło zero?] Jeśli $r = 0$ zakończ algorytm; odpowiedzią jest n.
3. [Upraszczanie] Wykonaj $m \leftarrow n$, $n \leftarrow r$ i wróć do kroku 1.

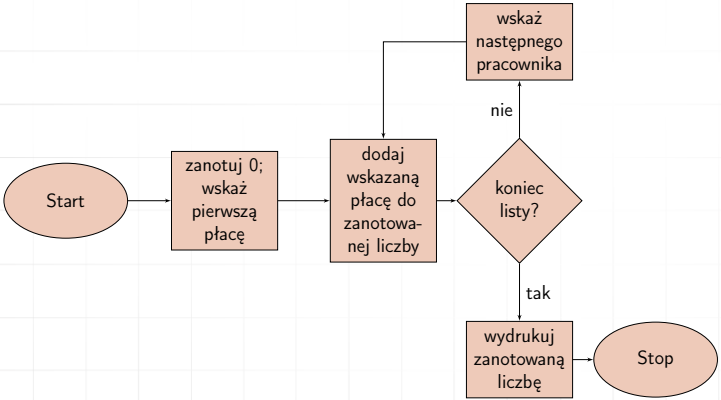
Algorytm Euklidesa

Schemat blokowy

Zapiszmy teraz Algorytm Euklidesa w postaci schematu blokowego:



Płace


wskaż następnego pracownika
nie
Start
zanotuj 0; wskaż pierwszą płacę
dodaj wskazaną płacę do zanotowanej liczby
koniec listy?
tak
wydrukuj zanotowaną liczbę
Stop

Tablice decyzyjne

- Alternatywa dla schematu blokowego.
- Bardzo wygodne w przypadku opisu problemów z ogromną ilością decyzji.
- Nieźle nadaje się do opisu problemów „z życia wziętych".
- Średnie zastosowanie w przypadku problemów obliczeniowych.
- Dziś już nieco zapomniane.

Tablice decyzyjne c.d

Pomysł tak, gdzieś z lat 50 (zeszłego wieku)!

Tablica decyzyjna to mieszanka warunków i decyzji które należy podejmować w zależności od ich spełnienia.

Drukarka

		Rules							
Warunki	drukarka nie drukuje	Y	Y	Y	Y	N	N	N	N
	miga czerwone światełko	Y	Y	N	N	Y	Y	N	N
	system „nie widzi" drukarki	Y	N	Y	N	Y	N	Y	N
Akcje	sprawdź kabel zasilający			X					
	sprawdź kabel łączący drukarkę z komputerem	X		X					
	upewnij się, że zainstalowałeś sterowniki	X		X		X		X	
	sprawdź/zmień atrament (toner)	X	X			X	X		
	sprawdź, czy nie zaciął się papier		X		X				

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania			
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne			
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny		
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne			
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne			
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne			
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj		
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj	Targuj się; kup jeżeli zbiłeś cenę	
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj	Targuj się; kup jeżeli zbiłeś cenę	Targuj się; kup niezależnie od wyników targów
Samochód nie spełnia wymagań			

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj	Targuj się; kup jeżeli zbiłeś cenę	Targuj się; kup niezależnie od wyników targów
Samochód nie spełnia wymagań	Rezygnuj		

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj	Targuj się; kup jeżeli zbiłeś cenę	Targuj się; kup niezależnie od wyników targów
Samochód nie spełnia wymagań	Rezygnuj	Rezygnuj	

Zakup samochodu

	cena nadmierna	cena OK	cena niedoszacowana
Samochód spełnia wszystkie wymagania	Targuj się; jak się nie uda — wróć następnego dnia i targuj się; kup nawet jak nie uda się zbić ceny	Targuj się; kup niezależnie od wyniku targów	kup
Samochód nie spełnia wszystkich wymagań ale stan i wyposażenie są akceptowalne	Rezygnuj	Targuj się; kup jeżeli zbiłeś cenę	Targuj się; kup niezależnie od wyników targów
Samochód nie spełnia wymagań	Rezygnuj	Rezygnuj	Targuj się o dodatki; kup jeśli cena dodatków jest rozsądna

Wkładanie palta

		R1	R2	R3
C1	Pada	T	T	N
C2	Zimno	T	N	T
A1	Włóż ocieplany płaszcz przeciwdeszczowy	X		
A2	Włóż zwykły płaszcz przeciwdeszczowy		X	
A3	Włóż ciepły płaszcz			X

Wkładanie palta c.d.

W tabeli pominęliśmy warunek:
Pada N
Zimno N
nie wymaga on żadnej specjalnej akcji, choć można by go dodać definiując akcję: „Nie wkładaj żadnego płaszcza".

Przykład: sklepik

Kolejny przykład to tablica decyzyjna opisująca działania związane z przyjęciem i realizacją zamówienia.

Tablica uwzględnia też politykę firmy, którą można opisać tak:

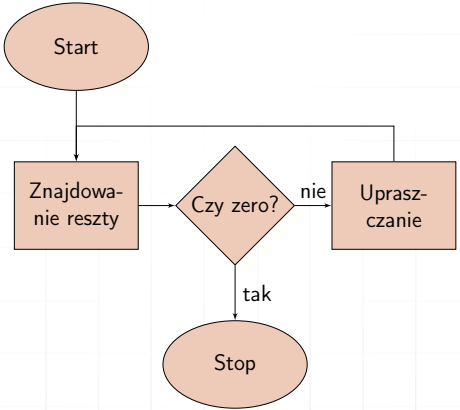
1. Firma obsługuje **tylko** zarejestrowanych klientów.
2. Firma dostarcza **tylko** towary znajdujące się na liście towarów.

Sklepik c.d.

C1	Towar na liście	T	N	—	T
C2	Klient zarejestrowany	T	—	N	T
C3	Wystarczający zapas towaru	T	—	—	N
A1	Zarezerwuj towar	X			
A2	Zarejestruj transakcję	X			
A3	Zapisz zamówienie na liście do realizacji				X
A4	Wyślij towar	X			
A5	Odrzuć transakcję		X	X	

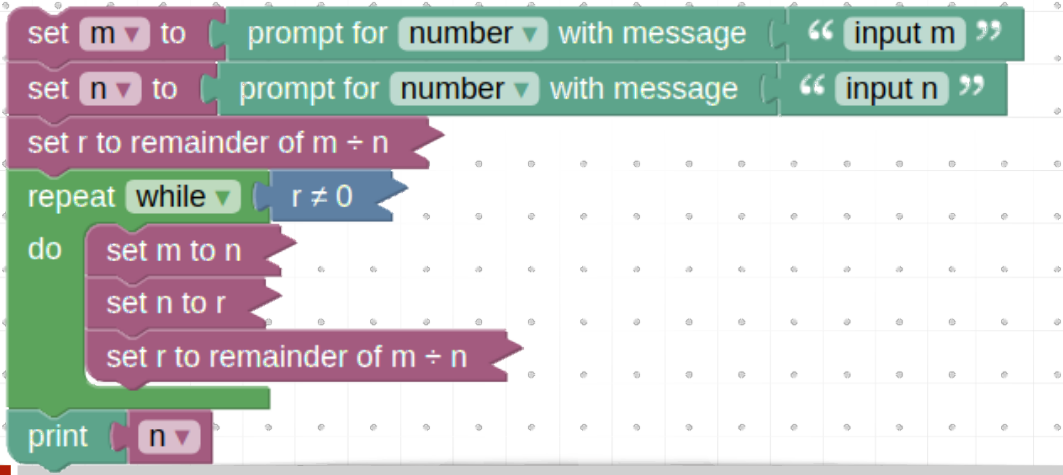
Języki programowania

Schemat blokowy



Języki programowania

Blockly



Języki programowania

Java Script

```
var m, n, r;
m = parseFloat(window.prompt('input m'));
n = parseFloat(window.prompt('input n'));
r = m % n;
while (r != 0) {
  m = n;
  n = r;
  r = m % n;
}
window.alert(n);
```

Języki programowania

Python

```
m = None
n = None
r = None
m = float(text_prompt('input m'))
n = float(text_prompt('input n'))
r = m % n
while r != 0:
    m = n
    n = r
    r = m % n
print(n)
```

Języki programowania

PHP

```
$m;
$n;
$r;
$m = floatval(readline('input m'));
$n = floatval(readline('input n'));
$r = $m % $n;
while ($r != 0) {
  $m = $n;
  $n = $r;
  $r = $m % $n;
}
print($n);
```

Języki programowania

Lua

```
m = tonumber(text_prompt('input m'), 10)
n = tonumber(text_prompt('input n'), 10)
r = m % n
while r ~= 0 do
  m = n
  n = r
  r = m % n
end
print(n)
```

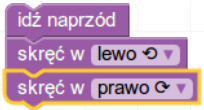
Języki programowania

Dart

```
var m, n, r;
main() {
  m = Math.parseDouble(Html.window.prompt('input m', ''));
  n = Math.parseDouble(Html.window.prompt('input n', ''));
  r = m % n;
  while (r != 0) {
    m = n;
    n = r;
    r = m % n;
  }
  print(n);
}
```

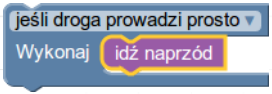
Struktura sterująca I

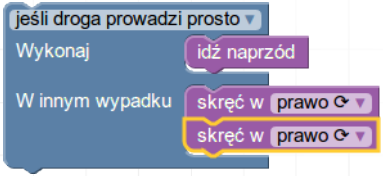
Tworząc algorytm zapisujemy go jako ciąg poleceń. Po cichu zakładamy, że poszczególne działania w sposób naturalny następują po sobie. (W algorytmach kucharskich będzie to coś takiego: "... Delikatnie dołóż czekoladę, [a potem] ponownie lekko podgrzej..."



Struktura sterująca II

Czasami będą pojawiać się działania warunkowe (`Jeśli` ... *coś*... `to` ... *coś*...).





Iteracje (pętle)

Iteracja ograniczona Niektóre algorytmy sugerują powtarzanie pewnych działań (`Wykonaj` *...coś...* `N razy`) ściśle określoną liczbę razy.

Iteracja warunkowa Czasami czynność trzeba powtarzać tak długo, aż zostanie spełniony jakiś warunek.

Pętle można zagnieżdżać jedne w drugich!

M — Instrukcja nie występuje!

Przekazanie sterowania I

Uwaga: W językach wysokiego poziomu może nie występować (bardzo często nie ma)!

Skok bezwarunkowy ma postać przejdź do „G" („G" to określone miejsce programu — algorytmu).

Skok warunkowy przejdź do „G" jeżeli spełniony jest jakiś warunek.

 — Instrukcje `Jump X`, `JumpI X`, `Skipcond X`

Bity 10 i 11 pola adresu (X) definiują warunek który jest badany. I tak gdy wartość ich jest 00 — instrukcja znaczy Skip (opuść/pomiń) jeżeli AC jest mniejsze od zera. Gdy 01 — opuść gdy AC jest równe zero, a gdy 10 — opuść gdy zawartość AC jest dodatnia.

Przykład (): instrukcja Skipcond 800 powoduje opuszczenie instrukcji występującej po niej gdy AC jest dodatnie (większe od zera).

Przekazanie sterowania II

12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	0	0	0	0	0	0	0
8				0				0			

Duża liczba skoków (do przodu, do tyłu) zmniejsza czytelność algorytmu.
Skoki powodują również problemy techniczne: czy można „wyskoczyć" ze środka pętli? Czy można do środka pętli wskoczyć?

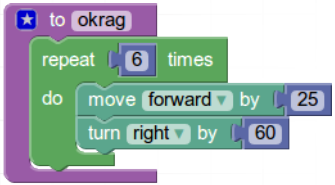
Podprogramy I

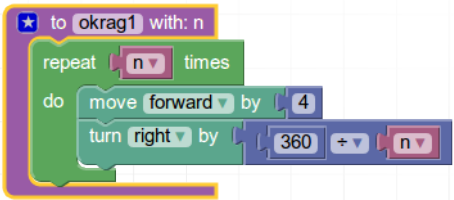
Podprogram — wyodrębniony zestaw instrukcji (osobny algorytm) realizujący pewne „zamknięte dzieło": ma dobrze[1] zdefiniowany zestaw danych (parametrów) wejściowych, zestaw danych wyjściowych i relację (zależność) pomiędzy nimi.

 — Instrukcje `JnS X` (*Zapisz aktualną wartość licznika rozkazów (PC) pod adresem X i przejdź do adresu X+1*)

- Oszczędność kodu (ta sama czynność wykonywana kilkakrotnie w różnych miejscach programu).
- Przejrzystość kodu („duży" problem dzielimy na mniejsze, dobrze zdefiniowane fragmenty).

Podprogramy II



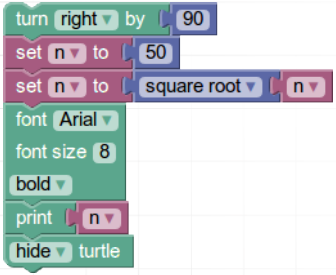


[1]Używając słowa „dobrze” będę miał na myśli „precyzyjnie”.

Zmienna I

Rodzaj pudełka do którego można coś włożyć, wyjąć. (Tak jak pokój hotelowy.)
Zmienna zazwyczaj ma jakąś nazwę (przez którą się do niej odwołujemy).
Wartość zmiennej (zawartość pokoju hotelowego) zmienia się w razie potrzeb.
W algorytmach używamy zmiennych do przechowywania (**pamiętania**) różnych wartości. Zakładamy wówczas, że to co tam wstawimy nie ulega zmianie (zniszczeniu) w sposób samorzutny.
Fakt przypisania wartości do zmiennej zaznaczamy w algorytmach w bardzo różny sposób. Najprostszy będzie taki: $X = 5$, $X := X + 1$ czy $X \leftarrow A + B$.
 — Zmienne to po prostu „etykiety" (adresy miejsc w pamięci). Dobrze nadać im jakąś wartość za pomocą dyrektyw assemblera.

Zmienna II


turn right by 90
set n to 50
set n to square root n
font Arial
font size 8
bold
print n
hide turtle

Wektor, tablica I

Zestaw wartości w których wprowadziliśmy rodzaj uporządkowania (dyskutując na temat wyszukiwania wartości maksymalnej mówiliśmy „weźmy pierwszą wartość" — albo jakoś tak). Jeżeli szereg zmiennych tego samego typu ustawimy jedną za drugą (i zaczniemy się do nich odwoływać przez numer: pierwsza, druga, ...) — mamy do czynienia z wektorem (zwanym czasami tablicą jednowymiarową). Jest to pewna analogia do piętra w hotelu: jest tam szereg pokoi.
Nazwę przypisujemy całemu wektorowi, do poszczególnych jego elementów będziemy odwoływać się przez numer, co zapisujemy najczęściej jakoś tak: $V(I)$ (zawartość I—tej komórki wektora V) albo tak $V[J]$. I (J) nazywany bywa indeksem wektora.

$V(1)$	$V(2)$	$V(3)$	$V(4)$	$V(5)$

Wektor, tablica II

Analogiczny zapis matematyczny:

V_1	V_2	V_3	V_4	V_5

 — Instrukcja `AddI X` (*Dodaj pośrednio: Traktuj wartość zapisaną w X jako adres elementu, który ma być dodany do AC*)

Tablica (wielowymiarowa) I

Struktura grupująca dane w sposób bardziej złożony.

Czasami mamy potrzebę grupowania danych w struktury bardziej złożone niż wektory. Struktura numerowana za pomocą dwu indeksów nazywana bywa tabelą.

Nasuwa się tu analogia do całego hotelu: pierwszym indeksem jest numer piętra drugim — numer pokoju na piętrze. My mówić będziemy o „wierszach" (poziome wektory) lub kolumnach (pionowe) tablicy.

Odwołanie do elementu tabeli zapisywane bywa tak: $W(I, J)$ albo $W[I, J]$ lub rzadziej jako $W[I][J]$.

Tablica (wielowymiarowa) II

W najprostszych sytuacjach tablice maja wszystkie wiersze (kolumny) jednakowej długości. Możliwe są jednak i bardziej ogólne przypadki.

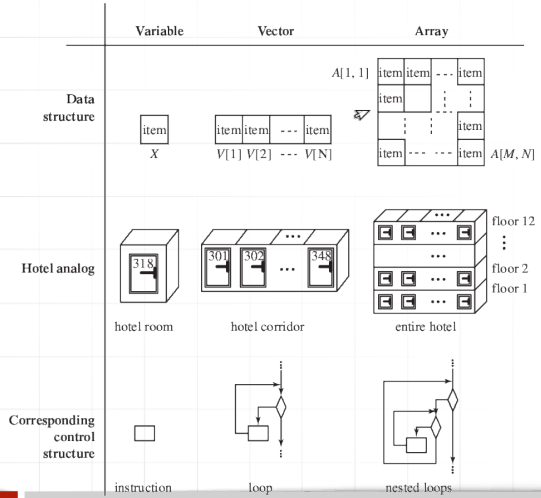
Zapis matematyczny:

$$\begin{matrix} v_{1,1} & v_{1,2} & v_{1,3} \\ v_{2,1} & v_{2,2} & v_{2,3} \\ v_{3,1} & v_{3,2} & v_{3,3} \\ v_{4,1} & v_{4,2} & v_{4,3} \end{matrix}$$

Zapis „algorytmiczny”:

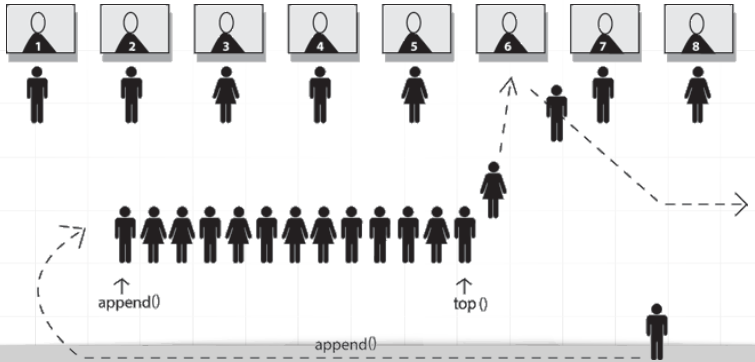
$$\begin{matrix} V(1,1) & V(1,2) & V(1,3) \\ V(2,1) & V(2,2) & V(2,3) \\ V(3,1) & V(3,2) & V(3,3) \\ V(4,1) & V(4,2) & V(4,3) \end{matrix}$$

Obsługa tablic



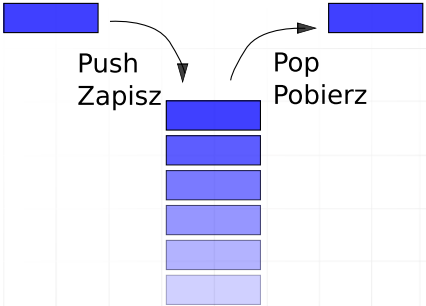
Kolejka

- Struktura bardzo podobna do wektora.
- Dane dostarczane są do jednego „końca”.
- Do odbioru danych służy drugi „koniec”.



Stos

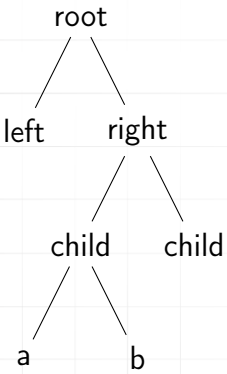
- Struktura podobna do wektora.
- Dane dostarczane są do jednego „końca”.
- Ten sam „koniec” służy do ich odbioru.



Drzewo albo hierarchia

- Struktura z porządkiem.
- Wyróżnia się specjalny obiekt będący „początkiem" całej struktury: **korzeń**.
- Inne elementy to „następniki" albo **potomstwo**.
- Każdy obiekt może mieć kilku **równoważnych** potomków.
- Każdy potomek to **węzeł** drzewa.
- Obiekty, które nie mają potomków to **liście**.
- **Gałąź** to droga od korzenia do liścia.

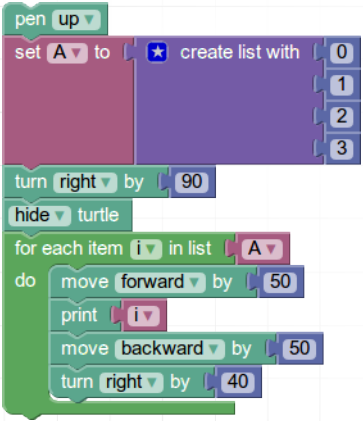
Drzewo

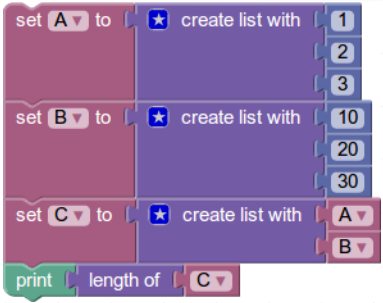


Inne struktury danych I

- Listy (pewne podobieństwo do wektorów czy tablic).
- Bazy danych (pewne podobieństwo do tablic).
- Grafy (pewne podobieństwo do drzew).

Listy w blockly





Część II

Dygresja

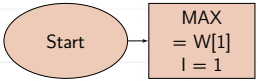
Raz jeszcze wybór osoby najwyższej

(na sali)

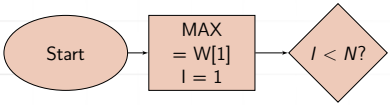
1. Wysokości wszystkich osób zostały zapisane w tablicy o nazwie **W**. (Powinno być wysokość, ale za dużo miejsca zajmuje.)
2. Zmienna **N** zawiera liczbę osób w sali (długość tablicy **W**).
3. Zmienna **MAX** po zakończeniu algorytmu zawiera największą wartość zapisaną w **W** (wysokość osoby najwyższej).
4. **I** — zmienna pomocnicza.

Największa wartość w tablicy

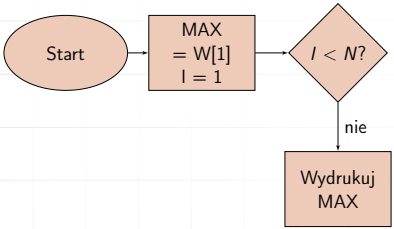
Największa wartość w tablicy



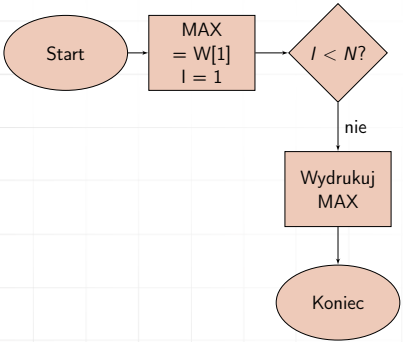
Największa wartość w tablicy



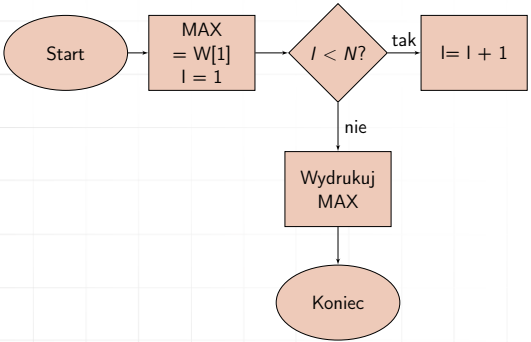
Największa wartość w tablicy



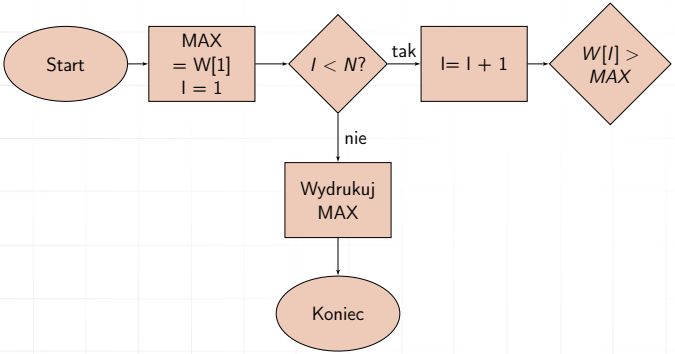
Największa wartość w tablicy



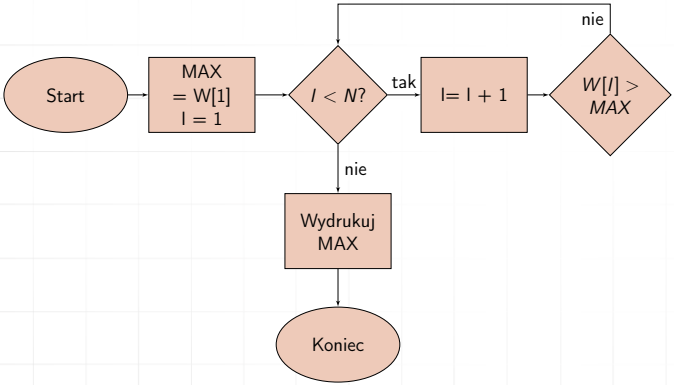
Największa wartość w tablicy



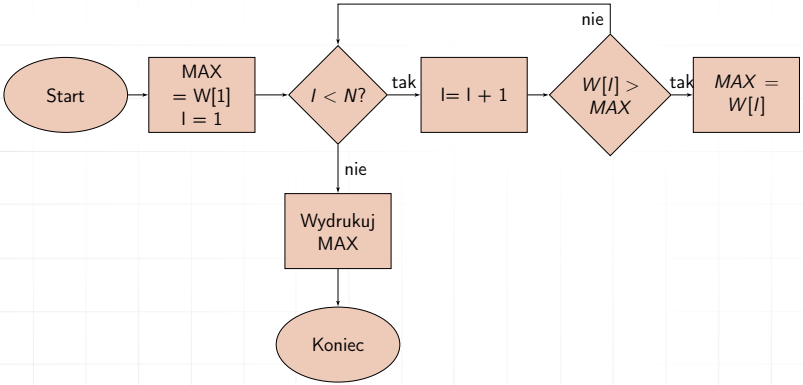
Największa wartość w tablicy



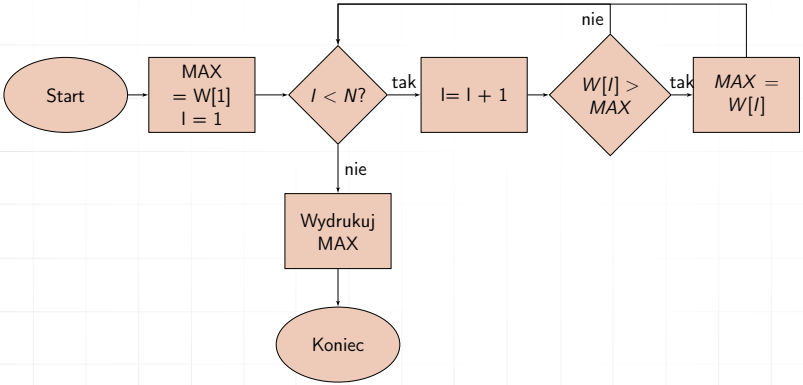
Największa wartość w tablicy



Największa wartość w tablicy



Największa wartość w tablicy



Banalny problem

1. Dopóki $X \neq 1$, dopóty wykonuj $X \leftarrow X - 2$.
2. Zatrzymaj się.

Banalny problem

1. Dopóki $X \neq 1$, dopóty wykonuj $X \leftarrow X - 2$.
2. Zatrzymaj się.

Gdy $X = 7$ algorytm „wygeneruje" następujące wartości X:
7, 5, 3, 1
i zatrzyma się...

Banalny problem

1. Dopóki $X \neq 1$, dopóty wykonuj $X \leftarrow X - 2$.
2. Zatrzymaj się.

Gdy $X = 7$ algorytm „wygeneruje" następujące wartości X:
7, 5, 3, 1
i zatrzyma się...

Gdy $X = 8$ będzie nieco inaczej:
8, 6, 4, 2, 0, -2, -4, -6, -8,...
i algorytm nie zatrzyma się.

Banalny problem

1. Dopóki $X \neq 1$, dopóty wykonuj $X \leftarrow X - 2$.
2. Zatrzymaj się.

Gdy $X = 7$ algorytm „wygeneruje" następujące wartości X:
7, 5, 3, 1
i zatrzyma się...

Gdy $X = 8$ będzie nieco inaczej:
8, 6, 4, 2, 0, -2, -4, -6, -8,...
i algorytm nie zatrzyma się.

Jak widać wszystko działa tylko dla liczb dodatnich nieparzystych, bo w przypadku liczb parzystych „udaje" nam się „minąć" jedynkę.

Problem bardziej skomplikowany...

Problem Collatza

1. Dopóki $X \neq 1$, dopóty wykonuj:
   1.1 Jeśli X jest liczbą parzystą to ustaw $X \leftarrow X/2$.
   1.2 W przeciwnym razie ustaw $X \leftarrow 3X + 1$
2. Zatrzymaj się

Problem bardziej skomplikowany...

Problem Collatza

1. Dopóki $X \neq 1$, dopóty wykonuj:
   1.1 Jeśli X jest liczbą parzystą to ustaw $X \leftarrow X/2$.
   1.2 W przeciwnym razie ustaw $X \leftarrow 3X + 1$
2. Zatrzymaj się

Zacznijmy od 7: 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

Problem bardziej skomplikowany...

Problem Collatza

1. Dopóki $X \neq 1$, dopóty wykonuj:
   1.1 Jeśli X jest liczbą parzystą to ustaw $X \leftarrow X/2$.
   1.2 W przeciwnym razie ustaw $X \leftarrow 3X + 1$
2. Zatrzymaj się

Zacznijmy od 7: 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
Jak będzie dla innych liczb?

Problem bardziej skomplikowany... I

1. Dopóki $X \neq 1$, dopóty wykonuj:
   1.1 Jeśli X jest liczbą parzystą to ustaw $X \leftarrow X/2$.
   1.2 W przeciwnym razie ustaw $X \leftarrow 3X + 1$
2. Zatrzymaj się

Problem bardziej skomplikowany... II

Okazuje się, że powyższy algorytm albo kończy działanie stosunkowo szybko, albo generuje nieskończony ciąg chaotycznych liczb.

Nie udało się nikomu ani udowodnić, że generowane sekwencje zaczynają się powtarzać (co oznacza, że algorytm nie zatrzyma się nigdy) ani dowieść, że dla jakiejś konkretnej wartości początkowej X algorytm zatrzyma się.

Uczeni w piśmie wierzą, że algorytm zatrzymuje się dla liczb dodatnich...