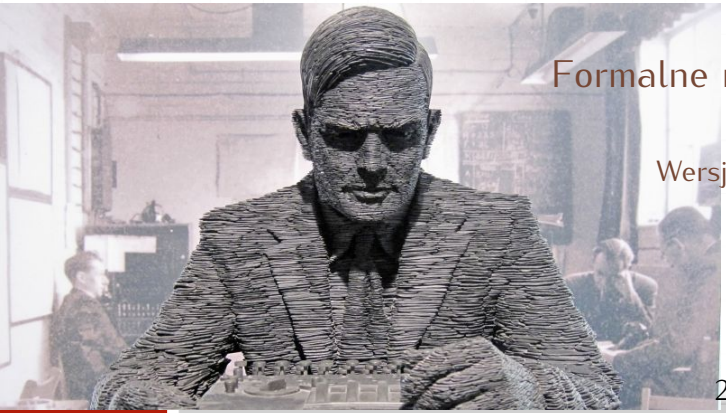


Algorytmy.
Formalne metody prezentacji algorytmu
Automat skończony

Wersja: 5 z drobnymi modyfikacjami!

Wojciech Myszka

2023-12-07 18:12:22 +0100

HR EXCELLENCE IN RESEARCH

Politechnika Wrocławska

Czym jest komputer?

1. Spojrzenie właściciela.
2. Spojrzenie von Neumanna.
3. Spojrzenie „systemowca”.

Co wybrać?

Co wybrać? Który opis jest najlepszy?

Co wybrać?

Co wybrać? Który opis jest najlepszy?
Zaczniemy dosyć dziwnie...

Maszyna Turinga

Maszyna Turinga składa się z:

1. skończonego **alfabetu** symboli;
2. skończonego zbioru **stanów** z wyróżnionymi **stanami końcowymi** (po osiągnięciu których maszyna zatrzymuje się);
3. nieskończonej **taśmy** z zaznaczonymi kwadratami, z których każdy może zawierać pojedynczy symbol;
4. ruchomej **głowicy** odczytująco-zapisującej, która może wędrować wzdłuż taśmy przesuwając się o jeden kwadrat na raz;
5. **diagramu przejść między stanami** (zazwyczaj zwanego **diagramem przejść** po prostu) zawierającego instrukcje, które powodują, że zmiany następują przy każdym zatrzymaniu się.

Maszyna Turinga

Opis formalny

Hopcroft i Ullman zdefiniowali formalnie maszyną Turinga jako następującą „siódemkę” (*7-tuple*): $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ gdzie:

- Q to skończny zbiór stanów,
- Γ to skończony zbiór symboli alfabetu
- $b \in \Gamma$ jest symbolem pustym
- $\Sigma \subseteq \Gamma \setminus \{b\}$ to zbiór symboli wejściowych
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, P\}$ jest „funkcją przejścia”, L onacza przesunięcie w lewo, a P przesunięcie w prawo.
- $q_0 \in Q$ To stan początkowy
- $F \subseteq Q$ jest zbiorem stanów końcowych

Każdy obiekt spełniający powyższe założenia nazywany być może maszyną Turinga.

Maszyna Turinga

Diagram przejść

Diagram przejść to graf skierowany, którego wierzchołki reprezentują stany.

Krawędź prowadzącą ze stanu *s* do *t* nazywa się **przejściem** i etykietuje kodem postaci: a/b, *kierunek*; gdzie a i b są symbolami (alfabetu), a *kierunek* to albo „w prawo" albo „w lewo" (*P*, *L*).

Część a jest **wyzwalaczem**[1] przejścia, a część b, *kierunek* **akcją**.[2]

[1] . . . jeżeli na taśmie spotkasz a. . .
[2] . . . to zapisz w kratce b i przesuń się w *kierunku*. . .

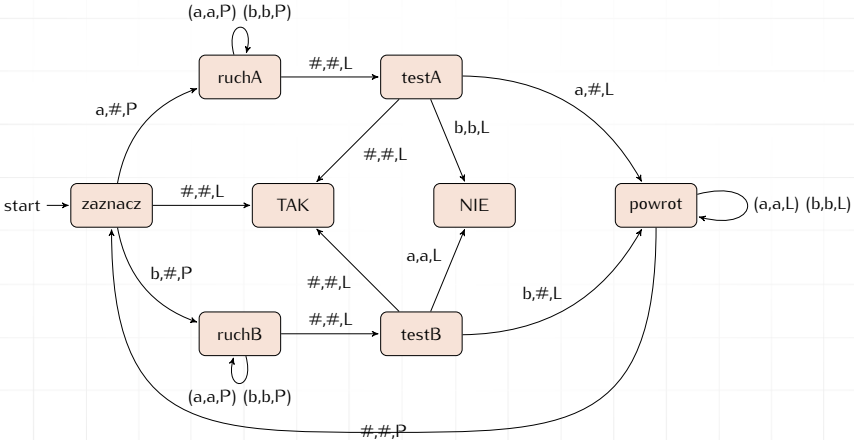
Przykładowa maszyna Turinga

- **Alfabet** — trzy symbole: **a**, **b** i **#** (co oznacza symbol pusty, „nic”)
- **Dane**

...	#	#	a	b	b	a	#	#	...

Przykładowa maszyna Turinga

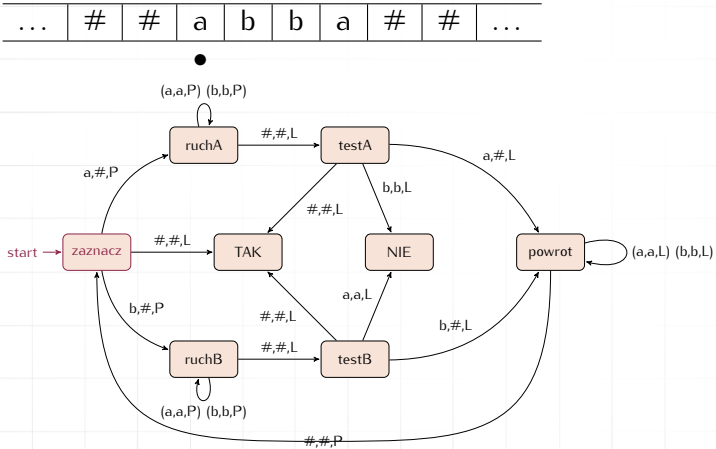
Diagram przejść, stany



Stany końcowe to stany opisane jako „TAK” i „NIE”

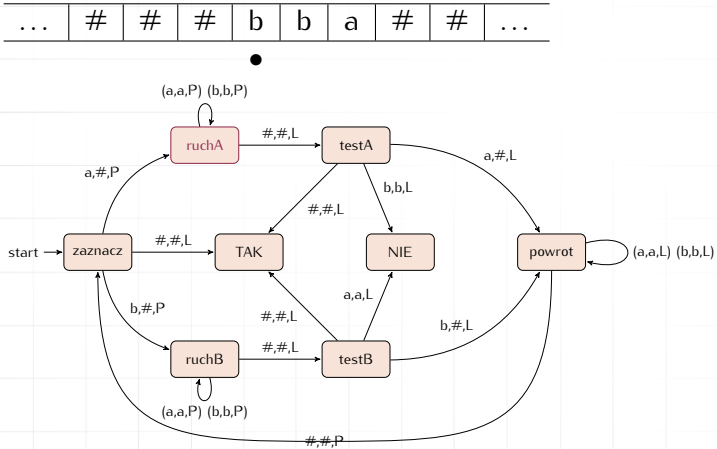
Maszyna Turinga

Przykład (1)



Maszyna Turinga

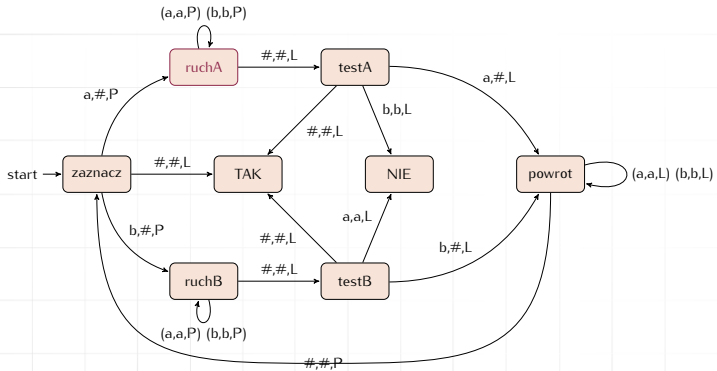
Przykład (1)



Maszyna Turinga

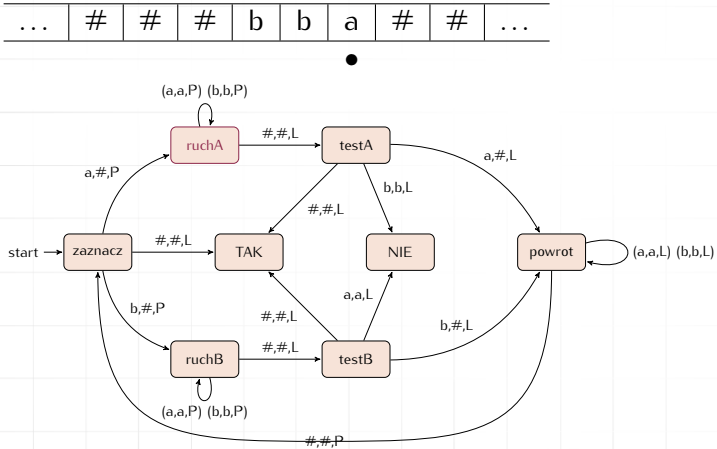
Przykład (1)

…	#	#	#	b	b	a	#	#	…



Maszyna Turinga

Przykład (1)

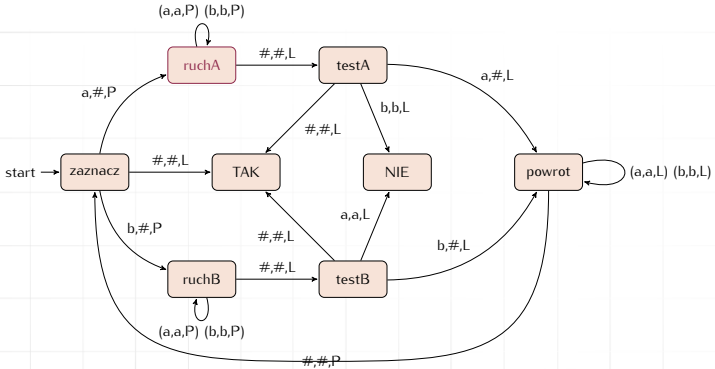


Maszyna Turinga

Przykład (1)

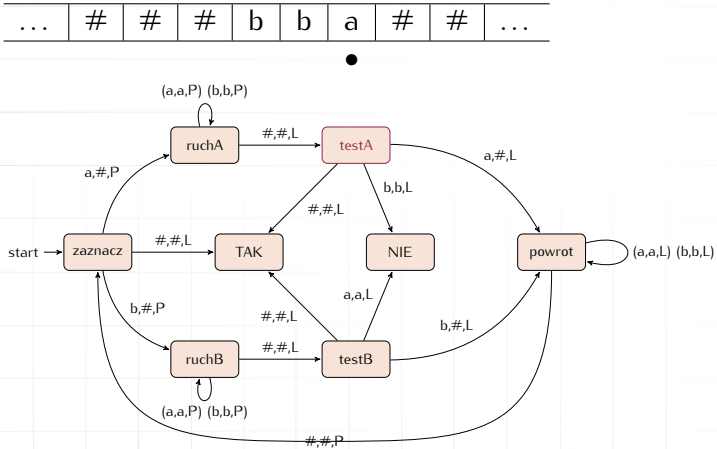
…	#	#	#	b	b	a	#	#	…

●



Maszyna Turinga

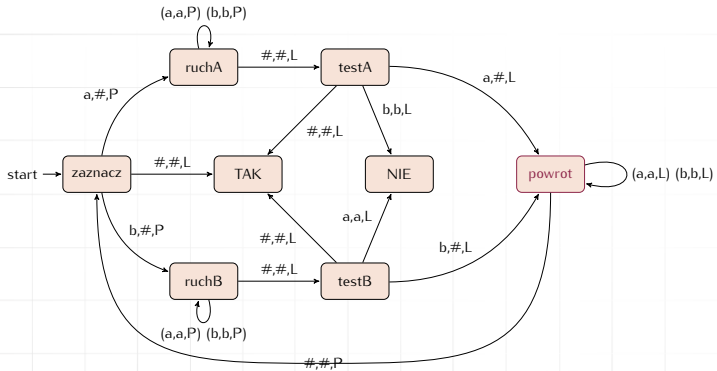
Przykład (1)



Maszyna Turinga

Przykład (1)

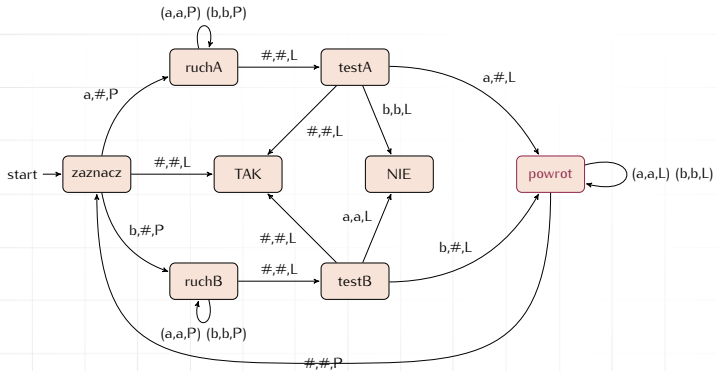
…	#	#	#	b	b	#	#	#	…



Maszyna Turinga

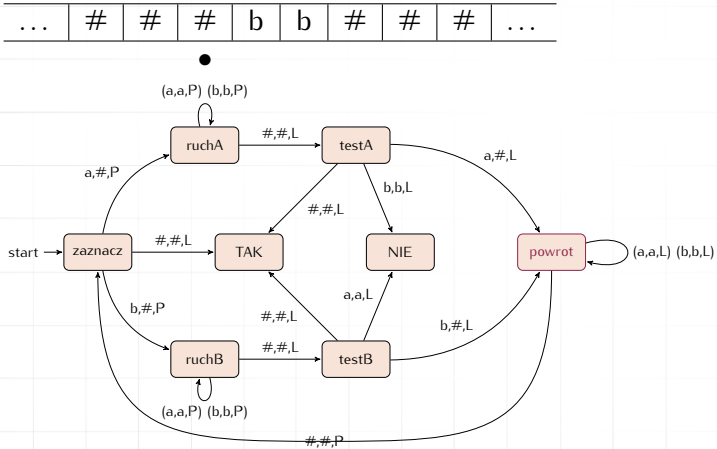
Przykład (1)

…	#	#	#	b	b	#	#	#	…



Maszyna Turinga

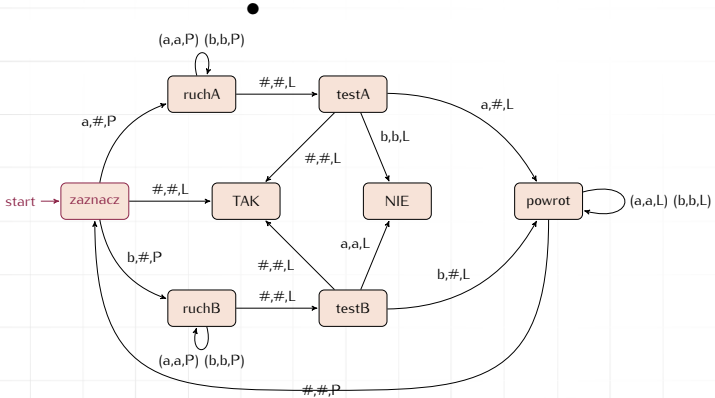
Przykład (1)



Maszyna Turinga

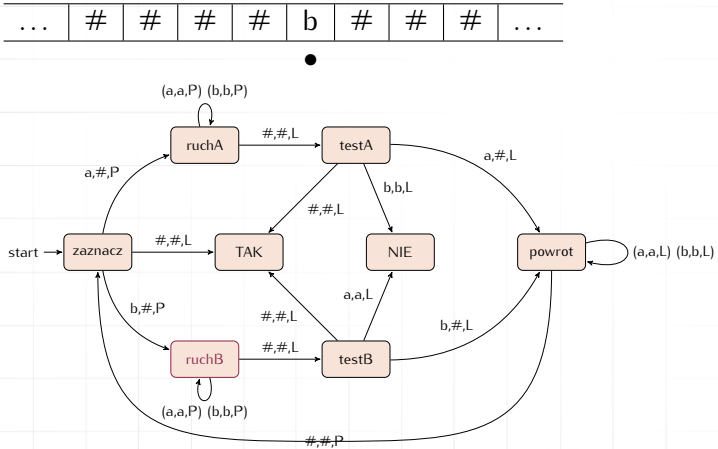
Przykład (1)

…	#	#	#	b	b	#	#	#	…



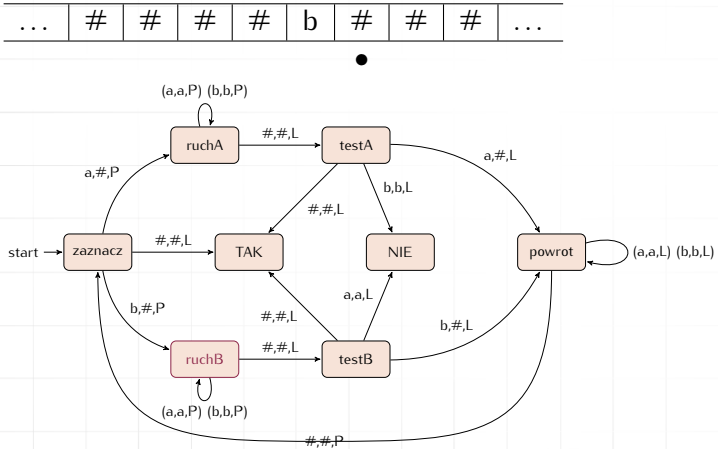
Maszyna Turinga

Przykład (1)



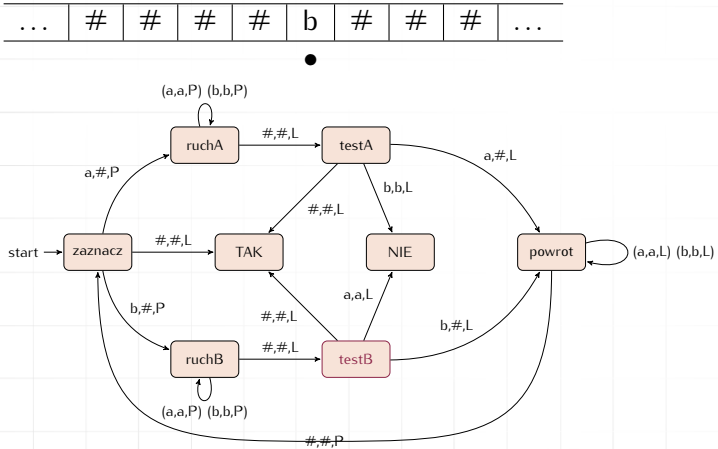
Maszyna Turinga

Przykład (1)



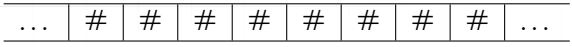
Maszyna Turinga

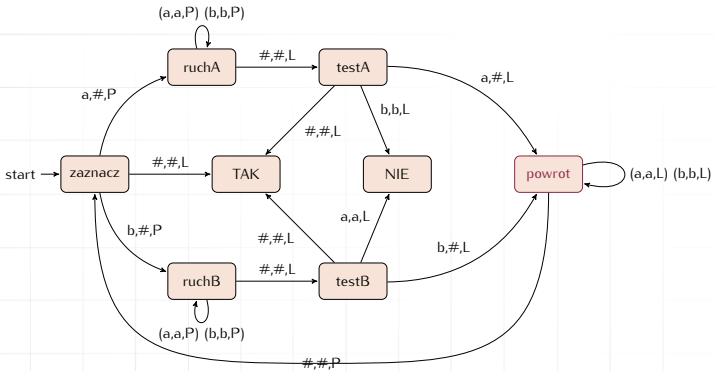
Przykład (1)



Maszyna Turinga

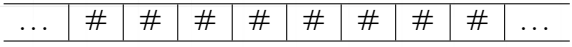
Przykład (1)

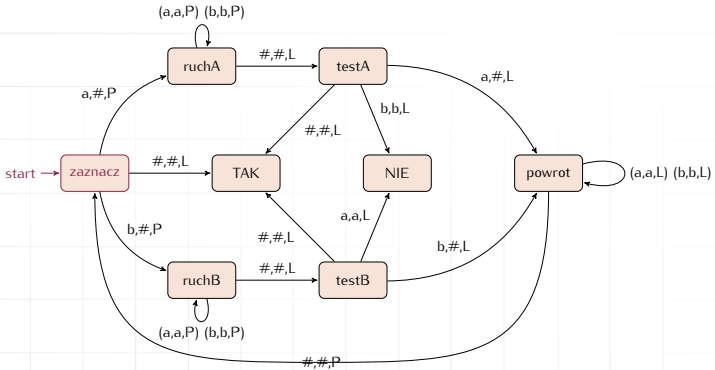




Maszyna Turinga

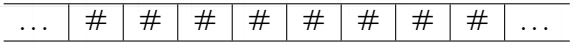
Przykład (1)

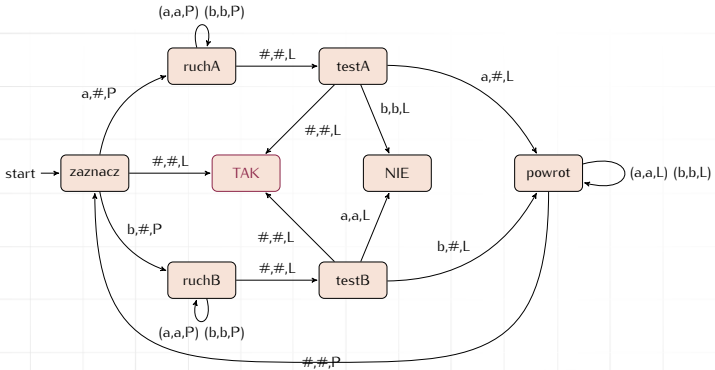




Maszyna Turinga

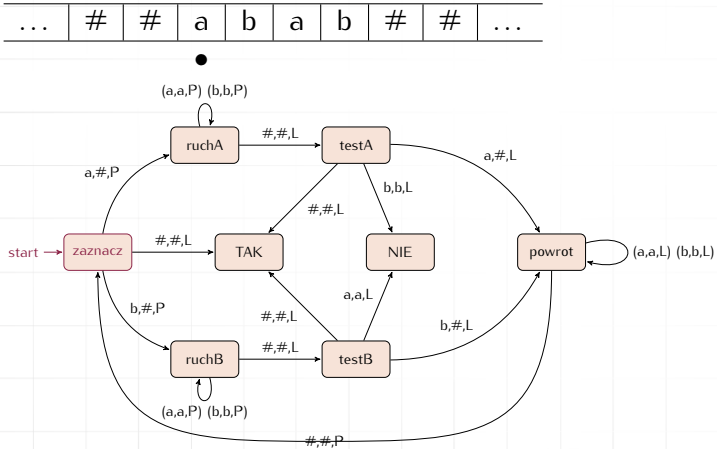
Przykład (1)





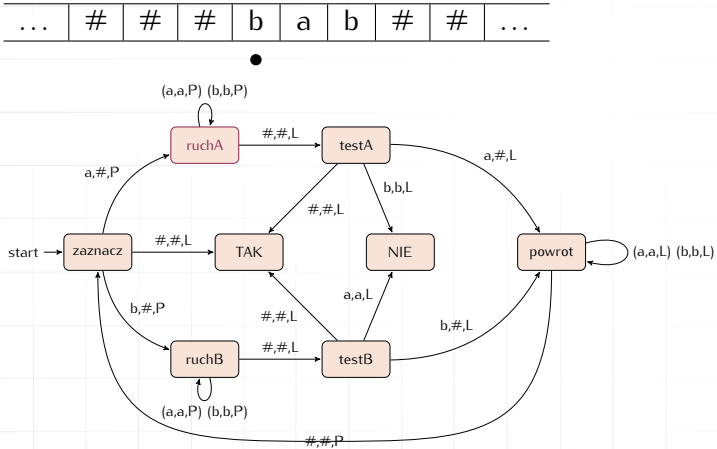
Maszyna Turinga

Przykład (2)



Maszyna Turinga

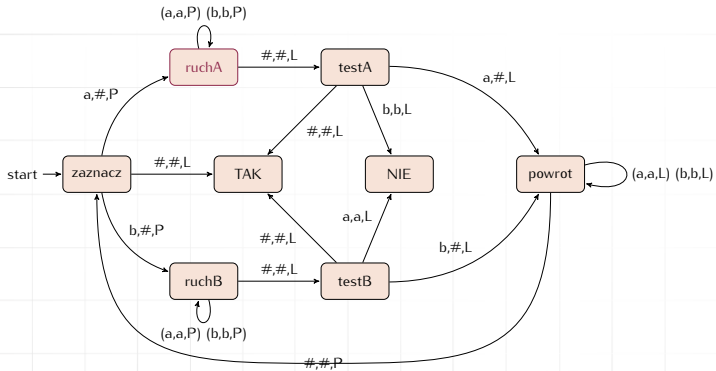
Przykład (2)



Maszyna Turinga

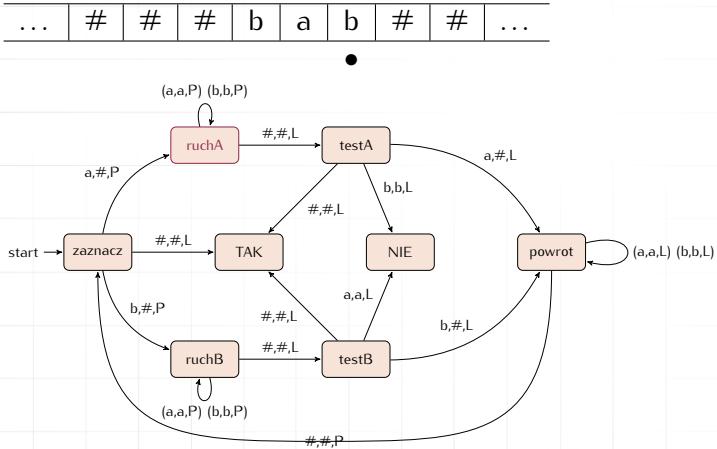
Przykład (2)

…	#	#	#	b	a	b	#	#	…



Maszyna Turinga

Przykład (2)

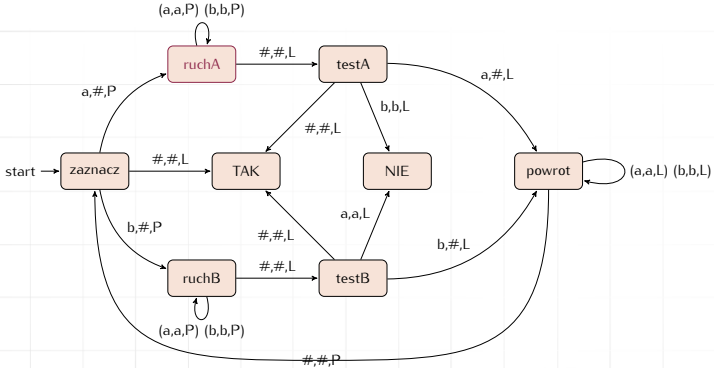


Maszyna Turinga

Przykład (2)

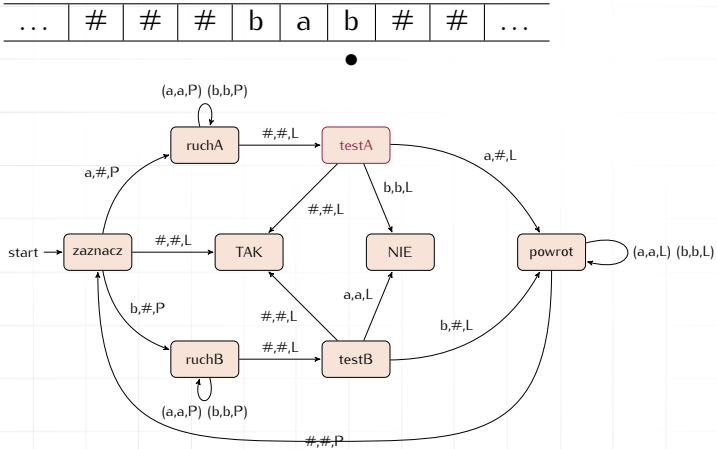
…	#	#	#	b	a	b	#	#	…

●



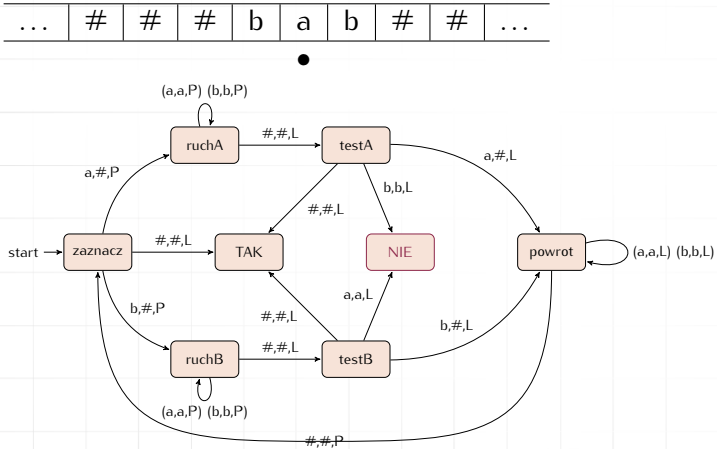
Maszyna Turinga

Przykład (2)



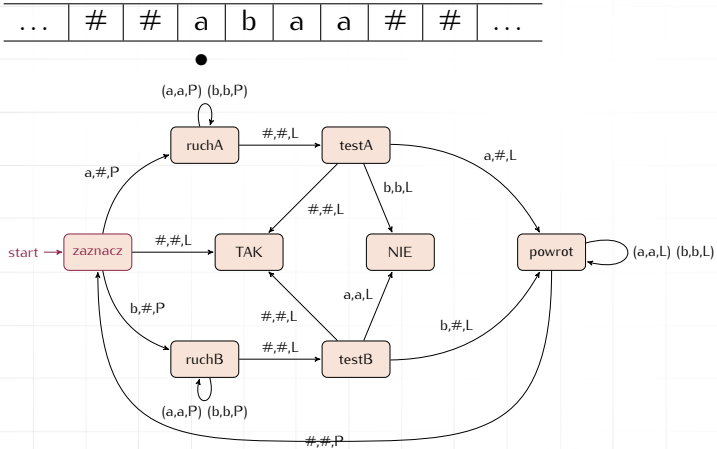
Maszyna Turinga

Przykład (2)



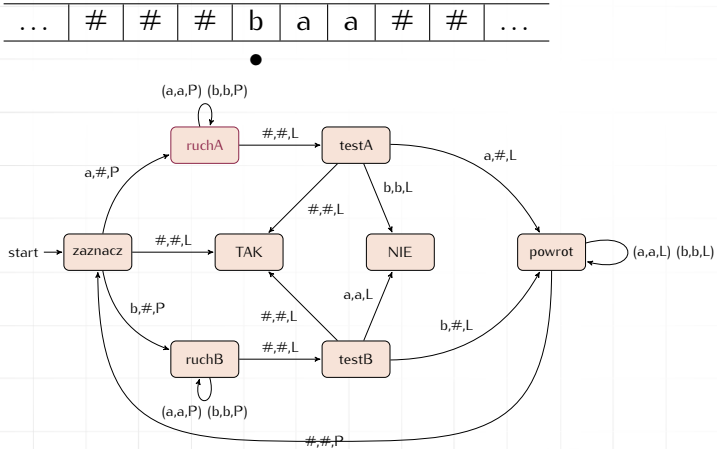
Maszyna Turinga

Przykład (3)



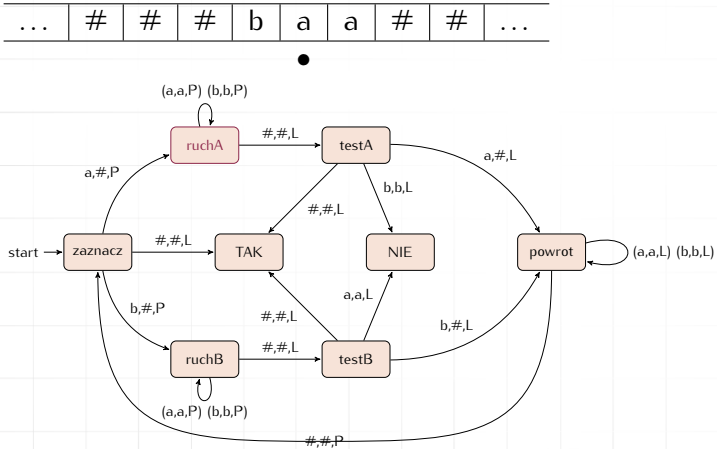
Maszyna Turinga

Przykład (3)



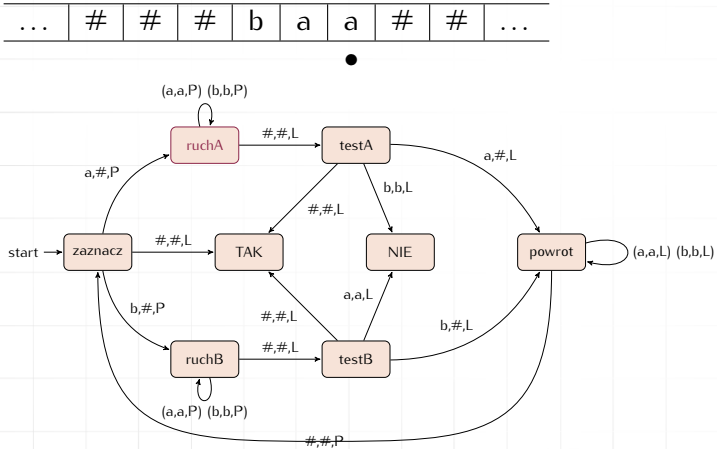
Maszyna Turinga

Przykład (3)



Maszyna Turinga

Przykład (3)

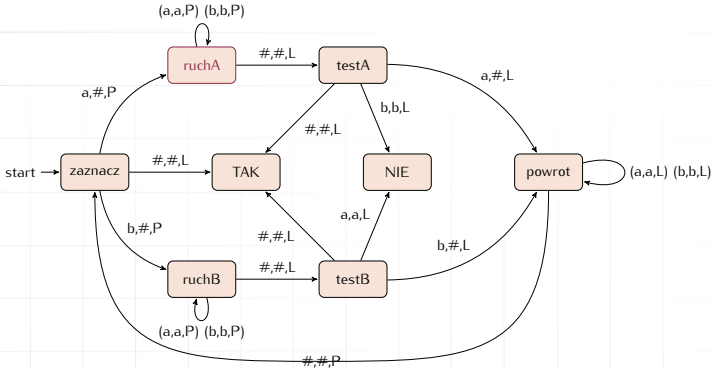


Maszyna Turinga

Przykład (3)

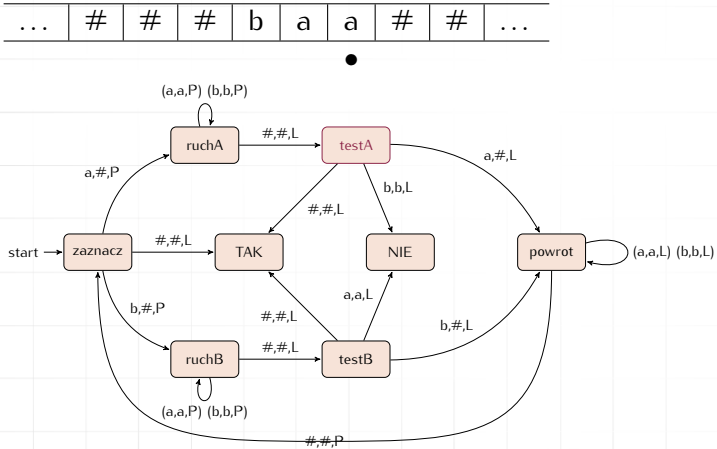
...	#	#	#	b	a	a	#	#	...

●



Maszyna Turinga

Przykład (3)

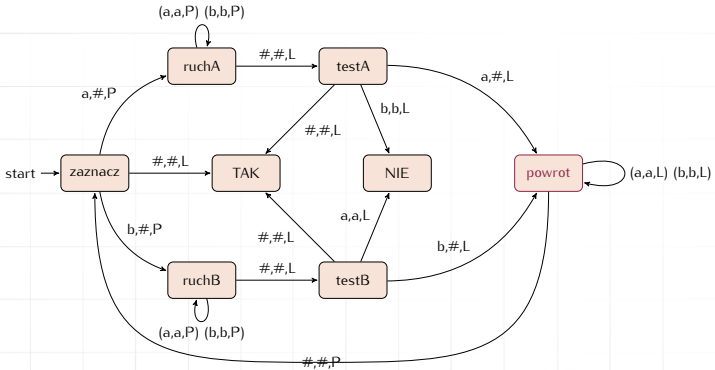


Maszyna Turinga

Przykład (3)

…	#	#	#	b	a	#	#	#	…

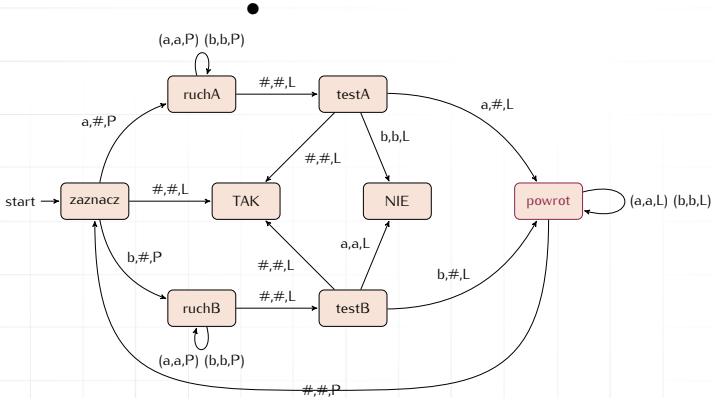
●



Maszyna Turinga

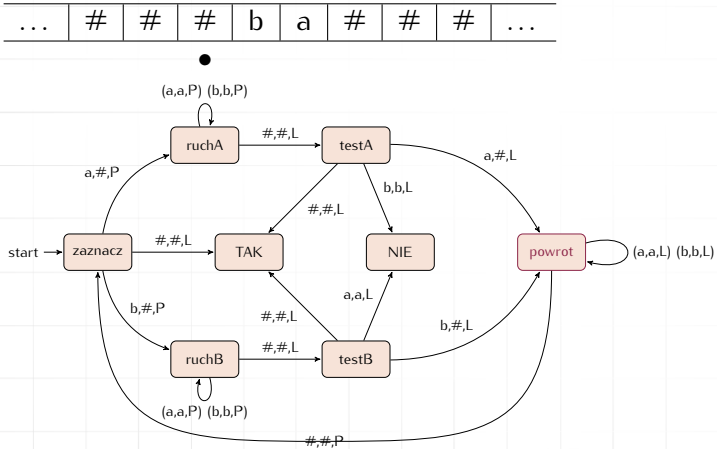
Przykład (3)

...	#	#	#	b	a	#	#	#	...



Maszyna Turinga

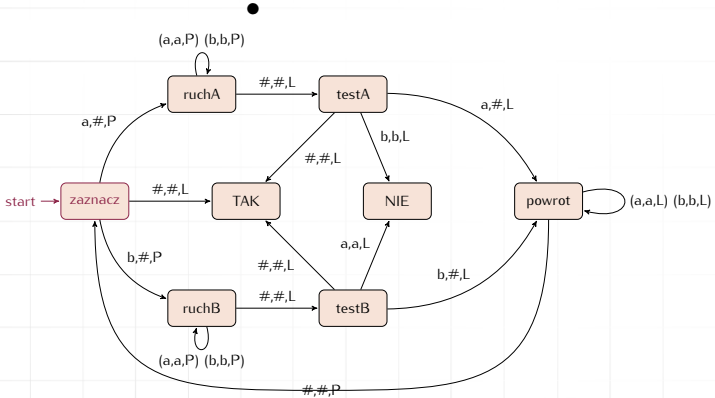
Przykład (3)



Maszyna Turinga

Przykład (3)

…	#	#	#	b	a	#	#	#	…

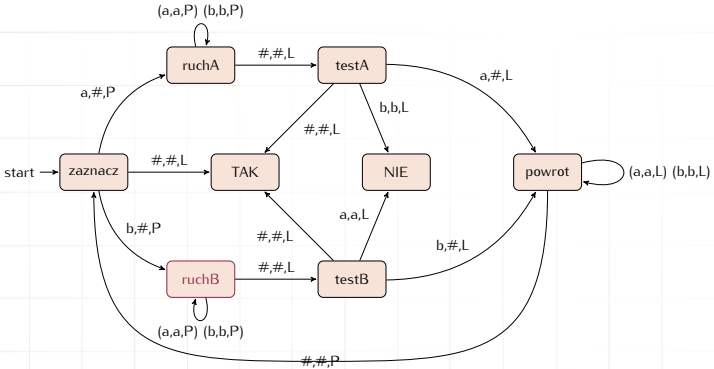


Maszyna Turinga

Przykład (3)

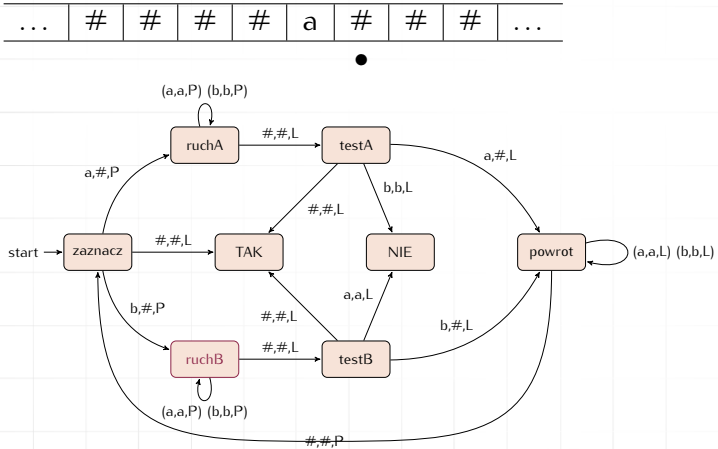
…	#	#	#	#	a	#	#	#	…

●



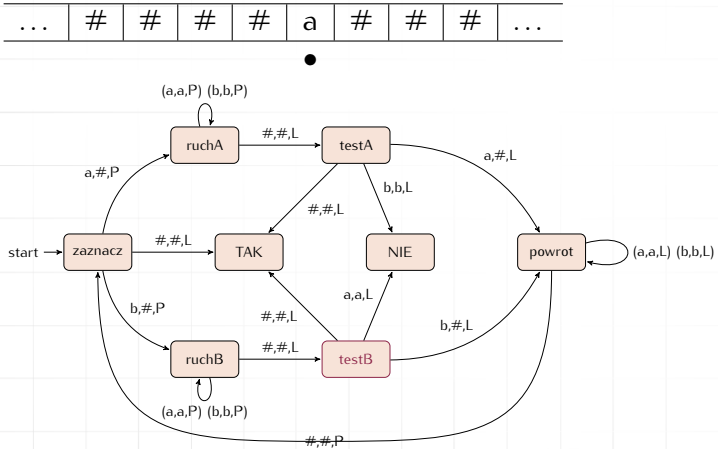
Maszyna Turinga

Przykład (3)



Maszyna Turinga

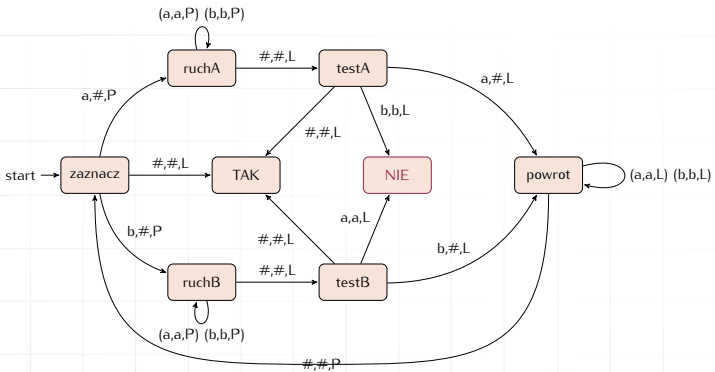
Przykład (3)



Maszyna Turinga

Przykład (3)

...	#	#	#	#	a	#	#	#	...



Maszyna Turinga

Warianty

- Taśma „jednostronnie” nieskończona (ma początek, nie ma końca).
- Dwie taśmy („wejściowa” i „wyjściowa”).
- Taśma dwuwymiarowa.
- Maszyna niedeterministyczna (w wariancie deterministycznym określony wyzwalacz w danym stanie powoduje zawsze taką samą reakcję; w wariancie niedeterministycznym dopuszcza się dla jednego wyzwalacza kilka różnych akcji wybieranych losowo).

Maszyna Turinga

Warianty

Wszystkie tak zmodyfikowane maszyny Turinga są sobie równoważne!

Maszyna Turinga

Warianty

Wszystkie tak zmodyfikowane maszyny Turinga są sobie równoważne! (to znaczy każdy problem, który można rozwiązać na jednej z nich można rozwiązać na każdej innej; albo inaczej: każda z tych maszyn może emulować każdą inną).

Maszyna Turinga

Warianty

Wszystkie tak zmodyfikowane maszyny Turinga są sobie równoważne! (to znaczy każdy problem, który można rozwiązać na jednej z nich można rozwiązać na każdej innej; albo inaczej: każda z tych maszyn może emulować każdą inną).

Pokazano, że komputer zaprojektowany przez Babbage'a — maszyna analityczna — jest równoważny z maszyną Turinga.

Maszyna Turinga

Do czego służy

1. Maszyna Turinga ma tylko skończoną liczbę stanów.
2. Programowanie jej nie musi być łatwe (spróbujcie skonstruować maszyną mnożącą liczby!)
3. Jej działanie zapewne będzie bardzo powolne, a sama symulacja (ręczna) jest dosyć nużąca.
4. Ale co tak na prawdę maszyna Turinga może zrobić?

Algorytm I

Przepis kucharski

- **Składniki:** 22 dag twardej czekolady półsłodkiej, 2 łyżki stołowe wody, $\frac{1}{4}$ filiżanki cukru pudru, 6 jajek rozdzielonych na żółtka i białka…
- **Przepis:**
  „Włóż czekoladę z dwiema łyżkami stołowymi wody do garnka o podwójnym dnie. Kiedy czekolada się rozpuści, domieszaj cukier puder; dodaj po trochu masło. Odstaw. Ubijaj żółtka około 5 minut, aż staną się gęste i nabiorą koloru cytrynowego. Delikatnie dołóż czekoladę. Ponownie lekko podgrzej, aby rozpuścić czekoladę, jeśli to będzie konieczne. Domieszaj rum i wanilię. Ubijaj białka aż do spienienia. Ubijając dodaj 2 łyżki stołowe cukru i ubijaj

Algorytm II

Przepis kucharski

dalej, aż utworzą się sztywne pagórki. Delikatnie połącz białka z masą czekoladowo-żółtkową. Wlej do oddzielnych naczyń, które będą podane na stół. Ochładzaj przez co najmniej 4 godziny. Wedle życzenia podawaj z bitą śmietaną. Wyjdzie z tego 6 do 8 porcji."

Algorytm Euklidesa

Oto jedna z jego wersji algorytmu Euklidesa:

Algorytm Euklidesa

Oto jedna z jego wersji algorytmu Euklidesa:
Dane są dwie dodatnie liczby całkowite m i n, należy znaleźć ich ***największy wspólny dzielnik (NWD)*** *tj. największą dodatnią liczbę całkowitą, która dzieli całkowicie zarówno m jak i n.*

Algorytm Euklidesa

Oto jedna z jego wersji algorytmu Euklidesa:
Dane są dwie dodatnie liczby całkowite m i n, należy znaleźć ich ***największy wspólny dzielnik (NWD)*** *tj. największą dodatnią liczbę całkowitą, która dzieli całkowicie zarówno m jak i n.*

1. *[Znajdowanie reszty] Podziel m przez n i niech r oznacza resztę z tego dzielenia. (Mamy $0 \leq r < n$.)*

Algorytm Euklidesa

Oto jedna z jego wersji algorytmu Euklidesa:
Dane są dwie dodatnie liczby całkowite m i n, należy znaleźć ich ***największy wspólny dzielnik (NWD)*** *tj. największą dodatnią liczbę całkowitą, która dzieli całkowicie zarówno m jak i n.*

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.*

Algorytm Euklidesa

Oto jedna z jego wersji algorytmu Euklidesa:
Dane są dwie dodatnie liczby całkowite m i n, należy znaleźć ich ***największy wspólny dzielnik (NWD)*** *tj. największą dodatnią liczbę całkowitą, która dzieli całkowicie zarówno m jak i n.*

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.*

Dzielenie z resztą

Dygresja

$$\frac{5}{3} =$$

Dzielenie z resztą

Dygresja

$$\frac{5}{3} = 1\frac{2}{3}$$

Dzielenie z resztą

Dygresja

$$\frac{5}{3} = 1\frac{2}{3} = 1{,}6666(6)$$

Dzielenie z resztą

Dygresja

$$\frac{5}{3} = 1\frac{2}{3} = 1{,}6666(6) = \overset{0{,}0001}{1{,}6667}$$

Dzielenie z resztą

Dygresja

$$\frac{5}{3} = 1\frac{2}{3} = 1{,}6666(6) = \overset{0{,}0001}{1{,}6667} = 1 \quad \text{reszta} \quad 2$$

Dzielenie z resztą

Dygresja

$$\frac{5}{3} = 1\frac{2}{3} = 1{,}6666(6) \overset{0{,}0001}{=} 1{,}6667 \quad = 1 \quad \text{reszta} \quad 2$$

bo

$$1 \times 3 + 2 = 5$$

Algorytm Euklidesa

Przykład

m	n	r
24	44	

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

Przykład

m	n	r
24	44	24

$24/44 = 0$ r 24

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

Przykład

m	n	r
24	44	24

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

Przykład

m	n	r
24	44	24
44		

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

Przykład

m	n	r
24	44	24
44	24	

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

Przykład

m	n	r
24	44	24
44	24	20

44/24 = 1 r 20

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

Przykład

m	n	r
24	44	24
44	24	20

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

Przykład

m	n	r
24	44	24
44	24	20
24		

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

Przykład

m	n	r
24	44	24
44	24	20
24	20	

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

Przykład

m	n	r
24	44	24
44	24	20
24	20	4

$24/20 = 1$ r 4

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

Przykład

m	n	r
24	44	24
44	24	20
24	20	4

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

Przykład

m	n	r
24	44	24
44	24	20
24	20	4
20		

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

Przykład

m	n	r
24	44	24
44	24	20
24	20	4
20	4	

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

Przykład

m	n	r
24	44	24
44	24	20
24	20	4
20	4	0

$20/4 = 5$ r 0

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

Przykład

m	n	r
24	44	24
44	24	20
24	20	4
20	4	0

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

Słowo „algorytm” jest bardzo nowe (w pewnym sensie).

Algorytm

Słowo „algorytm” jest bardzo nowe (w pewnym sensie).

Pochodzi od nazwiska Muhammed ibn Musa Alchwarizmi — perskiego matematyka (IX w) i pierwotnie oznaczało (każde) obliczenia w dziesiętnym systemie obliczeniowym.

Algorytm

Słowo „algorytm” jest bardzo nowe (w pewnym sensie).

Pochodzi od nazwiska Muhammed ibn Musa Alchwarizmi — perskiego matematyka (IX w) i pierwotnie oznaczało (każde) obliczenia w dziesiętnym systemie obliczeniowym.

Algorytm to jednoznaczny przepis przetworzenia w skończonym czasie pewnych danych wejściowych do pewnych danych wynikowych. (Wikipedia)

Algorytm

Słowo „algorytm” jest bardzo nowe (w pewnym sensie).

Pochodzi od nazwiska Muhammed ibn Musa Alchwarizmi — perskiego matematyka (IX w) i pierwotnie oznaczało (każde) obliczenia w dziesiętnym systemie obliczeniowym.

Algorytm to jednoznaczny przepis przetworzenia w skończonym czasie pewnych danych wejściowych do pewnych danych wynikowych. (Wikipedia)

Czasami rezygnuje się z żądania skończoności. Czasami, jeżeli algorytm się nie kończy — nazywamy go metodą obliczeniową.

Algorithm

In mathematics, computing, linguistics and related subjects, an **algorithm** is a sequence of finite instructions, often used for calculation and data processing. It is formally a type of effective method in which a list of well-defined instructions for completing a task will, when given an initial state, proceed through a well-defined series of successive states, eventually terminating in an end-state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as probabilistic algorithms, incorporate randomness.

Skończoność

Po pierwsze **powinien być skończony**; oznacza to, że po skończonej (być może bardzo dużej) liczbie kroków algorytm się zatrzyma.[3] Pytanie pomocnicze: Co gwarantuje, że algorytm Euklidesa zakończy się w skończonej liczbie kroków? Procedura, która ma wszystkie cechy algorytmu poza skończonością nazywana jest *metodą obliczeniową*. Podaj przykłady metod obliczeniowych realizowanych przez rzeczywiste komputery.

[3] Ale sama skończoność to jednak za mało — z praktycznego punktu widzenia **dobry** algorytm powinien gwarantować, że obliczenia zostaną zakończone w skończonym ale **rozsądnym** czasie!

Cechy algorytmu II

Dobre zdefiniowanie

Po drugie **powinien być „dobrze zdefiniowany”**. Każdy krok algorytmu musi być opisany precyzyjnie. Wszystkie możliwe przypadki powinny być uwzględnione, a podejmowane akcje dobrze opisane.[4] Oczywiście język naturalny nie jest wystarczająco precyzyjny — może to prowadzić do nieporozumień. z tego powodu używa się bardziej formalnych sposobów zapisu algorytmów, aż po języki programowania...

[4] Zwracam też uwagę, że algorytmy kucharskie nie są odpowiednio precyzyjne: co to znaczy „lekko podgrzej”?

Cechy algorytmu III

Dane wejściowe

Po trzecie **powinien mieć precyzyjnie zdefiniowane dane wejściowe**. Pewne algorytmy mogą nie mieć danych wejściowych (mieć zero danych wejściowych). Dane wejściowe to wartości, które muszą być zdefiniowane zanim rozpocznie się wykonanie algorytmu.

Cechy algorytmu IV

Dane wyjściowe

Po czwarte **zdefiniowane dane wyjściowe**. Daną wyjściową algorytmu Euklidesa jest liczna *n* która jest naprawdę największym wspólnym dzielnikiem danych wejściowych. Osobną sprawą jest pokazanie skąd wynika, że wynik algorytmu Euklidesa jest rzeczywiście NWD liczb *m* i *n*.

Cechy algorytmu V

Efektywność

Po piąte algorytm **powinien być określony efektywnie** to znaczy jego operacje powinny być wystarczająco proste by można je (teoretycznie?) wykonać w skończonym czasie z wykorzystaniem kartki i ołówka.

Teza Churcha-Turinga

Każdy problem algorytmiczny, dla którego możemy znaleźć algorytm dający się zaprogramować w pewnym — dowolnym języku, wykonujący się na pewnym, dowolnym komputerze, nawet na takim, którego jeszcze nie zbudowano, ale można zbudować, i nawet na takim, który wymaga nieograniczonej ilości czasu i pamięci dla coraz większych danych, jest także rozwiązywalny przez maszynę Turinga.

Teza Churcha-Turinga

Każdy problem algorytmiczny, dla którego możemy znaleźć algorytm dający się zaprogramować w pewnym — dowolnym języku, wykonujący się na pewnym, dowolnym komputerze, nawet na takim, którego jeszcze nie zbudowano, ale można zbudować, i nawet na takim, który wymaga nieograniczonej ilości czasu i pamięci dla coraz większych danych, jest także rozwiązywalny przez maszynę Turinga.

Uwaga!

Jest to TYLKO teza, nie twierdzenie!
Powyższe sformułowanie jest jednym z wielu!

Konsekwencje tezy Churcha-Turinga

1. „Domowy komputer” jest równoważny superkomputerowi z wielkiego centrum obliczeniowego (co nie znaczy, że rozwiąże ten sam problem w tym samym czasie!)
2. Wszystkie języki programowania są sobie równoważne (to znaczy to co można zaprogramować w jednym — można w każdym innym)
3. ...

Dziękuję za uwagę

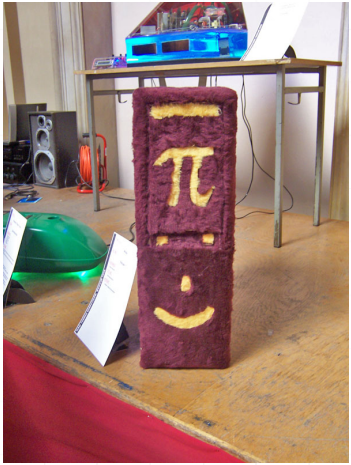
Dziękuję za uwagę

Co to jest komputer?

No właśnie, co?

Co to jest komputer?

Co to jest komputer?



Co to jest komputer?

Co to jest komputer?

Co to jest komputer?

Co to jest komputer?

Co to jest komputer?

Co to jest komputer?

Co to jest komputer?

Hmmm...

John von Neumann

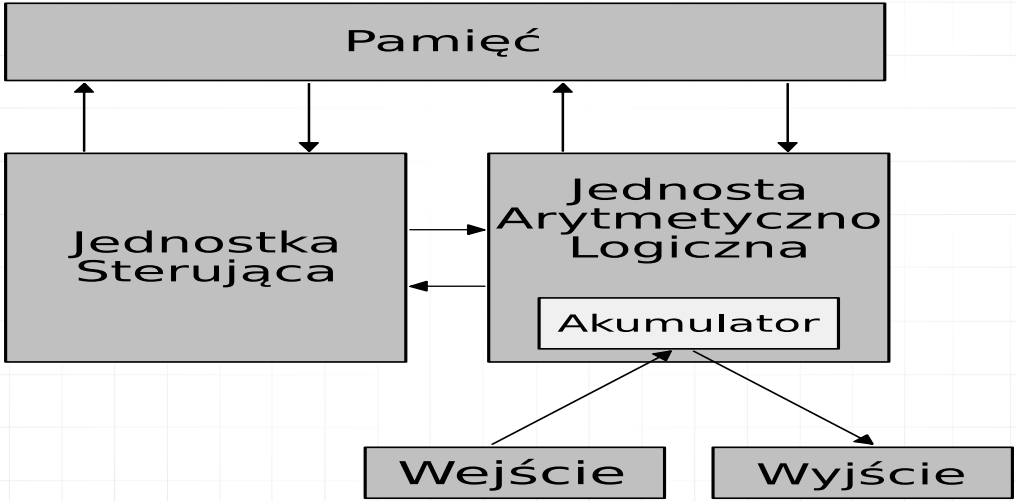
John von Neumann (ur. 28 grudnia 1903 w Budapeszcie, zm. 8 lutego 1957 w Waszyngtonie), inżynier chemik, fizyk, matematyk i informatyk. Wniósł znaczący wkład do wielu dziedzin matematyki, szczególnie teorii gier i uporządkował formalizm matematyczny mechaniki kwantowej. Uczestniczył w projekcie Manhattan. Przyczynił się do rozwoju numerycznych prognoz pogody.

Architektura von Neumanna

Architektura von Neumanna — rodzaj architektury komputera, przedstawionej po raz pierwszy w 1945 roku przez von Neumanna stworzonej wspólnie z Johnem W. Mauchly'ym i Johnem Presper Eckertem.
Polega na ścisłym podziale komputera na trzy podstawowe części:

- procesor (w ramach którego wydzielona bywa część sterująca oraz część arytmetyczno-logiczna)
- pamięć komputera (zawierająca dane i sam program)
- urządzenia wejścia/wyjścia

Architektura von Neumanna



Architektura von Neumanna

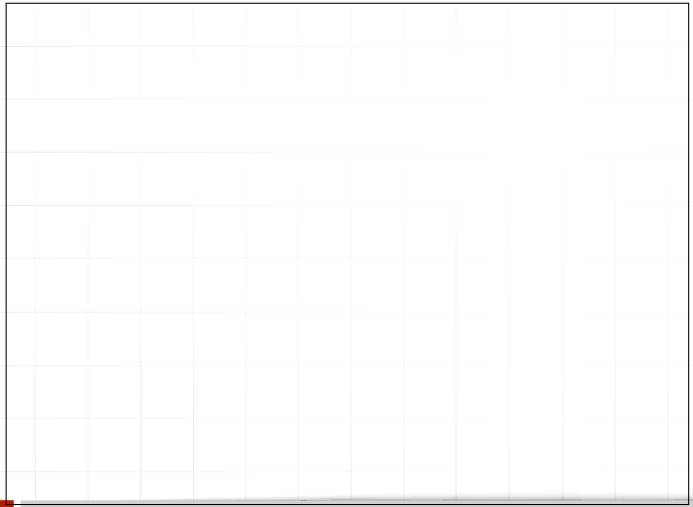
System komputerowy zbudowany w oparciu o architekturę von Neumanna powinien:

- mieć skończoną i funkcjonalnie pełną listę rozkazów
- mieć możliwość wprowadzenia programu do systemu komputerowego poprzez urządzenia zewnętrzne i jego przechowywanie w pamięci w sposób identyczny jak danych
- dane i instrukcje w takim systemie powinny być jednakowo dostępne dla procesora
- informacja jest tam przetwarzana dzięki sekwencyjnemu odczytywaniu instrukcji z pamięci komputera i wykonywaniu tych instrukcji w procesorze.

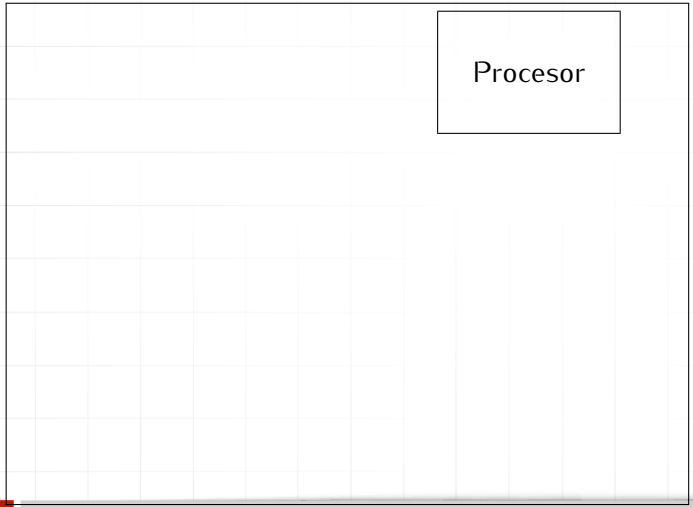
Architektura von Neumanna

Podane warunki pozwalają przełączać system komputerowy z wykonania jednego zadania (programu) na inne bez fizycznej ingerencji w strukturę systemu, a tym samym gwarantują jego uniwersalność.
System komputerowy von Neumanna nie posiada oddzielnych pamięci do przechowywania danych i instrukcji. Instrukcje jak i dane są zakodowane w postaci liczb. Bez analizy programu trudno jest określić czy dany obszar pamięci zawiera dane czy instrukcje. Wykonywany program może się sam modyfikować traktując obszar instrukcji jako dane, a po przetworzeniu tych instrukcji — danych — zacząć je wykonywać.

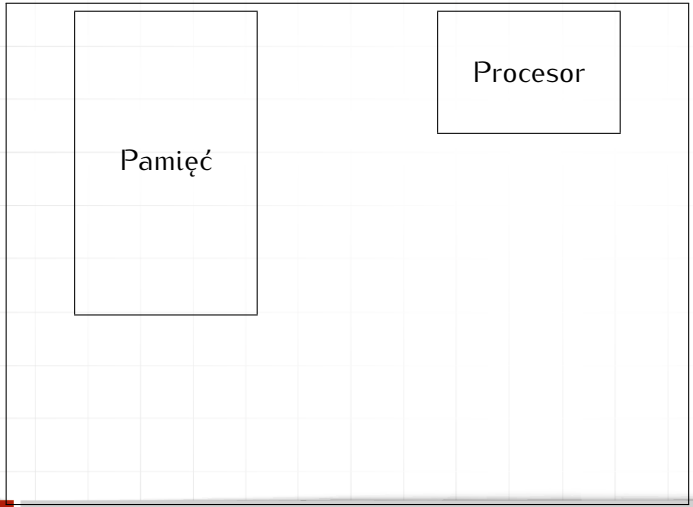
Schemat komputera



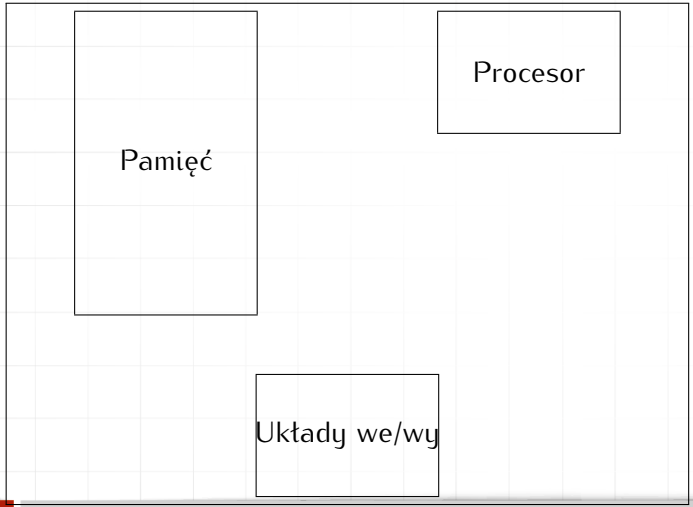
Schemat komputera



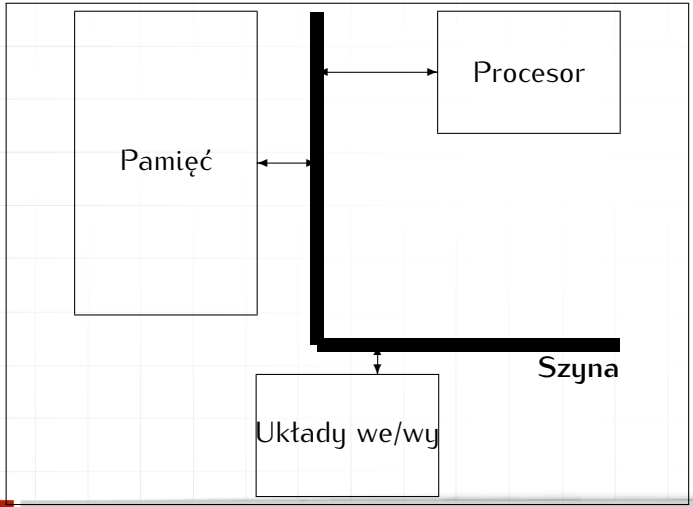
Schemat komputera



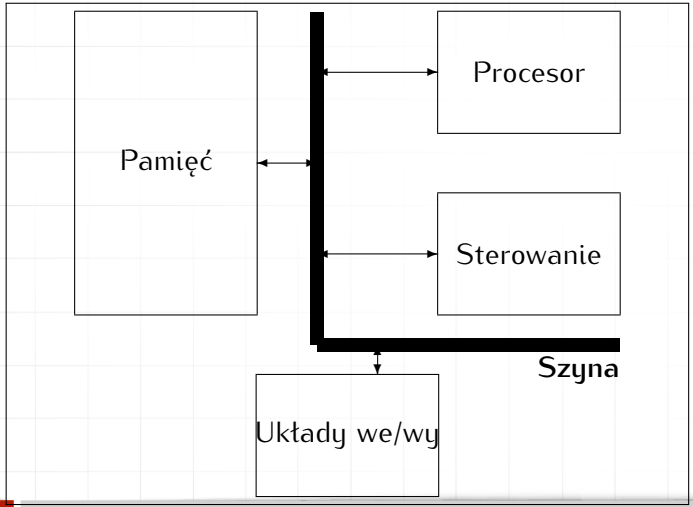
Schemat komputera



Schemat komputera



Schemat komputera



Schemat komputera I

W modelu tym wyróżniamy:

- Procesor.
- Pamięć (i pod tym pojęciem rozumiemy wszystkie rodzaje pamięci: Cache, RAM, ROM, dyski, dyskietki, dyski wymienne zapisywalne i nie).
- Urządzenie wejścia/wyjścia (wszystkie urządzenia pozwalające na kontakt ze „światem zewnętrznym”: klawiatura, mysz, karta graficzna, drukarka, czytniki różnego rodzaju).
- Sterowanie (wszystkie układy elektroniczne zapewniające właściwą komunikację wszystkich wymienionych wyżej urządzeń ze sobą i zapewniający uporządkowany przepływ informacji po szynie/szynach).
- Magistrala (szyna) — to wszystkie „drogi” łączące wymienione wyżej urządzenia.

Spojrzenie systemowca

Komputer to automat.

Spojrzenie systemowca

Komputer to automat.

1. Co to jest automat (cd)

   Automat *(2008) — urządzenie, maszyna lub ich zestaw, wykonujące samoczynnie cykl czynności lub operacji określony konstrukcją lub programem, nie wymagające bezpośredniego udziału człowieka.*

Spojrzenie systemowca

Komputer to automat.

1. Co to jest automat (cd)

   Automat *(2008) — urządzenie, maszyna lub ich zestaw, wykonujące samoczynnie cykl czynności lub operacji określony konstrukcją lub programem, nie wymagające bezpośredniego udziału człowieka.*

2. Automat (po angielsku)

   An ***automaton*** *(plural: automata or automatons) is a self-operating machine. The word is sometimes used to describe a robot, more specifically an autonomous robot. Used colloquially, it refers to a mindless follower.*

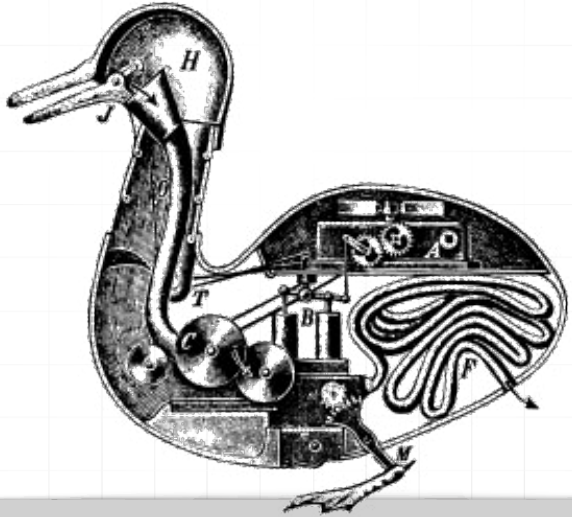
Automaty — też historia

The Canard Digérateur, or Digesting Duck, was an automaton in the form of duck, created by Jacques de Vaucanson in **1739**. The mechanical duck appeared to have the ability to eat kernels of grain, and to metabolise and defecate them. While the duck did not actually have the ability to do this—the food was collected in one inner container, and the feces being 'produced' from a second, so that no actual digestion took place—Vaucanson hoped that a truly digesting automaton could one day be designed.

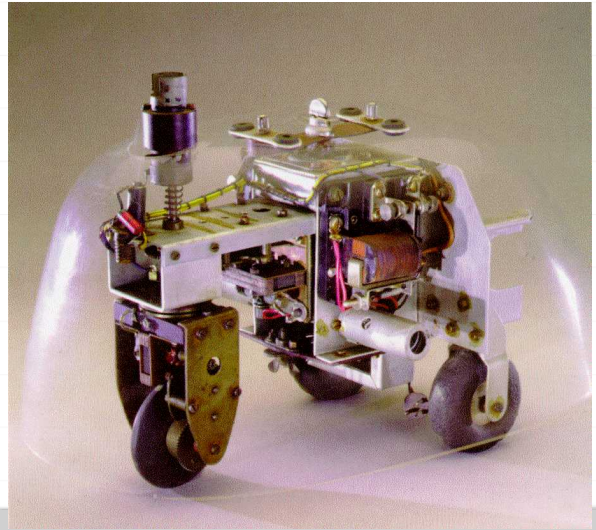
Voltaire wrote that "*without [...] the duck of Vaucanson, you have nothing to remind you of the glory of France.*" ("Sans...le canard de Vaucanson vous n'auriez rien qui fit ressouvenir de la gloire de la France.") This is often misquoted as "Without the shitting duck, we would have nothing to remind us of the glory of France."

Za Wikipedią (Digesting Duck)

Digesting Duck



Cybernetyczny żółw Greya Waltera



Trochę historii...

1. Krosna Jacquarda (1804)
2. Pierwszy program — Ada Lovelace dla maszyny analitycznej Charlesa Babbage'a (1843)
3. George Bool i algebra Boole'a (1847)
4. Zuse — Plankalkul — pierwszy prawdziwy język programowania komputerów (lata 40)
5. John Von Neumann (1945)

Back

Charles Babbage

ur. 26 grudnia 1791 r. – zm. 18 października 1871

Ukończył uniwersytet w Cambridge. Matematyk. Zajmował się różnymi sprawami, w tym organizacją produkcji w fabrykach. Zwolennik wysokiej specjalizacji siły roboczej.

Poświęcił 35 lat życia na konstruowanie maszyn liczących:

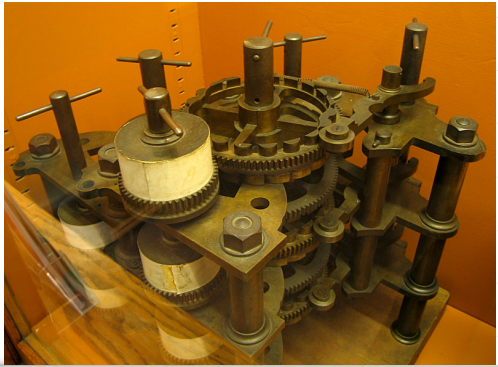
- Maszyna różnicowa
- Drukarka do maszyny różnicowej
- Maszyna analityczna

Żadna z nich nie ujrzałą światła dzennego, za jego życia.

Maszyna różnicowa

Rodzaj kalkulatora opertego na metodach różnicowych.

Zrobiona z brązu. Składała się z 25 000 elementów. Ważyła 13 600 kg. Nie została nigdy ukończona.



Fragment złożony po śmierci Babbage'a przez jego syna z części znalezionych w szopie.

Maszyna analityczna I

Mechaniczny komputer ogólnego zastosowania. Pierwsza idea — 1837, nie został ukończony do śmierci (1871).

Główne zadanie — tablicowanie funkcji logarytmicznych i trygonometrycznych przybliżanych wielomianami interpolacyjnymi.

Napędzany silnikiem parowym. Programowany za pomocą kart perforowanych (krosna Jacquarda!). Urządzenia wyjściowe: drukarka, ploter i dzwonek. Pamięć: 1000 liczb 50. cyfrowych. Arytmometr wykonywał cztery operacje, porównania i pierwiastki kwadratowe.

Język programowania zbliżony do współczesnych języków programowania (MARIE!).

Ocenia się, że koncept był znacznie bardziej zaawansowany niż późniejsze (lata 40. XX w) konstrukcje.

Maszyna analityczna II

Ideę Babbage'a najlepiej zrozumiałą Ada Augusta hrabina Lovelance — napisała pierwszy program (i została uznana za pierwszą programistkę).

Podjęto próbę realizacji maszyny. Zaangażowany w nią był syn Babbage'a.

Czasy współczesne

1. Podjęto szereg prób realizacji komputerów Babbage'a
   - Computer History Museum, Mountain Viev, CA i filmik(i) na YouTube
   - Z klocków Lego
   - Z elementów Meccano

Back

Kolofon

Prezentacja złożona w systemie LaTeX 2_ε z wykorzystaniem klasy beamer. Użyto fontu MS Trebuchet. Ilustracja na stronie tytułowej jest fragmentem zdjecia, przedstawiającego pomnik Alana Mathisona Turinga znajdujący się w Bletchley Park — centrum brytyjskiej kryptografii z czasów Drugiej Wojny Światowej. Coakley, Garrett. 2009. Alan Mathison Turing. Lipiec 18. Flickr. http://www.flickr.com/photos/garrettc/3777325029/.