

Wstęp do informatyki

Wykład 1

Uniwersytet Wrocławski

Instytut Informatyki

Program wykładu

- Program, literatura – p. zapisy.ii.uni.wroc.pl, kurs na skos.ii.uni.wroc.pl
- WdI a Wstęp do programowania
- Pytania?
- Komentarze?
- Uwagi?

Zasady


- Ćwiczenia – deklaracje, kolokwia (8 grudzień i 12 styczeń)
- Egzamin: premie za ocenę z ćwiczeń
- Szczegóły: p. kurs na skos.ii.uni.wroc.pl

Tematyka WdI?

- **Algorytmy** – projektowanie, **analiza**, techniki
- Złożoność algorytmu: czas, pamięć obliczeń
- **Struktury danych**
- **Program komputerowy** – struktura, implementacja, poprawność, testowanie
- Translacja programów i kod maszynowy
- Reprezentacja i arytmetyka binarna, cyfrowa reprezentacja danych (nie tylko liczb!)
- Grafy: pojęcia, problemy algorytmiczne
- Opis składni języków programowania
- Obliczalność...

Od problemu do rozwiązania

- Problem
- Specyfikacja
- Algorytm
- Program komputerowy
- Poprawność – zgodność ze specyfikacją
- Złożoność: czas, pamięć, energia, ...



Analiza
programu,
algorytmu

Problem

- Specyfikacja problemu:
 - **Wejście**
 - **Wyjście** („wynik”)
- Przykład 1:
 - **Wejście**: a, b – liczby
 - **Wyjście**: c – suma a i b
- Przykład 2:
 - **Wejście**: tekst T i słowo s
 - **Wyjście**: odpowiedź TAK wtedy i tylko wtedy gdy (wtw) s występuje w T

Specyfikacja problemu - przykład

PIERWIASTKI RÓWNANIA KWADRATOWEGO

Specyfikacja v.1:

Wejście: a, b, c – liczby rzeczywiste, $a \neq 0$

Wyjście: liczba rozwiązań równania $ax^2+bx+c=0$

Specyfikacja v.2 (bardziej formalna):

Wejście: a, b, c – liczby rzeczywiste, $a \neq 0$

Wyjście: liczba elementów zbioru

$$\{ x \in \mathbb{R} : ax^2+bx+c=0 \}$$

Specyfikacja problemu - przykład

WYSZUKIWANIE WZORCA W TEKŚCIE

Specyfikacja, v. 1

- **Wejście:** tekst T i słowo s
- **Wyjście:** odpowiedź TAK wtw s występuje w T

Specyfikacja, v. 2 (bardziej precyzyjna)

- **Wejście:** tekst $T = T_1 \dots T_n$ i słowo $s = s_1 \dots s_m$,
gdzie $T_1, \dots, T_n, s_1, \dots, s_m$ to litery
- **Wyjście:** odpowiedź TAK wtw istnieje $i \leq n - m + 1$,
takie że $T_i T_{i+1} \dots T_{i+m-1} = s_1 \dots s_m$

Specyfikacja problemu - przykład

MAXIMUM

Specyfikacja v.1:

Wejście: n – liczba naturalna większa od 0,
 a_1, a_2, \dots, a_n – ciąg liczb rzeczywistych

Wyjście: największy element ciągu a_1, a_2, \dots, a_n

Specyfikacja v.2:

Wejście: n – liczba naturalna większa od 0;
 a_1, a_2, \dots, a_n – ciąg liczb rzeczywistych

Wyjście: b takie, że $b \geq a_i$ dla każdego $i \in \{1, \dots, n\}$
oraz $b = a_k$ dla pewnego $k \in \{1, \dots, n\}$

Algorytm

- Algorytm: „skończony ciąg jasno zdefiniowanych **czynności**, koniecznych do **rozwiązania** problemu, w szczególności **przez komputer**”
- Jakie „**czynności**” potrafi wykonywać komputer?
 - Kontakt ze światem zewnętrznym: pobranie danych do **pamięci** (**wejście**) i udostępnienie wyników (**wyjście**)
 - Zapisywanie danych (liczb, tekstów, obrazków,...) i wyników obliczeń w **pamięci**
 - Operacje arytmetyczne (u nas: dodawanie, odejmowanie, mnożenie, dzielenie)
 - Porównania i wybór różnych dróg rozwiązania (rozgałęzienia)

Zapis algorytmu - schemat blokowy

Specyfikacja

Wejście:

a, b, c – liczby
rzeczywiste, $a \neq 0$

Wyjście:

liczba rozwiązań
równania

$$ax^2+bx+c=0$$

Zapis algorytmu - schemat blokowy

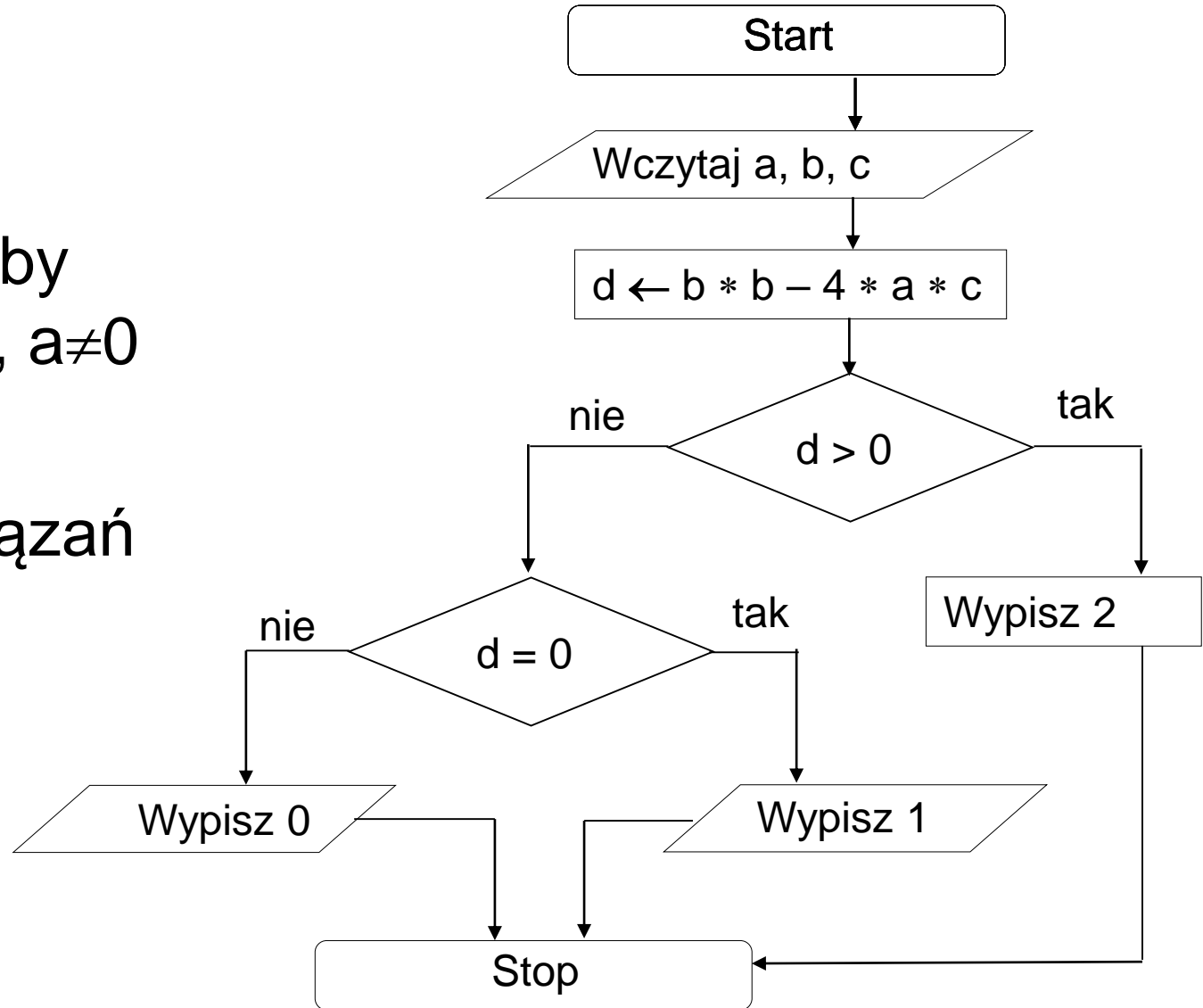
Specyfikacja

Wejście:

a, b, c – liczby
rzeczywiste, $a \neq 0$

Wyjście:

liczba rozwiązań
równania
 $ax^2+bx+c=0$



Zapis algorytmu - pseudokod

Specyfikacja

Wejście: a, b, c – liczby rzeczywiste, $a \neq 0$

Wyjście: liczba rozwiązań równania $ax^2+bx+c=0$

Algorytm

1. Wczytaj a, b, c
2. $d \leftarrow b * b - 4 * a * c$
3. Jeżeli ($d > 0$): wypisz 2, w przeciwnym razie:
 - Jeżeli ($d = 0$): wypisz 1
 - w przeciwnym razie wypisz 0

Zapis algorytmu

Schemat blokowy:

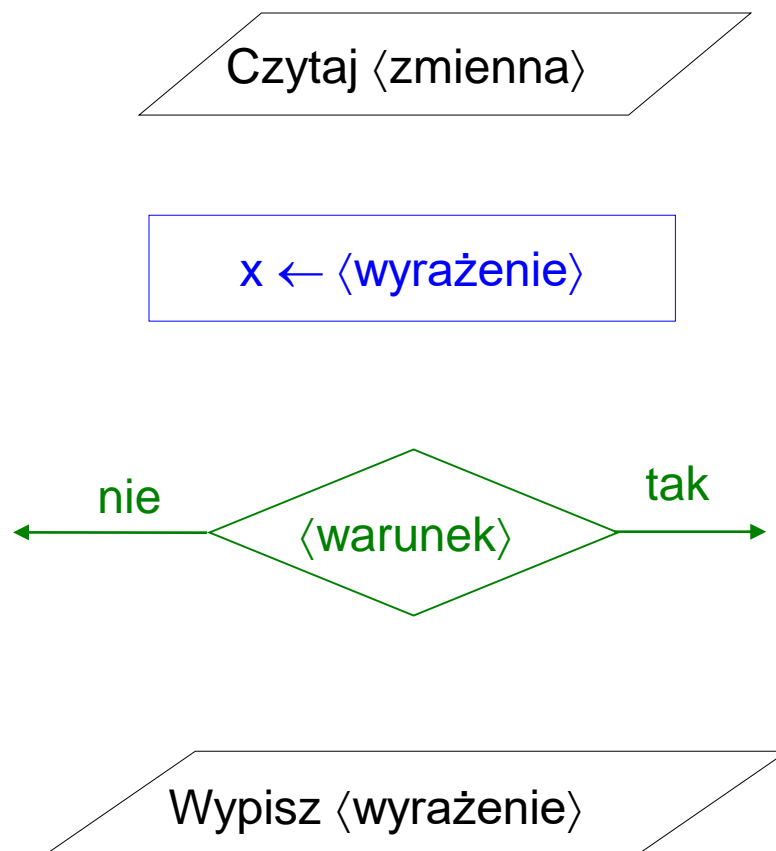
- Odczyt z „urządzenia wejściowego”
- Zapis na „urządzenie wyjściowe”
- Rozgałęzienia – porównania
- **Podstawienie** – nadaj zmiennej x wartość równą **wyrażeniu**

$x \leftarrow \langle \text{wyrażenie} \rangle$

zmienna - „miejsce w pamięci”
identyfikowane nazwą

Schemat blokowy formalnie

- Wartość z wejścia zapisz w $\langle \text{zmienna} \rangle$
- Wylicz wartość $\langle \text{wyrażenie} \rangle$, wynik zapisz w zmiennej x
- Przejdź w prawo/lewo gdy $\langle \text{warunek} \rangle$ jest/nie jest spełniony
- Wartość $\langle \text{wyrażenie} \rangle$ wypisz na wyjście



Schemat blokowy (nie)formalnie

UWAGA 1

Dla wygody wykładowcy, operacje wejścia i wyjścia będą czasem oznaczane prostokątami

Wypisz $\langle \text{wyrażenie} \rangle$

Czytaj $\langle \text{zmienna} \rangle$

UWAGA 2

Zastosowanie nawiasów $\langle \rangle$ oznacza pewną rodzinę pojęć/napisów.

Np. $\langle \text{wyrażenie} \rangle$ oznacza możliwość wpisania w podane miejsce dowolnego wyrażenia.

Algorytm z pętlą

Specyfikacja

Wejście: n – liczba naturalna większa od zera,
 a_1, \dots, a_n – ciąg liczb

Wyjście: największy element ciągu $a_1 \dots a_n$

Algorytm v.1

1. Czytaj n
2. Czytaj mx
3. **Powtórz** $n - 1$ razy:
 - Czytaj a
 - Jeżeli $a > mx$: $mx \leftarrow a$
4. Wypisz mx

Pytanie: dlaczego w algorytmie nie używamy notacji a_1, \dots, a_n ?

Algorytm a specyfikacja

Ważna uwaga!

Oznaczenia w specyfikacji są niezależne od nazw zmiennych algorytmu/programu

W szczególności:

W algorytmie nie używamy dolnych i górnych indeksów matematycznych”, np. notacji a_1, \dots, a_n !

(Dlaczego...?)

Algorytm z pętlą

Specyfikacja

Wejście: n – liczba naturalna większa od zera,
 a_1, \dots, a_n – ciąg liczb

Wyjście: największy element ciągu $a_1 \dots a_n$

Algorytm v.1

1. Czytaj n
2. Czytaj mx
3. **Powtórz** $n - 1$ razy:
 - Czytaj a
 - Jeżeli $a > mx$: $mx \leftarrow a$
4. Wypisz mx

Algorytm v.2

1. Czytaj n
2. Czytaj mx
3. **Dopóki** $n \neq 1$:
 - Czytaj a
 - Jeżeli $a > mx$: $mx \leftarrow a$
 - $n \leftarrow n - 1$
4. Wypisz mx

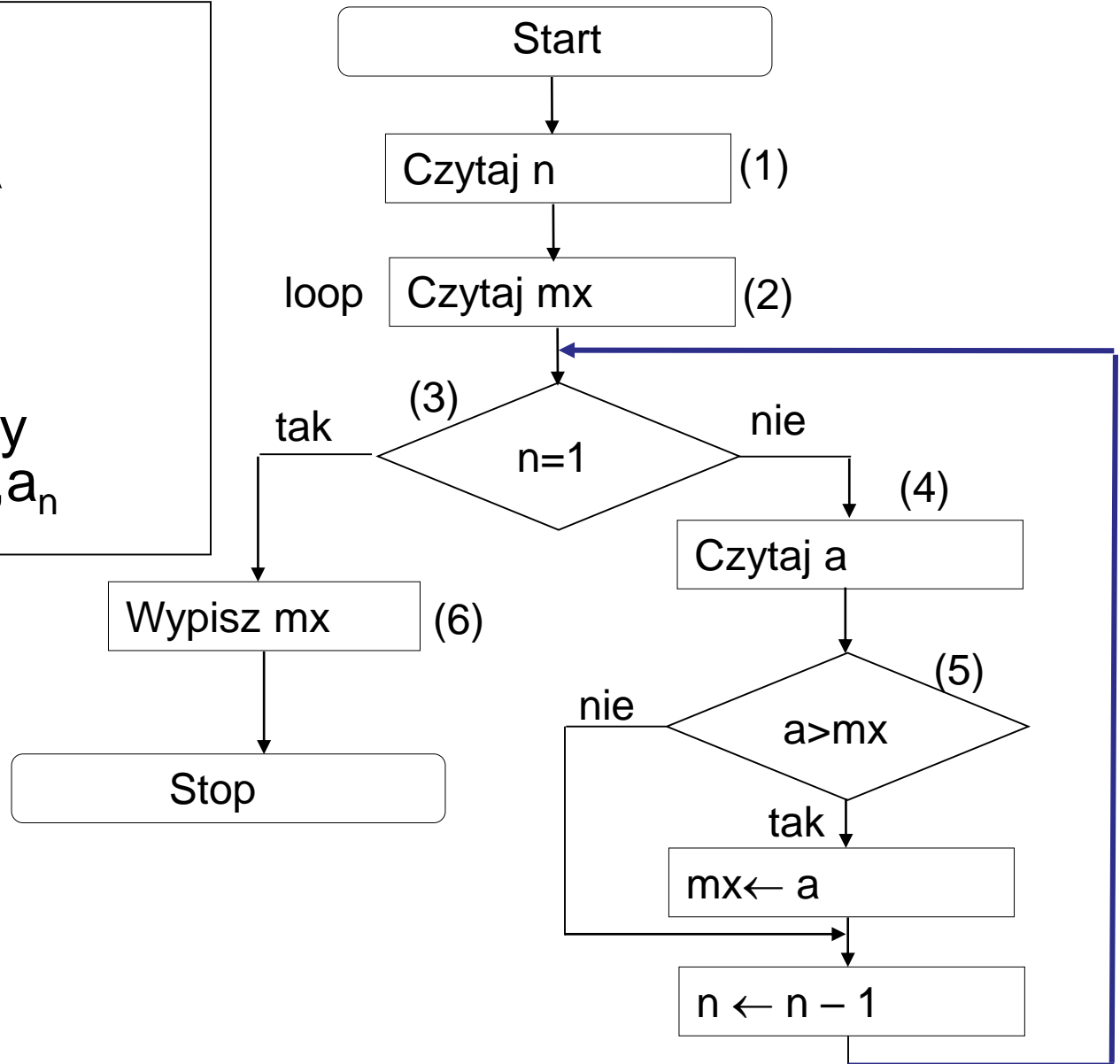
Algorytm z pętlą

Specyfikacja

Wejście:

n – liczba naturalna
większa od zera,
 a_1, \dots, a_n – ciąg liczb

Wyjście: największy
element ciągu a_1, \dots, a_n



Algorytm z **licznikiem** (i pętlą)

Specyfikacja

Wejście: n – liczba naturalna większa od zera,

a_1, \dots, a_n – ciąg liczb

Wyjście: liczba dodatnich elementów ciągu a_1, \dots, a_n

Algorytm v.1

1. Czytaj n
2. $ile \leftarrow 0$
3. **Powtórz** n razy:
 - Czytaj a
 - Jeżeli $a > 0$: $ile \leftarrow ile + 1$
4. Wypisz ile

Algorytm z licznikiem

Specyfikacja

Wejście: n – liczba naturalna większa od zera,

a_1, \dots, a_n – ciąg liczb

Wyjście: liczba dodatnich elementów ciągu a_1, \dots, a_n

Algorytm v.1

1. Czytaj n
2. $\text{ile} \leftarrow 0$
3. **Powtórz** n razy:
 - Czytaj a
 - Jeżeli $a > 0$: $\text{ile} \leftarrow \text{ile} + 1$
4. Wypisz ile

Algorytm v.2

1. Czytaj n
2. $\text{ile} \leftarrow 0$
3. **Dopóki** $n > 0$:
 - Czytaj a
 - Jeżeli $a > 0$:
 $\text{ile} \leftarrow \text{ile} + 1$
 - $n \leftarrow n - 1$
4. Wypisz ile

Algorytm z licznikiem

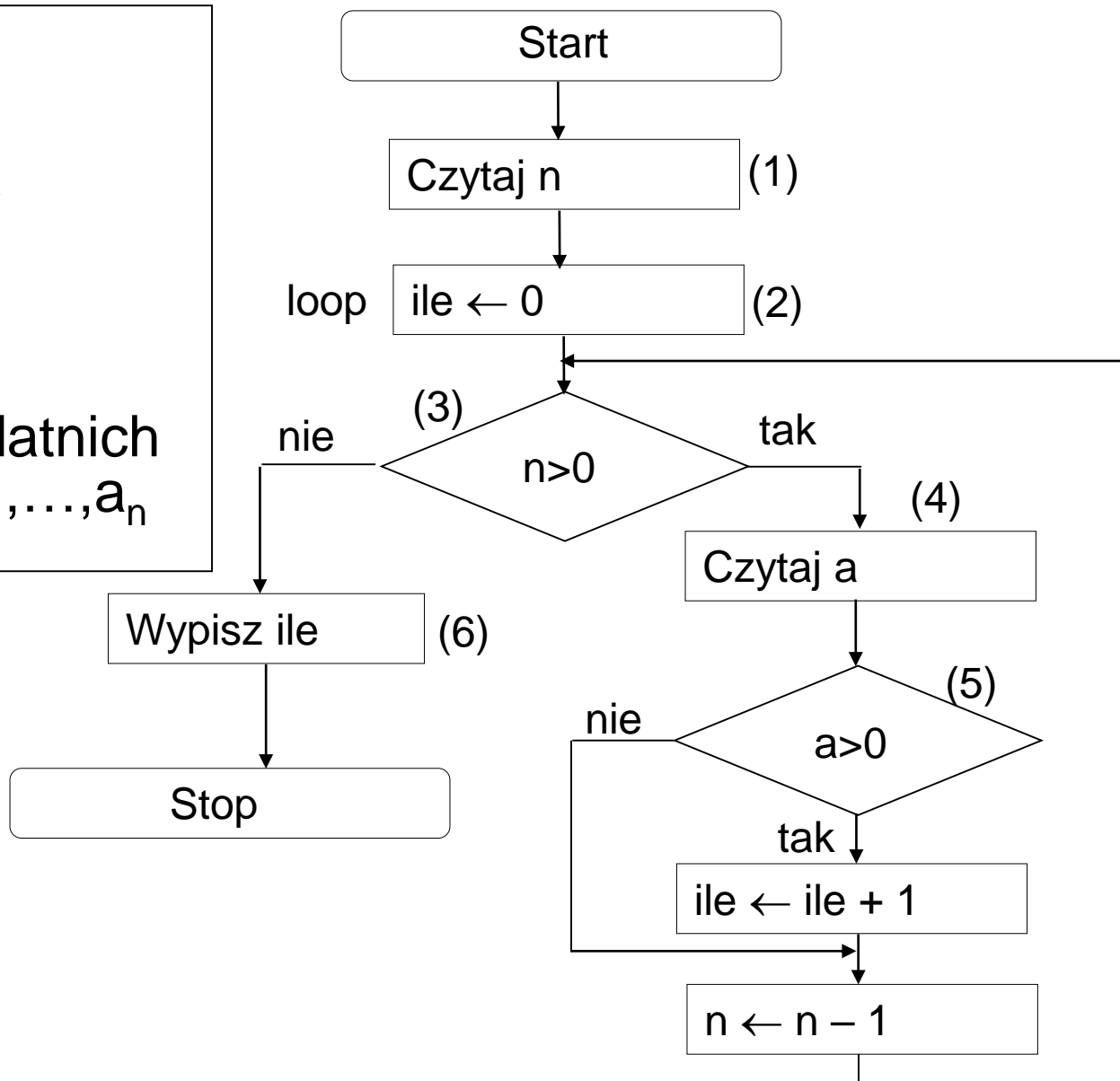
Specyfikacja

Wejście:

n – liczba naturalna
większa od zera,

a_1, \dots, a_n – ciąg liczb

Wyjście: liczba dodatnich
elementów ciągu a_1, \dots, a_n



Algorytmy – przykład historyczny

NAJWIĘKSZY WSPÓLNY DZIELNIK

Specyfikacja v. 1

Wejście: n, m – liczby naturalne

Wyjście: największy wspólny dzielnik n i m

Specyfikacja v. 2

Wejście: n, m – liczby naturalne

Wyjście: $\max \{ p : p \mid n \text{ oraz } p \mid m \}$

gdzie

- $a \mid b$ oznacza, że a jest dzielnikiem b .
- $\max \langle \text{zbiór} \rangle$ oznacza największy element w podanym zbiorze

Algorytmy – przykład historyczny

NAJWIĘKSZY WSPÓLNY DZIELNIK

Algorytm Euklidesa

Czytaj n, m

Jeżeli $n < m$: zamień(n, m)

Dopóki $m > 0$:

- $n \leftarrow n - m$

- Jeżeli $n < m$: zamień(n, m)

Wypisz n

Pyt. 1: Jak zrealizować zamień(n, m)? (ćwiczenia)

Pyt. 2: Czy ten algorytm działa poprawnie...?

Wyznacza $\text{nwd}(n, m)$?

O tym kiedy indziej...

Algorytm - złożoność

- **Złożoność czasowa:** liczba wykonanych instrukcji.
- **Złożoność pamięciowa:** liczba wykorzystanych zmiennych (**rozmiar** pamięci...?).

Od czego zależy złożoność czasowa/pamięciowa?

- **Rozmiar danych:** jak **dużo** elementów?, wartości elementów – wielkość?
- **Wartości danych:** **duże/male** liczby? liczby całkowite/rzeczywiste?
- Zestaw dostępnych „**instrukcji**”?

UWAGA: złożoność pamięciowa

złożoność pamięciowa i rozmiar danych to różne pojęcia, zazwyczaj zachodzi:

złożoność pamięciowa \neq rozmiar danych

!!!

Problem – rozmiar danych

1. Czy bardzo duża liczba całkowita zajmuje tyle samo miejsca ile liczba mała?
2. Liczby całkowite a liczby rzeczywiste?
3. Rozmiar tekstu? Obrazów? Wideo? Muzyki?

Ad. 1: dla uproszczenia, *zazwyczaj* będziemy ignorować wartości liczb (o ile nie mają wpływu na liczbę wykonanych instrukcji).

Ad. 2: o tym za parę tygodni...

Ad. 3: o tym raczej na innych przedmiotach.

Problem – rozmiar danych

Problem	Dane	Rozmiar
Suma dwóch liczb	a, b	2
Liczba pierwiastków równania kwadratowego	a, b, c	3
Wyszukiwanie wzorca w tekście	$n, m, T=T_1..T_n,$ $S=S_1..S_m$	$n + m + 2$
Maksimum ciągu liczb	n, a_1, \dots, a_n	$n + 1$
Największy wspólny dzielnik	n, m	2 ?? ???

Algorytm – złożoność

Złożoność czasowa

liczba wykonanych instrukcji („bloków” ze schematu blokowego)?

Problemy

Pyt.: czy operacje arytmetyczne na małych/dużych liczbach wykonują się tak samo szybko?

Odp.: niekoniecznie, ale zazwyczaj tak zakładamy.

Pyt.: które operacje naprawdę wykonują się „**w czasie jednostkowym**”? Jaki hardware (sprzęt)?

Pyt.: dużo operacji w jednym wyrażeniu arytmetycznym?

Algorytm – złożoność czasowa

MAXIMUM n liczb

Przykład 1:

$$a_1 < a_2 < \dots < a_n$$

Liczba instrukcji:

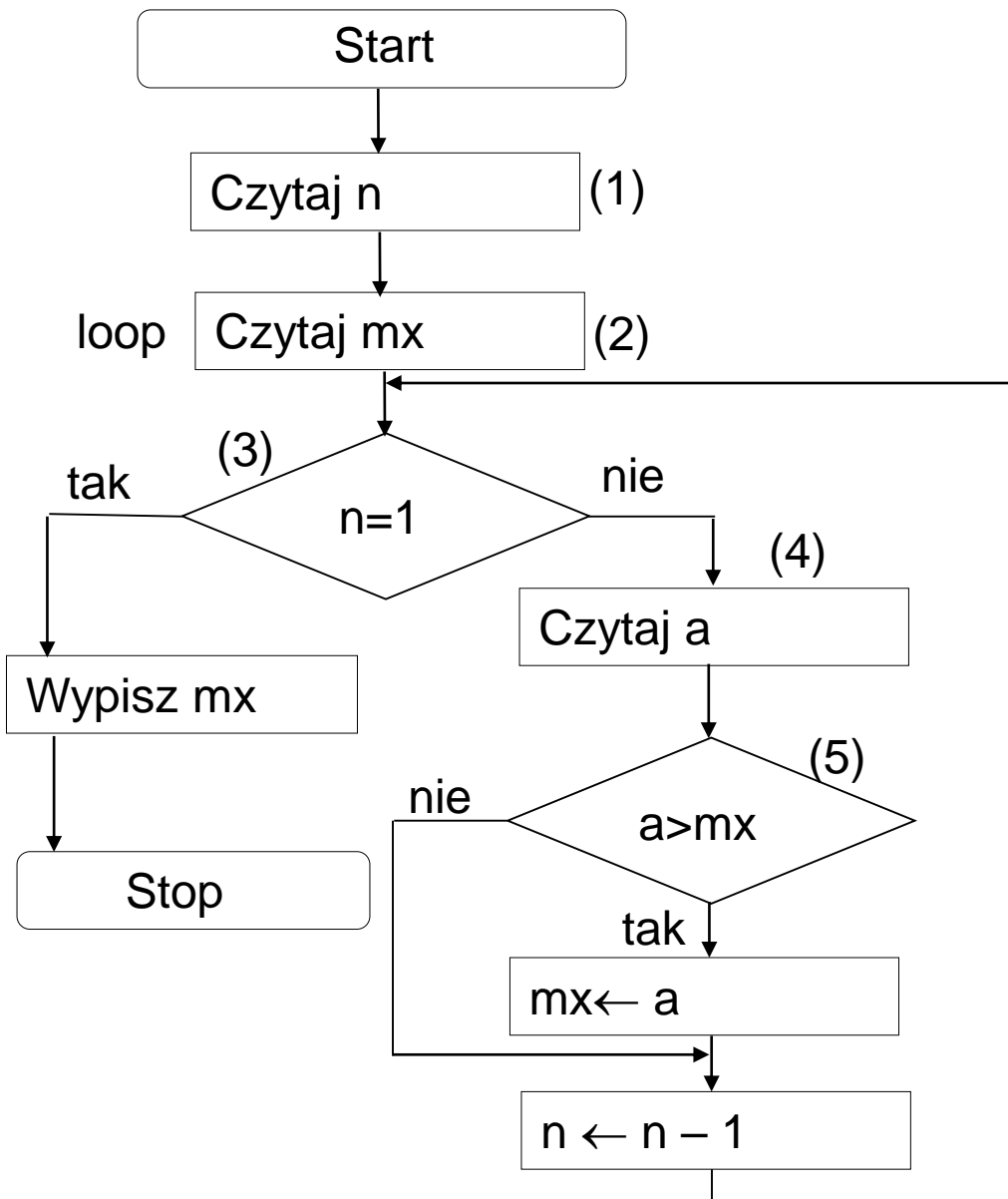
$$4 + 5(n - 1) = \mathbf{5n - 1}$$

Przykład 2:

$$a_1 > a_2 > \dots > a_n$$

Liczba instrukcji:

$$4 + 4(n - 1) = \mathbf{4n}$$



Algorytm – złożoność czasowa

Algorytm Euklidesa

Czytaj n, m

Jeżeli $n < m$: zamień(n, m)

Dopóki $m > 0$:

- $n \leftarrow n - m$

- Jeżeli $n < m$: zamień(n, m)

Wypisz n

Przykład 1: $n = m = 1\ 000\ 000$

Liczba instrukcji: **6** („mniej więcej”)

Przykład 2: $n = 1\ 000\ 000, m = 1$

Liczba instrukcji: **> 3 000 000**

Algorytm – złożoność czasowa

Problem	Rozmiar danych	Złożoność czasowa
Suma dwóch liczb	2	3
Liczba pierwiastków równania kwadrat.	3	4 lub 5
Maksimum ciągu n liczb	$n + 1$? $4n$? $5n - 1$
Największy wspólny dzielnik liczb n i m	2 ?? ???	??? $n+m$

Algorytm – złożoność czasowa

Złożoność najgorszego przypadku:

- Dla każdego rozmiaru danych n wyznaczamy „najgorsze” dane o tym rozmiarze, tj. takie, dla których algorytm działa najdłużej.
- Złożoność definiuje funkcja $T: \mathbb{N} \rightarrow \mathbb{N}$, taka że $T(n)$ jest równe czasowi działania algorytmu dla „najgorszych” danych rozmiaru n .

Przykład:

Złożoność najgorszego przypadku naszego algorytmu dla maximum, to $T(n)=5n - 1$.

Złożoność czasowa najgorszego przypadku

Problem	Rozmiar danych	Złożoność czasowa
Suma dwóch liczb	2	3
Liczba pierwiastków równania kwadrat.	3	4 lub 5
Maksimum ciągu liczb	$n + 1$	$4n$ $5n - 1$
Największy wspólny dzielnik	??	$4 + 3 n - m $

Podsumowanie: spis zagadnień

1. Problem i jego specyfikacja
2. Algorytm i używane w nim instrukcje: instrukcja warunkowa, podstawienie, instrukcje we/wy; operacje arytmetyczne
3. Konstrukcje algorytmiczne: pętla, licznik
4. Zapis algorytmu: schemat blokowy, pseudokod
5. Rozmiar danych dla problemu
6. Złożoność czasowa (i pamięciowa) algorytmu