POLITECHNIKA LUBELSKA
WYDZIAŁ ELEKTROTECHNIKI I INFORMATYKI
INFORMATYKA

Zintegrowany Program Rozwoju Politechniki Lubelskiej - część druga

WPROWADZENIE DO INFORMATYKI

Wykład organizacyjny

Dr hab. Małgorzata Charytanowicz

Fundusze Europejskie Wiedza Edukacja Rozwój
Rzeczpospolita Polska
Unia Europejska Europejski Fundusz Społeczny

Plan

- Organizacja zajęć
- Zasady zaliczenia
- Cele przedmiotu
- Treści programowe
- Literatura
- Ważne daty

Wprowadzenie do informatyki

Prowadzący:

- dr hab. Małgorzata Charytanowicz
- konsultacje: środa, 10.00-11.00, sobota 16.00-17.00, ul. Nadbystrzycka 36B, p. 110 (Pentagon)
- e-mail: m.charytanowicz@pollub.pl
- strona www: cs.pollub.pl/staff/mcharytanowicz/

Materiały:

- https://moodle.cs.pollub.pl/
- kurs „Wprowadzenie do informatyki 2019/2020”
- testy sprawdzające po wykładach, zamieszczane na platformie Moodle
- egzamin pisemny, zaliczenie od 51%
- skala ocen wg §19 Regulaminu studiów, www.pollub.pl/pl/news/get/id/8324

Zaliczenie przedmiotu

- Przed przystąpieniem do zaliczenia lub egzaminu student ma obowiązek okazać dokument ze zdjęciem dla potwierdzenia tożsamości i kartę okresowych osiągnięć, jeśli została wydana.
- Usprawiedliwieniem nieobecności na zaliczeniu lub egzaminie może być wyłącznie choroba, zdarzenie losowe albo inne ważne okoliczności uznane przez dziekana. Dokument stanowiący podstawę usprawiedliwienia powinien być dostarczony w ciągu 7 dni od daty zaliczenia i egzaminu, osobiście lub pocztą do dziekanatu.
- W przypadku nieobecności nieusprawiedliwionej na zaliczeniu lub egzaminie student otrzymuje ocenę niedostateczną.
- W przypadku niezaliczenia co najmniej jednej formy zajęć w ramach danego modułu lub przedmiotu, ocena końcowa z modułu lub przedmiotu jest oceną niedostateczną.

Zaliczenie przedmiotu

- W przypadku nieuzyskania zaliczenia zajęć student ma prawo do dwóch terminów zaliczeń poprawkowych z danego modułu i przedmiotu przed sesją lub w jej trakcie.
- Student, który nie przystąpił do zaliczenia poprawkowego, traci prawo do przywrócenia terminu poprawkowego i otrzymuje ocenę niedostateczną.
- Za zgodą prowadzącego zajęcia student może przystąpić do egzaminu przed sesją w terminie „zerowym”. Egzamin ten traktowany jest jako termin dodatkowy.
- Jeżeli podczas zaliczenia lub egzaminu prowadzący stwierdzi niesamodzielność pracy studenta, w szczególności korzystanie z niedozwolonych narzędzi lub materiałów, student uzyskuje ocenę niedostateczną.

Wprowadzenie do informatyki

Rodzaj przedmiotu	Kierunkowy
Rok/semestr	I/1
Liczba godzin	30 + 30
Liczba punktów ECTS	5
Sposób zaliczenia	Egzamin

Cele przedmiotu

- Zapoznanie studentów z podstawowymi technikami programowania i podstawami algorytmizacji, wprowadzenie do podstawowych algorytmów i struktur danych.
- Zapoznanie studentów z podstawowymi zagadnieniami teorii automatów i języków formalnych oraz maszyny Turinga.
- Zapoznanie studentów z elementami teorii złożoności obliczeniowej.

Treści programowe

- Informatyka, informacja, zadanie algorytmiczne, algorytm i sposoby jego zapisu.
- Podstawy algorytmizacji, schematy Nassi-Schneidermana.
- Język programowania, składnia, semantyka. Język proceduralny. Konstrukcje strukturalne i ich realizacja.
- Proste typy danych w językach programowania. Sposoby kodowania znaków. Liczby stałopozycyjne i zmiennopozycyjne. Strukturalne typy danych w językach programowania.
- Podejście zstępujące i wstępujące w programowaniu. Procedury i funkcje. Rekurencja w programowaniu. Problemy rozwiązywane z użyciem rekurencji. Modele danych.
- Podstawowe pojęcia ze złożoności obliczeniowej algorytmów. Złożoność obliczeniowa, funkcja złożoności obliczeniowej, rząd złożoności obliczeniowej.
- Podstawowe pojęcia lingwistyki matematycznej. Wzorce, automaty, wyrażenia regularne i gramatyki.
- Języki formalne, klasy P i NP, problemy NP-zupełne, teoria Turinga.

Literatura

- Abelson H., Sussman G. J., Sussman J., Struktura i interpretacja programów komputerowych. WNT, Warszawa 2002.
- Aho A. V., Ullman I. D., Projektowanie i analiza algorytmów. Helion, Gliwice 2003.
- Cormen T.H., Leirson Ch.E., Rivest R.L., Wprowadzenie do algorytmów. PWN, Warszawa 2013.
- Giaro K., Złożoność obliczeniowa algorytmów w zadaniach. Wyd. Politechniki Gdańskiej, Gdańsk 2011.
- Harel D., Rzecz o istocie informatyki. Algorytmika. WNT, Warszawa 2008.
- Homenda W., Elementy lingwistyki matematycznej i teorii automatów. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2005.

Literatura

- Hopcroft J.E., Motwani R., Ullman J.D., Wprowadzenie do teorii automatów, języków i obliczeń. PWN, Warszawa 2005.
- Kwiatkowska A., Łukasik E., Schematy zwarte NS. Przykłady i zadania. Mikom, Warszawa 2004.
- Martin R. C., Czysty kod. Podręcznik dobrego programisty. Helion, Warszawa 2004.
- Wirth N., Algorytmy i struktury danych = programy. WNT, Warszawa 2004.
- Ząbek Ś., Podstawy algorytmizacji i programowania. Wydawnictwo Uniwersytetu Marii Curie-Skłodowskiej, Lublin 2012.

Ważne daty

- Okres zajęć dydaktycznych

  01.10.2019 r. – 23.12.2019 r.
- Wakacje zimowe

  24.12.2019 r. – 07.01.2020 r.
- Okres zajęć dydaktycznych

  08.01.2020 r. – 02.02.2020 r.
- Sesja egzaminacyjna

  03.02.2020 r. – 16.02.2020 r.
- Przerwa międzysemestralna

  17.02.2020 r. – 21.02.2020 r.

Informacja i zasady jej zapisu

Plan

- Informacja i jej własności
- Teoria informacji
- Informatyka i jej działy
- Reprezentacja informacji
- Kodowanie znaków

Informacja

Informacja – łac. *Informatio* to wyobrażenie, wyjaśnienie (źródło: Encyklopedia PWN).

Informacja – pojęcie niedefiniowalne ze względu na jego pierwotny charakter, rozpatrywane w trzech aspektach: syntaktycznym, semantycznym, pragmatycznym.

Informacja – w języku potocznym konstatacja stanu rzeczy, wiadomość.

Podstawowe cechy informacji

- istnieje obiektywnie
- podlega przetworzeniu
- podlega przenoszeniu
- podlega powielaniu
- posiada różnorodne równoważne reprezentacje
- bywa fragmentaryczna
- podlega zniekształceniu
- podlega interpretacji

Jednostka informacji

- Bit (ang. binary digit, cyfra dwójkowa), oznaczany przez „b”. Wystarcza do zakomunikowania jednego z dwóch jednakowo prawdopodobnych zdarzeń.
- Bit przyjmuje jedną z dwóch wartości, które zwykle oznacza się jako „0” lub „1”.
- Jest to oznaczenie stosowane w matematyce:
  - „0” – fałsz,
  - „1” – prawda.

Teoria informacji

- **Claude Elwood Shannon** (30.04.1916 – 24.02.2001) amerykański matematyk i inżynier, profesor Massachusetts Institute of Technology .
- Jeden z twórców teorii informacji, zajmował się informacją w ujęciu ilościowym. Jako jeden z pierwszych docenił znaczenie kodu binarnego, twierdził, że ciągami zer i jedynek da się opisać tekst, obraz i dźwięk.
- W pracy *A mathematical theory of communication* (1948 r.) przedstawił najważniejsze zagadnienia z dziedziny teorii informacji. Jej celem były zastosowania techniczne związane z kodowaniem.
- Zajmując się zagadnieniem przepustowości linii telefonicznych, Shannon opracował ważne do dziś podstawy matematyczne, stanowiące podstawę teorii informacji.
- Jego twierdzenia nabrały szczególnego znaczenia praktycznego po wynalezieniu układów scalonych.
- Laureat Nagrody Kioto w dziedzinie nauk podstawowych z 1985 roku.

Teoria informacji

- Przesyłanie informacji wymaga ustalenia zrozumiałego zarówno przez nadajnik, jak i odbiornik zasobu znaków, co do których są one zgodne.
- Z punktu widzenia techniki najbardziej odpornym na zakłócenia i prostym w przetwarzaniu jest system binarny, w systemie tym możliwe jest prowadzenie rachunku logicznego – algebry Boole'a.
- Według teorii Shannona, każdemu znakowi odpowiada ciąg znaków dwójkowych tzw. bitów. Ciąg bitów jest nazywany **kodem** danego znaku lub **słowem kodowym,** liczba bitów to **długość słowa kodowego.**
- Słowa kodowe różnych komunikatów wytwarzanych przez dane źródło mogą być różnej długości i występować z różnymi prawdopodobieństwami.
- Liczbę jednostek informacji słowa kodowego występującego z prawdopodobieństwem p_i dana jest wzorem:

$$\log_2(1/ p_i)$$

Entropia

Entropia – średnia ilość informacji przypadająca na pojedynczą wiadomość ze źródła informacji,

tzn. średnia ważona ilości informacji niesionej przez pojedynczą wiadomość z wagami równymi prawdopodobieństwu p_i, i=1,...,n, nadania poszczególnych wiadomości

$$H = \sum_{i=1}^{n} p_i \log_2\left(\frac{1}{p_i}\right) \quad [\text{bit}] \ .$$

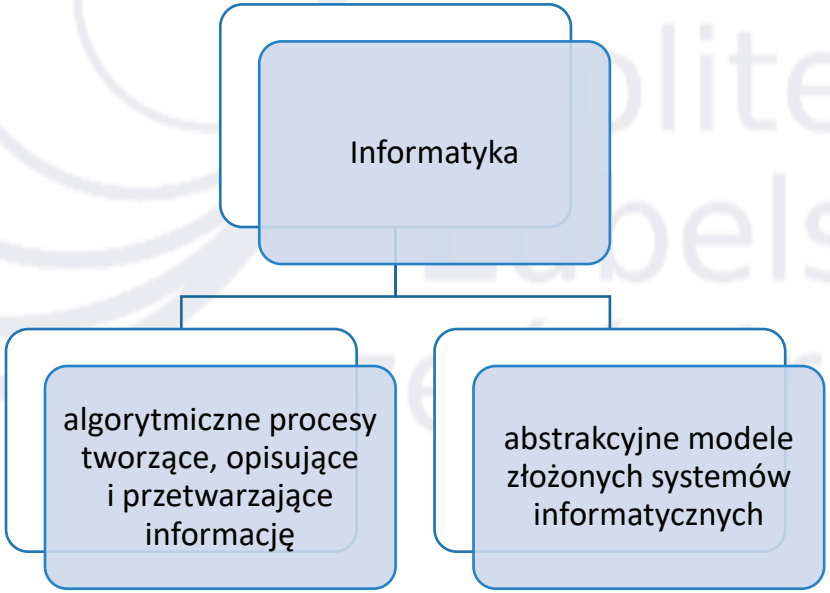
Entropia:

- jest nieujemna,
- jest maksymalna, gdy prawdopodobieństwa zajść zdarzeń są równe,
- jest równa 0, gdy stany systemu przyjmują wartości 0 albo 1,
- gdy dwa systemy są niezależne to entropia sumy systemów równa się sumie entropii, jest to tzw. własność superpozycji.

Informatyka

- Informatyka – kompleks dyscyplin teoretycznych (naukowo-badawczych) i praktycznych (technicznych), związanych bezpośrednio z zadaniami automatycznego przetwarzania informacji przy pomocy komputerów .
- **1962** – Francuzi zaproponowali termin *informatique*:

  informacja + automatyka.



Działy informatyki

- Administracja sieciowa
- Administracja systemów
- Algorytmika
- Architektura procesorów
- Bezpieczeństwo komputerowe
- Grafika komputerowa
- Inżynieria oprogramowania
- Języki programowania
- Programowanie
- Sprzęt komputerowy
- Symulacja komputerowa
- Sztuczna inteligencja
- Teoria informacji
- Webmastering

Reprezentacja informacji prostych

- **Znaki alfanumeryczne** (litery, cyfry, znaki interpunkcji, działań arytmetycznych itp.), za pomocą których człowiek komunikuje się z komputerem, zostają przetworzone automatycznie na zrozumiałe dla komputera znaki zapisane w systemie binarnym (dwójkowym).
- Najmniejszy element danych w komputerze nazywany jest **bitem**, która może przyjmować jedną z dwóch wartości, 0 lub 1. **Bajt** składa się z 8 bitów. Jest to najmniejsza jednostka pamięci komputera.
- Charakterystyczny dla danego komputera ciąg bitów będący wielokrotnością bajta to **słowo maszynowe**.
- Dla programistów praca z danymi na poziomie bitów jest uciążliwa. Dogodniej jest pracować z danymi w formie cyfr dziesiętnych i liter alfabetu łacińskiego oraz symboli specjalnych.
- Cyfry, litery i symbole specjalne nazywane są **znakami**. Każdy znak jest reprezentowany przez sekwencję zer i jedynek, do reprezentowania znaków w komputerze używany jest jednobajtowy typ znakowy (lub beznakowy) **char**. W sposób jednoznaczny znakom przyporządkowane są liczby od 0 do 255.

Kodowanie znaków

- **Kod ASCII** (ang. *American Standard Code for Information Interchange*), opracowany dla urządzeń dalekopisowych, później przyjęty dla komputerów. Jest to 7-bitowy kod przyporządkowujący liczby z zakresu 0-127 literom alfabetu angielskiego, cyfrom, znakom przestankowym i innym symbolom oraz poleceniom sterującym.

Dec	Hex	Char	Dec	Hex	Char
32	20	Spacja	65	41	A
33	21	!	66	42	B
34	22	”	67	43	C
48	30	0	68	44	D
49	31	1	97	61	a
50	32	2	98	62	b
51	33	3	99	63	c
52	34	4	100	64	d

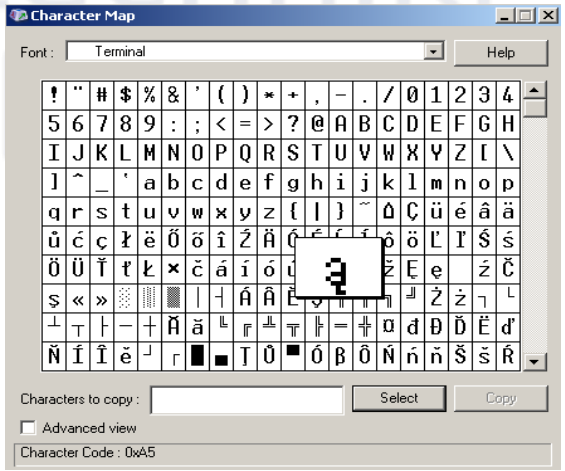
Kodowanie znaków

W 1981 r. IBM wprowadził kod rozszerzony do 8 bitów:
128-255 rozszerzone kody ASCII, w zależności od strony kodowej (kraju),
0 -127 podstawowy standard ASCII:

- litery, cyfry oraz inne znaki drukowane tworzą zbiór znaków ASCII, jest to 95 znaków o kodach od 32 do 126,
- pozostałe 33 kody (0-31 i 127) to tzw. kody sterujące służące do sterowania urządzeniem odbierającym komunikat.

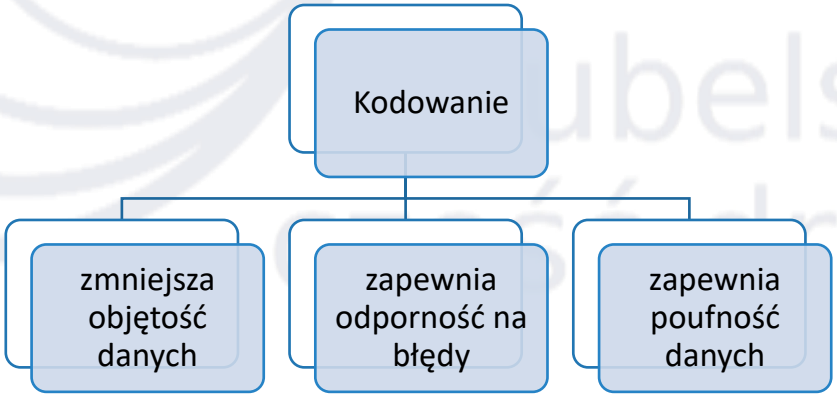
0-31, 127	znaki specjalne
32	spacja
48-57	cyfry
65-90	wielkie litery
97-122	małe litery

Pozostałe znaki: 33-47, 56-64, 91-96, 123-127.



Kod UNICODE

- 256 znaków alfanumerycznych jakie można zakodować za pomocą rozszerzonego kodu ASCII nie dawało możliwości zakodowania znaków diakrytycznych wielu języków np. japońskiego czy arabskiego.
- UNICODE ma obejmować wszystkie języki używane na świecie.
- Wprowadzono kod UNICODE o długości 16 bitów dla każdego znaku, co daje możliwość zakodowania 2^{16} czyli 65536 znaków.



Liczbowe systemy pozycyjne

Plan

- Systemy liczbowe
- System binarny
- Schemat Hornera
- Reprezentacja stałopozycyjna i zmiennopozycyjna
- Zapis obrazów cyfrowych

Liczbowe systemy pozycyjne

- Ogół zasad umożliwiających przedstawianie liczb za pomocą umownych znaków przyjęto nazywać systemem liczbowym, a znaki służące do zapisywania liczb – cyframi.
- Wyraz cyfra pochodzi od arabskiego wyrazu ***sifr***, oznaczającego zero. Zero przejęli Arabowie od Hindusów. Wprowadzenie tego symbolu ułatwiło zapisywanie liczb i wykonywanie działań.
- Wśród systemów liczbowych wyróżniamy:
  - systemy pozycyjne – znaczenie cyfry jest zależne od pozycji w liczbie, wartość cyfry na określonej pozycji obliczana jest poprzez pomnożenie tej cyfry przez odpowiednią potęgę podstawy systemu,
  - systemy inne niż pozycyjne – znaczenie cyfry jest niezależne od miejsca położenia w liczbie, np. liczbowy system rzymski pochodzenia etruskiego.

Liczbowe systemy pozycyjne

Twierdzenie 1.

Dla każdej liczby naturalnej $L \in N$ oraz liczby naturalnej $p > 1$ istnieje jednoznacznie wyznaczony ciąg

$$c_{n-1}c_{n-2}\ldots c_1 c_0\ ,\ c_i \in \{0,1,\ldots,p-1\},\ i = 0,1,\ldots,n-1,$$

taki, że

$$L = \sum_{i=0}^{n-1} c_i p^i$$

Cyfra o indeksie $n-1$ to cyfra najbardziej znacząca (ang. MSD – *most significant digit*). Ciąg $c_{n-1}c_{n-2}\ldots c_1 c_0$ nazywamy reprezentacją (rozwinięciem) liczby L w systemie u podstawy p.

Podstawa pozycyjnego systemu liczbowego odpowiada ilości cyfr wykorzystywanych do przedstawienia liczb w tym systemie.

Liczbowe systemy pozycyjne

Liczby z zakresu od 0 do 16 zapisane w systemie o postawie p			
$p = 10$	$p = 2$	$p = 8$	$p = 16$
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Liczbowe systemy pozycyjne

Schemat Hornera

$$L = \sum_{i=0}^{n-1} c_i \cdot 2^i = c_{n-1}2^{n-1} + c_{n-2}2^{n-2} + \ldots + c_2 2^2 + c_1 2 + c_0 =$$

$$= (c_{n-1}2^{n-1} + c_{n-2}2^{n-2} + \ldots + c_2 2 + c_1) \cdot 2 + c_0 = \ldots =$$

$$= (\ldots((c_{n-1}2 + c_{n-2}) \cdot 2 + c_{n-3}) \cdot 2 + \ldots + c_1) \cdot 2 + c_0$$

$$\begin{cases} w = c_{n-1} \\ w = w \cdot 2 + c_i \end{cases} \text{ dla } i = n-2, n-3, \ldots, 0 .$$

Przykład. $L = (01010110)_2 \qquad n = 8$

i	7	6	5	4	3	2	1	0
c_i	0	1	0	1	0	1	1	0
$p = 2$	**0**	1	2	5	10	21	43	**86**

Liczbowe systemy pozycyjne

W celu określenia maksymalnej wartości liczby całkowitej, jaką można zapisać należy za wszystkie cyfry c_i podstawić wartość maksymalną, równą $p-1$:

$$L_{\max} = \sum_{i=0}^{n-1}(p-1)p^i = (p-1)\sum_{i=0}^{n-1}p^i = (p-1)\frac{1-p^n}{1-p} = p^n - 1.$$

Ilość różnych liczb wynosi $L_{\max} + 1 = p^n$.

W systemach liczbowych powszechne uznanie zyskały systemy:

- dwójkowy (binarny), $p = 2$, $c_i \in \{0, 1\}$,
- ósemkowy, $p = 8$, $c_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$,
- dziesiętny (dziesiątkowy, decymalny), $p = 10$, $c_i \in \{0,1,2,3,4,5,6,7,8,9\}$,
- szesnastkowy (heksadecymalny), $p = 16$, $c_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$.

Liczbowe systemy pozycyjne

Zamiana liczby o podstawie 2 na liczbę o podstawie 16:

- grupujemy od prawej po 4 bity rozwinięcia binarnego,
- odczytujemy wartość dziesiętna każdej grupy bitów,
- podstawiamy odpowiednią cyfrę z systemu szesnastkowego.

Zamiana liczby o podstawie 16 na liczbę o podstawie 2:

- każdą cyfrę szesnastkową zapisujemy na 4 bitach
  (w razie potrzeby uzupełniamy od przodu odpowiednia liczbą zer).

Ćwiczenie 1.

Przedstawić liczby w zależności od cyfr i podstawy, podać wartości dziesiętne:

$(430)10 = 0 \cdot 100 + 3 \cdot 100 + 4 \cdot 102$

$(1101)2 = 1 \cdot 20 + 0 \cdot 21 + 1 \cdot 22 + 1 \cdot 23 = 13$

$(FF)16 = 15 \cdot 160 + 15 \cdot 161 = 255$

$(01010110)2 = 1 \cdot 21 + 1 \cdot 22 + 1 \cdot 24 + 1 \cdot 26 = 86$

Ćwiczenie 2.

Zapisać w systemie szesnastkowym (heksadecymalnym) liczbę $(101001)_2$

$(101001)_2 = (00101001)_2 = (29)_{16}$

System binarny

Ćwiczenie 3.
Zapisać w systemie dwójkowym liczbę $(41)_{10}$

n div 2	n mod 2	cyfra
41	1	c_0
20	0	c_1
10	0	c_2
5	1	c_3
2	0	c_4
1	1	c_5
0	**STOP**	

div – dzielenie całkowite

mod – reszta z dzielenia

$(41)_{10} = (101001)_2$

Należy wykonać dzielenie całkowite przez 2 aż do uzyskania ilorazu równego 0, cyfry liczby binarnej to kolejne reszty (zapisywane od końca, pierwsza wyliczona reszta to c_0, druga to c_1, itd.).

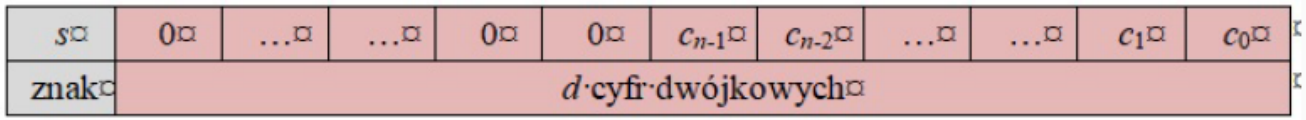
System binarny

Reprezentacja stałopozycyjna operuje na ustalonej liczbie cyfr. Liczbę całkowitą L przedstawiamy za pomocą rozwinięcia dwójkowego

$$L = s \sum_{i=0}^{n-1} c_i 2^i$$

gdzie s jest znakiem liczby (równym $+1$ lub -1), $c_{n-1} \neq 0$ dla liczby $L \neq 0$ oraz $c_i = 0$ lub $c_i = 1$, $i = n-2, n-3, \ldots, 0$.

Na reprezentację liczby przeznaczone jest słowo o skończonej długości, np. $d+1$ bitów. Jeżeli $n \leqslant d$, to liczba L jest reprezentowana w rozpatrywanej arytmetyce:



Liczby te należą do przedziału $[-2^d + 1, 2^d - 1]$.

System binarny

- System znak moduł
  - zero kodowane jest na dwa sposoby jako zero nieujemne i zero niedodatnie, trzeba pilnować, by nie potraktować ich jako różne,
  - przy dodawaniu trzeba ustalać znak wyniku.
- System znak moduł odwrotny
  - system podobny do poprzedniego z tą różnicą, że jeżeli pierwszy bit jest 1 to pozostałe reprezentują negatyw modułu liczby,
  - tu również występuje podwójne kodowanie zera.
- System uzupełnień do dwóch
  - każda wartość jest reprezentowana jednoznacznie.

Reprezentacja uzupełnieniowa

Reprezentacja uzupełnieniowa to najczęściej stosowana reprezentacja. Najstarszy bit n-bitowej reprezentacji uzupełnieniowej $c_{n-1}c_{n-2}\ldots c_1c_0$ traktowany jest jako -2^{n-1} i przyjmuje wartość 0, jeśli liczba jest dodatnia oraz 1, jeśli liczba jest ujemna. Pozostałe $n-1$ bity traktowane są tradycyjnie jako $2^{n-2},\ldots,2^0$. Zatem

- dla liczb dodatnich jest to reprezentacja binarna x, gdzie $0 \leqslant x \leqslant 2^{n-1}-1$,
- dla liczb ujemnych jest to reprezentacja binarna liczby $x_{uz} = 2^n - |x|$, gdzie $-2^{n-1} \leqslant x < 0$.

Wartość liczby o reprezentacji uzupełnieniowej $c_{n-1}c_{n-2}\ldots c_1c_0$ jest równa $\sum_{i=0}^{n-2} c_i 2^i - c_{n-1}2^{n-1}$.

Algorytmy zamiany U2

Reprezentacja uzupełnieniowa n-bitowa:

$$\text{reprezentacja(x)} = \begin{cases} \text{reprezentacja}(x) & 0 \leqslant x \leqslant 2^{n-1}-1 \\ \text{reprezentacja}(2^n - |x|) & -2^{n-1} \leqslant x < 0 \end{cases}$$

Schemat Hornera: liczba binarnej $c_{n-1}c_{n-2}\ldots c_0$ zapisana w n-bitowej reprezentacji uzupełnieniowej

$$\begin{cases} w = -c_{n-1} \\ w = w * 2 + c_i \text{ dla } i = n-2, n-3, \ldots, 0 \end{cases}$$

Algorytm zamiany na reprezentację uzupełnieniową
(dla liczb ujemnych)

- wyznaczyć n-bitową reprezentację binarną liczby |x| ,
- w uzyskanej reprezentacji zamienić 0 na 1 i 1 na 0,
- do wyniku dodać 1.

Aby z reprezentacji uzupełnieniowej wyznaczyć |x|

- od reprezentacji uzupełnieniowej odjąć 1,
- zamienić 0 na 1 i 1 na 0,
- wyznaczyć wartość dziesiętną.

Reprezentacja stałopozycyjna

- W systemie stałopozycyjnym liczby rzeczywiste mają część całkowitą i ułamkową. Liczba bitów tych części jest stała. Jest to nieekonomiczne, gdy operujemy na bardzo dużych lub bardzo małych liczbach.

Ułamek to liczba postaci $\mathbf{0.}c_{-1}c_{-2}\ldots c_{-k}$

- Wykorzystujemy ujemne potęgi dwójki (½, ¼, itp.)
  $(0.110)_2 = 1*2^{-1} + 1*2^{-2} + 0*2^{-3} = 0.5 + 0.25 + 0 = 0.75$
- Zamiana ułamka dziesiętnego na binarny:
  - Ułamek mnożymy przez 2 i wypisujemy kolejne cyfr przed kropką.
  - Liczbę złożoną z cyfr za kropką dziesiętna dalej mnożymy przez 2 aż do uzyskania wartości 0.
- Zwykle rozwinięcia te są nieskończone (okresowe).

$0.125*2 = \mathbf{0}.25$	0
$0.25*2 = \mathbf{0}.5$	0
$0.5*2 = \mathbf{1}.0$	1
0 STOP	

$0.125 = (0.001)_2$

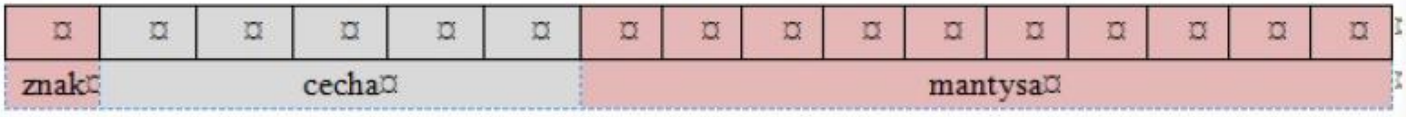
Reprezentacja zmiennopozycyjna

Każdą liczbę reczywistą x, różną od zera można zapisać jednoznacznie w postaci

$$x = s \cdot 2^c m$$

gdzie s równe $+1$ lub -1 jest znakiem liczby, c jest liczbą całkowitą zwaną cechą, natomiast $m \in [\frac{1}{2}, 1)$ jest liczbą rzeczywistą zwaną mantysą.

Cechę zapisujemy w sposób stałopozycyjny na $d - t$ bitach słowa maszynowego. Pozostałych t bitów przeznaczamy na zapis mantysy.



Reprezentacja zmiennopozycyjna

Przykład 1. Zapis liczby dziesiętnej 0.1 na 16 bitach w postaci $m_0\, c_1c_2c_3c_4c_5\, m_{-1}\ldots m_{-10}$

m_0	c_1	c_2	c_3	c_4	c_5	m_{-1}	m_{-2}	m_{-3}	m_{-4}	m_{-5}	m_{-6}	m_{-7}	m_{-8}	m_{-9}	m_{-10}
0	1	1	1	0	1	1	1	0	0	1	1	0	0	1	1
znak	cecha					mantysa									

- m_0 – znak mantysy (1 – liczba ujemna, 0 – liczba dodatnia),
- $m_{-1}\ldots m_{-10}$ – mantysa w postaci znormalizowanej, mantysę po normalizacji należy zaokrąglić (dodać 1 do 11-ego bitu mantysy i obciąć do 10 bitów),
- $c_1c_2c_3c_4c_5$ – cecha w postaci uzupełnieniowej (na 5 bitach).

Reprezentacja:

- Liczba rzeczywista pojedynczej precyzji – 4 bajty (8 bitów cechy, 23 bity mantysy).
- Liczba rzeczywista podwójnej precyzji – 8 bajtów (11 bitów cechy, 52 bity mantysy).

Zapis obrazów cyfrowych

- Obraz cyfrowy zapisywany jest w postaci tablicy pikseli.
- Do zapisu koloru każdego piksela wykorzystywany jest model *RGB* (*red, green, blue*). Najczęściej stosowany jest 24-bitowy (3 bajty) zapis kolorów, po 8 bitów na każdą z barw.
- W modelu RGB wartość 0 wszystkich składowych daje kolor czarny, natomiast 255 – kolor biały.
- Kolor *RGB* można obliczyć ze wzoru

  $$R \cdot 65536 + G \cdot 256 + B,$$

  gdzie składowe *R*, *G* i *B* przyjmują wartość od 0 do 255.
- Kolor piksela obrazu w odcieniach szarości jest zapisywany na 8 bitach i przyjmuje wartości 0-255.

Algorytmy i struktury danych

Plan

- Trochę historii
- Dane, rodzaje danych
- Podstawowe struktury danych
- Notacja ONP

Trochę historii

- XVII wiek – budowano pierwsze maszyny liczące.
- 1854 r. – George Boole, matematyk angielski, jako pierwszy tworzy podwaliny nowoczesnej logiki komputerowej, punktem wyjścia stał się system dwójkowy (Leibniz, 1679 r.). Boole znaczył ”1” jako prawda i ”0” jako fałsz, wprowadził pojęcie stanu logicznego ”i”, ”lub”, ”nie”.
- 1872 r. – w Stanach Zjednoczonych zbudowano maszynę sumującą wyposażoną w klawiaturę, wykonywała 4 podstawowe działania, następnie Wetzer (Niemcy) udoskonalił maszynę tak, by drukowała wyniki na pasku papieru.
- 1879 r. – w USA zbudowano pierwszą kasę rejestracyjną, by uniemożliwić kradzież pieniędzy przez pracowników. W późniejszym czasie połączono mechanizm liczący z blokadą szuflady z gotówką.
- 1880 -1886 r. – Hollerith pracując przy spisie ludności w USA zastosował kartę dziurkowaną jako nośnik danych, które mogły być czytane maszynowo.

Trochę historii

- 1909 r. – w Niemczech w czasie spisu powszechnego ludności wykorzystano udoskonaloną maszynę Holleritha.
- 1911 r. – urządzenie do przetwarzania danych przy pomocy kart dziurkowanych zastosowano w dużych przedsiębiorstwach.
- 1919 r. – fizycy Eccles i Jordan skonstruowali dwustanowy przełącznik.
- 1930 r. – w Cambridge (USA) pracuje pierwszy elektromagnetyczny komputer analogowy.
- 1932 r. – Austria, Tauschek buduje pierwszą pamięć bębnową.
- 1936 r. – Francja, Valtat zgłosił do opatentowania maszynę liczącą, której zasada działania oparta była na systemie dwójkowym.
- 1941 r. – prezentacja cyfrowej maszyny liczącej Zuse Z3, wartości liczbowe wprowadzane były z klawiatury w systemie dziesiętnym, maszyna realizowała wyznaczanie pierwiastka kwadratowego.
- 1942 r. – pierwsza sprawna maszyna licząca w technice lampowej, Atanasoff (USA).

Trochę historii

- 1943 r. – próba zastąpienia przekaźników elektromagnetycznych lampami elektronowymi.
- 1945 r. – Uniwersytet Pensylwania, uruchomiono pierwszą wielką maszynę liczącą ENIAC (Electronic Numerical Integrator and Computer), maszyna zajmowała 140 m^2 i ważyła 30 ton, twórcami byli J. Presper Eckert, J.W. Mauchly.
- 1948 r. – „krok binarny” określono jako bit (matematyk J. Tukey), znaczenia nabiera algebra Boole’a.
- 1949 r. – Uniwersytet Menchester (Anglia), skonstruowano komputer EDSAC (Electronic Delay Storage Automatic Computer) z programowaniem pamięciowym. Program i dane były zakodowane w pamięci, program zawierał rozkazy warunkowe.
- 1950 r. – wykorzystano pamięci zbudowane w oparciu o rdzenie ferrytowe.

Trochę historii

- 1950 r. – pamięć magnetyczna (peryferyjna) dla elektronicznych maszyn cyfrowych, zastępuje karty perforacyjne lub taśmę dziurkowaną.
- 1954 r. – J.W. Backhus rozwija język programowania Fortran, w niedługim czasie powstaje biblioteka programów.
- 1955 r. – Bell Laboratory (USA), tranzystorowa maszyna licząca TRADIC, komputer drugiej generacji.
- 1955 r. – angielska firma EMI, pierwszy skaner.
- 1958 r. – amerykańska firma Texas Instruments, pierwszy układ scalony.
- 1963 r. – IBM, szybka drukarka wierszowa.
- 1967 r. – N. Kitz, pierwsza biurowa maszyna matematyczna.
- 1975 r. – IBM, pierwsza drukarka laserowa „IBM 3800”.
- 1980 r. – Sharp, Casio, Sanyo, Panasonic, pierwsze komputery kieszonkowe, wyposażone w uproszczony BASIC.

Trochę historii

- 1980 r. – rozwój układów scalonych o dużej skali integracji (ROM, PROM, EPROM, RAM).
- 1983 r. – upowszechnienie dyskietek.
- 1987 r. – prace nad skonstruowaniem optoelekrycznego komputera.
- 1988 r. – tytuł arcymistrzowski dla Deep Thought, symulujący grę w szachy, przewidywał 10-11 ruchów, do 30 w sytuacji kryzysowej.
- 1989 r. – rozpowszechniają się wirusy komputerowe, w USA w ciągu ośmiu miesięcy zaatakowały 48 000 komputerów.
- 1996 r. – Deep Blue, zwycięstwo Kasparowa, 3 zwycięstwa, 2 remisy, jedna porażka.
- 1997 r. – wygrywa komputer, przeprowadzona partia zakończyła się przed rozegraniem 20 posunięcia . Komputer zawierał 256 procesorów.

Trochę historii

INTERNET – symbol ery informacyjnej

INTERNET = INTERconnection + NETwork.

- 1957 r. – wystrzelenie przez Związek Radziecki pierwszego sztucznego satelity Ziemi, Sputnika.
- Powołanie w ramach amerykańskiego Departamentu Obrony agencji ARPA (Advanced Research Project Agency), w polu zainteresowania ARPA znalazły się sieci komputerowe.
- 1969 r. – na uniwersytetach w USA w ramach eksperymentu finansowanego przez ARPA zainstalowano pierwsze węzły sieci ARPANET, bezpośredniego przodka Internetu.
- 1971 r. – pierwszy program poczty elektronicznej działający w sieci ARPANET; ostateczne specyfikacje: protokół telnet (1972 r.), protokół FTP (1973 r.), poczta elektroniczna (1977 r.).
- 1983 r. – komputery sieci ARPANET przeszły na stosowanie protokołu TCP/IP; pojawiła się wersja systemu Unix.
- 1984 r. – Internet został przekazany pod zarząd Narodowego Funduszu Nauki.
- 1991 r. – rozwój Internetu w Polsce.

Dana, rodzaje danych

- **Dana** w algorytmice to zapis informacji mający swą **tożsamość** (nazwę) i **treść** (wartość).
- Nazwa danej jest wyrażona za pomocą napisu złożonego ze znaków pewnego alfabetu.
- Wyróżniamy:
  - **dane stałe** (stałe) – nie mogą zmieniać swej wartości w czasie procesu przetwarzania,
  - **dane zmienne** (zmienne) – na ogół podlegają zmianom w miarę wykonywania poleceń zawartych w programie, jej wartość można zmienić przez przypisanie.
- W praktyce języki programowania dopuszczają do użytku takie wartości proste, które są liczbami, wartościami logicznymi lub wartościami znakowymi.

Liczby

- We współczesnych językach programowania stała, zmienna, wyrażenie, czy funkcja są określonego typu. Typ danych wyznacza:
  - zbiór wartości,
  - zbiór operacji, jakie można wykonać na elementach tego zbioru.
- Wśród skalarnych typów danych liczbowych wyróżniamy:
  - **typ integer** – obejmuje podzbiór zbioru liczb całkowitych, zwykle mniejszych co do wartości bezwzględnej od pewnej wartości maksymalnej; pomijając sytuację wystąpienie nadmiaru należy zauważyć, że wszystkie operacje wykonane na argumentach tego typu dają wartości dokładne (np. dodawanie, odejmowanie, mnożenie, dzielenie całkowite),
  - **typ real** – obejmuje skończony podzbiór zbioru liczb rzeczywistych, skutki wykonywanych obliczeń zależą od problemu i stosowanego algorytmu, wyniki przybliżają wyniki prawdziwe.

Dana, rodzaje danych

- Wartością **danej złożonej** jest zbiór danych prostszych.
- Skończony zbiór danych odróżnialnych od siebie, mających różne nazwy, nosi nazwę **struktury danych**. Wartość danej złożonej jest strukturą danych. Daną złożoną nazywa się również daną strukturalną.
- Dana, która sama nie wchodzi w skład wartości innej danej nazywana jest **daną całościową**. W przeciwnym razie nosi nazwę **danej składowej**.
- W większości języków programowania jako oznaczenia danej całościowej przyjęto używać zastrzeżonego dla niej **identyfikatora**, tzn. wybranego przez programistę napisu, rozpoczynającego się literą i złożonego z liter i/lub cyfr, nierozdzielonych żadnym innym znakiem, np. odstępem.

Instrukcja przypisania

Instrukcja przypisania ma kształt:

```
nazwa_zmiennej := wyrażenie
```

i nakazuje wykonanie następujących działań:

- obliczenie wartości wyrażenia zapisanego po prawej stronie znaku :=
- przypisanie tej wartości do nazwy zmiennej figurującej po lewej stronie tego znaku.

Fizycznie to przypisanie oznacza umieszczenie zapisu wartości w miejscu nośnika pamięciowego skojarzonego z tą nazwą i przechowanie jej tam do kolejnego przypisania.

Logicznie jest to skojarzenie nazwy danej zmiennej z tak obliczoną wartością, czyli utworzenie nowej (o innej wartości niż poprzednio) danej.

W części języków programowania operator przypisania oznaczany jest znakiem równości **=**.

Instrukcja przypisania

Przykład 1.

```
delta := b*b-4*a*c
```

Zmienna po lewej stronie znaku przypisania jest bierna, tzn. nie uczestniczy w obliczaniu wyrażenia po prawej stronie, jedynie przyjmuje nową wartość.

Przykład 2.

```
n := n+1
```

Zmienna o identyfikatorze **n** najpierw jest użyta jako źródło dostarczające wartości składnika sumy po prawej stronie znaku przypisania, następnie zaś jako bierne miejsce przechowania zwiększonej o 1 od dotychczasowej, wartości wynikowej wyrażenia **n+1**.

Wzajemna wymiana wartości

Wzajemna wymiana wartości między dwiema zmiennymi zawsze wymaga dodatkowej zmiennej pomocniczej (roboczej).

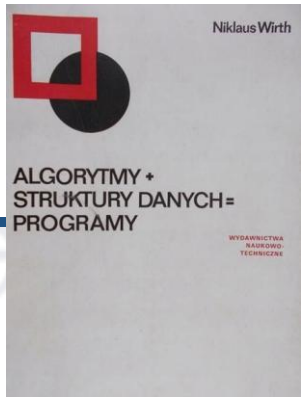
```
robocza := zmiennaPierwsza
zmiennaPierwsza := zmiennaDruga
zmiennaDruga := robocza
```

Zmienne robocze służą do tymczasowego przechowywania wartości częściowych wyników przetwarzania, które mają być wykorzystane albo w późniejszych fazach wykonania algorytmu, albo w bezpośrednio następnych wielokrotnie.

Struktury danych

- Algorytm operuje na danych różnego typu:
  - mają one przeważnie określoną formę zapewniającą pożądane właściwości,
  - do ich przechowywania i wykorzystania potrzebne stworzenie algorytmów zapewniających dostęp do wszystkich elementów i modyfikację zawartości.
- Struktury danych są pojemnikami na dane, które gromadzą je i układają w odpowiedni sposób.
- Ich różnorodność jest ogromna, a dla każdej znaleziono wiele zastosowań oraz algorytmów.
- Powszechnie spotykane jest używanie jednych struktur danych do przetwarzania informacji zgromadzonych w innych.
- Są one fundamentalnym narzędziem programisty, ich znajomość jest niezbędna.

Algorytmy i struktury danych



Niklaus Wirth – szwajcarski elektronik i informatyk.

- Wniósł duży wkład w zakresie języków programowania, analizy składniowej, konstrukcji kompilatorów i translatorów.
- Twórca języków programowania Pascal (1971), Modula (1973-1976), Modula-2 (1978–1980).
- Zaprojektował mikrokomputer Lilith.
- Od 1999 jest emerytowanym pracownikiem Instytutu Systemów Komputerowych Politechniki Federalnej w Zurychu.
- Autor książki *Algorytmy + Struktury Danych = Programy*
- Laureat Nagrody Turinga w 1984 r.

Podstawowe struktury danych

- Stos
- Kolejka
- Lista
- Drzewa

Stos LIFO – Last In First Out

Stos to abstrakcyjna struktura danych, do której dostęp możliwy jest tylko od strony wierzchołka, udostępniająca m.in. następujące operacje:

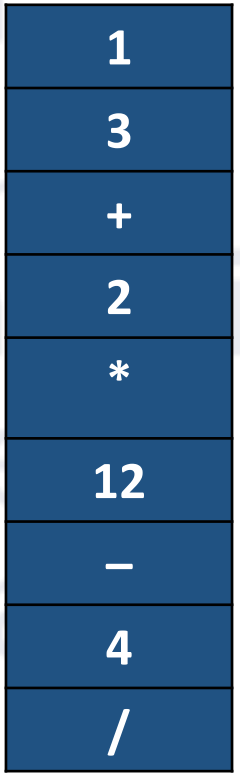
- empty – zwraca true jeżeli stos jest pusty,
- push – odkłada element na wierzchołek stosu,
- pop – zdejmuje element z wierzchołka stosu, rozważa się wersje stosu, w której operacja pop zwraca wartość usuwanego elementu.

Notacja ONP

- Odwrotna notacja polska jest sposobem zapisu wyrażeń arytmetycznych, w których znak wykonywanej operacji umieszczony jest po operandach (zapis postfiksowy) a nie pomiędzy nimi jak w zapisie algebraicznym (zapis infiksowy).
- Nie wymaga używania nawiasów.
- Obliczenia w ONP stają się bardzo łatwe do przeprowadzania na komputerze.
- Operacje obliczania wartości wyrażenia wykorzystują stos.

ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /

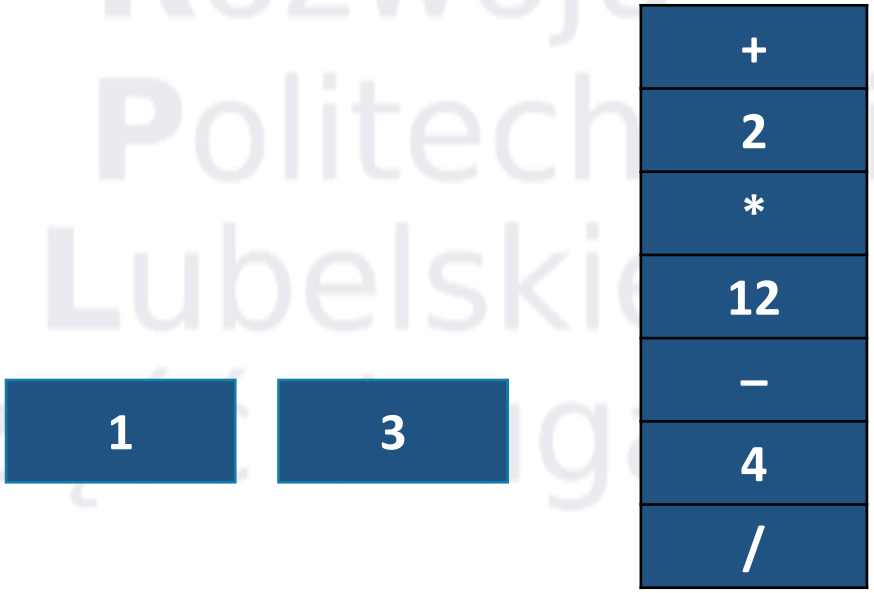


ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /

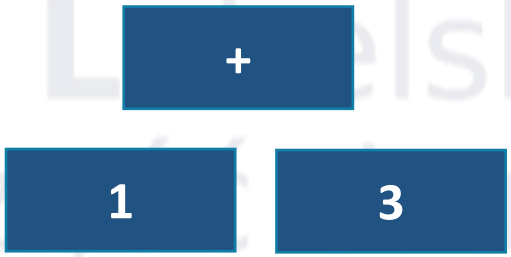
ONP

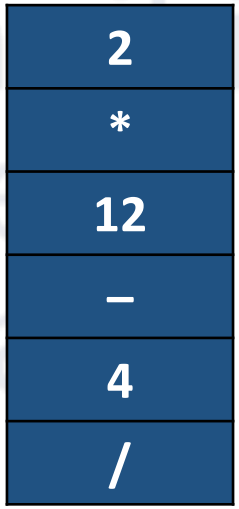
((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /



ONP

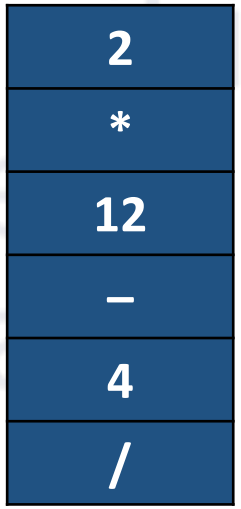
((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /





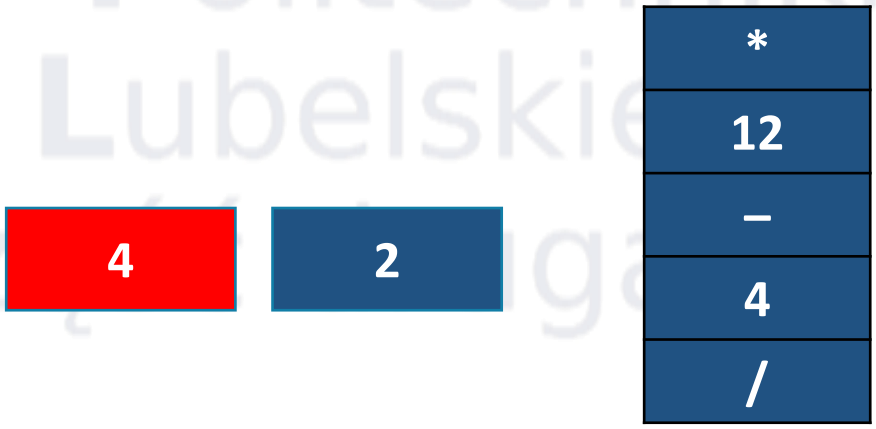
ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /



ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /



ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /

ONP

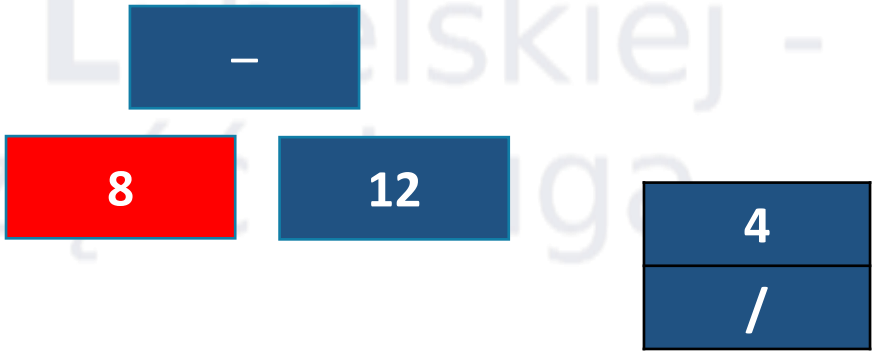
((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /

ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /

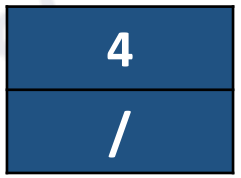
ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /



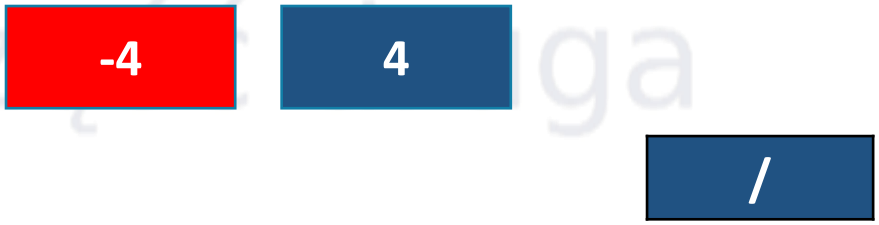
ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /



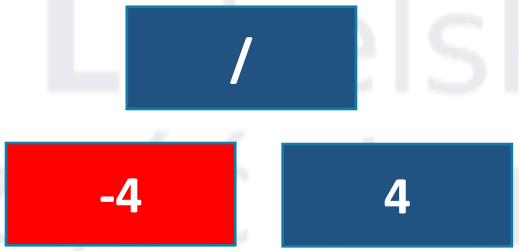
ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /



ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /



ONP

((1 + 3) * 2 – 12)/4 -> 1 3 + 2 * 12 – 4 /

-1

Kolejka FIFO – First In First Out

Kolejka to abstrakcyjny typ danych, w której nowe dane dopisywane są na końcu kolejki, a dane do dalszego przetwarzania pobierane są z początku kolejki. Dostęp do danych jest w takiej kolejności, w jakiej zostały zapisane. Kolejka udostępnia następujące operacje:

- empty – zwraca true, jeżeli kolejka jest pusta,
- push – wstawia element na koniec kolejki element,
- pop – usuwa pierwszy element; rozważa się wersje kolejki, w której operacja pop zwraca wartość usuwanego elementu,
- front – zwraca wartość pierwszego elementu.

Kolejki

- Kolejka priorytetowa – każda ze znajdujących się w kolejce danych ma przypisany priorytet, który modyfikuje kolejność późniejszego wykonania, oznacza to, że pierwsza na wyjściu pojawią się dane o największym priorytecie.
- Kolejka priorytetowa ma zastosowanie w systemach operacyjnych, przy przydzielaniu zasobów sprzętowych uruchomionym procesom.
- Kolejka cykliczna – pierwszy element o indeksie 0 uważany jest za następny w stosunku do ostatniego elementu.

Lista

Lista jest to liniowo uporządkowany zbiór elementów, z którego w dowolnym miejscu można usunąć element i w dowolne miejsce można dołączyć element.

- Rodzaje list:
  - jednokierunkowa,
  - dwukierunkowa,
  - cykliczna.

Drzewa

Drzewo jest to struktura abstrakcyjna, wzbogacona o kierunek. Jest to zbiór T jednego lub więcej elementów zwanych węzłami:

- Istnieje pierwszy, wyróżniony węzeł zwany korzeniem drzewa. Kolejne obiekty są jego potomkami i nazywane są węzłami. Pozostałe węzły są podzielone na *m* rozłącznych zbiorów T_1, …, T_m, z których każdy jest drzewem. Drzewa T_1, …, T_m nazywane są poddrzewami korzenia.
- Liście są to węzły nie mające potomków. Droga w drzewie to sekwencja węzłów odpowiadających przejściu w kierunku od korzenia do liścia.
- Długość ścieżki od korzenia do węzła x nazywamy głębokością węzła *x* w drzewie T.
- Największa głębokość węzła w drzewie T jest wysokością drzewa T.
- Każdy węzeł *y* leżący na ścieżce z korzenia do węzła *x* nazywamy przodkiem węzła *x*, węzeł *x* jest wtedy potomkiem węzła *y*. Bezpośredni potomkowie są synami, węzeł natomiast jest ojcem.

Drzewo binarne

Drzewo binarne – struktura abstrakcyjna, drzewo uporządkowane z korzeniem, w którym każdy węzeł ma co najwyżej dwóch potomków, lewego i prawego.

- Drzewa binarne mogą być wykorzystane do reprezentacji wyrażeń arytmetycznych.
- Drzewo wyrażeń jest to drzewo binarne, które przedstawia wyrażenie arytmetyczne. Liście drzewa reprezentują argumenty, a wewnętrzne węzły operacje arytmetyczne. Do tworzenia drzewa używamy notacji ONP, stosu i drzewa binarnego.

Drzewo wyrażeń algebraicznych

Czytamy argumenty znak po znaku odkładając je na stos.
W momencie pojawienia się jakiegoś operatora ze stosu zdejmowana jest odpowiednia liczba argumentów –
wynik operacji kładziemy na stos jako kolejny argument.

Przykład. Wyrażenie 1 3 + 2 * 12 – 4 /

1 3 + 2 * 12 – 4 /

tworzymy węzeł z argumentem 1 i na stosie umieszczamy wskaźnik do tego elementu

1 **3** + 2 * 12 – 4 /

tworzymy węzeł z argumentem 3 i na stosie umieszczamy wskaźnik do tego elementu

1 3 **+** 2 * 12 – 4 /

tworzymy węzeł dla operatora + . Ze stosu pobieramy adresy węzłów i tworzymy z nich lewego i prawego syna dla węzła z operatorem +, po czym na stosie umieszczamy wskaźnik do tego węzła.

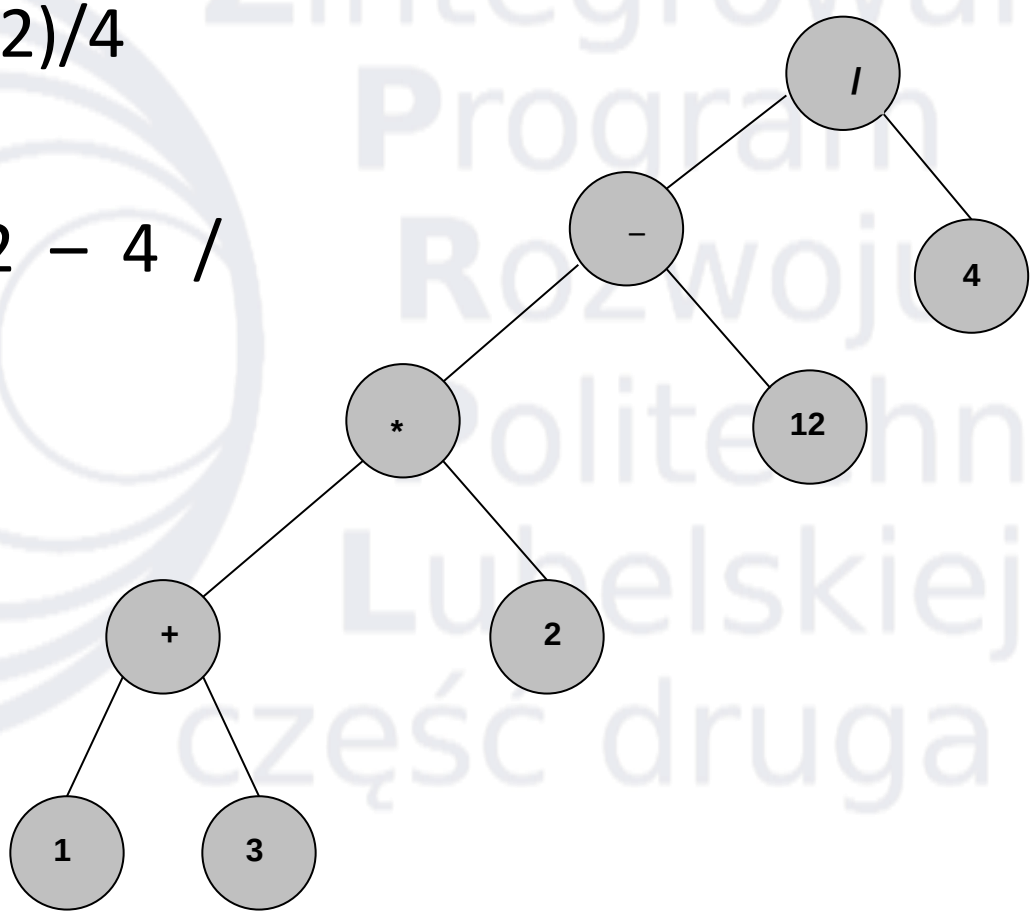
Postępowanie kontynuujemy aż do wyczerpania elementów.

Drzewo binarne, a wyrażenie algebraiczne

((1 + 3) * 2 – 12)/4

ONP:

1 3 + 2 * 12 – 4 /



Algorytmy i sposoby ich zapisu

Plan

- Algorytm, własności algorytmu
- Notacje zapisu algorytmów
- Algorytm Euklidesa
- Schematy N-S – zasady konstruowania
- Przykłady algorytmów
- Zapis algorytmu w języku programowania
- Język Python - wprowadzenie

Algorytm

- Słowo **algorytm** pochodzi od nazwiska perskiego matematyka o nazwisku Chuwarizmi Al- Muhammad Ibn Musa , które w formie zlatynizowanej brzmi Algorithmus[1].
- Chuwarizmi Al- Muhammad Ibn Musa urodził się ok. 780 roku w Chorezmie, zmarł ok. 850 r.; matematyk, astronom, geograf i kartograf. Dzięki jego pracom:
  - zaczęto stosować w Europie pochodzący z Indii dziesiętny system liczenia i pozycyjny system zapisu liczb,
  - cyfry arabskie wyparły cyfry rzymskie w Europie,
  - wprowadzono pojęcia zera, ułamków, funkcje trygonometryczne i elementy algebry.

[1] https://pl.wikipedia.org/wiki/Muhammad_ibn_Musa_al-Chuwarizm

Algorytm - definicje

- **Algorytm** jest to sposób rozwiązania danego problemu, podany w formie przepisu określającego skończoną liczbę operacji oraz kolejność w jakiej operacje te powinny być wykonywane.
- **Program** to algorytm zapisany w języku zrozumiałym dla komputera. Komputer rozwiązuje problemy dające się zapisać w postaci algorytmu.
- Własności algorytmu:
  - **Określoność** – znane są wszystkie przypadki, jakie mogą zaistnieć w czasie realizacji algorytmu.
  - **Skończoność** – wszystkie obliczenia zawarte w algorytmie powinny zostać zrealizowane w skończonej liczbie kroków.
  - **Wykonalność** – właściwe zdefiniowanie poszczególnych kroków algorytmu, tak aby możliwe było jego efektywne wykonanie.
- Zasady budowy algorytmu:
  - prostota budowy,
  - minimalizacja liczby wykonywanych działań (optymalizacja),
  - dokładność przeprowadzonych obliczeń.

Wybór algorytmu

- Należy implementować algorytm najprostsze, które wykonują określone zadanie.
  - łatwiejsza implementacja, czytelniejszy kod ,
  - łatwość testowania,
  - łatwość pisania dokumentacji.
- Algorytm powinien być efektywny
  - zajętość pamięci,
  - czas potrzebny na jego wykonanie,
  - obciążenie sieci komputerowej,
  - liczba danych odczytywanych i zapisywanych na dysku.

Notacje zapisu algorytmów

- Język naturalny, zapis słowny
  - pozwala określić kierunek działań i odpowiedzieć na pytanie, czy zagadnienie jest możliwe do rozwiązania.
- Lista kroków
  - zapis bardziej konkretny, wymaga określenia danych i wyniku oraz zapisania kolejnych kroków.
- Zapis graficzny:
  - Schematy blokowe (*sieć działań*), schematy N-S – *Nassi Schneidermana*
- Pseudo-kod
- Języki programowania

Algorytmy i ich rodzaje

Algorytm liniowy

- ma postać ciągu kroków, które muszą zostać bezwarunkowo wykonane jeden po drugim,
- nie zawiera żadnych warunków ani rozgałęzień.

Przykład

- Sformułowanie zadania: oblicz sumę S dwóch liczb naturalnych a, b.
- Dane wejściowe: dwie liczby a i b.
- Cel obliczeń: obliczenie sumy S = a + b.

Dodatkowe ograniczenia:

- sprawdzenie warunku dla danych wejściowych np. czy a, b są naturalne. Sprawdzenie warunków sprawia, że algorytm przestaje być liniowy.

Algorytm Euklidesa

- Pomysłodawcą algorytmu był Eudoksos z Knidos, w IV wieku p.n.e. Euklides zawarł ten algorytm i udowodnił jego poprawność w dziele *Elementy.*
- Największy Wspólny Dzielnik (NWD) dwóch liczb jest największą liczbą naturalną spośród tych, które dzielą obie te liczby bez reszty.

Algorytm

WE: dwie dodatnie liczby całkowite **`a`** i **`b`**

WY: największy wspólny dzielnik NWD(**`a`**,**`b`**).

Krok 1. Podziel **`a`** przez **`b`** i niech **`r`** oznacza resztę z tego dzielenia (**`r < b`**):

```
r = a % b
```

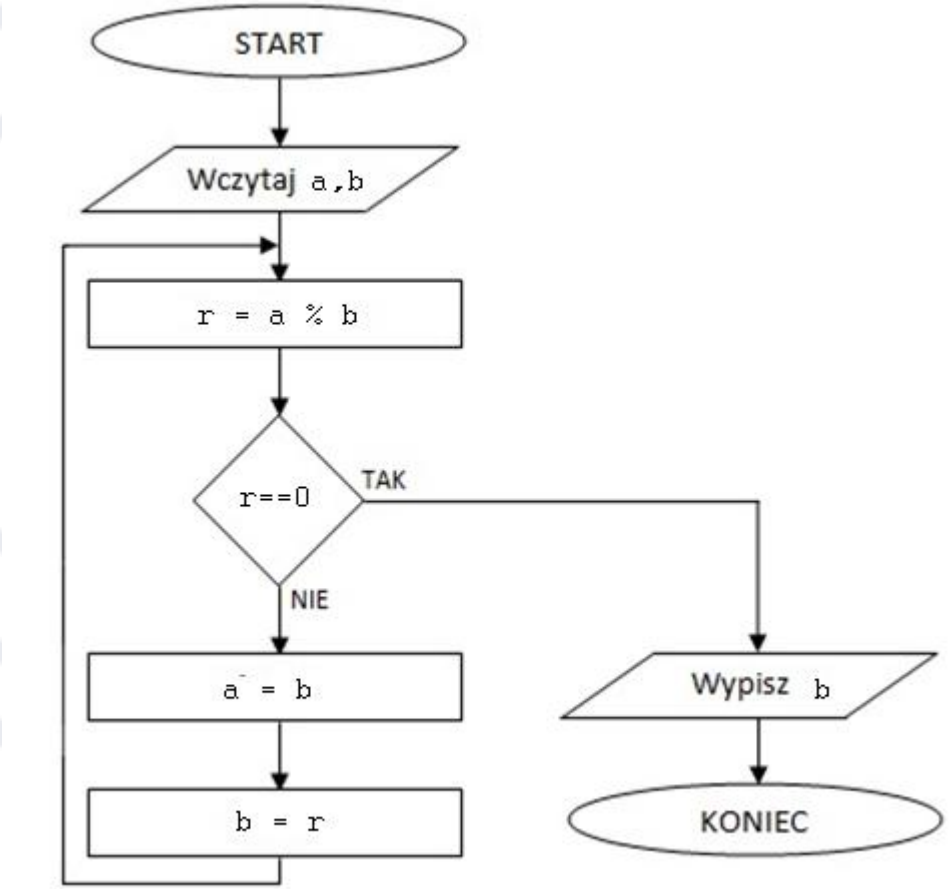
Krok 2. Jeżeli **`r == 0`** zakończ algorytm; wynikiem jest **`b`**.

Krok 3. W przeciwnym razie wykonaj

```
a = b
b = r
```

i wróć do **Kroku 1**.

Algorytm Euklidesa



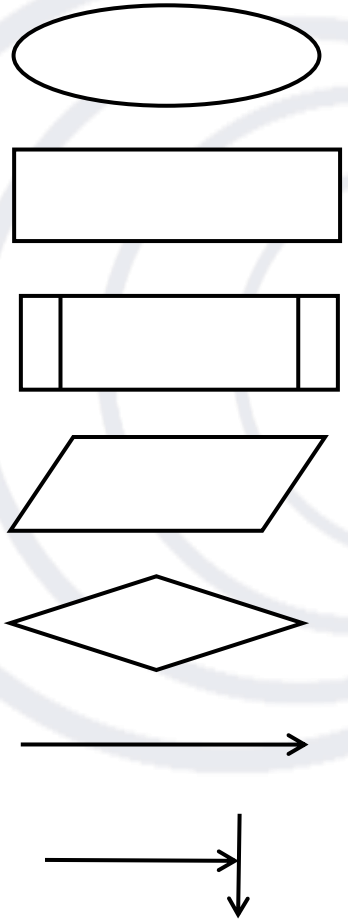
Instrukcje

- Sekwencja
- Instrukcje warunkowe (wariantowe)
- Instrukcje iteracyjne (pętle)

1966 r. – Corrado Bohm i Giuseppe Jacopini udowodnili, że te trzy rodzaje instrukcji wystarczają dla wyrażenia każdego sensownego algorytmu.

Twierdzenie to stało się fundamentem **strukturalnych** metod i technik programowania.

Podstawowe symbole w schematach blokowych



Początek, Koniec (Start, Stop)

Przetwarzanie

Proces uprzednio zdefiniowany

WE / WY

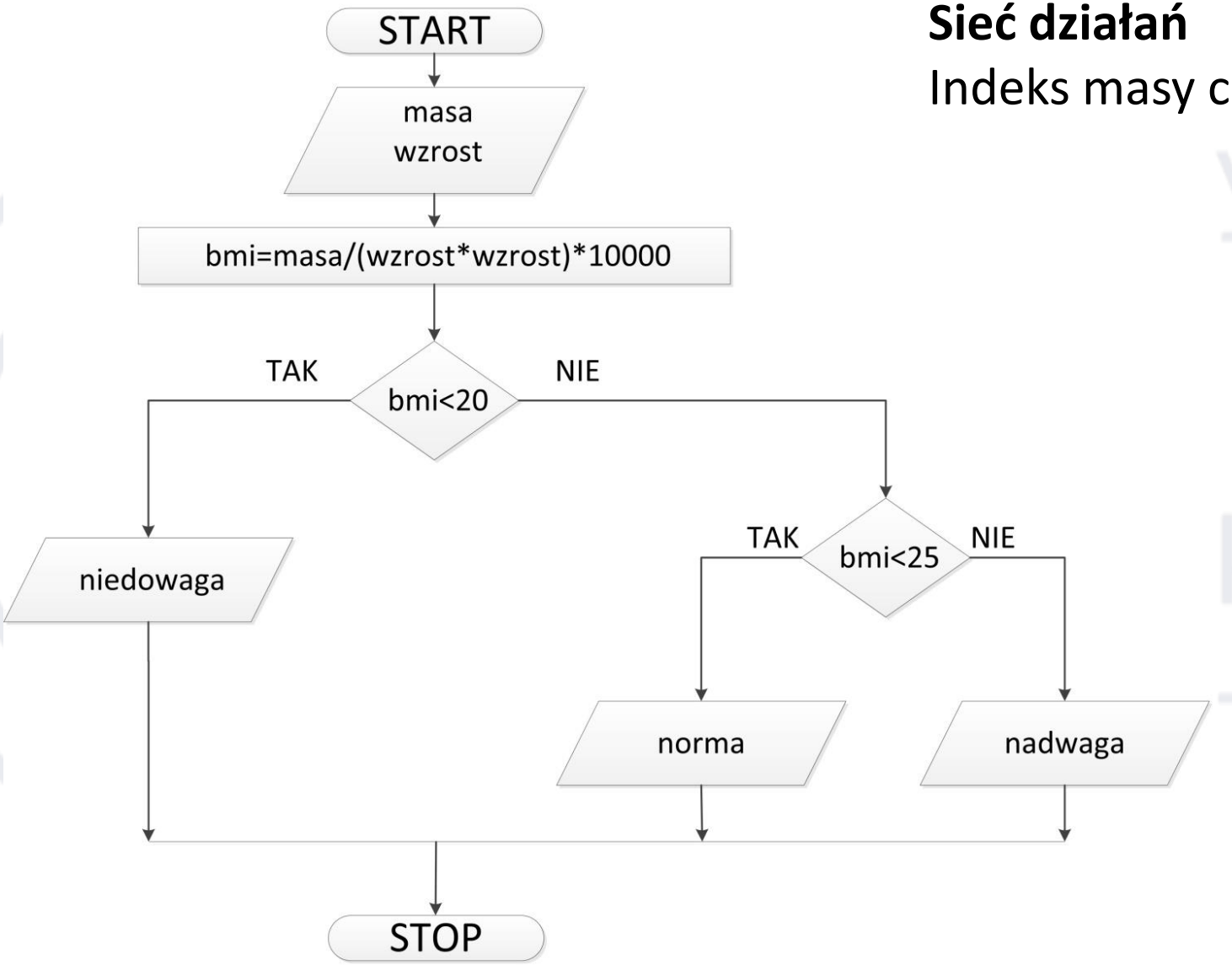
Decyzja

Kierunek przepływu danych

Łączenie dróg przepływu danych

Sieć działań

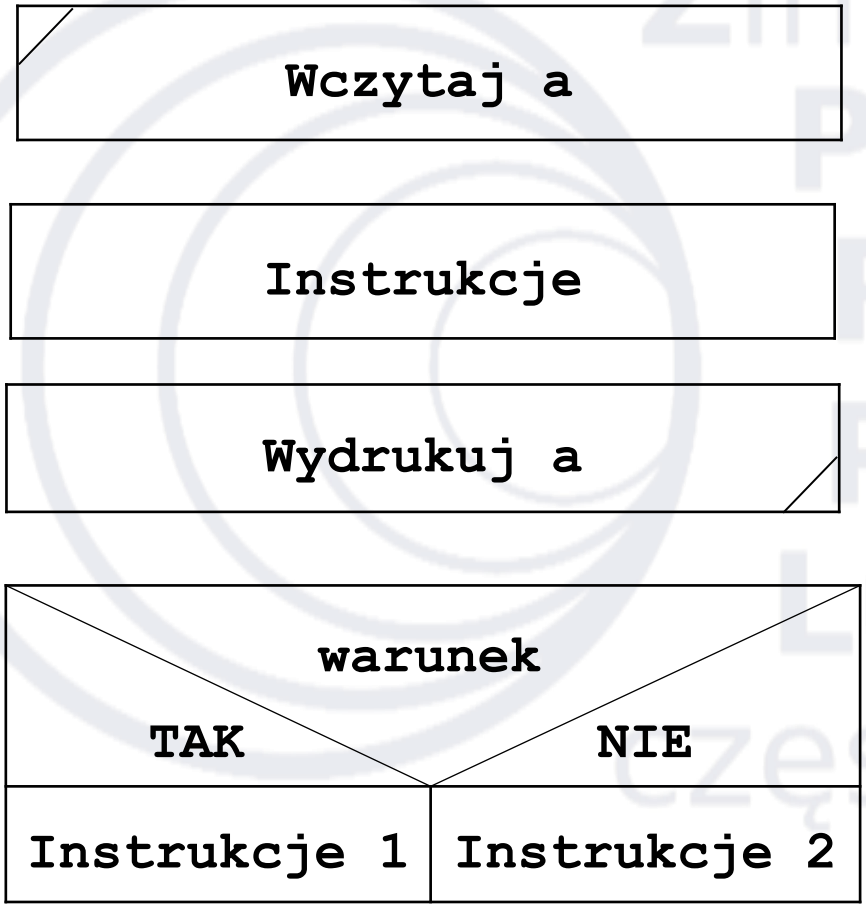
Indeks masy ciała



Schematy N-S

- Schematy N-S składają się ze zwartych bloków uniemożliwiają zapis skoków wewnątrz algorytmu, zmuszając programistę do myślenia strukturalnego.
- Sposób rozwiązania danego problemu zapisany w postaci schematu zwartego jest łatwiejszy do zrozumienia. Zapis algorytmu w postaci schematu N-S ułatwia sprawdzenie jego poprawności, programy pisane według tego schematu zawierają mniej błędów.
- Konstruowanie schematu zaczynamy od prostokąta symbolizującego cały algorytm, o wystarczająco dużych rozmiarach – cały arkusz z pozostawieniem marginesu na komentarze.
- Kolejne podziały klatki dokonujemy kolejno według schematu sekwencji, proporcjonalnie do przewidywanych rozmiarów zawartości.
- Również proporcjonalnie planujemy podziały liniami pionowymi.
- Nie umieszczamy zbyt wiele podklatek na jednym rysunku, w razie potrzeby wyodrębniamy podschematy dla większych fragmentów algorytmu.

Podstawowe symbole w schematach N-S



Instrukcja WE

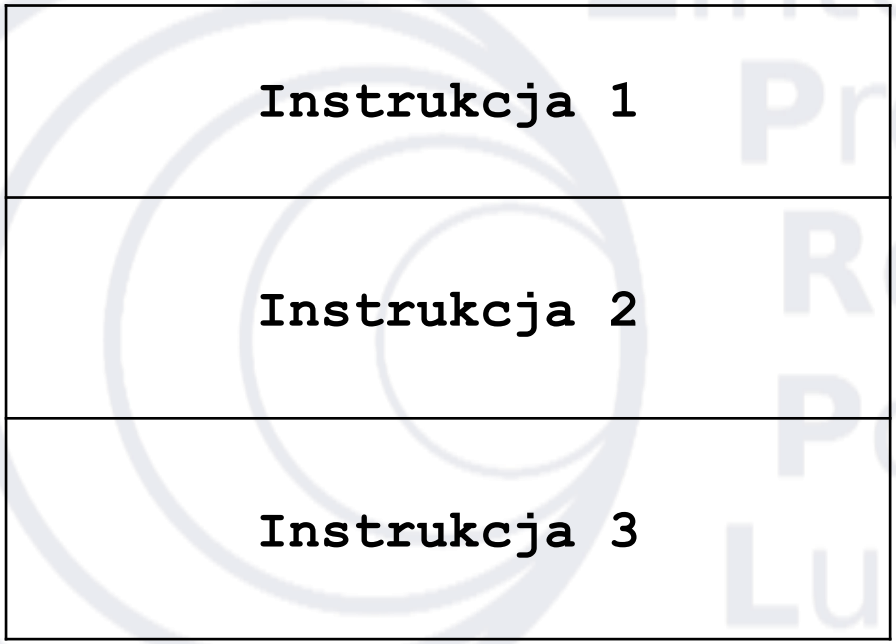
Przetwarzanie

Instrukcja WY

Instrukcja warunkowa (dwuwariantowa)

```
jeżeli (warunek) to
  Instrukcje1;
w przeciwnym razie
  Instrukcje2;
```

Sekwencja

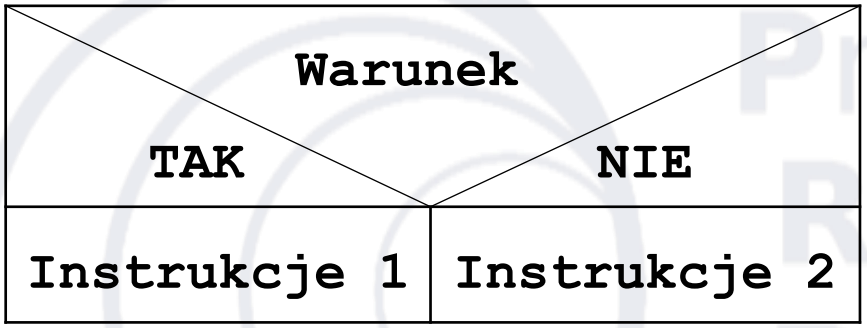


Instrukcja 1
a następnie

Instrukcja 2
a następnie

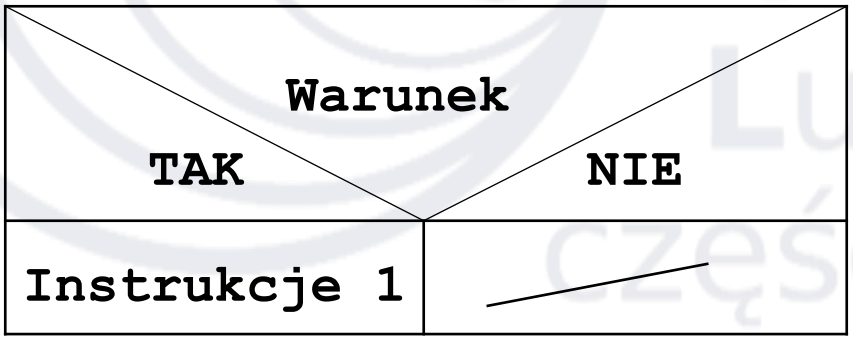
Instrukcja 3

Instrukcje warunkowe



dwuwariantowa

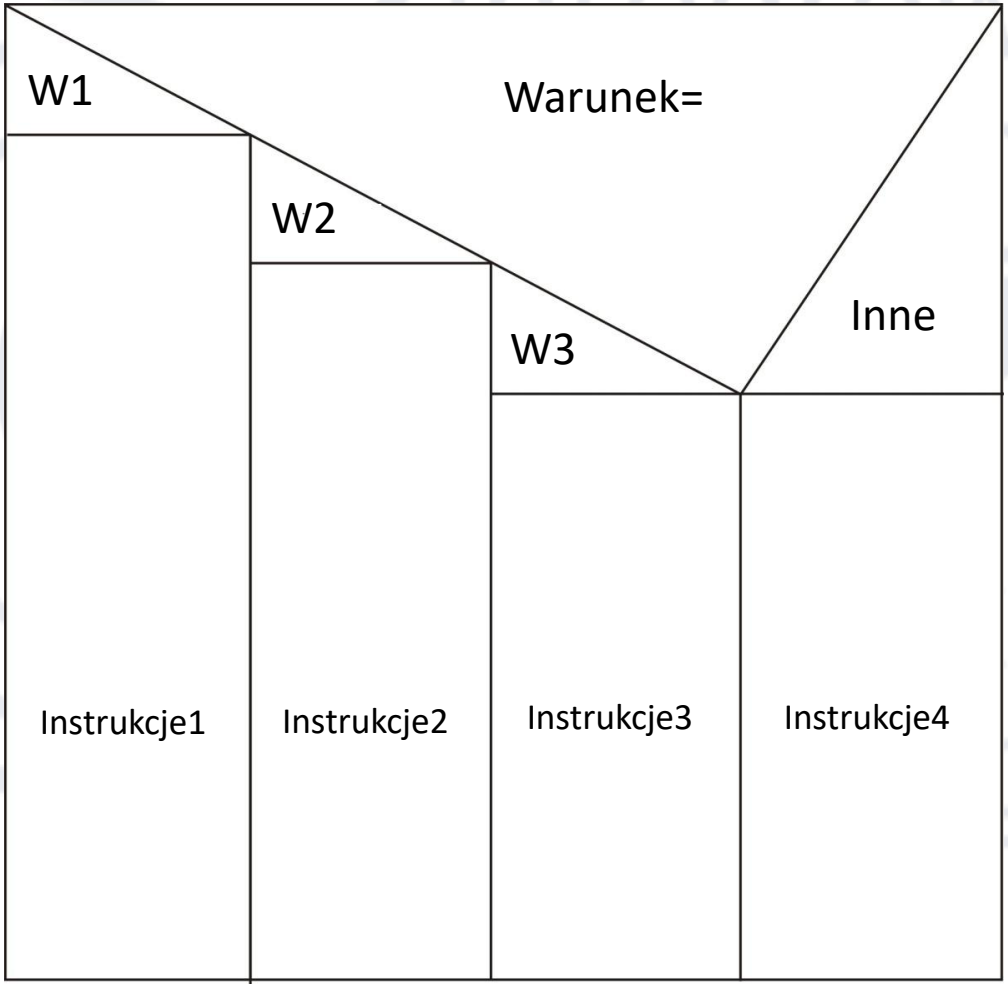
```
Jeżeli (warunek) to
  Instrukcje1;
w przeciwnym razie
  Instrukcje2;
```



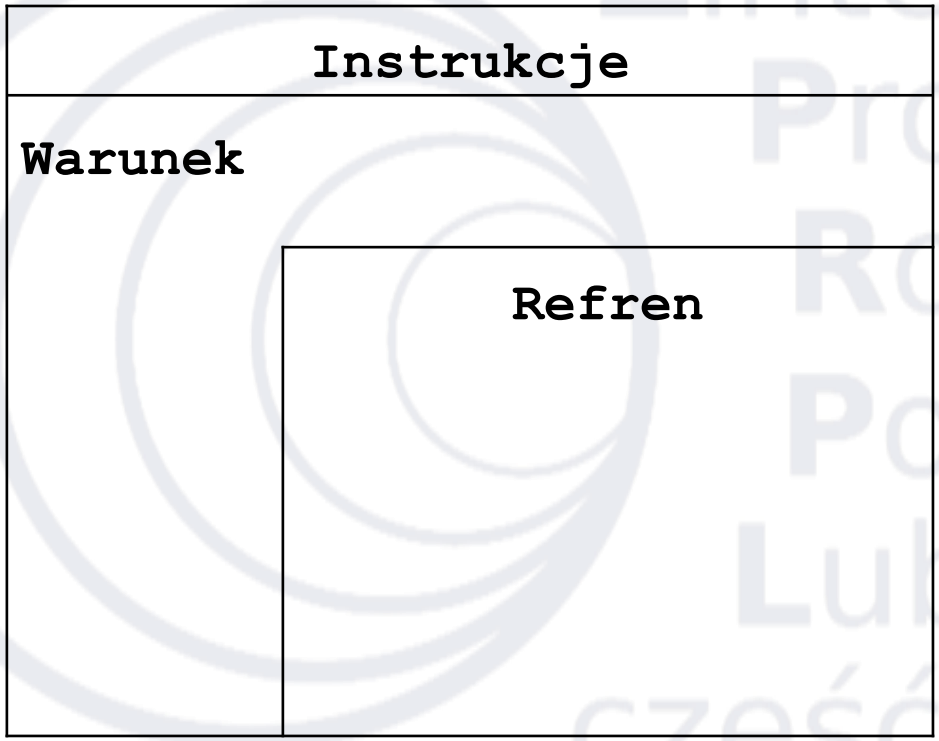
z jednym wariantem pustym

```
Jeżeli (warunek) to
  Instrukcje1;
```

Instrukcje warunkowe – wybór wielowariantowy



Pętla „dopóki”

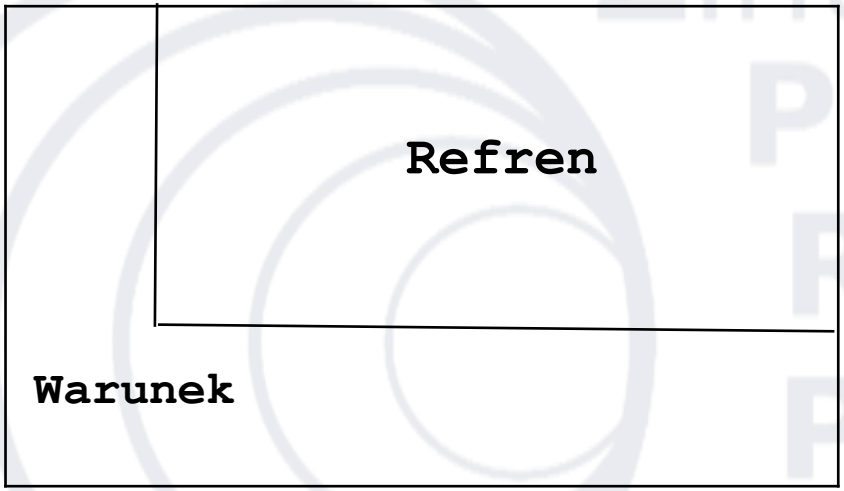



```
dopóki (warunek)
{
 refren pętli
}
```

Konstrukcja schematu N-S dla pętli „dopóki”

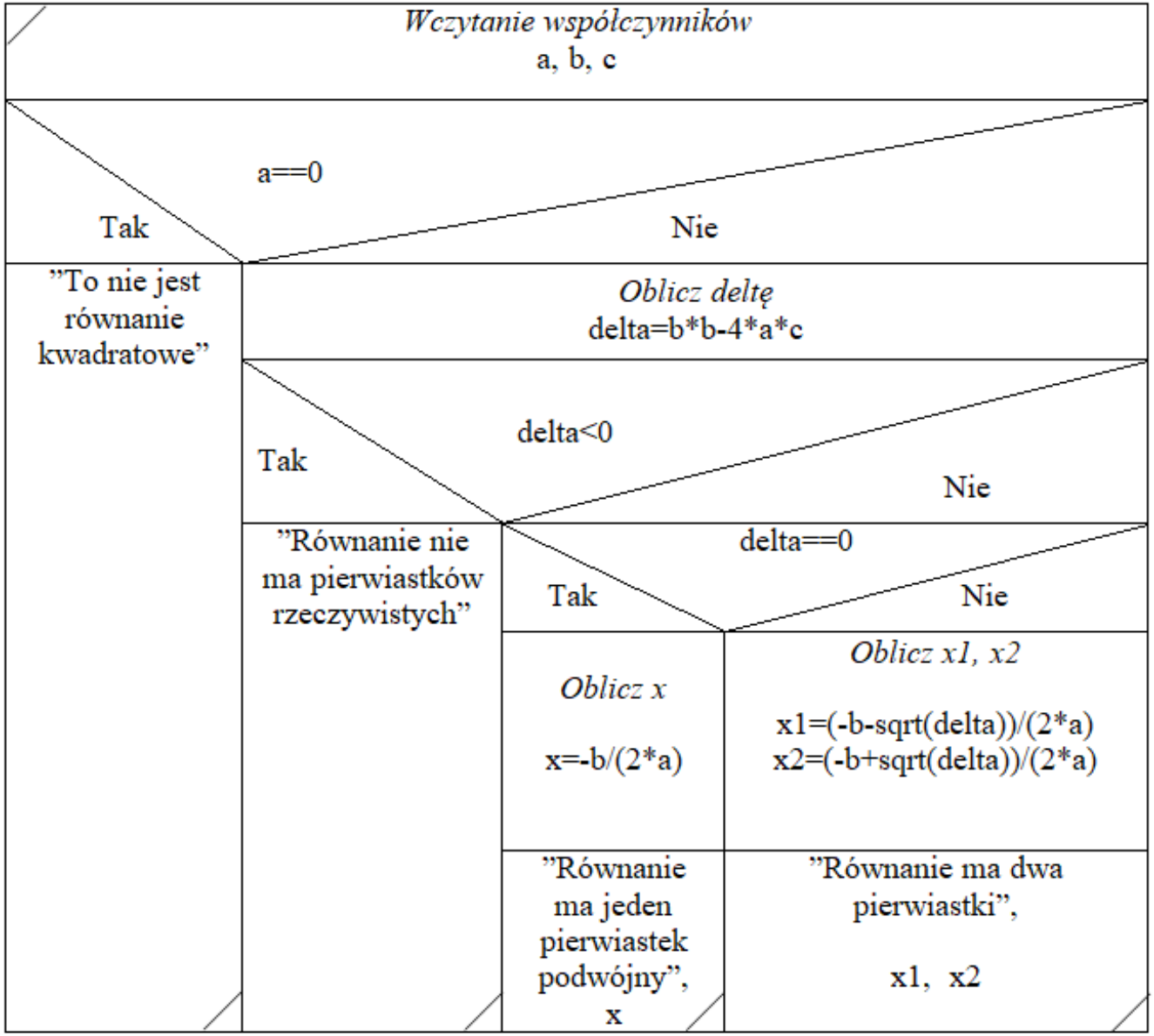
dopóki Warunek jest spełniony powtarzaj Refren

Pętla „aż do”



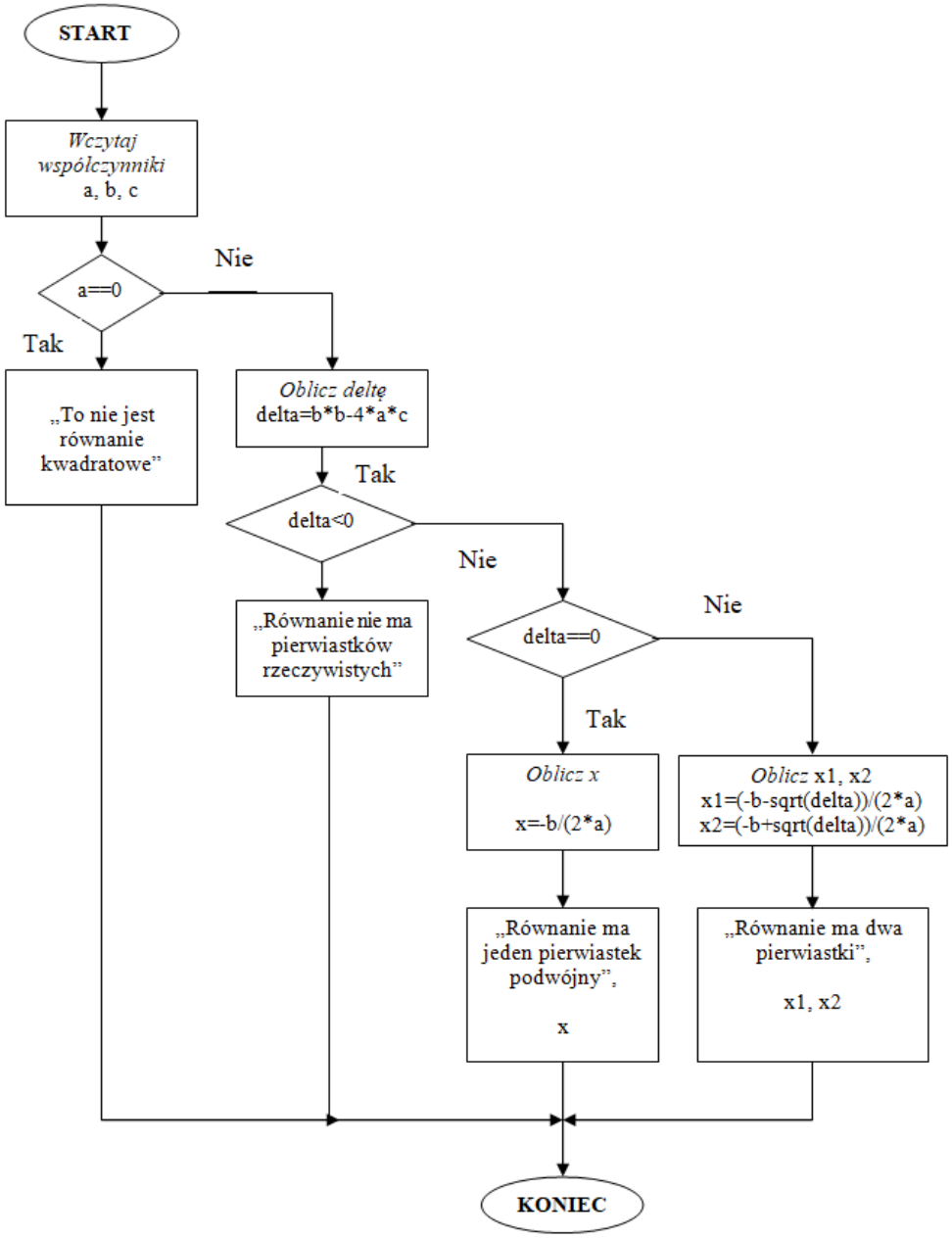
Konstrukcja schematu N-S dla pętli „aż do”

powtarzaj Refren aż do chwili,
gdy stwierdzisz, że Warunek jest spełniony

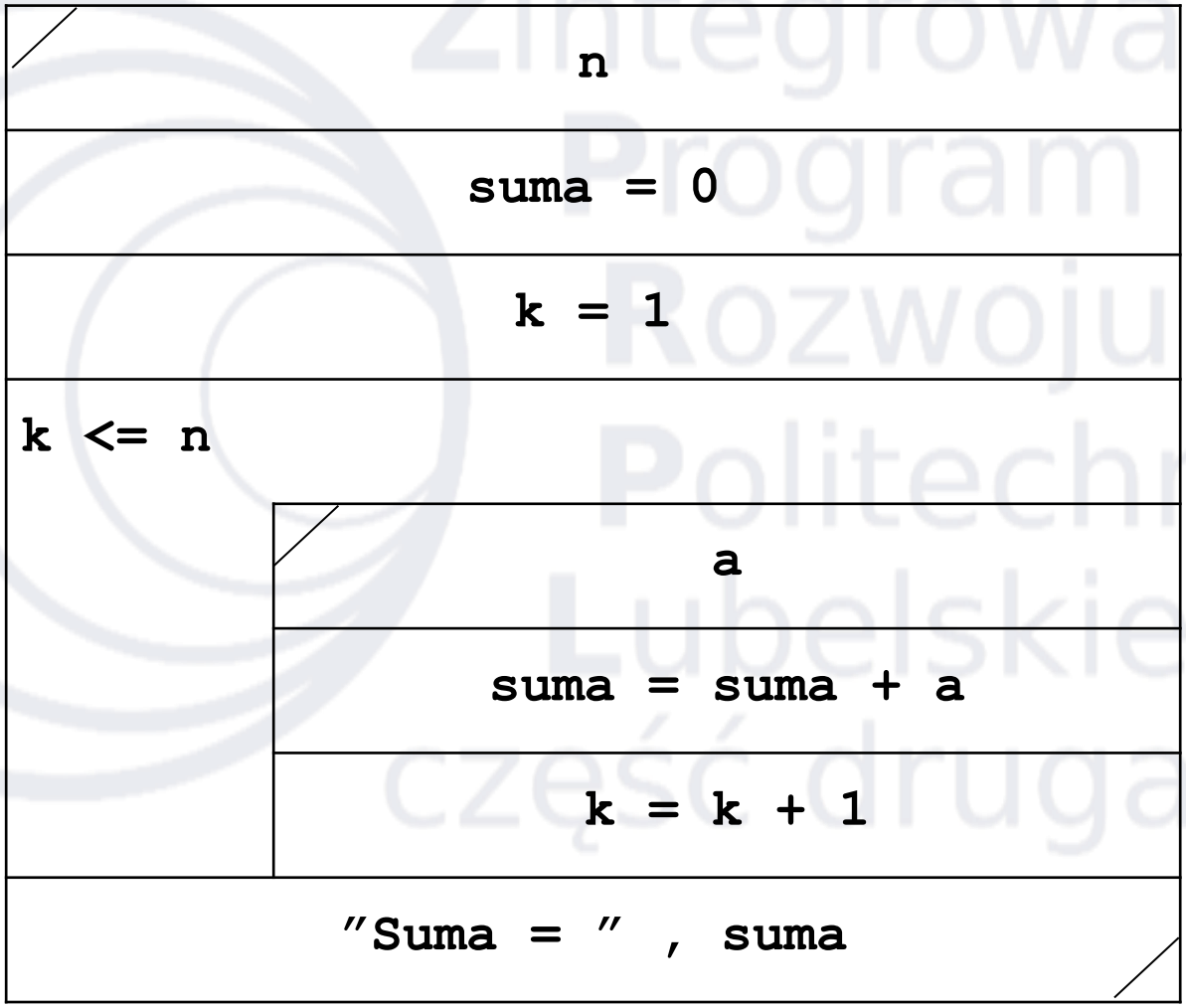


Rozwiązać równanie kwadratowe $ax^2+bx+c=0$

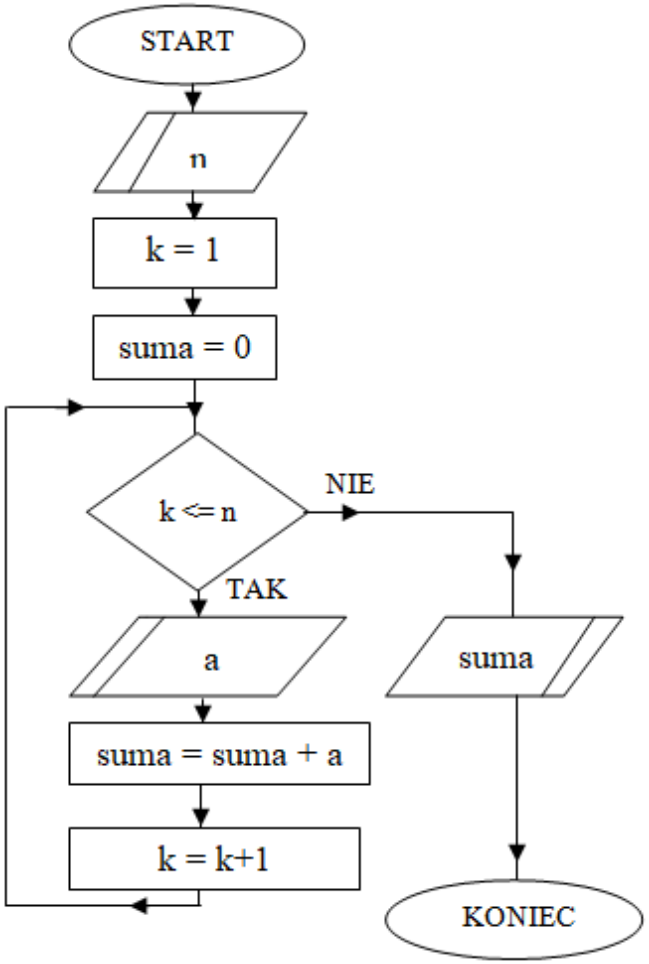
Sieć działań



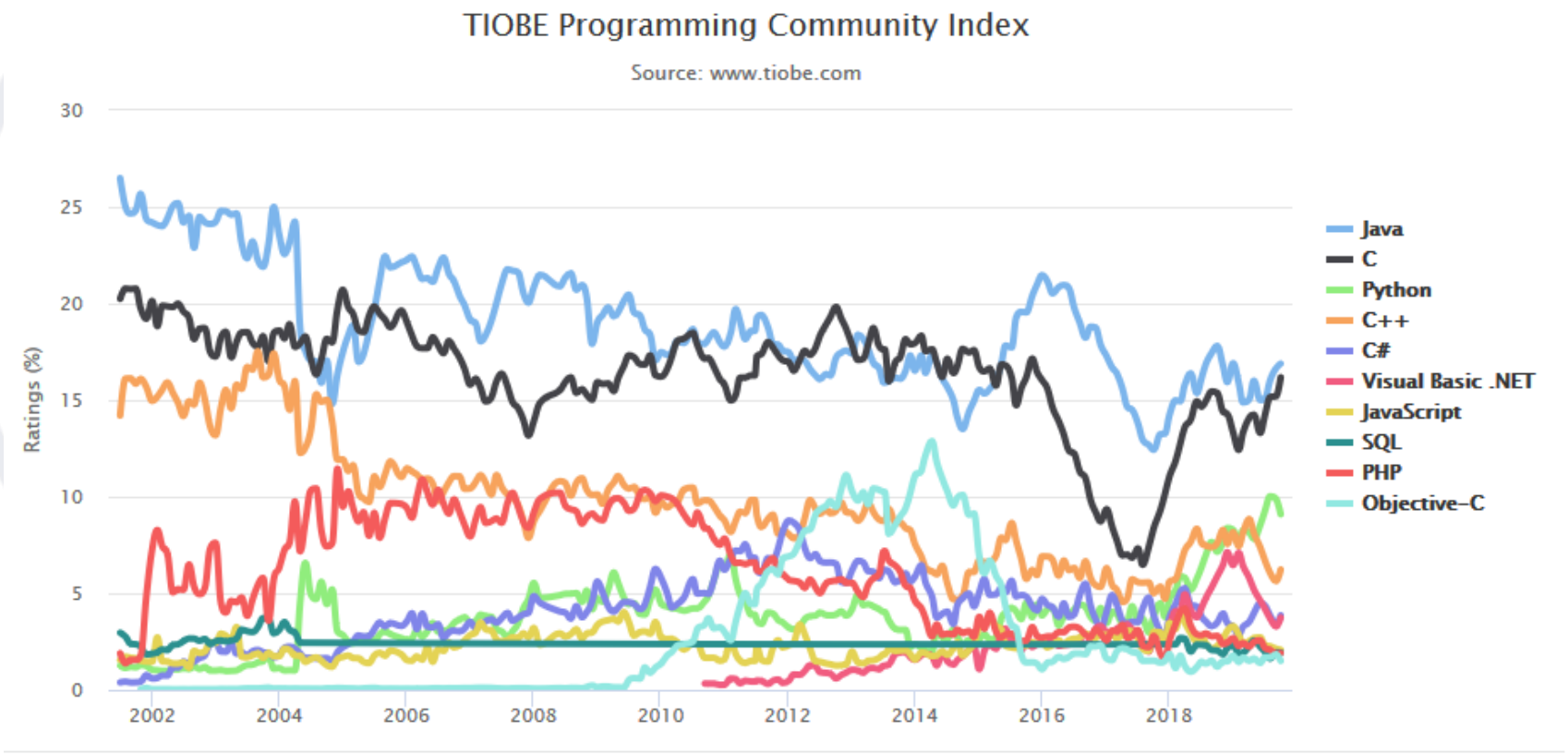
Schemat N-S (suma n wczytywanych liczb)



Schemat blokowy (sieć działań)



Zapis algorytmu w języku programowania

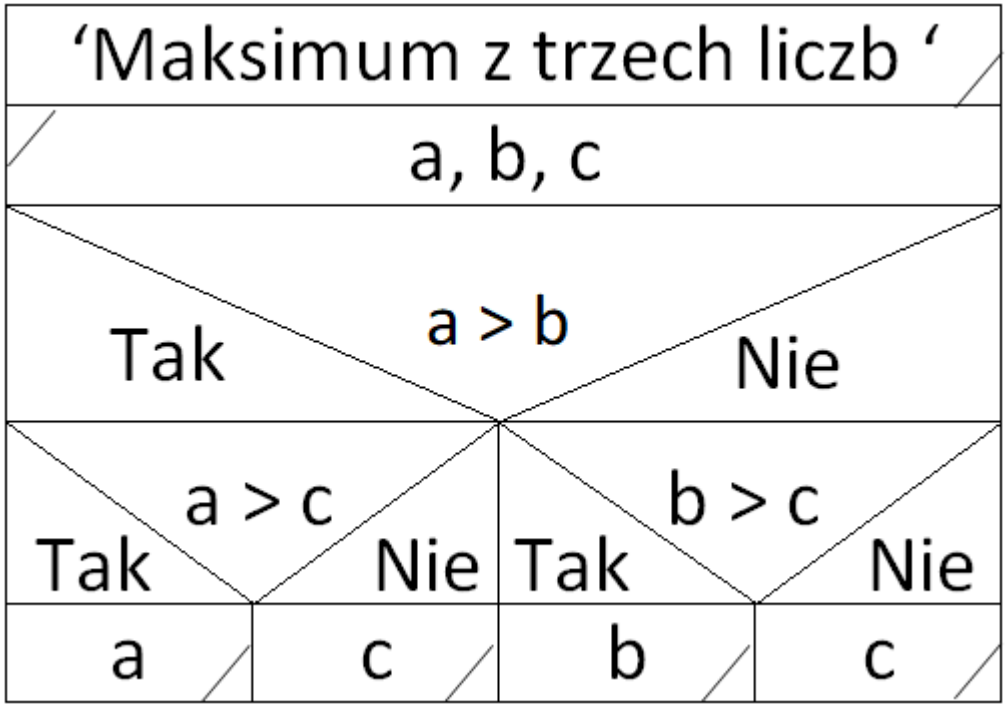


1. Obliczenie objętości kuli

‘Obliczanie objętości kuli ‘
r
pi = 3.14159265
v = 4.0/3.0 * pi * r * r* r
v

```
print('Obliczanie objętości  kuli  ')
r = float(input('r= '))
pi = 3.14159265
v = 4.0/3.0 * pi * r * r * r
print('v = ',v)
```

2. Maksimum z trzech liczb

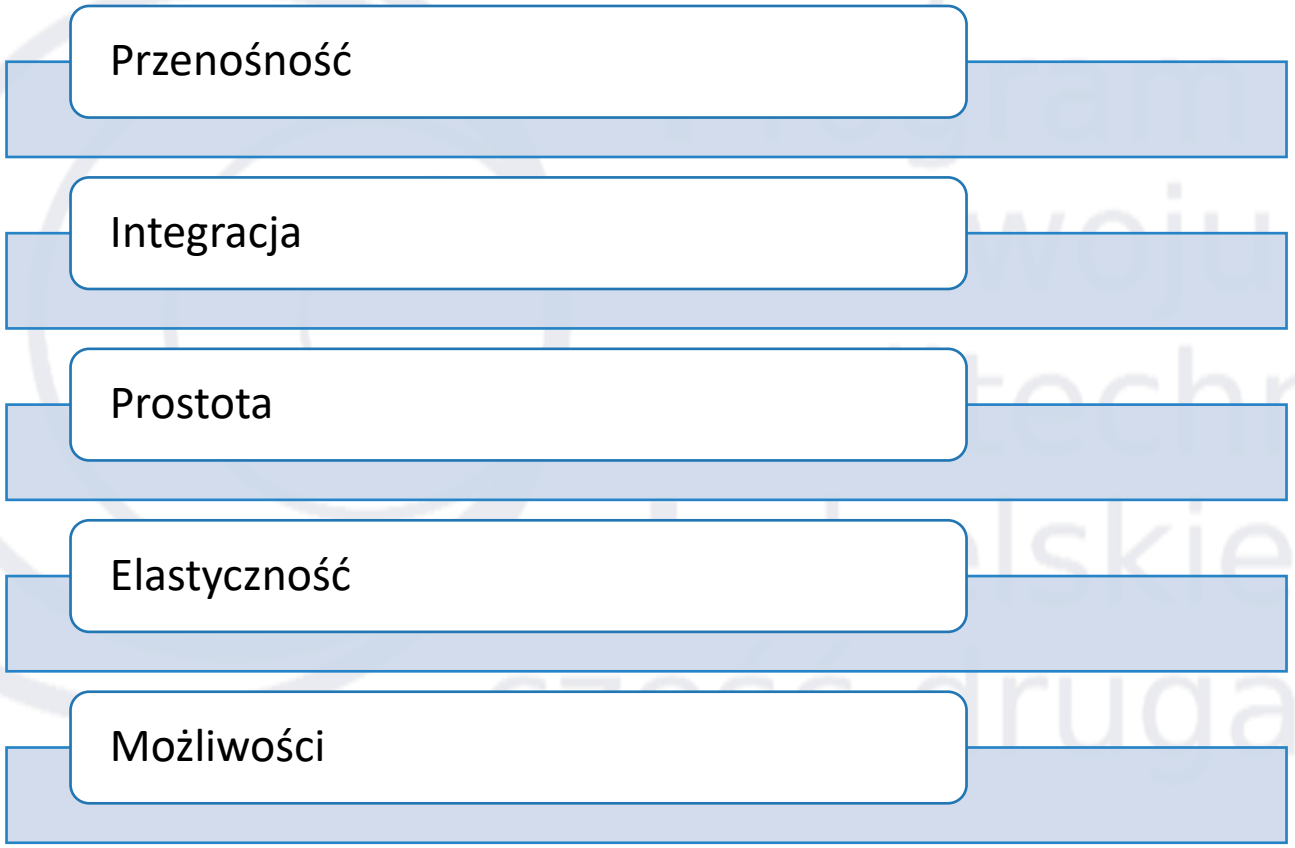



```
print('Maksimum z trzech liczb')
a = int(input('a= '))
b = int(input('b= '))
c = int(input('c= '))
if a > b:
    if a > c:
        print(a)
    else:
        print(c)
else:
    if b > c:
        print(b)
    else:
        print(c)
```

Wprowadzenie do języka Python

- Twórcą języka Python jest holenderski programista **Guido van Rossum**,
- interpretowalny język skryptowy, typu *open source,* zoptymalizowany pod kątem jakości, wydajność, przenośności i integracji, zwiększa wydajność programistów,
- zorientowany obiektowo, posiadający czytelną składnię, bogaty zbiór interfejsów, bibliotek i narzędzi programistycznych,
- łatwy w użyciu, pasuje do naszego toku myślenia.

Dlaczego warto używać Pythona?



Organizacja kodu

- Krótkie komentarze zaznacza się znakiem # na początku wiersza

```
# komentarz
```

- Bloki instrukcji wyróżniane są za pomocą wcięcia, nie stosuje się żadnych znaków specjalnych, jak w innych językach programowania.
- Każdy kolejny poziom zagnieżdżenia w bloku kodu poprzedza odstęp w postaci wielokrotności czterech spacji. Jest to pojedyncza wartość wcięcia.

Liczby

Systemy liczbowe

- Można używać innych systemów niż dziesiętny.
- Liczbę w systemie ósemkowym zapisujemy poprzedzając jej wartość znakiem zera.
- Liczbę w systemie szesnastkowym zapisujemy poprzedzając jej wartość dwuznakiem 0x .

Typy liczb

- liczby całkowite,
- liczby rzeczywiste: 6.24, 6.1e-3,
- liczby zespolone: 2+4j.

Zmienne

Zmienne posiadają swoje nazwy, które je identyfikują.

- Pierwszym znakiem identyfikatora musi być mała lub duża litera alfabetu albo znak podkreślenia, lecz takie zmienne mają specjalne znaczenie.
- Pozostałe znaki mogą zawierać małe lub duże litery alfabetu łacińskiego, znak podkreślenia oraz cyfry (0–9).
- Wielkość znaków w identyfikatorze jest ważna, dlatego
  - `nazwaZmiennej`
  - `nazwazmiennej`

  to dwie różne zmienne.

Zmienne

- Python pozwala używać zmiennych do przechowywania wartości dowolnego typu.
- Utworzenie zmiennej polega na nadaniu jej wartości początkowej:

```
a = 4
x = float(a)
y = int(x)
```

- Funkcje konwersji:

  int(x) na liczby całkowite,

  long(x) na duże liczby całkowite,

  float(x) na liczby wymierne,

  complex(x) na liczby zespolone.

Napisy

- Łańcuchy można umieszczać w apostrofach lub cudzysłowach
  - w napisach ograniczonych apostrofami możemy używać cudzysłowów,
  - w napisach ograniczonych cudzysłowami możemy używać apostrofów.
- W łańcuchach można używać znaków specjalnych, np. znaku nowej linii \n :

  "Pierwszy wiersz\nDrugi wiersz"

Operatory matematyczne

dodawanie	+	`a = a + 2 lub a += 2`
odejmowanie	–	`a = a - 2 lub a -= 2`
mnożenie	*	`a = a * 2 lub a *= 2`
dzielenie	/	`a = a / 2 lub a /= 2`
potęgowanie	**	
dzielenie całkowite	//	
reszta z dzielenia	%	
operacja przypisania	=	

Operatory relacji

większe od	>
mniejsze od	<
większe lub równe z	>=
mniejsze lub równe z	<=
równe z	==
różne od	!=

Operatory logiczne

suma logiczna	or	alternatywa
iloczyn logiczny	and	koniunkcja
negacja	not	

Priorytety i kolejność wykonywania operatorów:

1. Operatory porównania <, >, <=, >=, ==, !=
2. Operator negacji (not)
3. Operator iloczynu logicznego (and)
4. Operator sumy logicznej (or).

Operatory – przykłady użycia

$\frac{a+b}{c+d}$	(a+b)/(c+d)
$x \in [0,6]$	x >= 0 and x <= 6
$x \in (-\infty,-2] \cup [2,\infty)$	x <= –2 or x >= 2
$\frac{-b}{2a}$	–b/(2.0*a)
$\frac{-b}{2a}$	–b/2.0/a
Sprawdzenie, czy liczba a jest liczbą parzystą	a%2 == 0
Wyznaczenie cyfry jedności liczby a	cyfra = a%10

Instrukcja warunkowa

```
if WARUNEK1:
   Instrukcja1
   Instrukcja2
elif WARUNEK2:
   Instrukcja3
   Instrukcja4
else:
   Instrukcja5
   Instrukcja6
Instrukcja7
```

Jeśli jest spełniony wykonaj blok instrukcji

Jeśli nie to zbadaj kolejny warunek

Jeśli nie jest spełniony wykonaj blok instrukcji

Instrukcja następna po instrukcji warunkowej

Części **elif** oraz **else** są opcjonalne**.**

Każda część instrukcji warunkowej kończy się dwukropkiem.

Zagnieżdżanie

Instrukcje warunkowe można zagnieżdżać, co zaznaczone jest poprzez odpowiednio duże wcięcia

```
if WARUNEK1:
  instrukcji
  Instrukcja1
  if WARUNEK2:
       Instrukcja2
  else:
       Instrukcja3
else:
  Instrukcja4
```

Jeśli jest spełniony wykonaj blok

Zagnieżdżona instrukcja warunkowa

Zakończenie bloku instrukcji sygnalizowane jest przez zmniejszenie wcięcia.

Instrukcja pętli for … in

Instrukcja pętli `for … in` służy do iterowania po dowolnej sekwencji obiektów, gdzie sekwencja to uporządkowany zbiór elementów:

```
for i in range(1,5):
    Instrukcja1
    Instrukcja2
else:
    Instrukcja3
Instrukcja4
```

i przyjmie wart. od 1 do 4

Element else jest opcjonalny

Instrukcja3 po zakończeniu pętli

Instrukcja następna po pętli

for i in range(x,y) w Pythonie odpowiada instrukcji
for (int i=x; i<y; i++) w C

Instrukcja pętli while

Instrukcja pętli while pozwala na kilkukrotne wywołanie bloku poleceń tak długo, jak długo warunek będzie prawdziwy:

```
while WARUNEK:
    Instrukcja1
    Instrukcja2
else:
    Instrukcja3
```

Jeśli jest spełniony wykonaj blok

Element else jest opcjonalny
Instrukcja3 wykonana po pętli

Przykład.

```
d = 3
while d:
    d = d-1
print(d)
```

Algorytm Euklidesa

```
print("Podaj dwie liczby naturalne ")
a = int(input("a = "))
b = int(input("b = "))
# ustalenie, która spośród nich jest mniejsza
if  a > b:
    w = a
    m = b
else:
    w = b
    m = a
r = w % m
while r:
    w = m
    m = r
    r = w % m
print("NWD liczb %i i %i wynosi %i,  ich NWW wynosi %i" % (a, b, m, a*b/m))
```

POLITECHNIKA LUBELSKA
WYDZIAŁ ELEKTROTECHNIKI I INFORMATYKI

INFORMATYKA

Zintegrowany Program Rozwoju Politechniki Lubelskiej - część druga

Materiały zostały opracowane w ramach projektu
„Zintegrowany Program Rozwoju Politechniki Lubelskiej – część druga”,
umowa nr **POWR.03.05.00-00-Z060/18-00**
w ramach Programu Operacyjnego Wiedza Edukacja Rozwój 2014-2020
współfinansowanego ze środków Europejskiego Funduszu Społecznego

Fundusze Europejskie Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska Europejski Fundusz Społeczny