Informatyka 1

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia niestacjonarne I stopnia

Rok akademicki 2017/2018

Wykład nr 9 (08.06.2018)

dr inż. Jarosław Forenc

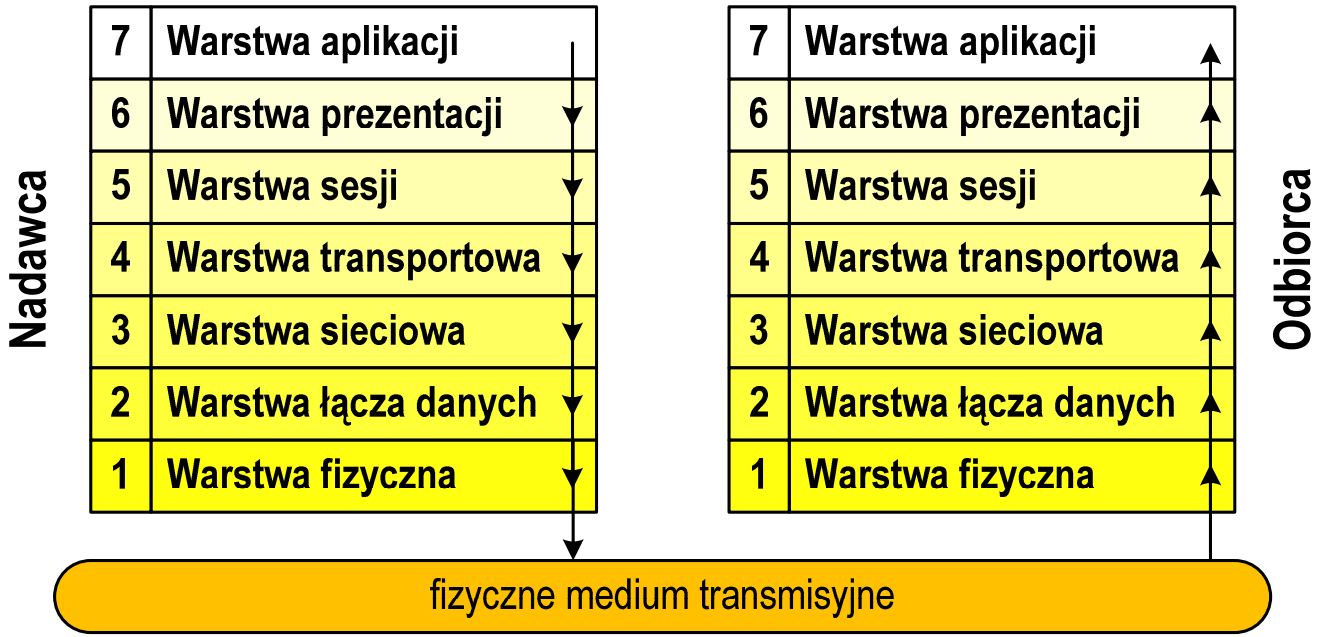
Plan wykładu nr 9

- Sieci komputerowe
  - model referencyjny ISO/OSI, model protokołu TCP/IP
- Algorytmy komputerowe
  - definicje, podstawowe cechy, sposoby opisu
  - rekurencja, złożoność obliczeniowa
- Algorytmy sortowania
  - proste wstawianie
  - proste wybieranie
  - bąbelkowe
  - Quick-Sort

Model ISO/OSI

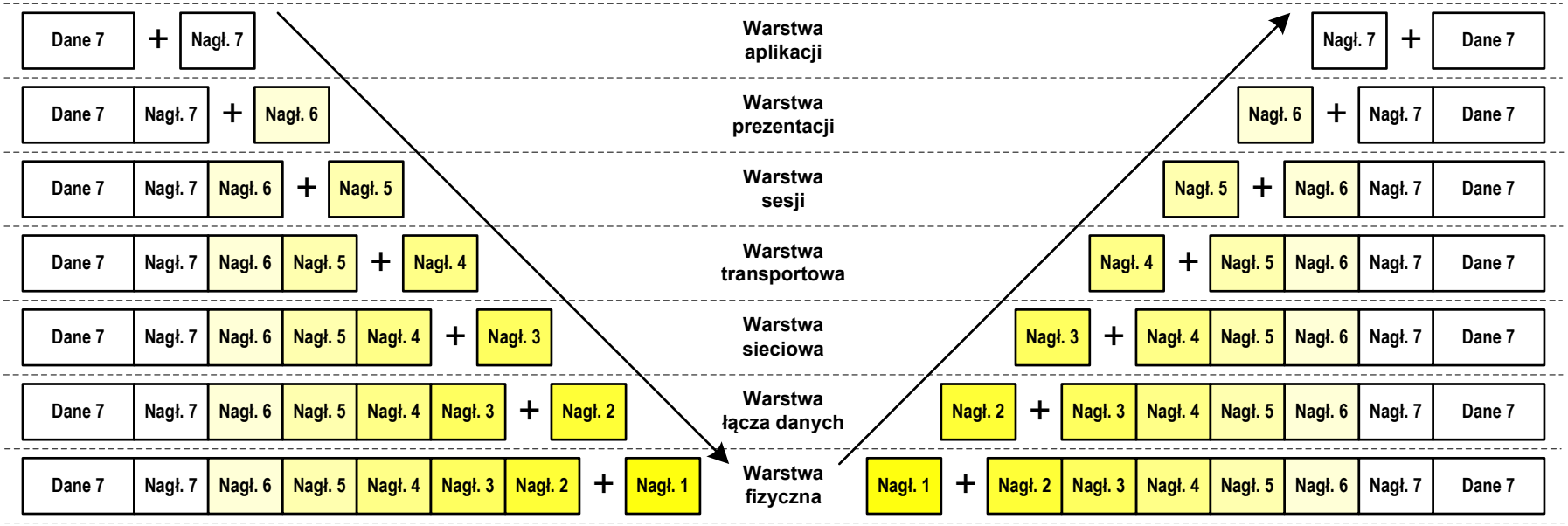
- w latach 70-tych nie istniały ogólne standardy dotyczące sieci komputerowych - każdy producent tworzył własną sieć
- w 1984 roku Międzynarodowa Organizacja Normalizacyjna (ISO) przyjęła model sieciowy, dzięki któremu producenci mogliby opracowywać współpracujące ze sobą rozwiązania sieciowe
- ISO OSI RM - ISO Open Systems Interconnection Reference Model
- głównym założeniem modelu jest podział systemów sieciowych na współpracujące ze sobą 7 warstw (layers)
- struktura tworzona przez warstwy nazywana jest stosem protokołu wymiany danych

Model ISO/OSI



- wierzchołek stosu odpowiada usługom świadczonym bezpośrednio użytkownikowi przez aplikacje sieciowe, zaś dół odpowiada sprzętowi realizującemu transmisję sygnałów
- dane przekazywane są od wierzchołka stosu nadawcy przez kolejne warstwy, aż do warstwy pierwszej, która przesyła je do odbiorcy

Model ISO/OSI



- przy przechodzeniu do warstwy niższej, warstwa dokleja do otrzymanych przez siebie danych nagłówek z informacjami dla swojego odpowiednika na odległym komputerze (odbiorcy)
- warstwa na odległym komputerze interpretuje nagłówek i jeśli trzeba przekazać dane wyżej - usuwa nagłówek i przekazuje dane dalej

Model ISO/OSI a model TCP/IP

- w przypadku protokołu TCP/IP tworzącego Internet stosuje się uproszczony model czterowarstwowy

7	Warstwa aplikacji
6	Warstwa prezentacji
5	Warstwa sesji
4	Warstwa transportowa
3	Warstwa sieciowa
2	Warstwa łącza danych
1	Warstwa fizyczna

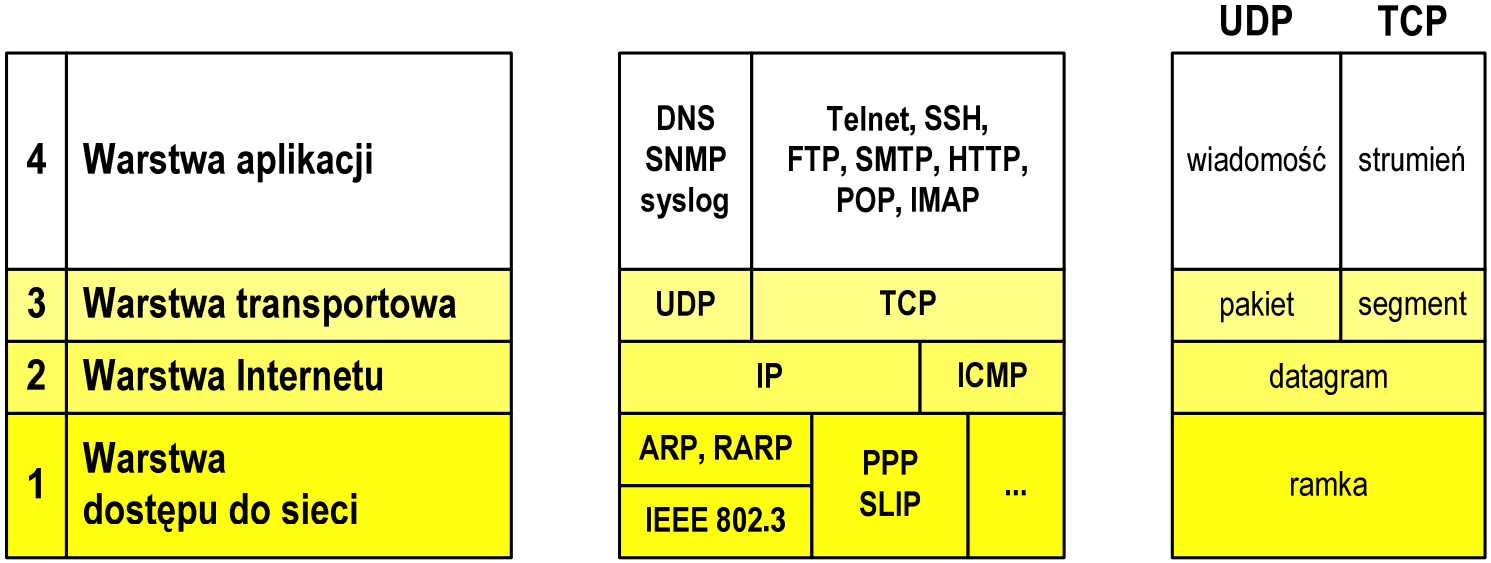
Model ISO/OSI

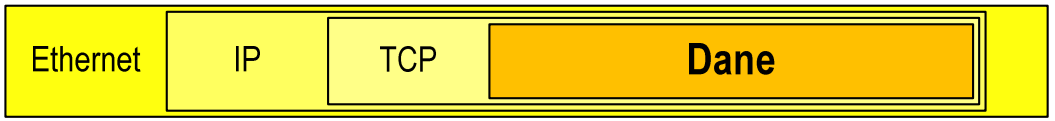
Warstwa aplikacji	4
Warstwa transportowa	3
Warstwa Internetu	2
Warstwa dostępu do sieci	1

Model TCP/IP

Model TCP/IP

- z poszczególnymi warstwami związanych jest wiele protokołów
- protokół - zbiór zasad określających format i sposób przesyłania danych



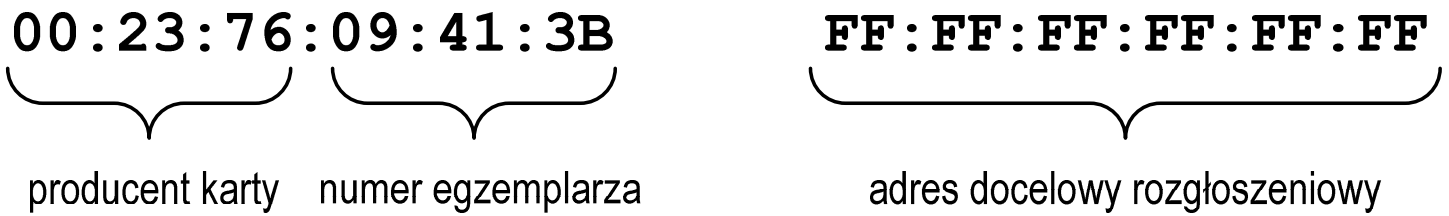


Warstwa dostępu do sieci

- standard IEEE 802.3 (Ethernet) - 1985 r.
- dane przesyłane w postaci ramek Ethernet, format ramki Ethernet II (DIX):

8B	6B	6B	2B	46 - 1500 B	4B
Preambuła	**Adres docelowy**	**Adres źródłowy**	**Typ**	**Dane**	**FCS**

- Preambuła - naprzemienny ciąg bitów 1 i 0 informujący o ramce
- Adres docelowy / źródłowy - 6-bajtowe liczby będące adresami sprzętowymi komunikujących się interfejsów sieciowych (MAC - Media Access Control)



Warstwa dostępu do sieci

- format ramki Ethernet II (DIX)

8B	6B	6B	2B	46 - 1500 B	4B
Preambuła	**Adres docelowy**	**Adres źródłowy**	**Typ**	**Dane**	**FCS**

- Typ - numer protokołu warstwy wyższej, która odbierze dane po zakończeniu obróbki przez standard Ethernet
- Dane - przesyłane dane, jeśli ilość danych jest mniejsza od 46 bajtów, wprowadzane jest uzupełnienie jedynkami (bitowo)
- FCS (Frame Check Sequence) - 4 bajty kontrolne (CRC - Cyclic Redundancy Check) wygenerowane przez interfejs nadający i sprawdzane przez odbierający

Warstwa dostępu do sieci

- format ramki Ethernet II (DIX)

8B	6B	6B	2B	46 - 1500 B	4B
Preambuła	**Adres docelowy**	**Adres źródłowy**	**Typ**	**Dane**	**FCS**

- wysłanie ramki wymaga znajomości adresu MAC odbiorcy
- do określenia adresu MAC na podstawie numeru IP stosowany jest protokół ARP (Address Resolution Protocol)
- protokół ARP stosowany jest także do zapobiegania zdublowaniu adresów IP
- aktualną tablicę translacji ARP wyświetla polecenie: arp -a

Warstwa Internetu

- najważniejsza część Internetu to protokół IP (Internet Protocol):
  - definiuje format i znaczenie pól datagramu IP
  - określa schemat adresowania stosowany w Internecie
  - zapewnia wybór trasy przesyłania datagramu (routing)
  - zapewnia podział danych na fragmenty i łączenie ich w całość w przypadku sieci nie akceptujących rozmiaru przenoszonych danych
- cechy protokołu:
  - bezpołączeniowy - nie ustanawia połączenia i nie sprawdza gotowości odbiorcy danych
  - niepewny - nie zapewnia korekcji i wykrywania błędów transmisji

Warstwa Internetu - datagram IP

- Wersja (Version) - numer wersji protokołu IP (IPv4, nowsza - IPv6)
- IHL (Internal Header Length) - długość nagłówka w 32-bitowych słowach
- Typ usługi (Type of Service) - opisuje wymaganą jakość usługi (pole najczęściej ignorowane przez routery)
- Długość całkowita (Datagram Length) - długość pakietu IP w bajtach (Nagłówek + Dane)

Warstwa Internetu - datagram IP

- Identyfikator (Identification), Flagi (Flags), Przesunięcie fragmentacji (Fragment offset) - pola używane w przypadku podziału datagramu na części (fragmenty)
- Czas życia TTL (Time-to-Live) - maksymalny czas (w sekundach) pozostawania datagramu w Internecie, przejście datagramu przez każdy router zmniejsza wartość o 1
- Protokół (Protocol) - numer protokołu warstwy wyższej, do którego zostaną przekazane dane z tego pakietu

Warstwa Internetu - datagram IP

- Suma kontrolna (Header checksum) - suma kontrolna nagłówka
- Adres źródła (Source Address) - adres IP źródła danych
- Adres przeznaczenia (Destination Address) - adres IP odbiorcy danych
- Opcje (Options) - dodatkowe opcje
- Uzupełnienie (Padding) - uzupełnienie pola opcji do pełnego słowa (32 bitów)

Warstwa Internetu - adresy IP

- adres IP komputera zajmuje 4 bajty (32-bitowa liczba całkowita)
- najczęściej zapisywany jest w postaci 4 liczb z zakresu od 0 do 255 każda, oddzielonych kropkami, np.

```
213.33.95.114
```

```
11010100.00100001.01011111.01110010
```

- adres składa się z dwóch części:
  - identyfikującej daną sieć w Internecie
  - identyfikującej konkretny komputer w tej sieci
- do roku 1997 wyróżnienie części określającej sieć i komputer w sieci następowało na podstawie tzw. klas adresów IP

Warstwa Internetu - klasy adresów IP

Klasa A

`0nnnnnnn . hhhhhhhh . hhhhhhhh . hhhhhhhh`

sieć (max. 126) — komputer (max. 16 777 214)

Zakres IP
od: 1.0.0.0
do: 126.255.255.255

Klasa B

`10nnnnnn . nnnnnnnn . hhhhhhhh . hhhhhhhh`

sieć (max. 16 382) — komputer (max. 65 534)

Zakres IP
od: 128.1.0.0
do: 191.255.255.255

Klasa C

`110nnnnn . nnnnnnnn . nnnnnnnn . hhhhhhhh`

sieć (max. 2 097 150) — komputer (max. 254)

Zakres IP
od: 192.0.0.0
do: 223.255.255.255

Klasa D

`1110xxxx . xxxxxxxx . xxxxxxxx . xxxxxxxx`

multicast - adresy transmisji grupowej, np. wideokonferencje

Zakres IP
od: 224.0.0.0
do: 239.255.255.255

Klasa E

`1111xxxx . xxxxxxxx . xxxxxxxx . xxxxxxxx`

zarezerwowane na potrzeby badawcze

Zakres IP
od: 240.0.0.0
do: 255.255.255.255

Warstwa Internetu - maska sieci

- klasy adresów IP zostały zastąpione bezklasowym routowaniem międzydomenowym CIDR (Classless Inter-Domain Routing)
- do określenia liczby bitów odpowiadających sieci i liczby bitów odpowiadających hostowi stosowana jest maska sieci

IP:	212.33.95.114	11010100.00100001.01011111.01110010
Maska:	255.255.255.192	11111111.11111111.11111111.11000000

Adres sieci:	212.33.95.64	11010100.00100001.01011111.01000000
Broadcast:	212.33.95.127	11010100.00100001.01011111.01111111

Pierwszy host:	212.33.95.65	11010100.00100001.01011111.01000001
Ostatni host:	212.33.95.126	11010100.00100001.01011111.01111110

Warstwa Internetu - adresy IP

- adresy specjalne

`0.0.0.0`	- adres sieci dla całego Internetu
`255.255.255.255`	- adres rozgłoszeniowy dla całego Internetu
`127.0.0.1`	- adres pętli (loop-back address) - stosowany do komunikacji z lokalnym komputerem (localhost)

- adresy prywatne (nierutowalne) - nie są przekazywane przez routery

`10.0.0.0 – 10.255.255.255`	- klasa A
`172.16.0.0 – 172.31.255.255`	- klasa B
`192.168.0.0 – 192.168.255.255`	- klasa C

Warstwa transportowa - porty

- protokoły warstwy transportowej zapewniają dostarczenie danych do konkretnych aplikacji (procesów) w odpowiedniej kolejności i formie
- identyfikacja przynależności danej transmisji do procesu odbywa się na podstawie numeru poru (liczba 16-bitowa, zakres: 0 ÷ 65535)
- numery portów przydzielane są przez organizację IANA (Internet Assigned Numbers Authority):
  - 0 ÷ 1023 - zakres zarezerwowany dla tzw. dobrze znanych portów (well-know port number)
  - 1024 ÷ 49151 - porty zarejestrowane (registered)
  - 49152 ÷ 65535 - porty dynamiczne/prywatne (dynamic/private)
- połączenie numeru IP komputera i portu, na którym odbywa się komunikacja, nazywa się gniazdem (socket)

Warstwa transportowa - porty

- wybrane dobrze znane porty:

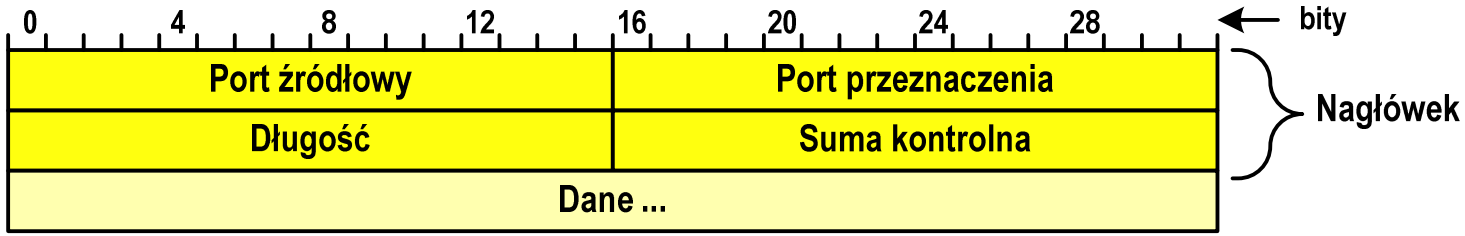
port	protokół
20	FTP (dane)
21	FTP (polecenia)
22	SSH
23	Telnet
25	SMTP (mail)

port	protokół
53	DNS
80	HTTP (www)
110	POP3 (mail)
119	NNTP (news)
143	IMAP (mail)

- w warstwie transportowej funkcjonują dwa podstawowe protokoły:
  - UDP (User Datagram Protocol)
  - TCP (Transmission Control Protocol)

Warstwa transportowa - protokół UDP

- UDP wykonuje usługę bezpołączeniowego dostarczania datagramów:
  - nie ustanawia połączenia
  - nie sprawdza gotowości odbiorcy do odebrania przesyłanych danych
  - nie sprawdza poprawności dostarczenia danych
- jednostką przesyłanych danych jest pakiet



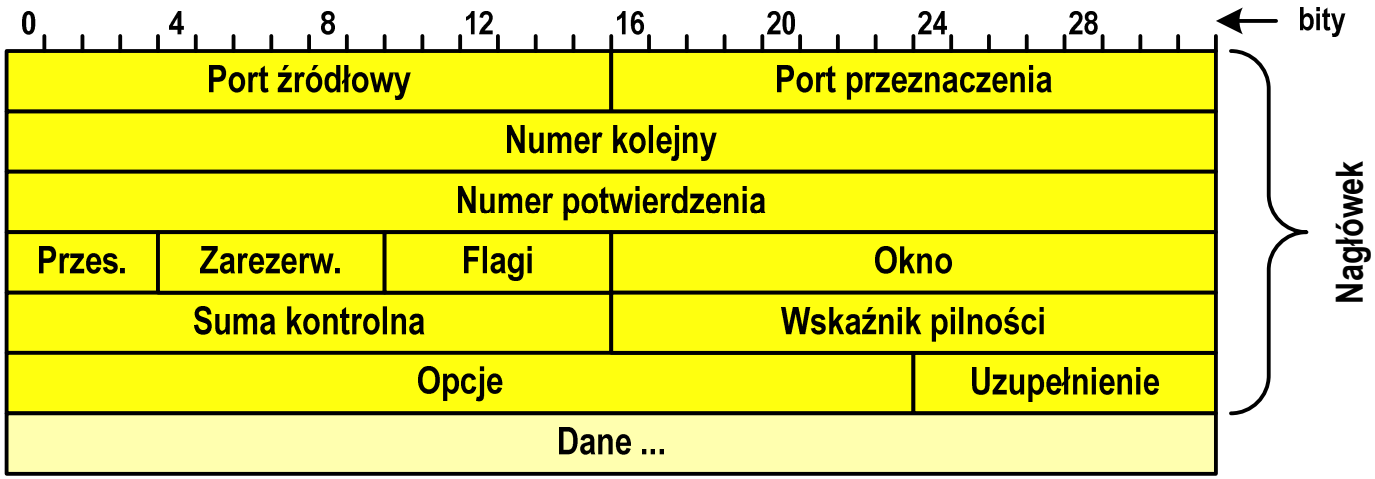
- Port źródłowy (Source port) - numer portu nadawcy
- Port przeznaczenia (Destination port) - numer portu odbiorcy
- Długość (Length) - całkowita długość pakietu w bajtach (nagłówek + dane)
- Suma kontrolna (Checksum) - tworzona na podstawie nagłówka i danych

Warstwa transportowa - protokoły UDP i TCP

- UDP stosowany jest, gdy ilość przesyłanych danych w pakiecie jest niewielka
- pakiet UDP zawiera bardzo mało informacji kontrolnych, zatem opłacalne jest jego stosowanie w powiązaniu z aplikacjami samodzielnie dbającymi o kontrolę poprawności transmisji

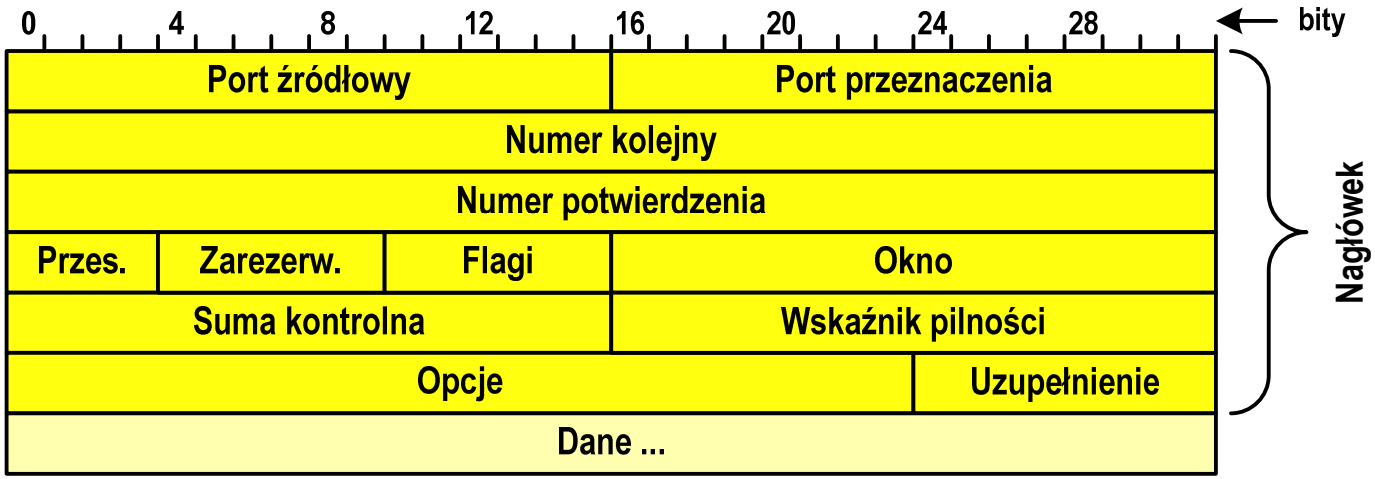
- TCP (Transmission Control Protocol) jest protokołem niezawodnym i połączeniowym, działa na strumieniach bajtów
- TCP sprawdza czy dane zostały dostarczone poprawnie i w określonej kolejności
- jednostką przesyłanych danych stosowaną przez TCP jest segment

Warstwa Internetu - segment TCP



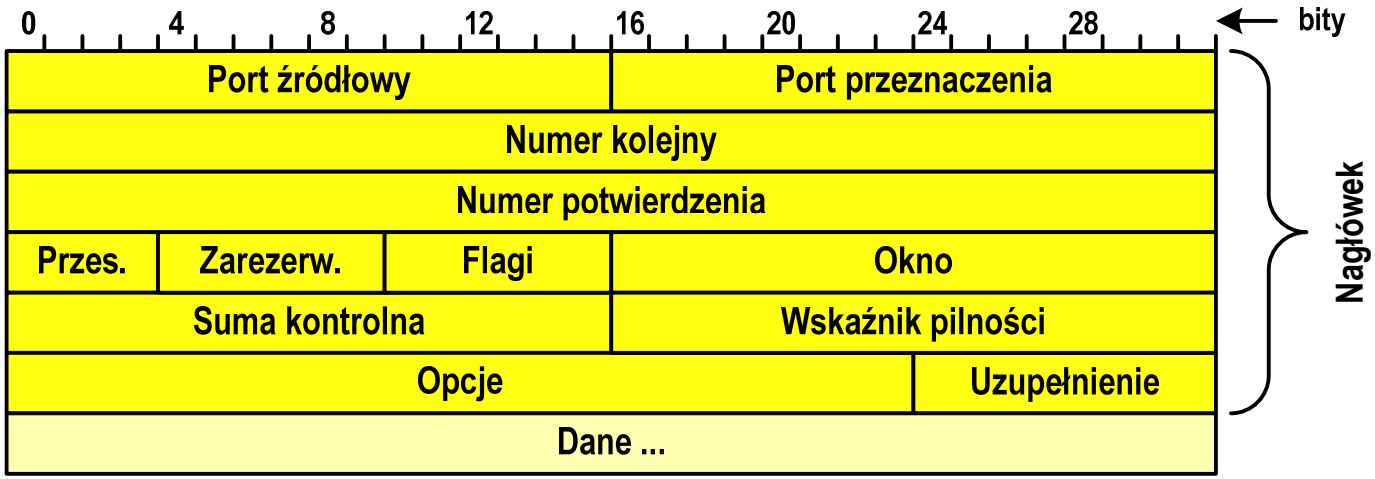
- Port źródłowy (Source port) - numer portu nadawcy
- Port przeznaczenia (Destination port) - numer portu odbiorcy
- Numer kolejny (Sequence number) - identyfikator określający miejsce segmentu przed fragmentacją
- Numer potwierdzenia (Acknowledgment number) - identyfikator będący potwierdzeniem otrzymania danych przez odbiorcę

Warstwa Internetu - segment TCP



- Przesunięcie (Data offset) - liczba 32-bitowych słów w nagłówku TCP
- Zarezerwowane (Reserved) - zarezerwowane do przyszłych zastosowań
- Flagi (Flags) - flagi dotyczące bieżącego segmentu
- Okno (Window) - określa liczbę bajtów, które aktualnie odbiorca może przyjąć (0 - wstrzymanie transmisji)

Warstwa Internetu - segment TCP



- Suma kontrolna (Checksum) - suma kontrolna nagłówka i danych
- Wskaźnik pilności (Urgent pointer) - jeśli odpowiednia flaga jest włączona (URG), to informuje o pilności pakietu
- Opcje (Options) - dodatkowe opcje
- Uzupełnienie (Padding) - uzupełnienie pola opcji do pełnego słowa (32 bitów)

Warstwa aplikacji

- zawiera szereg procesów (usług, protokołów) wykorzystywanych przez uruchamiane przez użytkownika aplikacje do przesyłania danych
- większość usług działa w architekturze klient-serwer (na odległym komputerze musi być uruchomiony serwer danej usługi)

DNS (Domain Name System)

- świadczy usługi zamieniania (rozwiązywania) nazwy komputera na jego adres IP

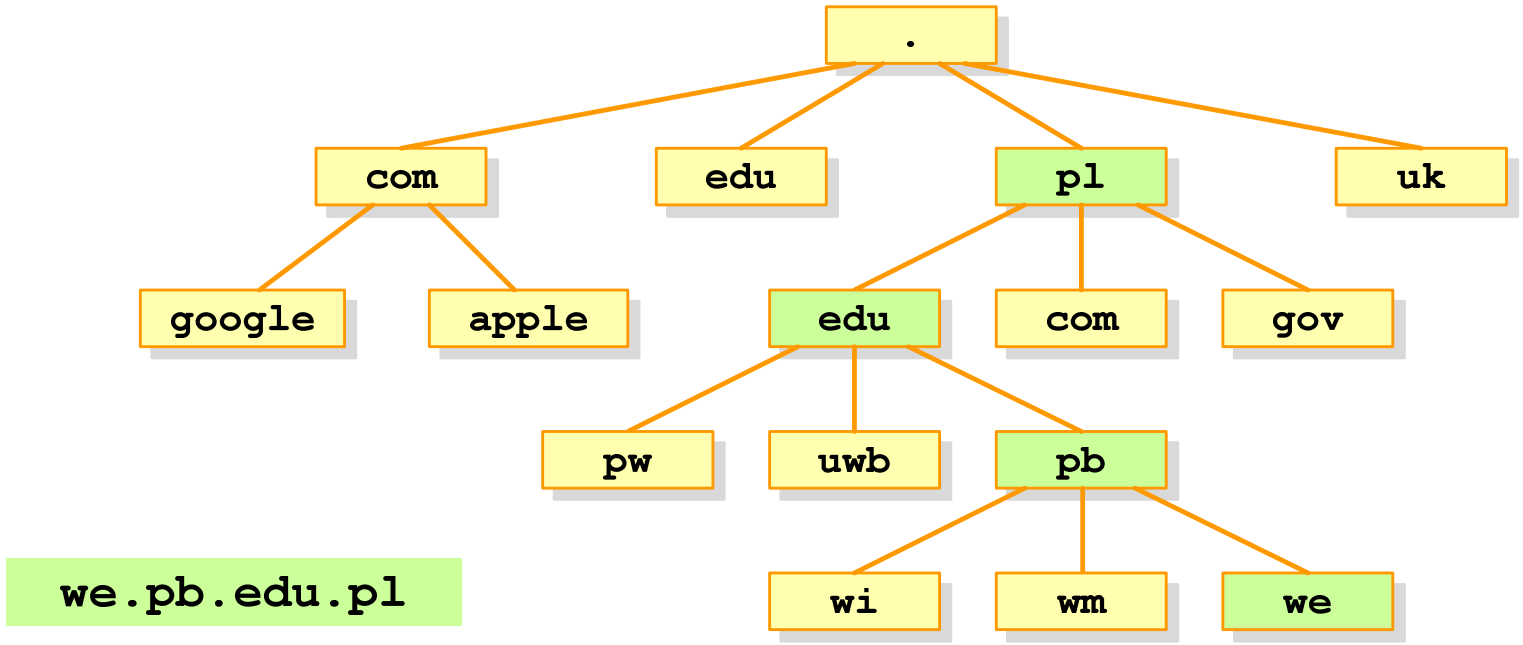
`we.pb.edu.pl` → `213.33.95.2`

- wykorzystuje port o numerze 53

Warstwa aplikacji

DNS (Domain Name System)

- przestrzeń nazw w Internecie oparta jest na modelu odwróconego drzewa



- zarządzaniem przestrzenią nazw domenowych zajmuje się w świecie ICANN, zaś w Polsce - NASK

Warstwa aplikacji

SMTP (Simple Mail Transfer Protocol)

- umożliwia wysyłanie (ale nie odbieranie) i transport poczty elektronicznej e-mail poprzez różnorodne środowiska systemowe
- podczas przesyłania e-maila każdy serwer SMTP dodaje swój nagłówek
- wykorzystuje port o numerze 25

POP (Post Office Protocol)

- umożliwia odbieranie poczty ze zdalnego serwera na komputer lokalny
- ma wiele ograniczeń: każda wiadomość jest pobierana z załącznikami, nie pozwala przeglądać oczekujących w kolejce wiadomości
- ostatnia wersja to POP3
- wykorzystuje port o numerze 110

Warstwa aplikacji

IMAP (Internet Message Access Protocol)

- następca POP3
- pozwala na umieszczenie wiadomości na serwerze w wielu folderach
- umożliwia zarządzanie wiadomościami (usuwanie, przenoszenie pomiędzy folderami) oraz ściąganie tylko nagłówków wiadomości
- wykorzystuje port o numerze 143

FTP (File Transfer Protocol)

- umożliwia wysyłanie i odbiór plików z odległego systemu oraz wykonywanie operacji na tych plikach
- umożliwia dostęp anonimowy - login: anonymous, password: e-mail
- dwa tryby pracy: aktywny (active) i pasywny (passive)
- wykorzystuje dwa porty: 21 (polecenia), 20 (dane)

Algorytm - definicje

Definicja 1

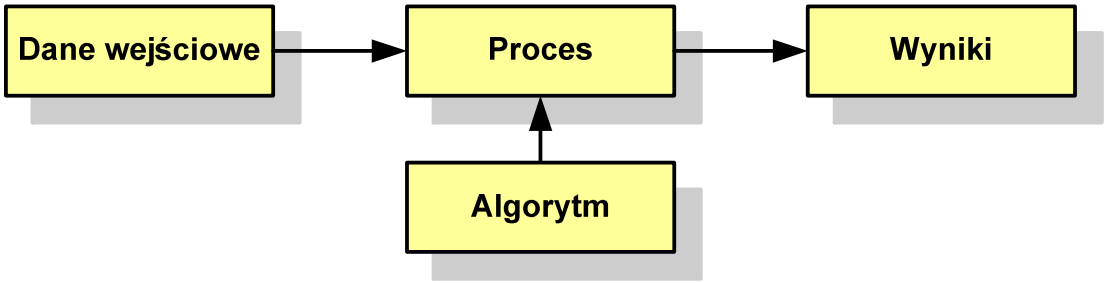
- Skończony, uporządkowany ciąg jasno zdefiniowanych czynności, koniecznych do wykonania pewnego zadania

Definicja 2

- Metoda rozwiązania zadania

Definicja 3

- Ściśle określona procedura obliczeniowa, która dla właściwych danych wejściowych zwraca żądane dane wyjściowe zwane wynikiem działania algorytmu



Algorytmy

- Słowo „algorytm” pochodzi od nazwiska Muhammada ibn-Musy al-Chuwarizmiego (po łacinie pisanego jako Algorismus), matematyka perskiego z IX wieku
- Badaniem algorytmów zajmuje się algorytmika
- Algorytm może zostać zaimplementowany w postaci programu komputerowego
- Przetłumaczenie algorytmu na wybrany język programowania nazywane jest też kodowaniem algorytmu
- Ten sam algorytm może być zaimplementowany (zakodowany) w różny sposób przy użyciu różnych języków programowania.

Podstawowe cechy algorytmu

- Posiada dane wejściowe (w ilości większej lub równej zero) pochodzące z dobrze zdefiniowanego zbioru
- Zwraca wynik
- Jest precyzyjne zdefiniowany (każdy krok algorytmu musi być jednoznacznie określony)
- Poprawność (dla każdego z założonego dopuszczalnego zestawu danych wejściowych)
- Kończy działanie po skończonej liczbie kroków (powinna istnieć poprawnie działająca reguła stopu algorytmu)
- Efektywność (jak najkrótszy czas wykonania i jak najmniejsze zapotrzebowanie na pamięć).

Sposoby opisu algorytmów

1. Opis słowny w języku naturalnym lub w postaci listy kroków (opis w punktach)
2. Schemat blokowy
3. Pseudokod (nieformalna odmiana języka programowania)
4. Wybrany język programowania

Opis słowny algorytmu

- Podanie kolejnych czynności, które należy wykonać, aby otrzymać oczekiwany efekt końcowy
- Przypomina przepis kulinarny z książki kucharskiej lub instrukcję obsługi urządzenia, np.

Algorytm:	Tortilla („Podróże kulinarne” R. Makłowicza)
Dane wejściowe:	0,5 kg ziemniaków, 100 g kiełbasy Chorizo, 8 jajek
Dane wyjściowe:	gotowa Tortilla
Opis algorytmu:	Ziemniaki obrać i pokroić w plasterki. Kiełbasę pokroić w plasterki. Ziemniaki wrzucić na gorącą oliwę na patelni i przyrumienić z obu stron. Kiełbasę wrzucić na gorącą oliwę na patelni i przyrumienić z obu stron. Ubić jajka i dodać do połączonych ziemniaków i kiełbasy. Dodać sól i pieprz. Usmażyć z obu stron wielki omlet nadziewany chipsami ziemniaczanymi z kiełbaską.

Lista kroków

- Uporządkowany opis wszystkich czynności, jakie należy wykonać podczas realizacji algorytmu
- Krok jest to pojedyncza czynność realizowana w algorytmie
- Kroki w algorytmie są numerowane, operacje wykonywane są zgodnie z rosnącą numeracją kroków
- Jedynym odstępstwem od powyższej reguły są operacje skoku (warunkowe lub bezwarunkowe), w których jawnie określa się numer kolejnego kroku
- Przykład (instrukcja otwierania wózka-specerówki):

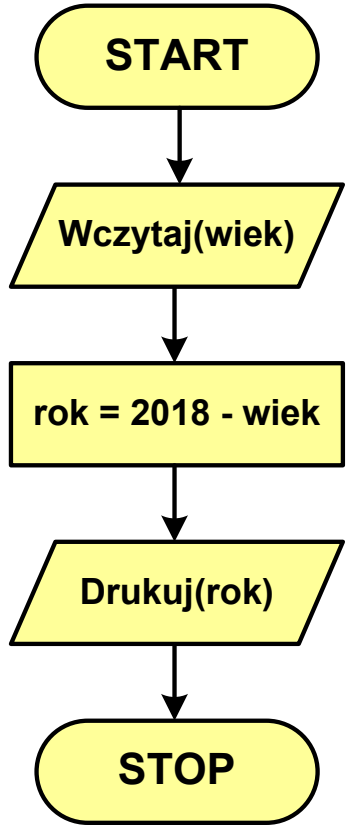
  Krok 1: Zwolnij element blokujący wózek

  Krok 2: Rozkładaj wózek w kierunku kółek

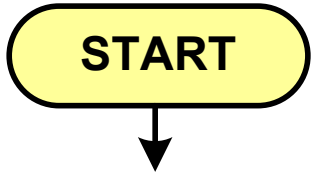
  Krok 3: Naciskając nogą dolny element blokujący aż do zatrzaśnięcia, rozłóż wózek do pozycji przewozowej

Schemat blokowy

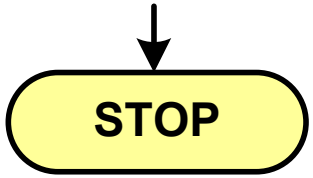
- Zawiera plan algorytmu przedstawiony w postaci graficznej
- Na schemacie umieszczane są bloki oraz linie przepływu (strzałki)
- Blok zawiera informację o wykonywanej operacji
- Linie przepływu (strzałki) określają kolejność wykonywania bloków algorytmu
- Przykład: wyznaczanie roku urodzenia na podstawie wieku (algorytm liniowy)



Schemat blokowy - symbole graficzne

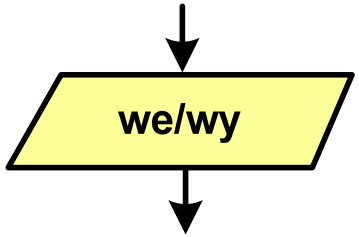


- blok startowy, początek algorytmu
- wskazuje miejsce rozpoczęcia algorytmu
- ma jedno wyjście
- może występować tylko jeden raz

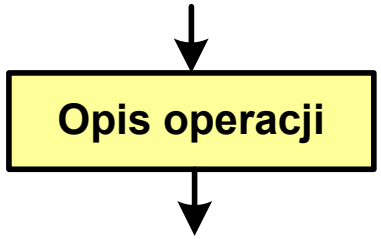


- blok końcowy, koniec algorytmu
- wskazuje miejsce zakończenia algorytmu
- ma jedno wejście
- musi występować przynajmniej jeden raz

Schemat blokowy - symbole graficzne

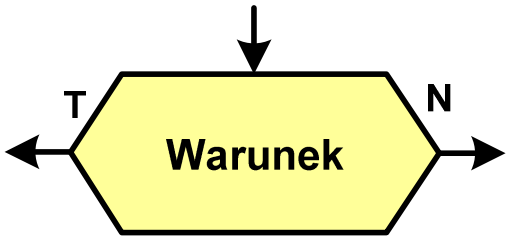


- blok wejścia-wyjścia
- poprzez ten blok wprowadzane są (czytane) dane wejściowe i wyprowadzane (zapisywane) wyniki
- ma jedno wejście i jedno wyjście

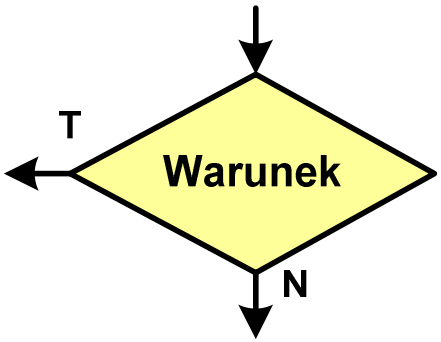


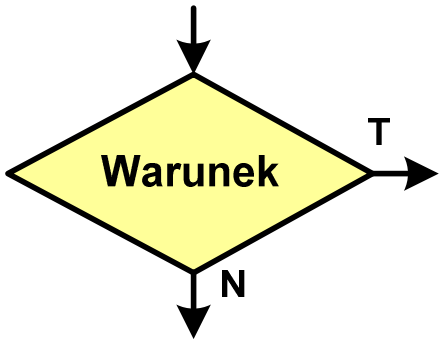
- blok wykonawczy, blok funkcyjny, opis procesu
- zawiera jedno lub kilka poleceń (elementarnych instrukcji) wykonywanych w podanej kolejności
- instrukcją może być np. operacja arytmetyczna, podstawienie
- ma jedno wejście i jedno wyjście

Schemat blokowy - symbole graficzne

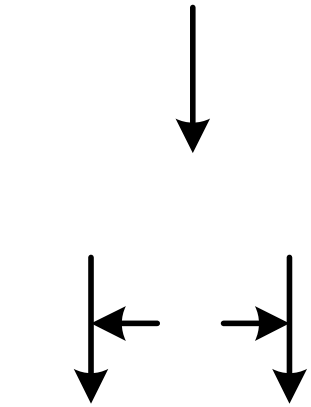


- blok warunkowy (decyzyjny, porównujący)
- wewnątrz bloku umieszcza się warunek logiczny
- na podstawie warunku określana jest tylko jedna droga wyjściowa
- połączenia wychodzące z bloku:
  - T lub TAK - gdy warunek jest prawdziwy
  - N lub NIE - gdy warunek nie jest prawdziwy
- wyjścia mogą być skierowane na boki lub w dół

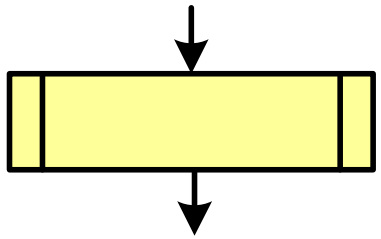




Schemat blokowy - symbole graficzne

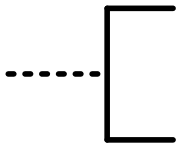


- linia przepływu, połączenie, linia
- występuje w postaci linii zakończonej strzałką
- określa kierunek przemieszczania się po schemacie
- łączy inne bloki występujące na schemacie
- linie pochodzące z różnych części algorytmu mogą zbiegać się w jednym miejscu

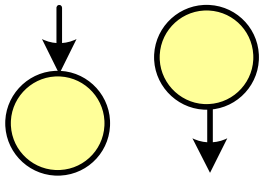


- podprogram
- wywołanie wcześniej zdefiniowanego fragmentu algorytmu (podprogramu)
- ma jedno wejście i jedno wyjście

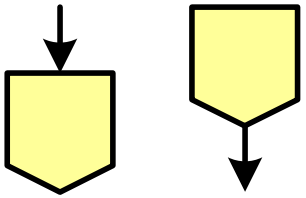
Schemat blokowy - symbole graficzne



- komentarz
- dodanie do schematu dodatkowego opisu



- łącznik stronicowy (wewnętrzny)
- połączenie dwóch odrębnych części schematu znajdujących się na tej samej stronie
- łączniki opisywane są etykietami



- łącznik międzystronicowy (zewnętrzny)
- połączenie dwóch odrębnych części schematu znajdujących się na różnych stronach
- łączniki opisywane są etykietami

Pseudokod i język programowania

Pseudokod:

- Pseudokod (pseudojęzyk) - uproszczona wersja języka programowania
- Często zawiera zwroty pochodzące z języków programowania
- Zapis w pseudokodzie może być łatwo przetłumaczony na wybrany język programowania

Opis w języku programowania:

- Zapis programu w konkretnym języku programowania
- Stosowane języki: Pascal, C, C++, Matlab, Python
  (kiedyś - Fortran, Basic)

Największy wspólny dzielnik - algorytm Euklidesa

- NWD - największa liczba naturalna dzieląca (bez reszty) dwie (lub więcej) liczby całkowite

NWD(1675,3752) = ?

Algorytm Euklidesa - przykład

a	b	Dzielenie większej liczby przez mniejszą	Zamiana
1675	3752	b/a = 3752/1675 = 2 reszta 402	b = 402
1675	402	a/b = 1675/402 = 4 reszta 67	a = 67
67	402	b/a = 402/67 = 6 reszta 0	b = 0
67	0	KONIEC	

NWD(1675,3752) = 67

Algorytm Euklidesa - lista kroków

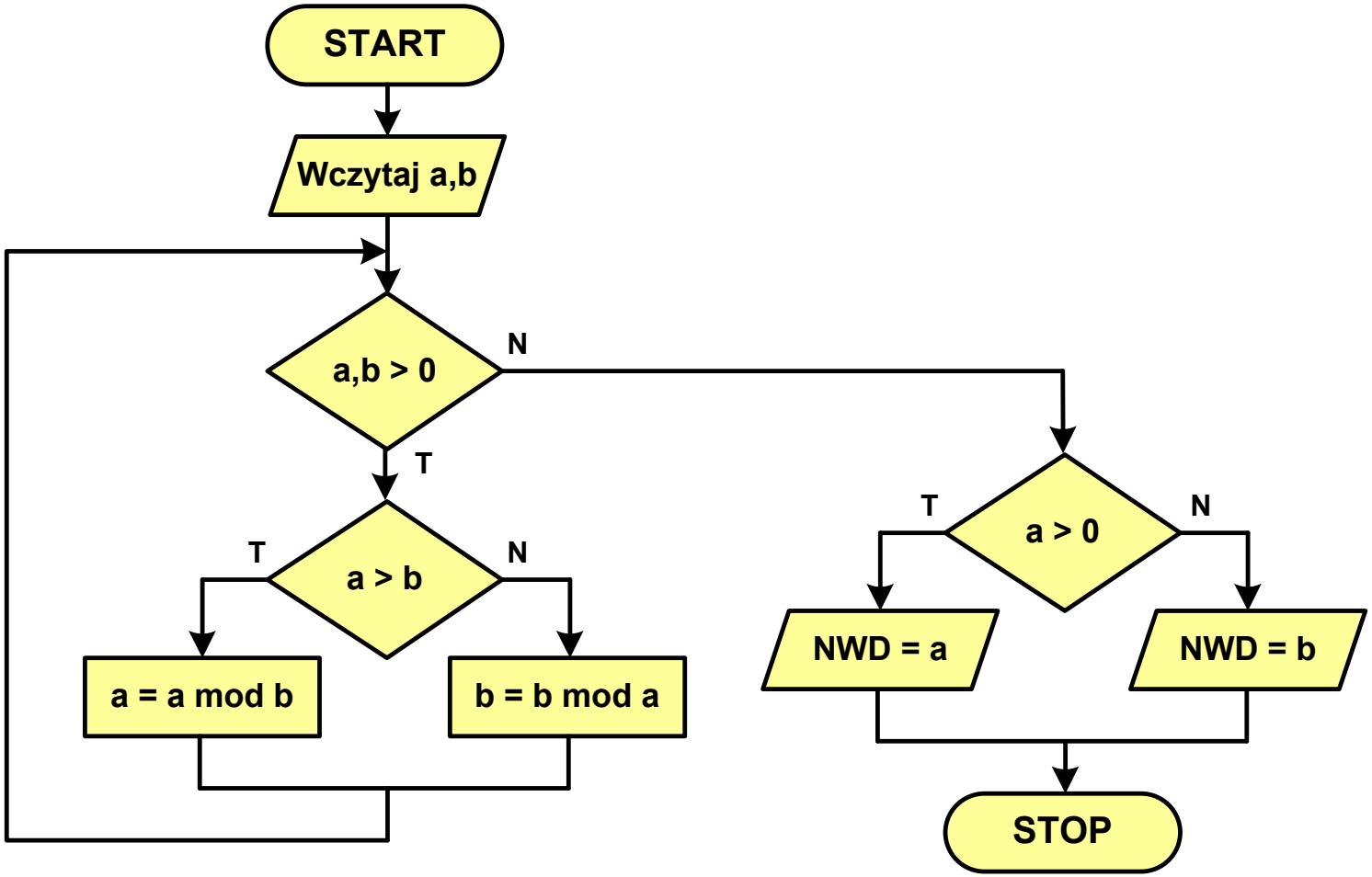
Dane wejściowe: niezerowe liczby naturalne a i b

Dane wyjściowe: NWD(a,b)

Kolejne kroki:

1. Czytaj liczby a i b
2. Dopóki a i b są większe od zera, powtarzaj krok 3, a następnie przejdź do kroku 4
3. Jeśli a jest większe od b, to weź za a resztę z dzielenia a przez b, w przeciwnym razie weź za b resztę z dzielenia b przez a
4. Przyjmij jako największy wspólny dzielnik tę z liczb a i b, która pozostała większa od zera
5. Drukuj NWD(a,b)

Algorytm Euklidesa - schemat blokowy



Algorytm Euklidesa - pseudokod

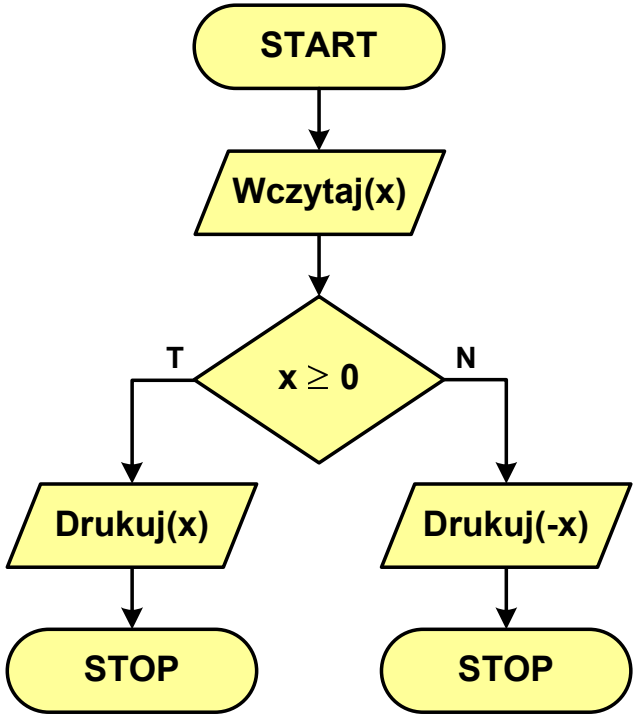
```
NWD(a,b)
while a>0 i b>0
 do if a>b
       then a ← a mod b
       else b ← b mod a
if a>0
   then return a
   else return b
```

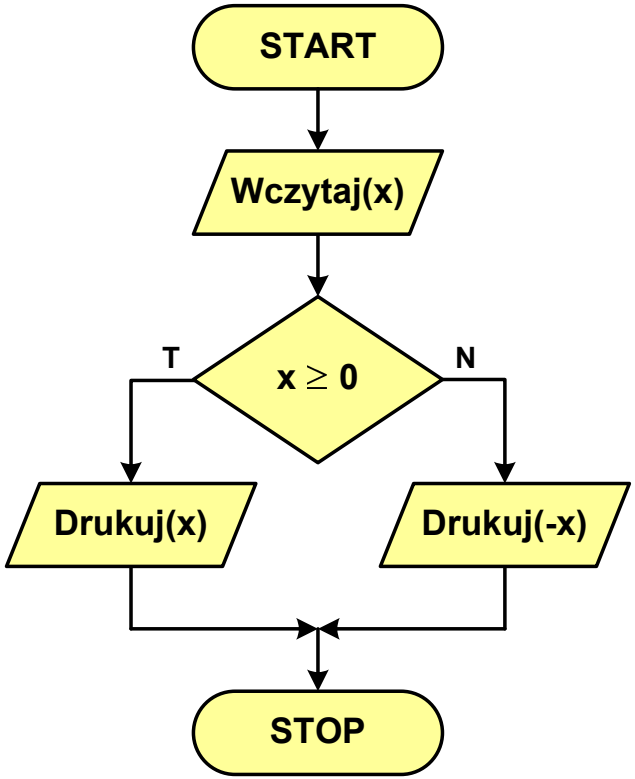
Algorytm Euklidesa - język programowania (C)

```c
#include <stdio.h>

int main(void)
{
    int a = 1675, b = 3752, NWD;

    while (a>0 && b>0)
        if (a>b)
            a = a % b;
        else
            b = b % a;

    if (a>0)
        NWD = a;
    else
        NWD = b;

    printf("NWD = %d\n",NWD);
}
```

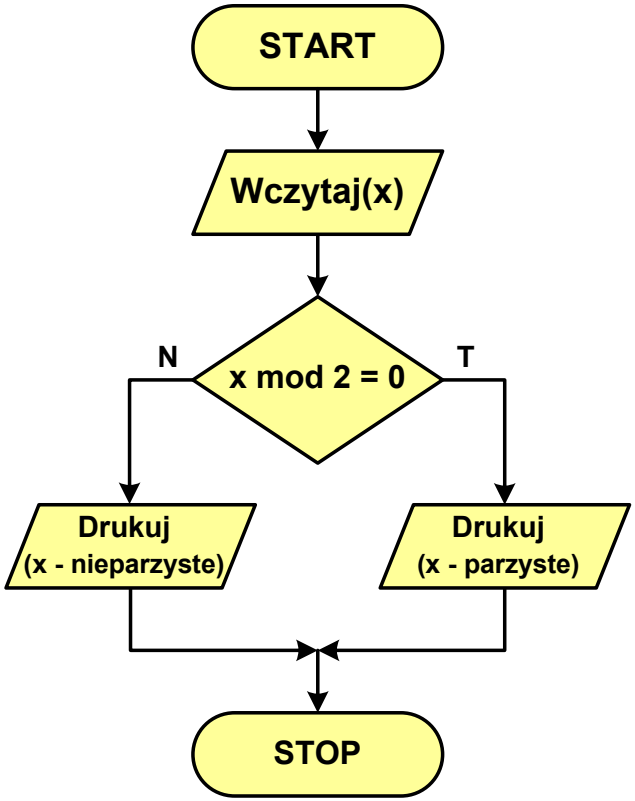
Wartość bezwzględna liczby - schemat blokowy

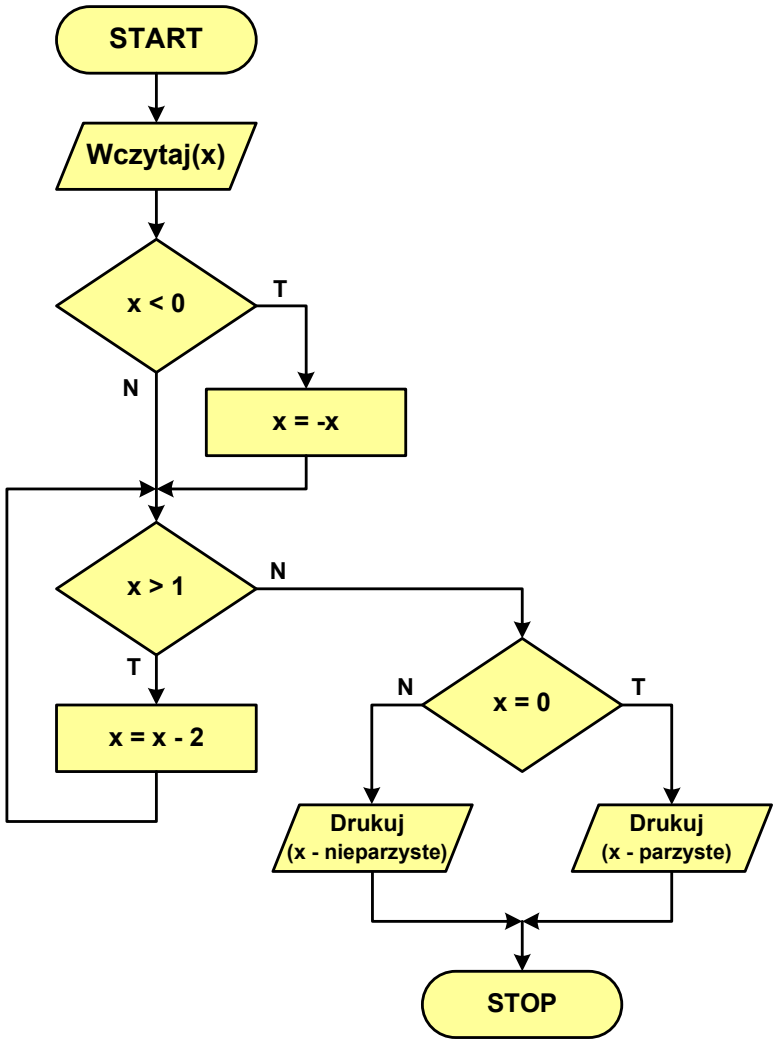
$$|x| = \begin{cases} x \ \ dla \ \ x \geq 0 \\ -x \ \ dla \ \ x < 0 \end{cases}$$





Parzystość liczby - schemat blokowy





Równanie kwadratowe - schemat blokowy

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

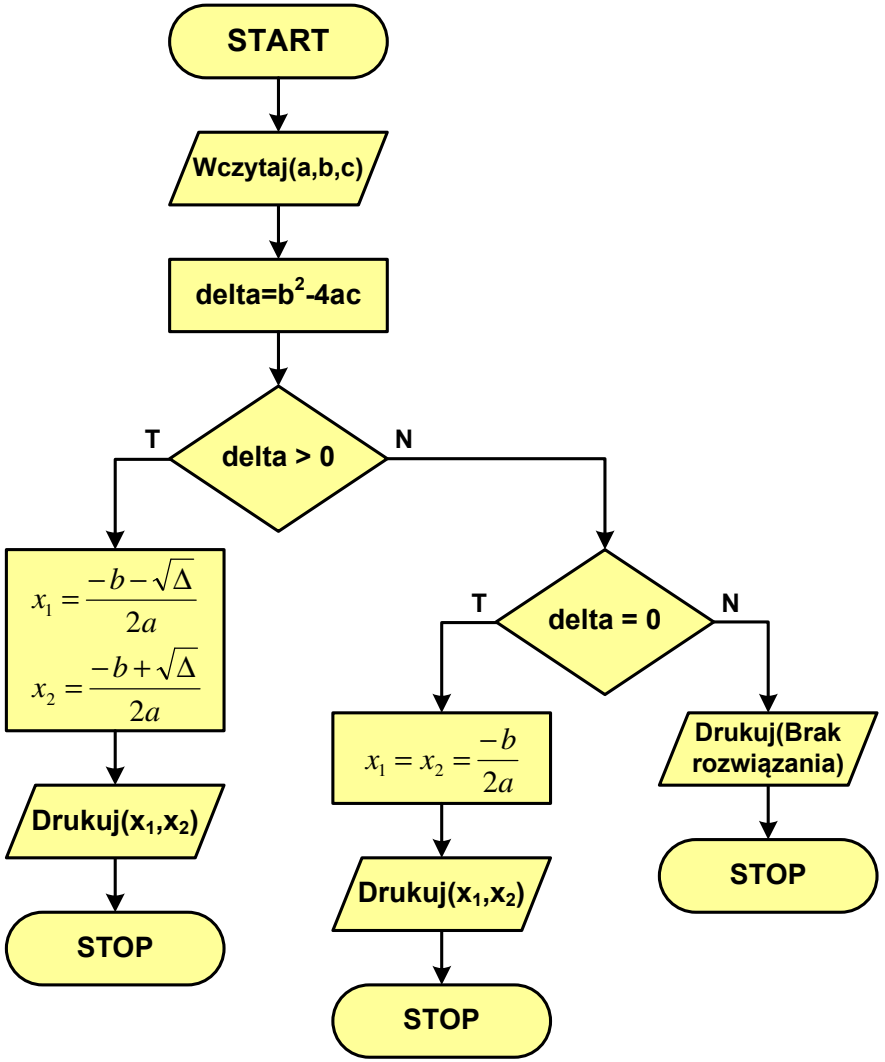
$$\Delta = b^2 - 4ac$$

$\Delta > 0$:

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a}, \quad x_2 = \frac{-b + \sqrt{\Delta}}{2a}$$

$\Delta = 0$:

$$x_1 = x_2 = \frac{-b}{2a}$$



Silnia - schemat blokowy

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

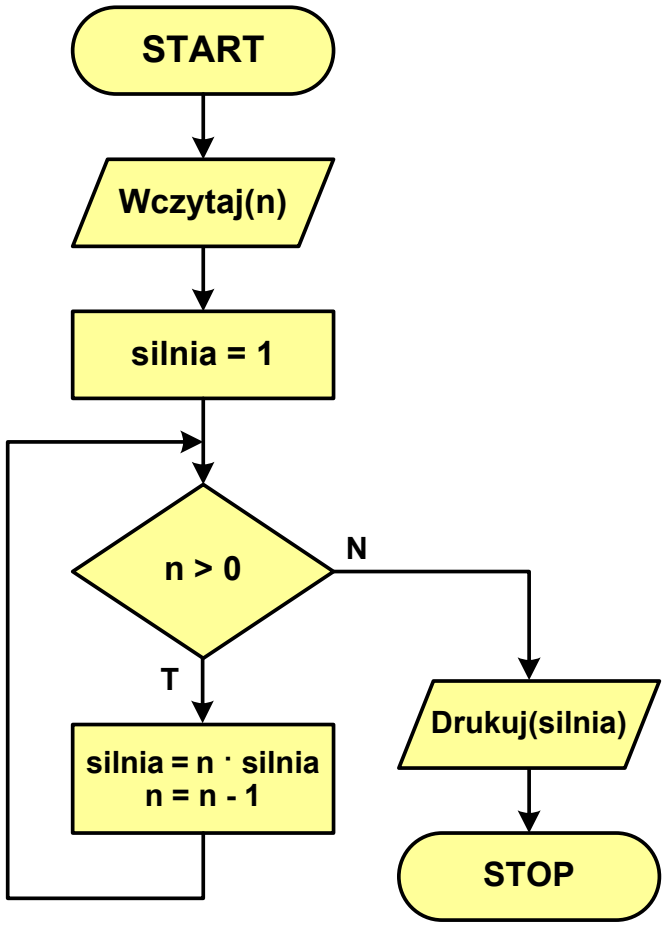
$$0! = 1$$

$$1! = 1$$

$$2! = 1 \cdot 2$$

$$3! = 1 \cdot 2 \cdot 3$$

$$\ldots$$

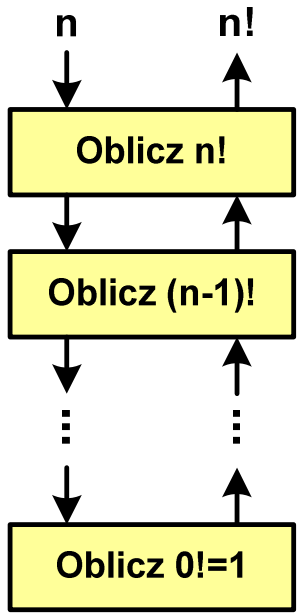


Rekurencja

- Rekurencja lub rekursja - jest to odwoływanie się funkcji lub definicji do samej siebie
- Rozwiązanie danego problemu wyraża się za pomocą rozwiązań tego samego problemu, ale dla danych o mniejszych rozmiarach
- W matematyce mechanizm rekurencji stosowany jest do definiowania lub opisywania algorytmów
- Silnia:

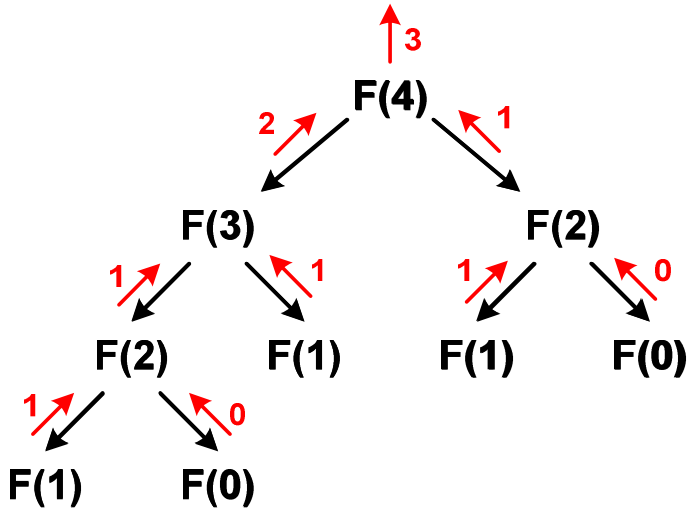
$$n! = \begin{cases} 1 & dla \quad n = 0 \\ n(n-1)! & dla \quad n \geq 1 \end{cases}$$

```
int silnia(int n)
{
    return n==0 ? 1 : n*silnia(n-1);
}
```



Rekurencja - ciąg Fibonacciego

$$F_n = \begin{cases} 0 & dla \quad n = 0 \\ 1 & dla \quad n = 1 \\ F_{n-1} + F_{n-2} & dla \quad n > 1 \end{cases}$$




```
int F(int n)
{
    if (n==0) return 0;
    else
        if (n==1) return 1;
        else
            return F(n-1) + F(n-2);
}
```

Rekurencja - algorytm Euklidesa

$$NWD(a,b) = \begin{cases} a & dla \quad b = 0 \\ NWD(b, a \bmod b) & dla \quad b \geq 1 \end{cases}$$

```
int NWD(int a, int b)
{
    if (b==0)
        return a;
    else
        return NWD(b,a % b);
}
```

Złożoność obliczeniowa

- W celu rozwiązania danego problemu obliczeniowego szukamy algorytmu najbardziej efektywnego czyli:
  - najszybszego (najkrótszy czas otrzymania wyniku)
  - o możliwie małym zapotrzebowaniu na pamięć
- Problem: Jak ocenić, który z dwóch różnych algorytmów rozwiązujących to samo zadanie jest efektywniejszy?
- Do oceny efektywności służy złożoność obliczeniowa algorytmu (koszt algorytmu)
- Złożoność obliczeniowa algorytmu to ilość zasobów potrzebnych do jego działania (czas, pamięć)

Złożoność obliczeniowa

Złożoność czasowa

- Czas wykonania algorytmu wyrażony w jednostkach czasu, liczbie cykli procesora, liczbie wykonywanych operacji
- Jej miarą jest zazwyczaj liczba podstawowych operacji (dominujących) - pozostałe operacje są pomijane
- Podstawowe operacje - porównanie, podstawienie, operacja arytmetyczna

Złożoność pamięciowa

- Jest miarą wykorzystania pamięci (liczba komórek pamięci)
- Wyrażana jest w liczbie bajtów lub liczbie zmiennych określonego typu

Złożoność obliczeniowa

- Jeśli wykonanie algorytmu zależne jest od zestawu danych wejściowych, to wyróżnia się:

Złożoność optymistyczna

- Odpowiada danym najbardziej sprzyjającym dla algorytmu

Złożoność średnia

- Złożoność uśredniona po wszystkich możliwych zestawach danych, występująca dla „typowych” (losowych) danych wejściowych

Złożoność pesymistyczna

- Odpowiada danym najbardziej niesprzyjającym dla algorytmu

- Przykład: poszukiwanie określonej wartości w N-elementowej tablicy liczb

Złożoność obliczeniowa

- Złożoność obliczeniowa algorytmu jest funkcją rozmiaru danych, na których pracuje ten algorytm
- Złożoność obliczeniowa wyznaczana jest poprzez zliczanie operacji
- W praktyce stosuje się oszacowanie powyższej funkcji - są to tzw. notacje (klasy złożoności):
  - O (duże O)
  - Ω (omega)
  - Θ (theta)

Notacja O („duże O”)

- Wyraża złożoność matematyczną algorytmu
- Po literze O występuje wyrażenie w nawiasach zawierające literę n, która oznacza liczbę elementów, na której działa algorytm
- Do wyznaczenia złożoności bierze się pod uwagę liczbę wykonywanych w algorytmie podstawowych operacji

Przykład:

O(n) - złożoność algorytmu jest prostą funkcją liczby elementów (jeśli sumowanie 10.000 elementów zajmuje 5 s, to sumowanie 20.000 elementów zajmie 10 s)

$O(n^2)$ - czas konieczny do wykonania algorytmu rośnie wraz z kwadratem liczby elementów (przy podwojeniu liczby elementów ich obsługa będzie trwała cztery razy dłużej)

Notacja O („duże O”)

- Porównanie najczęściej występujących złożoności:

Elementy (n)	O(log n)	O(n)	O(n log n)	$O(n^2)$	$O(n^3)$	$O(2^n)$
10	3	10	33	100	1 000	1024
100	7	100	664	10 000	1 000 000	$1{,}27 \cdot 10^{30}$
1 000	10	1 000	9 966	1 000 000	10^9	$1{,}07 \cdot 10^{301}$
10 000	13	10 000	132 877	10^8	10^{12}	$1{,}99 \cdot 10^{3010}$

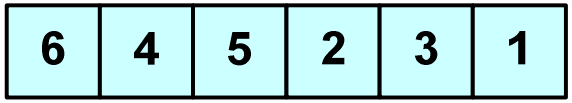
O(log n) - logarytmiczna (np. przeszukiwanie binarne)
O(n) - liniowa (np. porównywanie łańcuchów znaków)
O(n log n) - liniowo-logarytmiczna (np. sortowanie szybkie)
$O(n^2)$ - kwadratowa (np. proste algorytmy sortowania)
$O(n^3)$ - sześcienna (np. mnożenie macierzy)
$O(2^n)$ - wykładnicza (np. problem komiwojażera)

Sortowanie

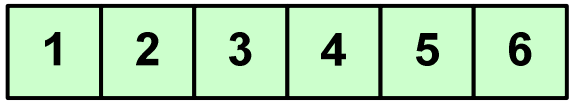
- Sortowanie polega na uporządkowaniu zbioru danych względem pewnych cech charakterystycznych każdego elementu tego zbioru (wartości każdego elementu)
- W przypadku liczb, sortowanie polega na znalezieniu kolejności liczb zgodnej z relacją $\leq$ lub $\geq$

Przykład:

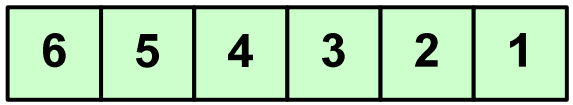
- Tablica nieposortowana:



- Tablica posortowana zgodnie z relacją $\leq$ (od najmniejszej do największej liczby):



- Tablica posortowana zgodnie z relacją $\geq$ (od największej do najmniejszej liczby):



Sortowanie

- W przypadku słów sortowanie polega na ustawieniu ich w porządku alfabetycznym (leksykograficznym)

Przykład:

- Tablica nieposortowana:

Paweł	Piotr	Adrian	Ela	Ola	Henryk

- Tablice posortowane:

Adrian	Ela	Henryk	Ola	Paweł	Piotr

Piotr	Paweł	Ola	Henryk	Ela	Adrian

Sortowanie

- W praktyce sortowanie sprowadza się do porządkowanie danych na podstawie porównania - porównywany element to klucz

Przykład:

- Tablica nieposortowana (imię, nazwisko, wiek):

Piotr	Ola	Paweł	Jan	Ela	Magda
Kowalski	Nowak	Wójcik	Kamiński	Król	Mazur
25	18	23	20	22	15

- Tablica posortowana (klucz - nazwisko):

Jan	Piotr	Ela	Magda	Ola	Paweł
Kamiński	Kowalski	Król	Mazur	Nowak	Wójcik
20	25	22	15	18	23

- Tablica posortowana (klucz - wiek):

Magda	Ola	Jan	Ela	Paweł	Piotr
Mazur	Nowak	Kamiński	Król	Wójcik	Kowalski
15	18	20	22	23	25

Sortowanie

Po co stosować sortowanie?

- Posortowane elementy można szybciej zlokalizować
- Posortowane elementy można przedstawić w czytelniejszy sposób

Klasyfikacje algorytmów sortowania

- Złożoność obliczeniowa algorytmu - zależność liczby wykonywanych operacji od liczebności sortowanego zbioru n
- Złożoność pamięciowa - wielkość zasobów zajmowanych przez algorytm (sortowanie w miejscu - wielkość zbioru danych podczas sortowania nie zmienia się lub jest tylko nieco większa)
- Sortowanie wewnętrzne (odbywa się w pamięci komputera) i zewnętrzne (nie jest możliwe jednoczesne umieszczenie wszystkich elementów zbioru sortowanego w pamięci komputera)

Klasyfikacje algorytmów sortowania

- Algorytm jest stabilny, jeśli podczas sortowania zachowuje kolejność występowania elementów o tym samym kluczu

Przykład:

- Tablica nieposortowana (imię, nazwisko, wiek):

Piotr	Ola	Paweł	Jan	Ela	Magda
Kowalski	Nowak	Wójcik	Kamiński	Król	Mazur
20	18	23	20	18	15

- Tablica posortowana algorytmem stabilnym (klucz - wiek):

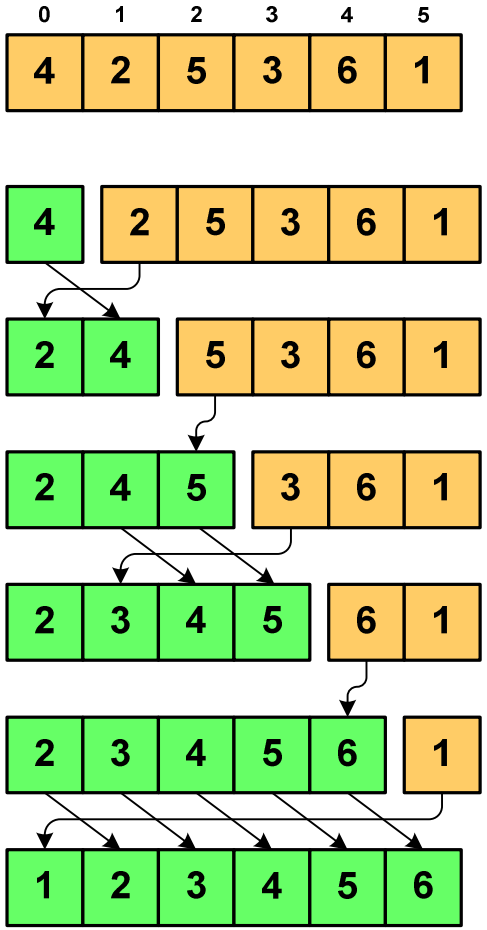
Magda	Ola	Ela	Piotr	Jan	Paweł
Mazur	Nowak	Król	Kowalski	Kamiński	Wójcik
15	18	18	20	20	23

- Tablica posortowana algorytmem niestabilnym (klucz - wiek):

Magda	Ela	Ola	Jan	Piotr	Paweł
Mazur	Król	Nowak	Kamiński	Kowalski	Wójcik
15	18	18	20	20	23

Proste wstawianie (insertion sort)

Przykład:



Program w języku C:

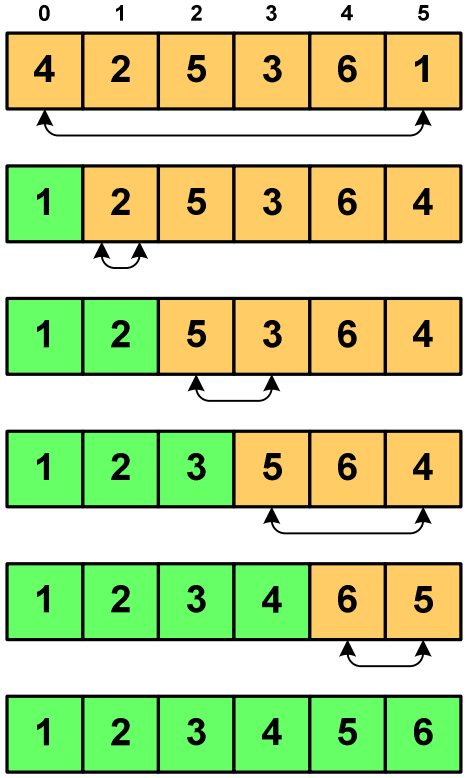
```
int main(void)
{
   int tab[N],i,j,tmp;
   // ...
   for (i=1; i<N; i++)
   {
      j=i;
      tmp=tab[i];
      while (tab[j-1]>tmp && j>0)
      {
         tab[j]=tab[j-1];
         j--;
      }
      tab[j]=tmp;
   }
}
```

Proste wstawianie (insertion sort)

- Złożoność algorytmu: $O(n^2)$
  - \+ wydajny dla danych wstępnie posortowanych
  - \+ wydajny dla zbiorów o niewielkiej liczebności
  - \+ małe zasoby zajmowane podczas pracy (sortowanie w miejscu)
  - \+ stabilny
  - \+ prosty w implementacji
  - – mała efektywność dla normalnej i dużej ilości danych.

Proste wybieranie (selection sort)

Przykład:



Program w języku C:

```
int main(void)
{
   int tab[N],i,j,k,tmp;
   // ...
   for (i=0;i<N-1;i++)
   {
      k=i;
      for (j=i+1; j<N; j++)
         if (tab[k]>=tab[j])
            k = j;
      tmp = tab[i];
      tab[i] = tab[k];
      tab[k] = tmp;
   }
}
```

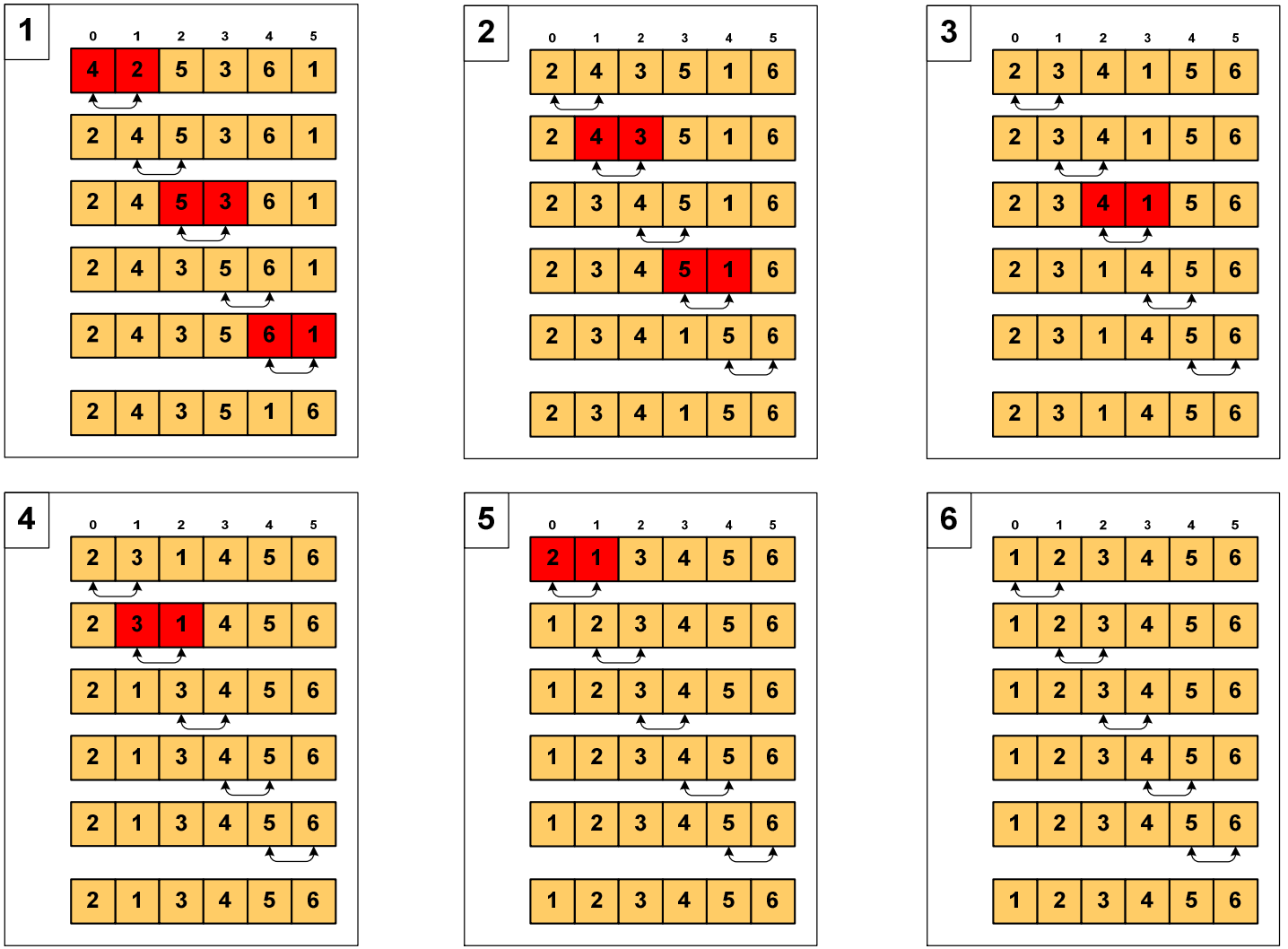
Proste wybieranie (selection sort)

- Złożoność algorytmu: $O(n^2)$
  - \+ szybki w sortowaniu niewielkich tablic
  - \+ małe zasoby zajmowane podczas pracy (sortowanie w miejscu)
  - \+ prosty w implementacji
  - – liczba porównań elementów jest niezależna od początkowego rozmieszczenia elementów w tablicy
  - – w algorytmie może zdarzyć się, że wykonywana jest zamiana tego samego elementu ze sobą.

Bąbelkowe (bubble sort)

- Sortowanie bąbelkowe (ang. bubble sort), nazywane jest także:
  - sortowaniem pęcherzykowym
  - sortowaniem przez prostą zamianę (ang. straight exchange)
- Metoda ta polega na porównywaniu dwóch kolejnych elementów i zamianie ich kolejności jeśli jest to konieczne
- Nazwa metody wzięła się stąd, że kolejne porównania powodują „wypychanie” kolejnego największego elementu na koniec („wypłynięcie największego bąbelka”)

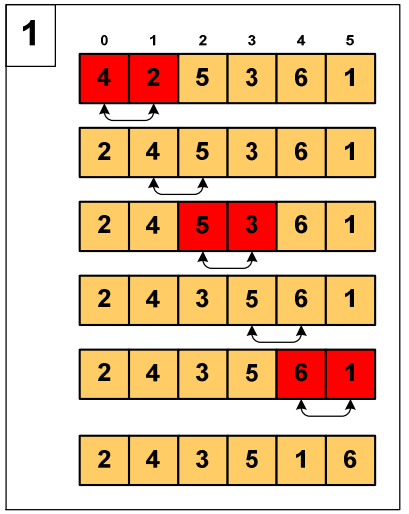
Bąbelkowe (bubble sort)

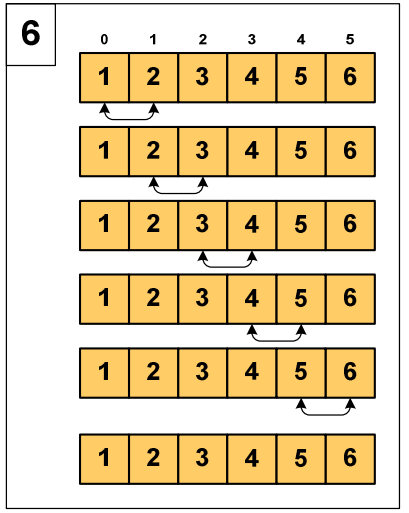


Bąbelkowe (bubble sort)

Program w języku C:

```
int main(void)
{
   int tab[N],i,j,tmp,koniec;
   // ...
   do {
      koniec=1;
      for (i=0;i<N-1;i++)
         if (tab[i]>tab[i+1])
         {
            tmp=tab[i];
            tab[i]=tab[i+1];
            tab[i+1]=tmp;
            koniec=0;
         }
   } while (!koniec);
}
```





Bąbelkowe (bubble sort)

- Złożoność algorytmu: $O(n^2)$
  - \+ prosta realizacja
  - \+ wysoka efektywność użycia pamięci (sortowanie w miejscu)
  - \+ stabilny
  - – mała efektywność.

Sortowanie szybkie (Quick-Sort) - faza dzielenia

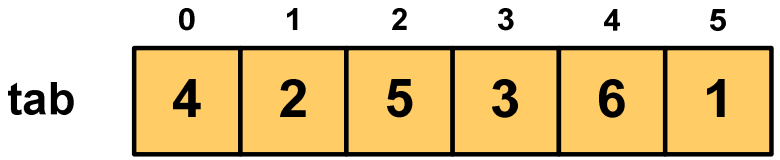
- Tablica jest dzielona na dwie części wokół pewnego elementu x (nazywanego elementem centralnym)
- Jako element centralny x najczęściej wybierany jest element środkowy (choć może to być także element losowy)
- Przeglądamy tablicę od lewej strony, aż znajdziemy element $a_i \geq x$, a następnie przeglądamy tablicę od prawej strony, aż znajdziemy element $a_j \leq x$
- Zamieniamy elementy a_i i a_j miejscami i kontynuujemy proces przeglądania i zamiany, aż nastąpi spotkanie w środku tablicy
- W ten sposób otrzymujemy tablicę podzieloną na lewą część z wartościami mniejszymi lub równymi x i na prawą część z wartościami większymi lub równymi x

Sortowanie szybkie (Quick-Sort) - faza sortowania

- Zawiera dwa rekurencyjne wywołania tej samej funkcji sortowania: dla lewej i dla prawej części posortowanej tablicy
- Rekurencja zatrzymuje się, gdy wielkość tablicy wynosi 1

Przykład:

- Sortujemy 6-elementową tablicę tab:

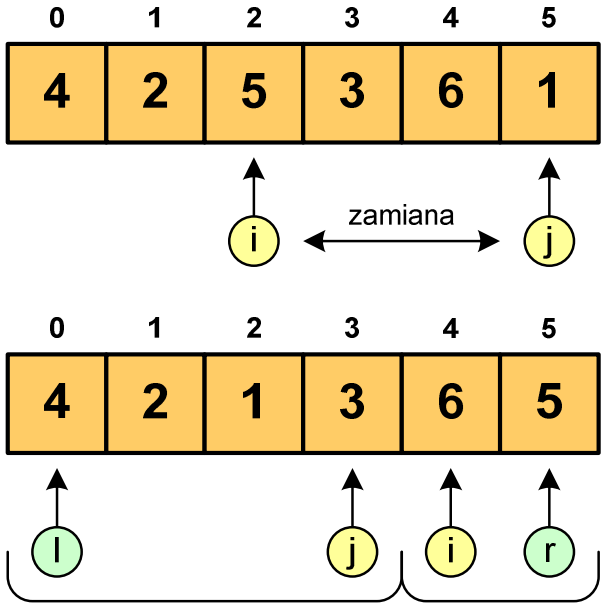


- Wywołanie funkcji QS() ma postać:

```
QS(tab,0,5);
```

Sortowanie szybkie (Quick-Sort) - QS(tab,0,5)

- Element środkowy: (0+5)/2 = 2, x = tab[2] = 5
- Od lewej szukamy tab[i] ≥ x,
  a od prawej szukamy tab[j] ≤ x,
  zamieniamy elementy miejscami
- Poszukiwania kończymy,
  gdy indeksy i, j mijają się

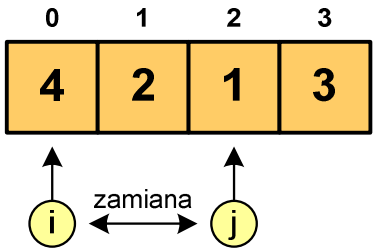


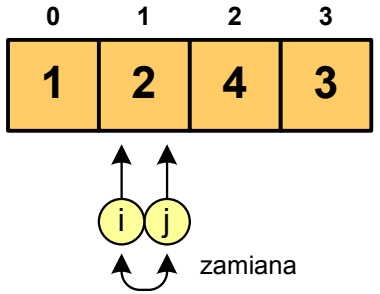
- Wywołujemy rekurencyjnie funkcję QS() dla elementów
  z zakresów [l,j] i [i,r]:

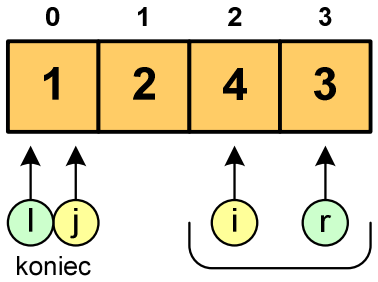
```
QS(tab,0,3);      QS(tab,4,5);
```

Sortowanie szybkie (Quick-Sort) - QS(tab,0,3)

- Element środkowy: (0+3)/2 = 1, x = tab[1] = 2
- Od lewej szukamy tab[i] ≥ x,
  a od prawej szukamy tab[j] ≤ x,
  zamieniamy elementy miejscami
- Poszukiwania kończymy,
  gdy indeksy i, j mijają się





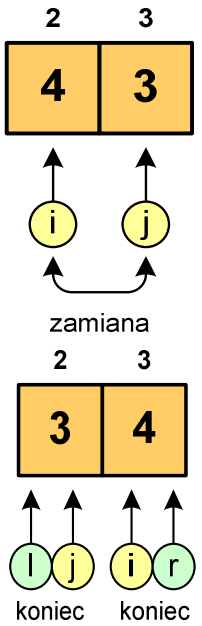


- Wywołanie QS() tylko dla elementów z zakresu [2,3], gdyż po lewej stronie rozmiar tablicy do posortowania wynosi 1:

```
QS(tab,2,3);
```

Sortowanie szybkie (Quick-Sort) - QS(tab,2,3)

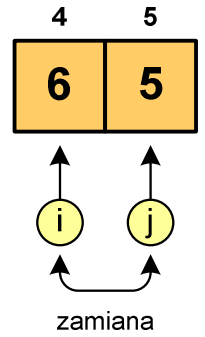
- Element środkowy: (2+3)/2 = 2, x = tab[2] = 4
- Od lewej szukamy tab[i] ≥ x,
  a od prawej szukamy tab[j] ≤ x,
  zamieniamy elementy miejscami
- Poszukiwania kończymy,
  gdy indeksy i, j mijają się



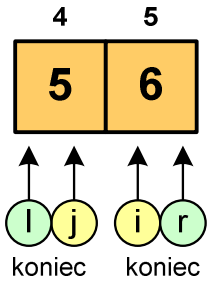
- Rozmiar obu tablic do posortowania wynosi 1 więc nie ma nowych wywołań funkcji QS()

Sortowanie szybkie (Quick-Sort) - QS(tab,4,5)

- Element środkowy: (4+5)/2 = 4, x = tab[4] = 6
- Od lewej szukamy tab[i] ≥ x,
  a od prawej szukamy tab[j] ≤ x,
  zamieniamy elementy miejscami



- Poszukiwania kończymy,
  gdy indeksy i, j mijają się



- Rozmiar obu tablic do posortowania wynosi 1 więc nie ma nowych wywołań funkcji QS()

Sortowanie szybkie (Quick-Sort)

Funkcja w języku C:

```
void QuickSort(int tab[], int l, int r)
{
   int i,j,x,y;

   i=l;
   j=r;
   x=tab[(l+r)/2];
   do
   {
      while (tab[i]<x) i++;
      while (x<tab[j]) j--;
      if (i<=j)
      {
         y=tab[i];
         tab[i]=tab[j];
         tab[j]=y;
         i++; j--;
      }
   } while (i<=j);
   if (l<j) QuickSort(tab,l,j);
   if (i<r) QuickSort(tab,i,r);
}
```

Koniec wykładu nr 9

Dziękuję za uwagę!

(następny wykład: 15.06.2018)

Zaliczenie nr 2 (EK2, EK3)!