Algorytmy

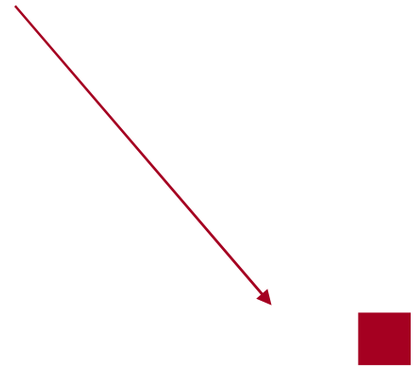
Instrukcje iteracyjne i tablice

Wprowadzenie

Mirosław Ochodek
Miroslaw.Ochodek@pwsz.pila.pl
Miroslaw.Ochodek@cs.put.poznan.pl

Instrukcja

- **Wyrażenie zakończone znakiem ;**



```
;

int a =3;

System.out.println(”Napis”);

if ( z < 11){

        z ++;

}
```

Instrukcja złożona

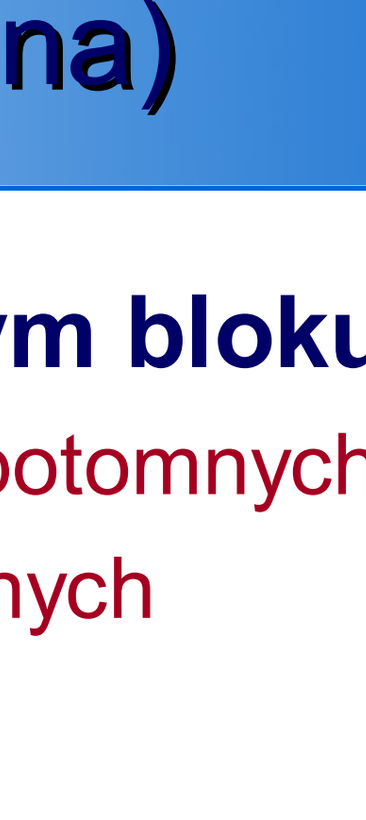
- Instrukcja złożona jest sekwencją instrukcji ograniczoną znakami '}' o '{'
- Może być użyta w instrukcjach warunkowych i pętlach w miejsce pojedynczej instrukcji

```
if (wiek < 13) {
    szkola = "Podstawówka";
    System.out.println(szkola);
} else {
    szkoła = "Inna";
    System.out.println(szkola);
}
```

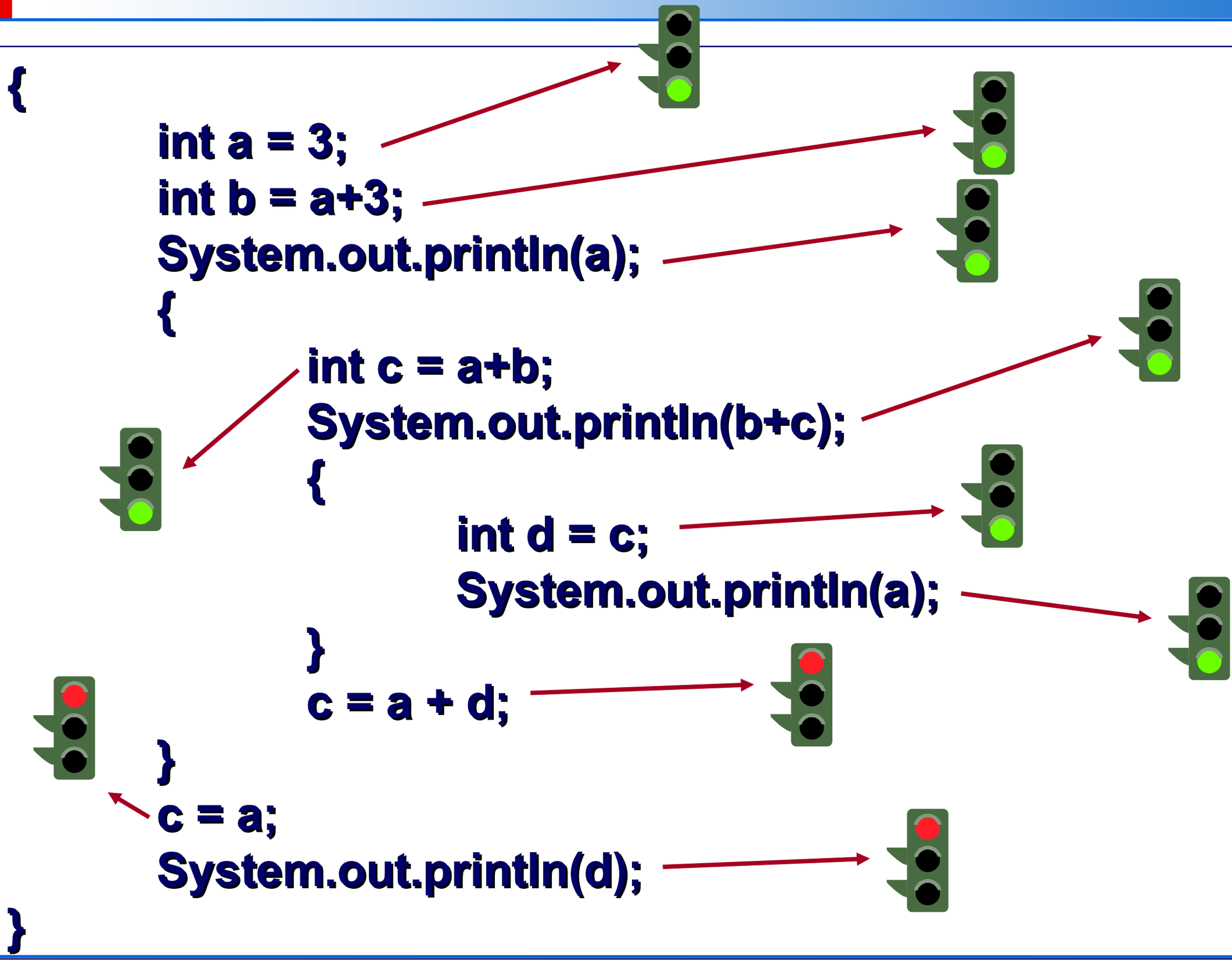
Blok instrukcji (instrukcja złożona)

- **Zmienne zadeklarowane w danym bloku { }**
  - Są widoczne w tym bloku i blokach potomnych
  - Nie są widoczne w blokach nadrzędnych

```
{
        int a = 34;
        System.out.println(a);
        {
                int b = 32;
                System.out.println(a);
                System.out.println(b);
        }
        System.out.println(b);
}
```



Blok instrukcji (instrukcja złożona)

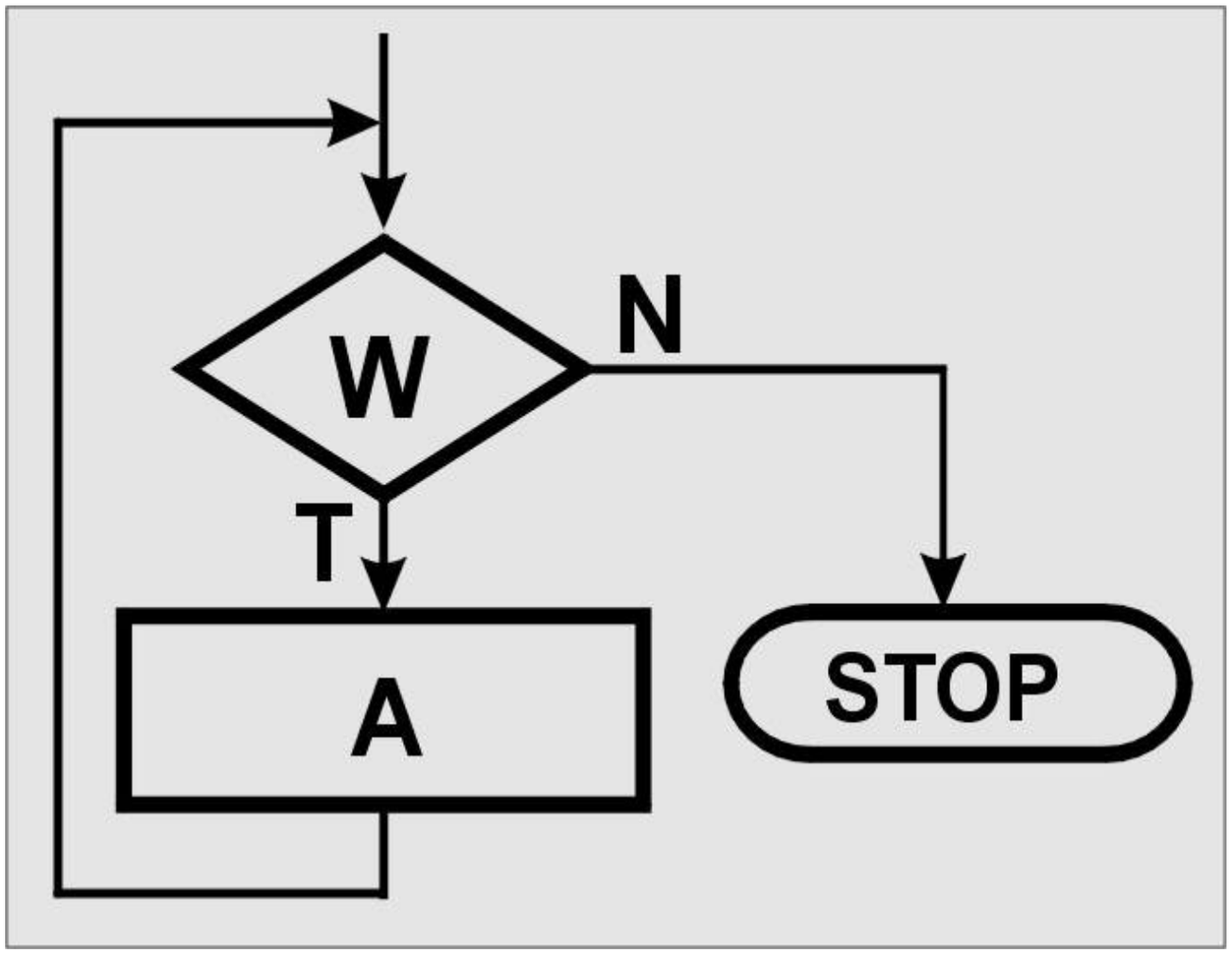


Zakres widoczności przykład

- Spróbujcie napisać program:

```
String imie = ”Jan”;
{
   String nazwisko = ”Kowalski”;
}

System.out.println(imie+nazwisko);
```

Jaki to algorytm?



- Sprawdź W
  - Tak – wykonaj A
  - Nie – koniec
- …

- Dopóki warunek W wykonuj A

Instrukcja while

- **Instrukcja *while* pozwala wykonywać określone polecenia dopóki spełniony jest warunek kontynuacji sprawdzany na początku pętli**

warunek kontynuacji

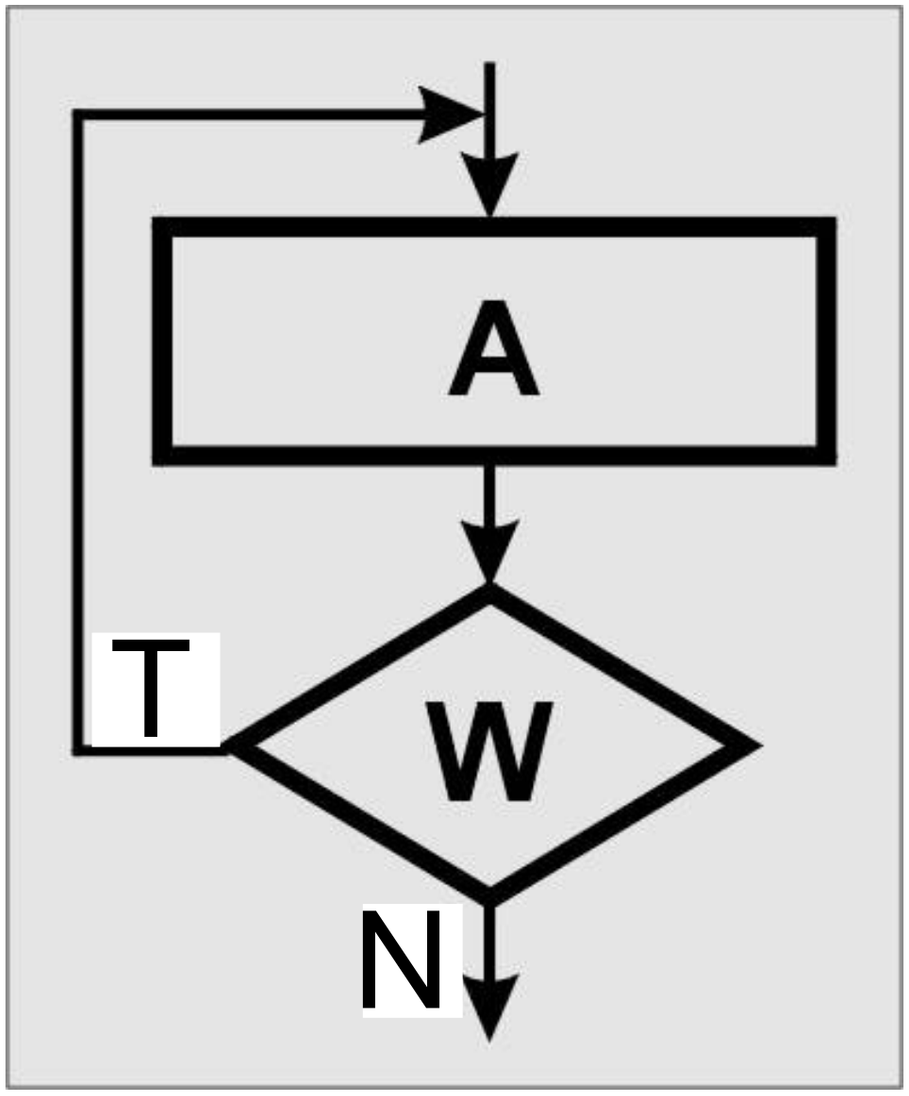
```
while (licznik < 25){
System.out.println("Licznik=" + licznik++);

}
```

Ćwiczenie

- Wejście
  - n – liczba całkowita
- Wyjście
  - Wyświetl wszystkie liczby całkowite od 0 do n

- Zrealizuj przy pomocy pętli while

Jaki to algorytm?



- Wykonaj A
- Sprawdź W
  - Tak – wykonaj A
  - Nie – koniec
- …
- Wykonuj A dopóki warunek W

Instrukcja do..while

- Instrukcja *do..while* pozwala wykonywać określone polecenia dopóki spełniony jest warunek kontynuacji sprawdzany na końcu pętli
- *do..while* różni się od *while* tym, że zostanie wykonana przynajmniej raz

```
do {
    System.out.println("Licznik=" + licznik++);
} while (licznik < 25);
```

warunek kontynuacji

Ćwiczenie

- Wejście
  - n – liczba całkowita
- Wyjście
  - Wyświetl wszystkie liczby całkowite od 0 do n

- Zrealizuj przy pomocy pętli do…while

Instrukcja for

- Instrukcja *for* pozwala wykonywać określone polecenia w pętli, wykorzystując zmienne sterujące
- Pętla *for* pozwala na realizację wielu typów pętli

instrukcja początkowa pętli

warunek kontynuacji

```
for (int licznik = 0; licznik < 10; licznik++) {
    System.out.println("Licznik=" + licznik);

}
```

instrukcja postępu pętli

Ćwiczenie

- Wejście
  - n – liczba całkowita
- Wyjście
  - Wyświetl wszystkie liczby całkowite od 0 do n

- Zrealizuj przy pomocy pętli for

Ćwiczenie

- Wejście
  - n – liczba całkowita
- Wyjście
  - Wyświetl wszystkie liczby całkowite od n do 0

- Zrealizuj przy pomocy pętli for

Ćwiczenie

- Wejście
  - a – liczba całkowita
  - b – liczba całkowita
- Wyjście
  - Wszystkie liczby parzyste z przedziału <a, b>

Ćwiczenie

- Wejście
  - N – liczba całkowita określająca długość ciągu
  - Ciąg liczb (wczytywanych kolejno)
- Wyjście
  - Średnia

Ćwiczenie

- Wejście
  - N – liczba całkowita określająca długość ciągu
  - Ciąg liczb (wczytywanych kolejno)
- Wyjście
  - Średnia
  - **Maksymalna wartość podana**
  - **Minimalna wartość podana**

Ćwiczenie

- Wejście
  - n – liczba całkowita (długość ciągu)
  - div – liczba całkowita
  - Ciąg liczb (wczytywanych kolejno)
- Wyjście
  - Wszystkie liczby z podanego ciągu, których kwadrat podzielny jest przez **div**

Ćwiczenie

- Wejście
  - h – liczba całkowita (wysokość trójkąta)
- Wyjście
  - Trójkąt prostokątny o wysokości h złożony ze znaków *
- **Przykład**
  - h=3

```
*

**

***
```

Ćwiczenie

- Wejście
  - n – liczba całkowita
- Wyjście
  - Wartość funkcji silnia n!

$$\cdot ! : \mathbb{N} \cup \{0\} \to \mathbb{N}$$

$$n! = \prod_{k=1}^{n} k \qquad \text{dla } n \geq 1.$$

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

Ćwiczenie

- Wejście
  - n – liczba całkowita
- Wyjście
  - Wartość n-tego wyrazu ciągu Fibonacciego

$$F_n := \begin{cases} 0 & \text{dla } n = 0; \\ 1 & \text{dla } n = 1; \\ F_{n-1} + F_{n-2} & \text{dla } n > 1. \end{cases}$$

Ćwiczenie

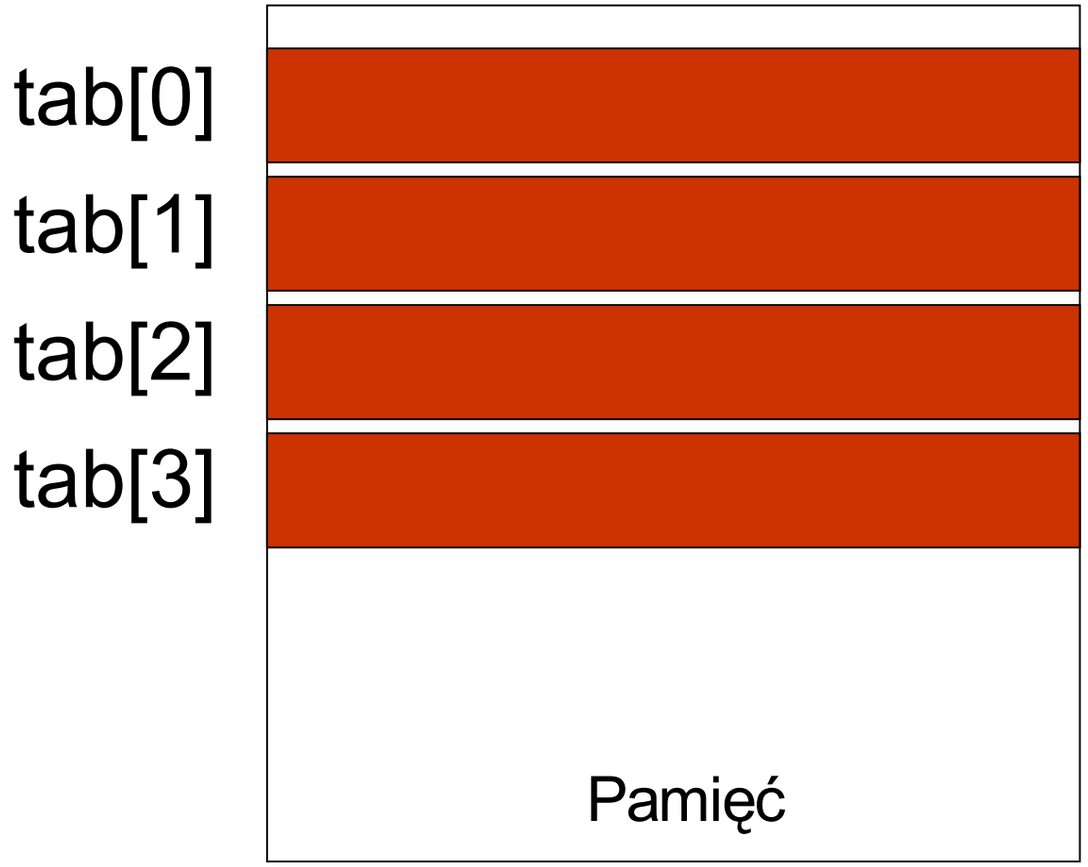
- Wejście
  - N – liczba całkowita określająca długość ciągu
  - Ciąg liczb (wczytywanych kolejno)
  - a – liczba całkowita
  - b – liczba całkowita
- Wyjście
  - Wszystkie liczby z ciągu wejściowego podzielne przez a i przez b
  - Suma i średnia liczba podzielnych przez a i b

Problem

- Napisać program, który:
  - Wejście – ciąg liczb (wczytany raz)
  - Wyjście – wypisanie liczb:
    - W porządku takim jak były podane
    - W porządku odwrotnym
- Jakie są wnioski?
- Chcemy przechowywać wiele zmiennych tego samego typu
- Nie wiemy w trakcie pisania ile takich zmiennych będzie nam potrzeba

Tablica

- Ciągły obszar w pamięci
- Przeznaczona na zmienne danego typu
- Rozmiar deklarowany podczas tworzenia
- Możliwość odwoływania do jej konkretnych komórek za pomocą operatora []



Deklaracje tablic

Deklaracja tablicy ma postać:

typ_tablicy[ ] nazwa_tablicy;

lub alternatywnie:

typ_tablicy nazwa_tablicy[ ];

Tworzenie tablic

Sama tablica tworzona jest za pomocą operatora **new** o postaci:

new typ_tablicy[liczba_elementów];

Możliwe jest także jednoczesne zadeklarowanie i utworzenie tablicy, stosując konstrukcję:

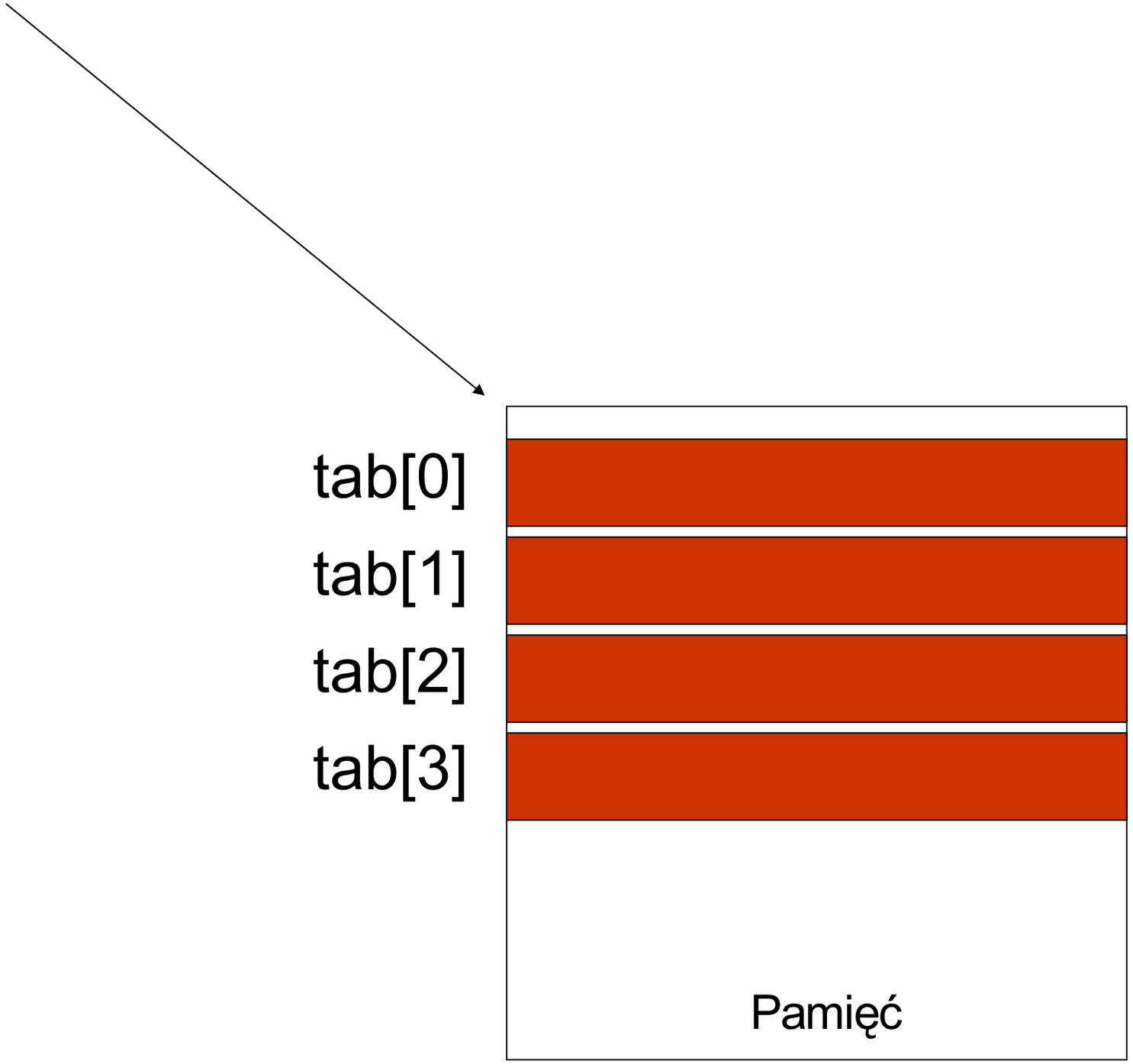
typ_tablicy[ ] nazwa_tablicy = new typ_tablicy [liczba_elementów];

lub

typ_tablicy nazwa_tablicy[ ] = new typ_tablicy [liczba_elementów];

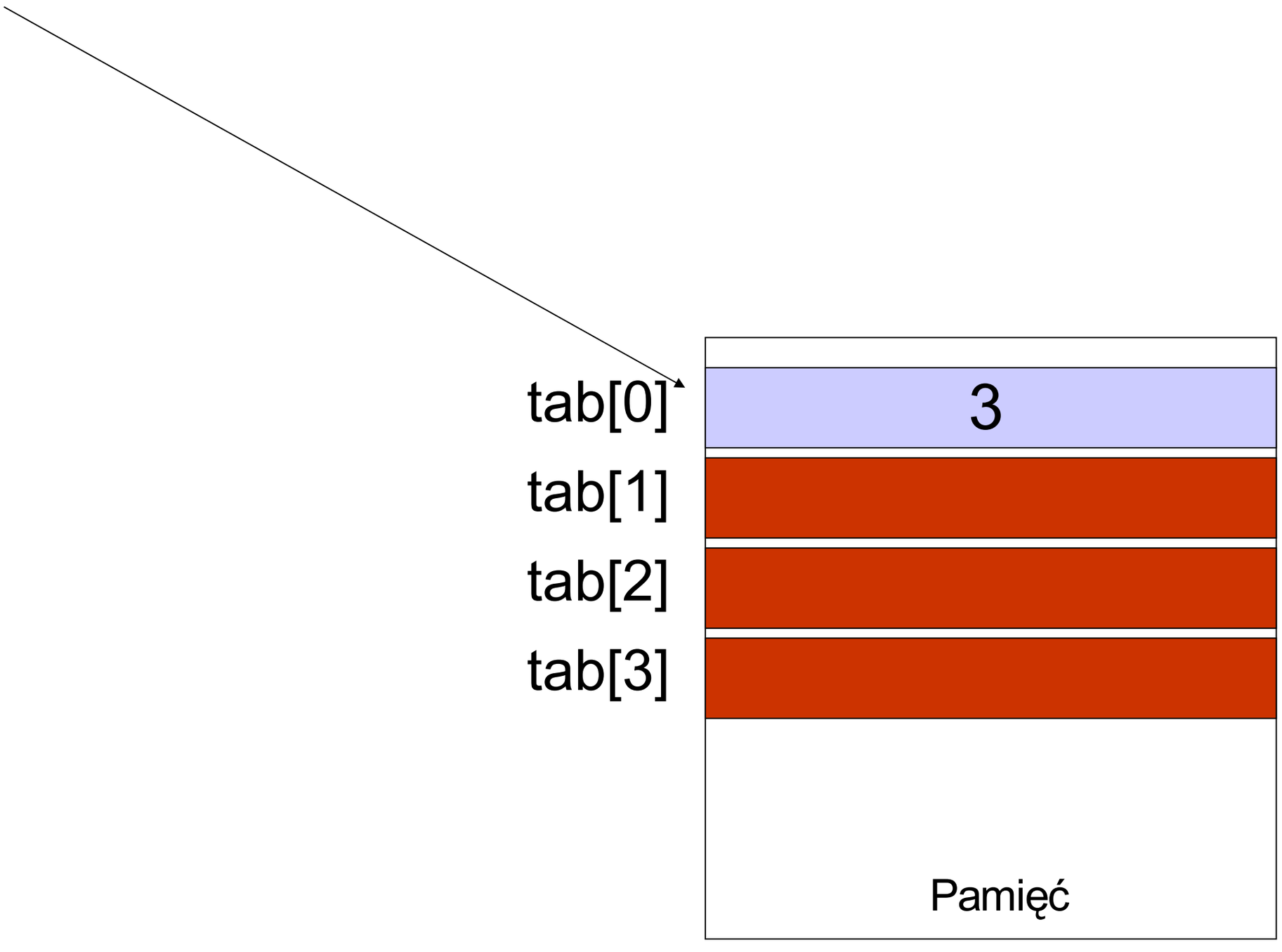
Tablica

```
int[] tab = new int[4];
```



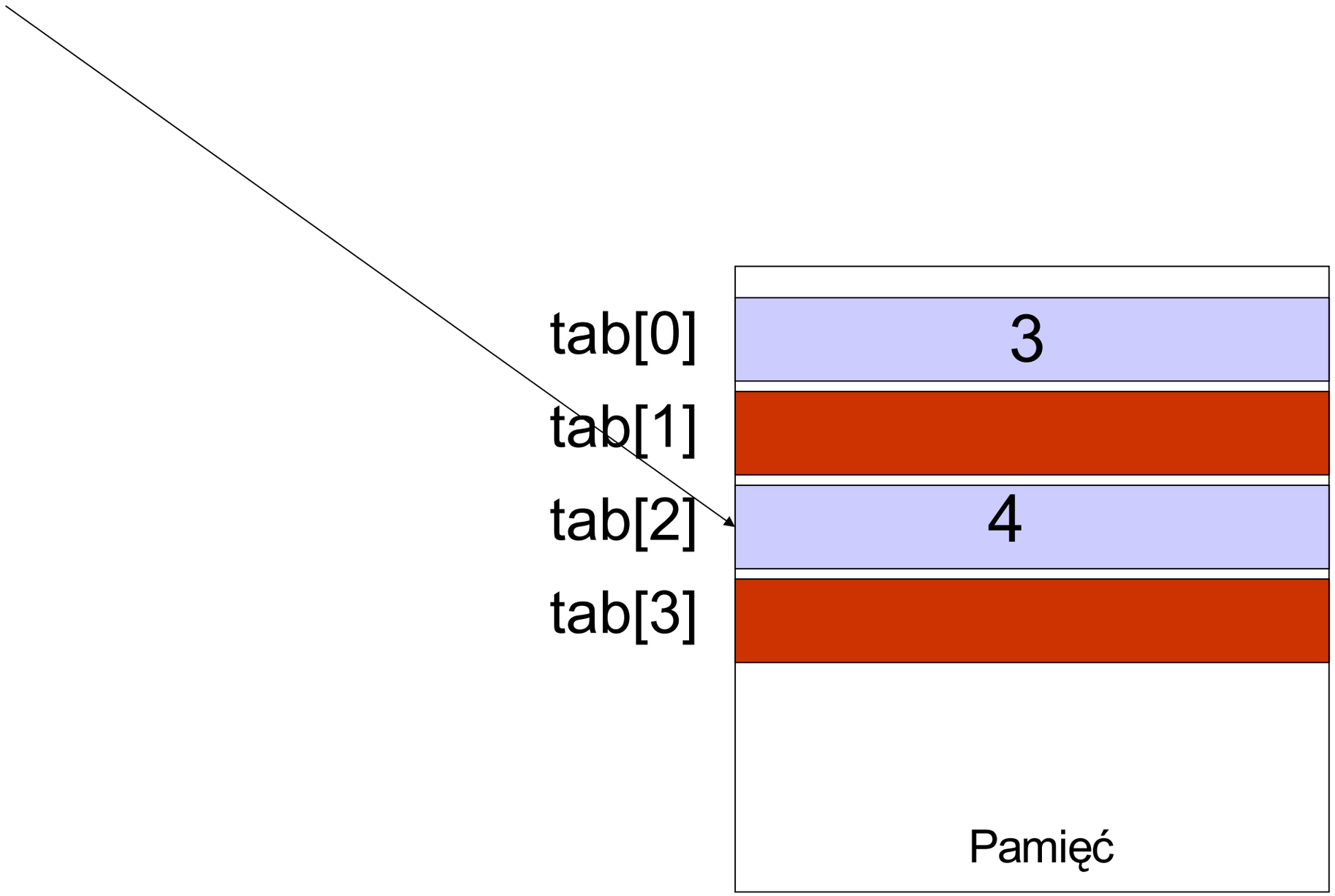
Tablica

```
int[] tab = new int[4];
tab[0] = 3;
```



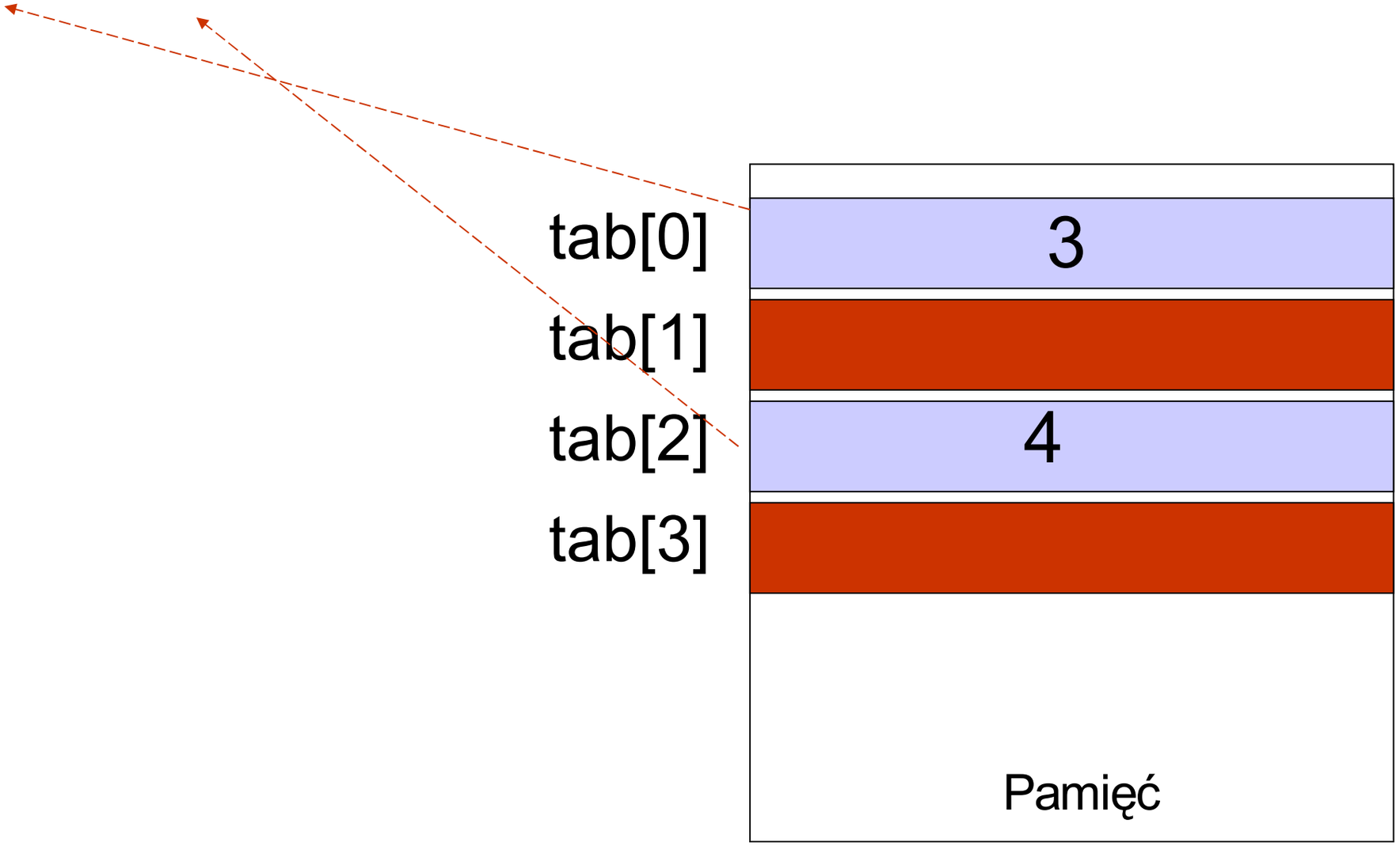
Tablica

```
int[] tab = new int[4];
tab[0] = 3;
tab[2] = 4;
```



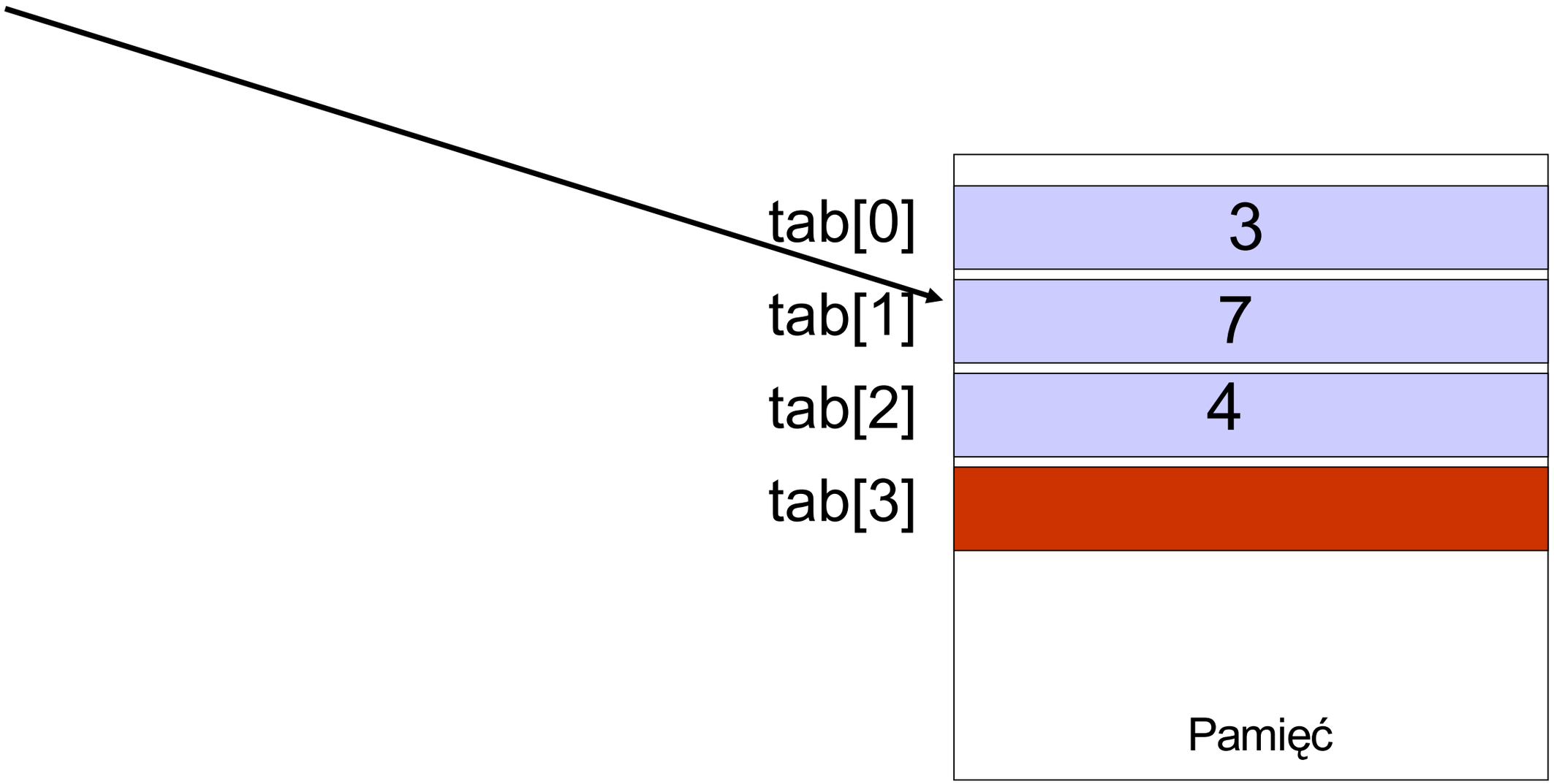
Tablica

```
int[] tab = new int[4];
tab[0] = 3;
tab[2] = 4;
tab[1] = tab[0] + tab[2];
```



Tablica

```
int[] tab = new int[4];
tab[0] = 3;
tab[2] = 4;
tab[1] = tab[0] + tab[2];
```



Przykład

```
class Main
{
    public static void main (String args[ ])
    {
        int tab[ ] = new int[1];
        tab[0] = 10;
        System.out.println(„Pierwszy element

        tablicy ma wartość: ” + tab[0]);
    }
}
```

Ćwiczenie

- Zadeklaruj zmienną tablicową przechowującą wartości typu int o wielkości 6
- Przypisz kolejnym komórkom tablicy kolejne wartości -> 0, 1, 2, 3, 4, 5

- Bez wykorzystania pętli

Inicjacja tablicy

Niewielkie tablice można zainicjować, ujmując wartości, które mają się znaleźć w jej komórkach, w nawiasy klamrowe.
Nie trzeba wówczas korzystać z operatora **new**.

typ_tablicy nazwa_tablicy[ ] = {wartość_1, wartość_2, ... , wartość_n}

Na przykład: deklarację sześcioelementowej tablicy liczb całkowitych **int** i wypełnienie kolejnych komórek wartościami od 1 do 6 można wykonać przy pomocy instrukcji:

```
int tablica[ ] = {1, 2, 3, 4, 5, 6}
```

Ćwiczenie

- Stwórz tablicę wartości typu String, która będzie miała na początku następujące wartości -> imie, nazwisko, kolor oczu

- Wykorzystaj operator { }

Właściwość length

Każda tablica posiada pole length, która określa liczbę jej komórek.

Przykład:

```
int tablica[ ] = new int[10];
for (int i = 0; i < tablica.length; i++){
    tablica[i];
}
```

Ćwiczenie

- Wejście
  - N – liczba całkowita
- Przetwarzanie:
  - Zadeklaruj zmienną tablicową przechowującą wartości typu int o wielkości N
  - Przypisz kolejnym komórkom tablicy kolejne wartości -> 0, 1, 2, 3, 4, 5 … (konieczna pętla)

Ćwiczenie

- Wejście
  - N – liczba całkowita
  - Ciąg liczb o długości N
- Przetwarzanie:
  - Zadeklaruj zmienną tablicową przechowującą wartości typu int o wielkości N
  - Wczytaj wartości do tablicy
- Wyjście
  - Wypisz liczby w kolejności podawania
  - Wypisz liczby od końca

Ćwiczenie

- Wejście
  - N – liczba wartości
  - Kolejne wartości (w liczbie N)
- Przetwarzanie:
  - Zadeklaruj zmienną tablicową przechowującą wartości typu int o wielkości N
  - Wczytaj wartości do tablicy
- Wyjście
  - Średnia $\frac{a_1 + a_2 + \cdots + a_n}{n}$.
  - Odchylenie standardowe $\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}}$

Tablice wielowymiarowe

Deklaracja

regularnych tablic wielowymiarowych

odbywa się analogicznie jak

tablic jednowymiarowych,

dodawane są jedynie kolejne

pary nawiasów klamrowych:

typ_tablicy[ ] [ ] nazwa_tablicy

Inicjowanie tablicy wielowymiarowej

```
typ_tablicy nazwa_tablicy[ ] [ ] =
{
    {wartość_1, wartość_2, ... , wartość_n},
    {wartość_1, wartość_2, ... , wartość_n}
}
```

Przykład dwuwymiarowej tablicy liczb całkowitych, wypełnionej danymi:

```
int tab[ ] [ ] =
{
  {1, 2, 3, 4},
  {5, 6, 7, 8}
}
```

Tworzenie tablicy wielowymiarowej poprzez op. new

new typ_tablicy[liczba_elementów] [liczba elementów];

Przykład:

```
int tab[ ] [ ] = new int[2] [4];
int count = 1;
for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 4; j++)
    {
        tab[i] [j] = count++;
    }
}
```

Ćwiczenie

- Wejście
  - N – liczba wierszy
  - M – liczba kolumn
- Przetwarzanie
  - Wypełnij przekątną wartością 1 i pozostałe komórki wartością 0
- Wyjście
  - Wypisz tablicę na ekran

Ćwiczenie

- Wejście
  - N – liczba wierszy
  - M – liczba kolumn
- Przetwarzanie
  - Wczytaj zawartość tablicy (intów) od użytkownika pytając o wartości poszczególnych komórek
- Wyjście
  - Wypisz tablicę na ekran

Ćwiczenie

- Wejście
  - N – liczba wierszy
  - M – liczba kolumn
  - S – liczba zmiennoprzecinkowa
- Przetwarzanie
  - Wczytaj zawartość tablicy (double) od użytkownika pytając o wartości poszczególnych komórek
  - Przemnóż powstałą macierz przez wartość S
- Wyjście
  - Wypisz tablicę na ekran

Ćwiczenie

- Wejście:
  - Predefiniowana tablica String'ów 3x3 zapełniona wartościami:
    - X – gracz krzyżyk
    - O – gracz kółko
    - - - pole puste
- Wyjście:
  - Wypisz stan planszy
  - Sprawdź czy któryś z graczy nie wygrał
    - Wygrana w pionie

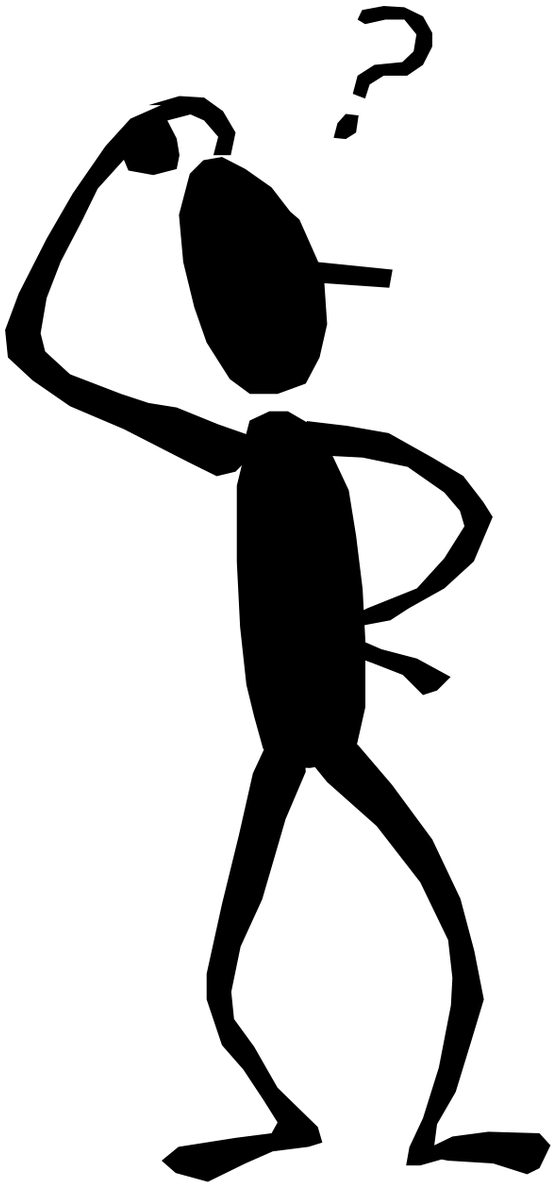
Ćwiczenie

- Wejście:
  - Predefiniowana tablica String'ów 3x3 zapełniona wartościami:
    - X – gracz krzyżyk
    - O – gracz kółko
    - - - pole puste
- Wyjście:
  - Wypisz stan planszy
  - Sprawdź czy któryś z graczy nie wygrał
    - Wygrana w pionie
    - **Wygrana w poziomie**

Ćwiczenie

- Wejście:
  - Predefiniowana tablica String'ów 3x3 zapełniona wartościami:
    - X – gracz krzyżyk
    - O – gracz kółko
    - - - pole puste
- Wyjście:
  - Wypisz stan planszy
  - Sprawdź czy któryś z graczy nie wygrał
    - Wygrana w pionie
    - Wygrana w poziomie
    - **Wygrana po przekątnej**

Koniec



Dziękuje za uwagę

☺