

Informatyka 2 (ES1D300 017)

Politechnika Białostocka - Wydział Elektryczny
Elektrotechnika, semestr III, studia stacjonarne I stopnia
Rok akademicki 2019/2020

Wykład nr 1 (01.10.2019)

dr inż. Jarosław Forenc

Dane podstawowe

- dr inż. Jarosław Forenc
- Politechnika Białostocka, Wydział Elektryczny,
Katedra Elektrotechniki Teoretycznej i Metrologii
ul. Wiejska 45D, 15-351 Białystok
WE-204
- e-mail: j.forenc@pb.edu.pl
- tel. (0-85) 746-93-97
- <http://jforenc.prv.pl>
 - Dydaktyka - slajdy prezentowane na wykładzie
- Konsultacje
 - poniedziałek, godz. 10:00-11:30, WE-204
 - piątek, godz. 10:00-11:30, WE-204

Program wykładu (1/2)

1. Tablice dwu- i wielowymiarowe w języku C. Tablice o zmiennym rozmiarze (VLA).
2. Łańcuchy znaków. Plik nagłówkowy string.h.
3. Struktury w języku C, inicjalizacja zmiennej strukturalnej, odwołania do pól struktury. Pola bitowe i unie.
4. Wskaźniki, operacje na wskaźnikach. Dynamiczny przydział pamięci w języku C. Dynamiczne struktury danych.
5. Funkcje w języku C, ogólna struktura funkcji, deklaracja i definicja funkcji, przekazywanie argumentów do funkcji przez wartość i wskaźnik.
6. Klasy zmiennych i funkcji. Programy wielomodułowe.
7. Operacje wejścia-wyjścia w języku C: znakowe, łańcuchowe, sformatowane, rekordowe.

Program wykładu (2/2)

8. Pliki tekstowe i binarne.
9. Sprawdzian nr 1.
10. System operacyjny. Zarządzanie procesami i dyskowymi operacjami wejścia-wyjścia.
11. Systemy plików (FAT, NTFS, ext).
12. Zarządzanie pamięcią operacyjną.
13. Sieci komputerowe. Topologie i media transmisyjne.
14. Model referencyjny ISO/OSI i model protokołu TCP/IP.
15. Sprawdzian nr 2.

Literatura (1/2)

1. S. Prata: „Język C. Szkoła programowania. Wydanie VI”. Helion, Gliwice, 2016.
2. B.W. Kernighan, D.M. Ritchie: „Język ANSI C. Programowanie. Wydanie II”. Helion, Gliwice, 2010.
3. S.G. Kochan: „Język C. Kompendium wiedzy. Wydanie IV”. Helion, Gliwice, 2015.
4. R. Reese: „Wskaźniki w języku C. Przewodnik”. Helion, Gliwice, 2014.
5. R. Kawa, J. Lembas: „Wykłady z informatyki. Wstęp do informatyki”. PWN, Warszawa 2017.
6. G. Coldwin: „Zrozumieć programowanie”. PWN, Warszawa, 2015.

Literatura (2/2)

7. A.S. Tanenbaum, H. Bos: „Systemy operacyjne. Wydanie IV”. Helion, Gliwice, 2015.
8. W. Stallings: „Systemy operacyjne. Architektura, funkcjonowanie i projektowanie. Wydanie IX”. Helion, Gliwice, 2018.
9. A.S. Tanenbaum, D.J. Wetherall: „Sieci komputerowe. Wydanie V”. Helion, Gliwice, 2012.
10. K. Krysiak: „Sieci komputerowe. Kompendium. Wydanie II”. Helion, Gliwice, 2005.

Efekty kształcenia i system ich oceniania

Podstawę do zaliczenia przedmiotu (uzyskanie punktów ECTS) stanowi stwierdzenie, że każdy z założonych **efektów kształcenia** został osiągnięty w co najmniej minimalnym akceptowalnym stopniu.

EK1	zna w stopniu podstawowym zasady stosowania tablic, struktur, funkcji, plików i wskaźników w programach w języku C
EK2	opisuje podstawowe zadania systemu operacyjnego oraz strukturę sieci komputerowych

Zaliczenie wykładu - efekty kształcenia (EK1)

- Student, który zaliczył przedmiot:

zna w stopniu podstawowym zasady stosowania tablic, struktur, funkcji, plików i wskaźników w programach w języku C

- Student, który zalicza na ocenę **dostateczny (3)**:
 - opisuje sposób deklaruwania i inicjalizacji tablic dwuwymiarowych (macierzy) w języku C oraz metody wykonywania podstawowych operacji na tych tablicach
 - opisuje sposób deklaruwania, inicjalizacji oraz przechowywania łańcuchów znaków (napisów)
 - omawia sposób deklaruwania struktur, inicjalizacji zmiennych strukturalnych oraz odwoływania się do pól struktury
 - wyjaśnia pojęcie wskaźnika, podaje jak deklaruje się wskaźniki i przypisuje im wartości

Zaliczenie wykładu - efekty kształcenia (EK1)

- Student, który zalicza na ocenę **dostateczny (3)** (c.d.):
 - opisuje funkcje do dynamicznego przydzielania i zwalniania pamięci w języku C
 - charakteryzuje elementy definicji funkcji w języku C
 - opisuje znakowe, łańcuchowe, sformatowane i blokowe operacje wejścia-wyjścia
 - charakteryzuje tryby otwarcia pliku w języku C oraz opisuje schemat przetwarzania pliku
 - podaje różnice pomiędzy plikami tekstowymi i binarnymi

Zaliczenie wykładu - efekty kształcenia (EK1)

- Student, który zalicza na ocenę **dobry (4)** (oprócz wymagań na ocenę 3):
 - charakteryzuje deklarację, inicjalizację i sposób odwoływania się o elementów tablic wielowymiarowych
 - wyjaśnia sposób deklarowania oraz przeznaczenie pól bitowych i unii
 - opisuje związek tablic ze wskaźnikami w języku C
 - wyjaśnia czym różni się deklaracja od definicji funkcji
 - podaje różnice w przekazywaniu parametrów do funkcji przez wartość i wskaźnik
 - wyjaśnia w jaki sposób w programach wielomodułowych można odwoływać się do zmiennych i funkcji zdefiniowanych w innych modułach

Zaliczenie wykładu - efekty kształcenia (EK1)

- Student, który zalicza na ocenę **bardzo dobry (5)** (oprócz wymagań na ocenę 4):
 - charakteryzuje tablice o zmiennym rozmiarze (VLA) w języku C
 - opisuje wybraną metodę przydziału pamięci dla macierzy
 - opisuje strukturę programu w pamięci komputera
 - wyjaśnia sposób przekazywania do funkcji tablic oraz struktur
 - charakteryzuje klasy zmiennych i klasy funkcji w języku C

Zaliczenie wykładu - efekty kształcenia (EK2)

- Student, który zaliczył przedmiot:

opisuje podstawowe zadania systemu operacyjnego
oraz strukturę sieci komputerowych

- Student, który zalicza na ocenę **dostateczny (3)**:
 - podaje definicję i wymienia podstawowe zadania systemu operacyjnego
 - opisuje wybraną metodę przydziału pamięci dyskowej
 - wyjaśnia podstawowe pojęcia związane z sieciami komputerowymi
 - charakteryzuje wybrane media transmisyjne i urządzenia sieciowe

Zaliczenie wykładu - efekty kształcenia (EK2)

- Student, który zalicza na ocenę **dobry (4)** (oprócz wymagań na ocenę 3):
 - podaje strukturę dysku logicznego w wybranym systemie plików (FAT, NTFS, ext)
 - wyjaśnia pojęcia stronicowania i segmentacji pamięci oraz opisuje zasadę działania pamięci wirtualnej
 - charakteryzuje podstawowe protokoły sieciowe oraz topologie sieci komputerowych

- Student, który zalicza na ocenę **bardzo dobry (5)** (oprócz wymagań na ocenę 4):
 - opisuje sposób przechowywania informacji o położeniu pliku na dysku w wybranym systemie plików (FAT, NTFS, ext)
 - opisuje modele ISO/OSI i TCP/IP stosowane w sieciach komputerowych

Zaliczenie wykładu

- Dwa sprawdziany pisemne:
 - sprawdzian 1: **26.11.2019** (wtorek), godz. 14:15-15:00, WE-030
 - sprawdzian 2: **28.01.2020** (wtorek), godz. 14:15-15:00, WE-030
 - poprawa: termin do ustalenia (sesja egzaminacyjna)
- Za każdy sprawdzian można otrzymać od 0 do 100 pkt.
- Na podstawie otrzymanych punktów wystawiana jest ocena:

Punkty	Ocena	Punkty	Ocena
91 - 100	5,0	61 - 70	3,5
81 - 90	4,5	51 - 60	3,0
71 - 80	4,0	0 - 50	2,0

Zaliczenie wykładu

- Ocena końcowa wyznaczana jest na podstawie sumy otrzymanych punktów za efekty kształcenia EK1 i EK2:

Punkty	Ocena	Punkty	Ocena
182 - 200	5,0	122 - 141	3,5
162 - 181	4,5	102 - 121	3,0
142 - 161	4,0	0 - 101	2,0

Plan wykładu nr 1

- Tablice w języku C
 - jednowymiarowe - wektory (przypomnienie)
 - dwuwymiarowe - macierze
 - wielowymiarowe

- Tablice o zmiennym rozmiarze (VLA)

Język C - tablica elementów

- **Tablica** - ciągły obszar pamięci, w którym umieszczone są elementy tego samego typu

wektor

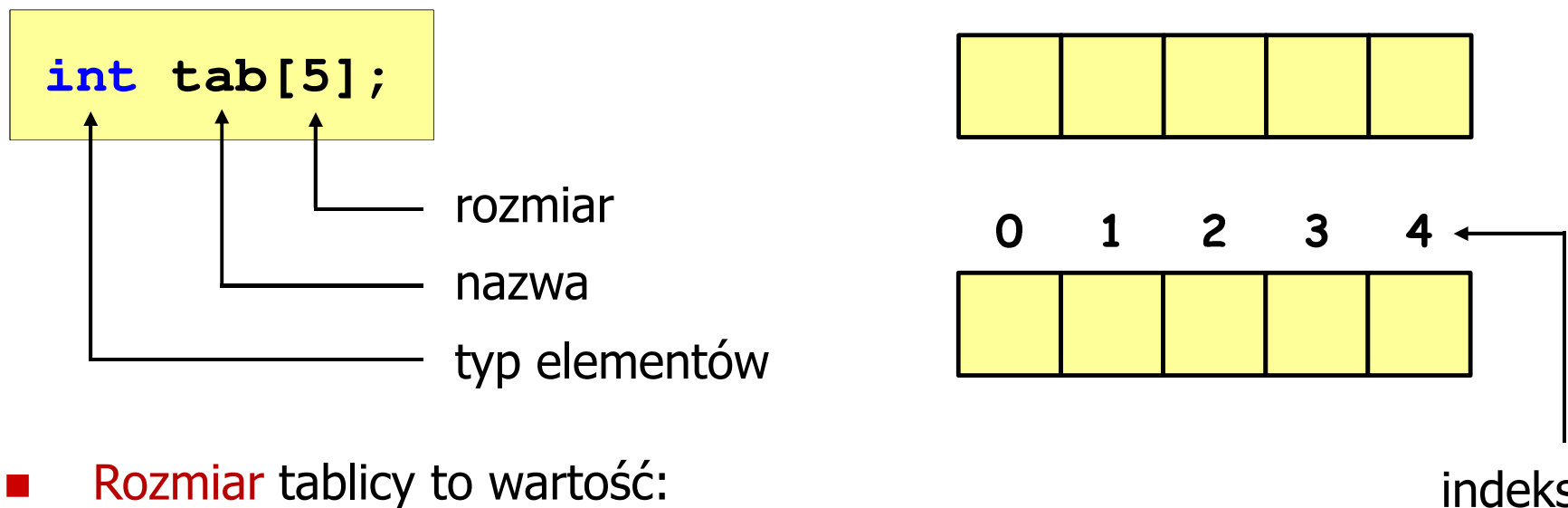
5	3	-2	1	-4
---	---	----	---	----

macierz

a	c	d	m
p	d	q	l
a	t	x	v

1.2	2.5	2.0	10.0
-0.1	4.3	6.2	-5.1
0.0	12.2	4.1	-2.2

Język C - deklaracja tablica jednowymiarowej



- **Rozmiar** tablicy to wartość:
 - całkowita, dodatnia
 - znana na etapie kompilacji programu
(stała liczbowa: `5`, `#define N 5`, `const int n = 5;`)

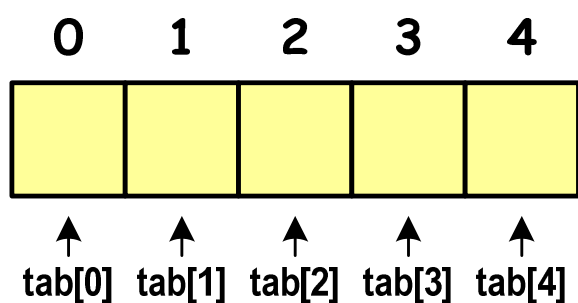
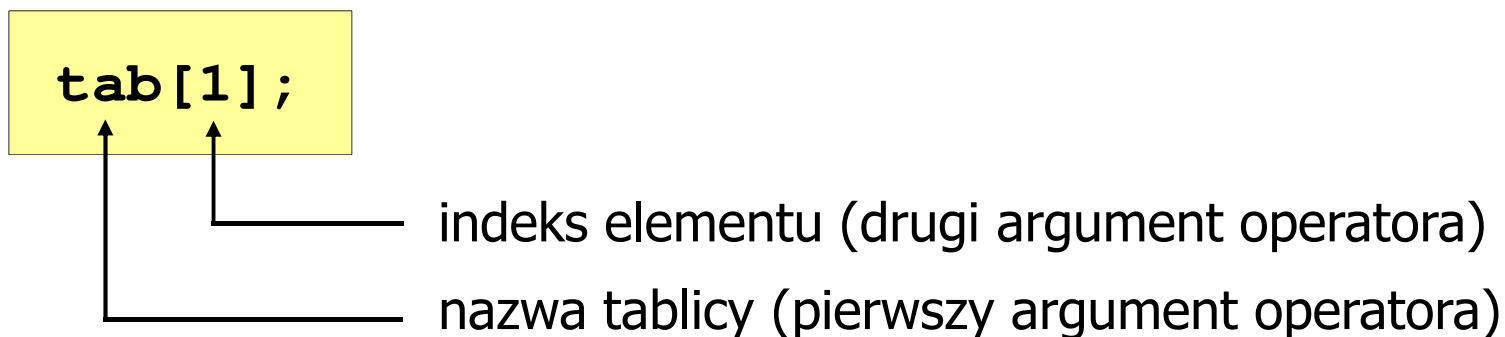
```
int tab[5];
```

```
int tab[N];
```

```
int tab[n];
```

Język C - odwołania do elementów tablicy

[] - dwuargumentowy operator indeksowania



- Indeks:
 - stała liczbowa, np. **0**, **1**, **10**
 - nazwa zmiennej, np. **i**, **idx**
 - wyrażenie, np. **i*j+5**

Język C - inicjalizacja tablicy jednowymiarowej

```
int tab[5] = {1, 2, 3, 4, 5};
```

0	1	2	3	4
1	2	3	4	5

```
int tab[5] = {1, 2, 3};
```

0	1	2	3	4
1	2	3	0	0

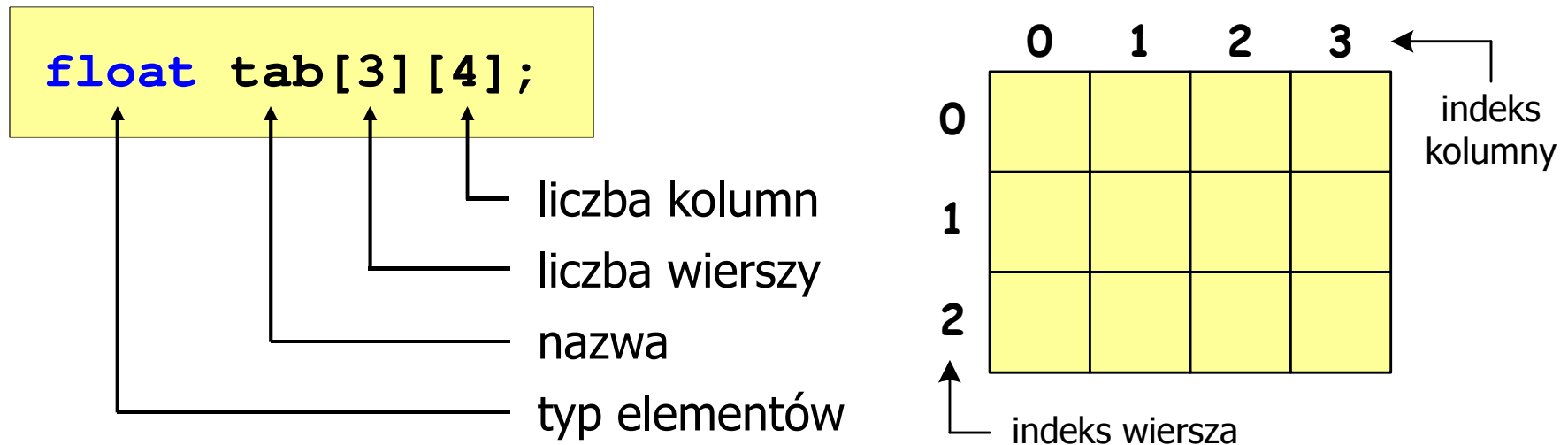
```
int tab[5] = {1, 2, 3, 4, 5, 6};
```

- błąd kompilacji

```
int tab[] = {1, 2, 3, 4, 5};
```

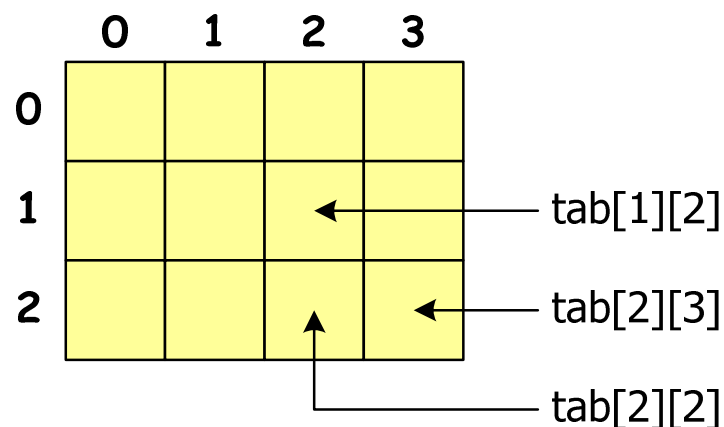
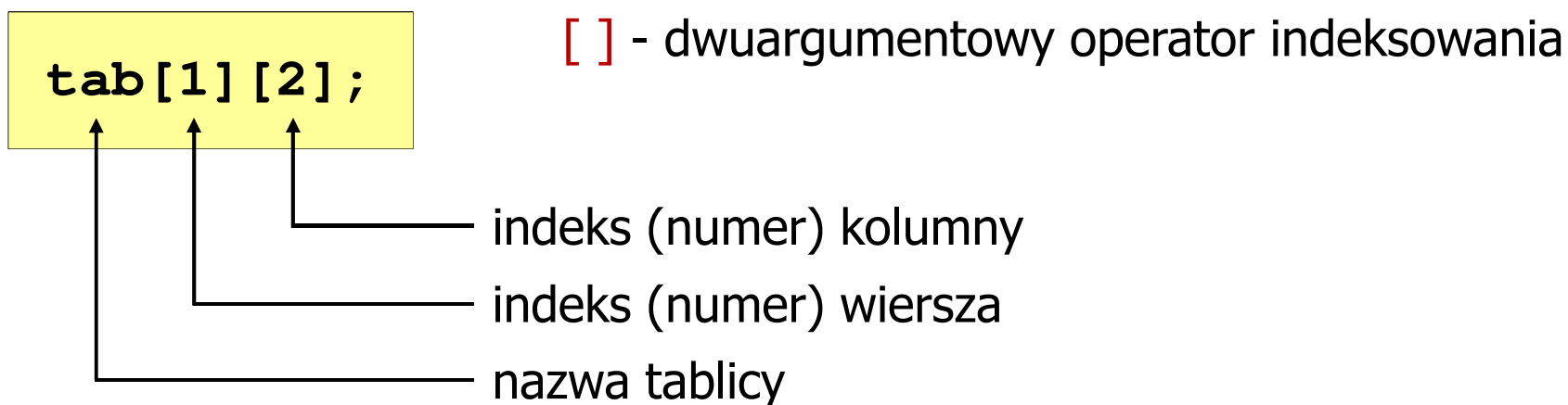
0	1	2	3	4
1	2	3	4	5

Język C - deklaracja tablica dwuwymiarowej



- **Rozmiar** tablicy (liczb wierszy i kolumn) to wartość:
 - całkowita, dodatnia
 - znana na etapie kompilacji programu
(stała liczbowa: `5`, `#define N 5`, `const int n = 5;`)

Język C - odwołania do elementów macierzy



- Indeks:
 - stała liczbowa, np. **0**, **1**, **10**
 - nazwa zmiennej, np. **i**, **idx**
 - wyrażenie, np. **i*j+5**
- Brak sprawdzania poprawności indeksów!

Język C - inicjalizacja elementów macierzy

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

	0	1	2
0	1	2	3
1	4	5	6

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

	0	1	2
0	1	2	3
1	4	0	0

```
int T[2][3] = {1, 2, 3, 4};
```

	0	1	2
0	1	0	0
1	4	5	0

```
int T[2][3] = {{1}, {4, 5}};
```

Język C - inicjalizacja elementów macierzy

```
int T[2][3] = {0};
```

wyzerowanie elementów macierzy

	0	1	2
0	0	0	0
1	0	0	0

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

pominięcie liczby wierszy

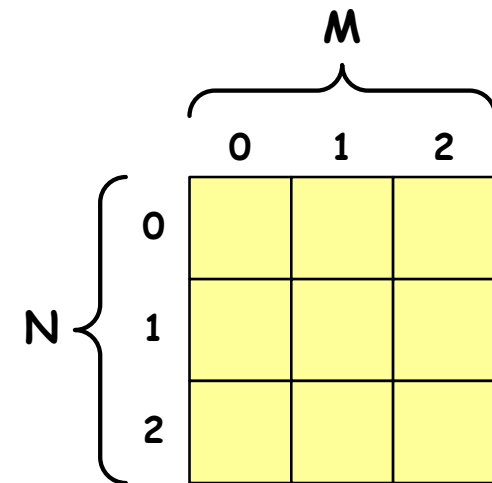
	0	1	2
0	1	2	3
1	4	5	6

Język C - operacje na macierzy

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

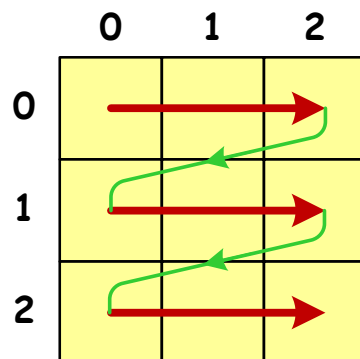
#define N 3      /* liczba wierszy */
#define M 3      /* liczba kolumn */

int main(void)
{
    int tab[N][M];
    int i, j;
```

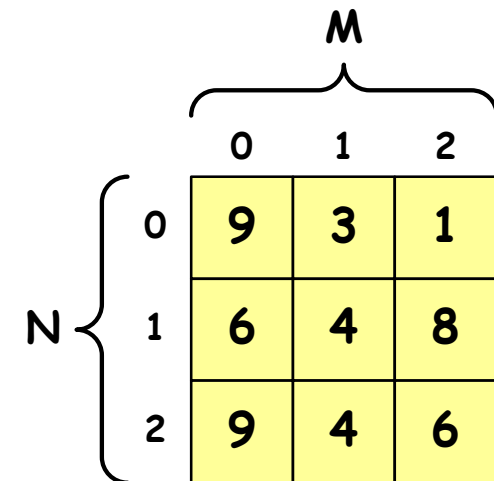


Język C - operacje na macierzy

```
/* generowanie pseudolosowe elementow macierzy */  
  
srand((unsigned int) time(NULL));  
for (i=0; i<N; i++)  
    for (j=0; j<M; j++)  
        tab[i][j] = rand() % 10;
```



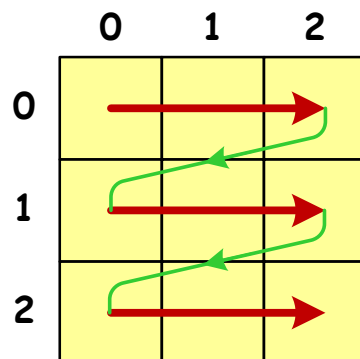
kolejność zapisywania
wartości elementów
macierzy



Język C - operacje na macierzy

```
/* wyświetlenie elementów macierzy */  
  
for (i=0; i<N; i++)  
{  
    for (j=0; j<M; j++)  
        printf("%3d", tab[i][j]);  
    printf("\n");  
}
```

9	3	1
6	4	8
9	4	6

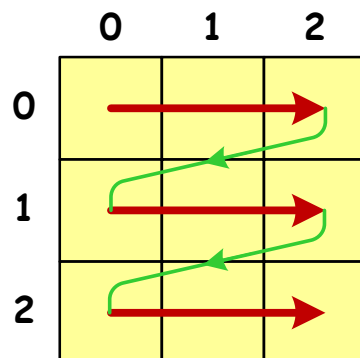


	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

Język C - operacje na macierzy

```
/* poszukiwanie elementu o wartosci minimalnej */  
  
int min = tab[0][0];  
for (i=0; i<N; i++)  
    for (j=0; j<M; j++)  
        if (tab[i][j] < min)  
            min = tab[i][j];  
printf("Wartosc min: %d\n",min);
```

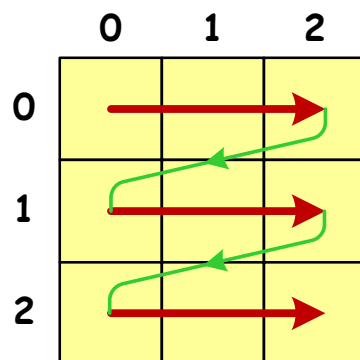
Wartosc min: 1



	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

Język C - operacje na macierzy

```
/* suma i srednia arytmetyczna elementow */  
  
int suma = 0;  
for (i=0; i<N; i++)  
    for (j=0; j<M; j++)  
        suma = suma + tab[i][j];  
float srednia = (float) suma / (N*M);  
printf("Suma:      %d\n", suma);  
printf("Srednia:  %f\n\n", srednia);
```

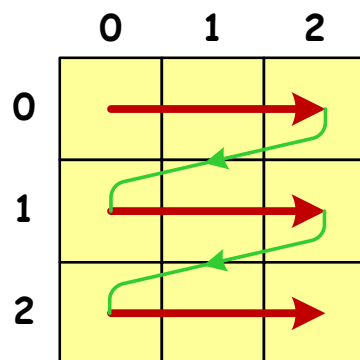


	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

Suma: 50
Srednia: 5.555555

Język C - operacje na macierzy

```
/* sumy elementow w poszczegolnych wierszach */  
for (i=0; i<N; i++)  
{  
    suma = 0;  
    for (j=0; j<M; j++)  
        suma = suma + tab[i][j];  
    printf("Suma wiersza %d = %d\n", i, suma);  
}
```

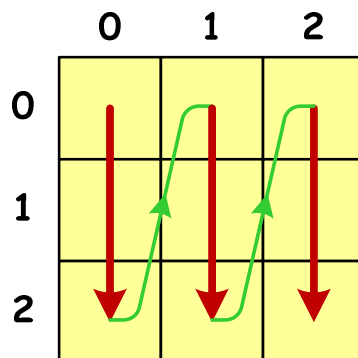


	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

Suma wiersza 0 = 13
Suma wiersza 1 = 18
Suma wiersza 2 = 19

Język C - operacje na macierzy

```
/* sumy elementow w poszczegolnych kolumnach */  
for (j=0; j<M; j++)  
{  
    suma = 0;  
    for (i=0; i<N; i++)  
        suma = suma + tab[i][j];  
    printf("Suma kolumny %d = %d\n", j, suma);  
}
```



	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

Suma kolumny 0 = 24
Suma kolumny 1 = 11
Suma kolumny 2 = 15

Język C - operacje na macierzy

```
/* sumy elementow nad, na i ponizej przekatnej */  
  
suma = suma1 = suma2 = 0;  
for (i=0; i<N; i++)  
    for (j=0; j<M; j++)  
    {  
        if (i < j) suma1+=tab[i][j]; /* nad */  
        if (i > j) suma2+=tab[i][j]; /* pod */  
        if (i == j) suma+=tab[i][j]; /* na */  
    }  
  
printf("Suma nad: %d\n", suma1);  
printf("Suma na: %d\n", suma);  
printf("Suma pod: %d\n", suma2);
```

```
Suma nad: 12  
Suma na: 19  
Suma pod: 19
```


Język C - operacje na macierzy

		j		
		0	1	2
i	0	0,0	0,1	0,2
	1	1,0	1,1	1,2
	2	2,0	2,1	2,2

$i < j$

		j		
		0	1	2
i	0	0,0	0,1	0,2
	1	1,0	1,1	1,2
	2	2,0	2,1	2,2

$i = j$

		j		
		0	1	2
i	0	0,0	0,1	0,2
	1	1,0	1,1	1,2
	2	2,0	2,1	2,2

$i > j$

	0	1	2
0			
1			
2			

	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

Suma nad: 12
 Suma na: 19
 Suma pod: 19

Język C - tablice wielowymiarowe

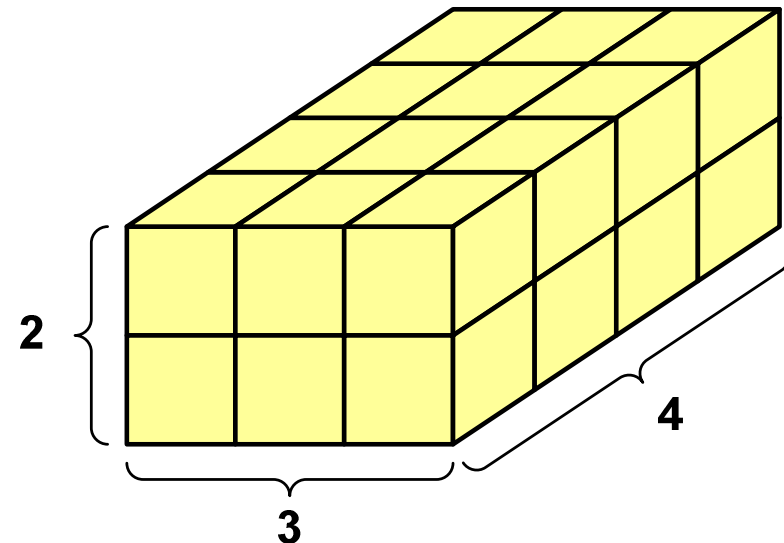
- Deklaracja tablicy wielowymiarowej

```
typ nazwa[wymiar_1][wymiar_2]...[wymiar_N]
```

- Deklaracja tablicy trójwymiarowej

```
int tab[4][2][3];
```

- Inicjalizacja i odwoływanie się do elementów są analogiczne jak w przypadku macierzy



Język C - tablice wielowymiarowe

```
#include <stdio.h>
```

```
#define X 3
```

```
#define Y 2
```

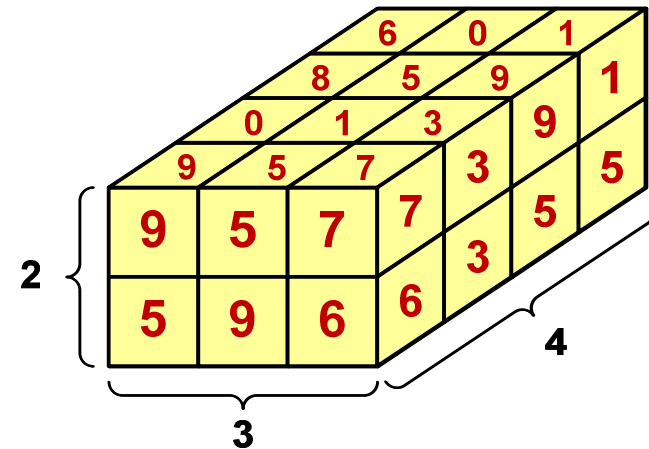
```
#define Z 4
```

```
int main(void)
```

```
{
```

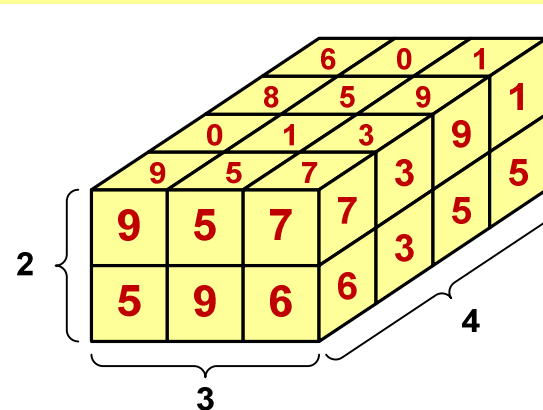
```
    int x, y, z;
```

```
    int tab[Z][Y][X] = {{{9, 5, 7}, {5, 9, 6}},  
                        {{0, 1, 3}, {7, 4, 3}},  
                        {{8, 5, 9}, {1, 3, 5}},  
                        {{6, 0, 1}, {8, 2, 5}}};
```



Język C - tablice wielowymiarowe

```
for (z=0; z<Z; z++)  
{  
    for (y=0; y<Y; y++)  
    {  
        for (x=0; x<X; x++)  
            printf ("%3d", tab[z][y][x]);  
        printf ("\n");  
    }  
    printf ("\n");  
}  
  
return 0;  
}
```



9	5	7
5	9	6
0	1	3
7	4	3
8	5	9
1	3	5
6	0	1
8	2	5

Tablice o zmiennym rozmiarze (VLA)

- **VLA** (ang. variable length array) - tablice, których rozmiar określany jest na etapie wykonywania programu (np. jako rozmiar może wystąpić nazwa zmiennej)

```
int n;  
n = 10;  
int T[n];
```

```
int n;  
scanf("%d", &n);  
int T[n];
```

- Rozmiar tablicy, a standardy języka C:
 - do standardu C99 rozmiar tablicy musiał być stałym wyrażeniem całkowitym (stała liczbowa: 5, #define N 5, const int n = 5;)
 - w standardzie C99 wprowadzono tablice o zmiennym rozmiarze
 - w standardzie C11 tablice o zmiennym rozmiarze określane są jako opcjonalne dla implementacji

Tablice VLA (VC++ 2008)

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    int n, i;

    printf("Rozmiar wektora: ");
    scanf("%d", &n);

    float T[n];

    for (i=0; i<n; i++)
        T[i] = sqrt((float)i);

    for (i=0; i<n; i++)
        printf("T[%d] = %f\n", i, T[i]);

    return 0;
}
```

Tablice VLA (VC++ 2008)

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    int n, i;

    printf("Rozmiar wektora: ");
    scanf("%d", &n);

    float T[n];

    for (i=0; i<n; i++)
        T[i] = sqrt((float)i);

    for (i=0; i<n; i++)
        printf("T[%d] = %f\n", i, T[i]);

    return 0;
}
```

error C2057: expected constant expression
error C2466: cannot allocate an array of constant size 0
error C2133: 'T' : unknown size

Tablice VLA (Dev-C++, Code::Blocks)

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    int n, i;

    printf("Rozmiar wektora: ");
    scanf("%d", &n);

    float T[n];

    for (i=0; i<n; i++)
        T[i] = sqrt((float)i);

    for (i=0; i<n; i++)
        printf("T[%d] = %f\n", i, T[i]);

    return 0;
}
```

```
Rozmiar wektora: 8
T[0] = 0.000000
T[1] = 1.000000
T[2] = 1.414214
T[3] = 1.732051
T[4] = 2.000000
T[5] = 2.236068
T[6] = 2.449490
T[7] = 2.645751
```


Tablice VLA

- Tablica VLA może być także tablicą dwu- lub wielowymiarową

```
int n = 5, m = 6;  
int T1[n][m], T2[n][m][n];
```

- Nie można modyfikować rozmiaru tablic VLA po deklaracji
- Tablice VLA nie mogą być inicjalizowane podczas deklaracji
 - błędy i ostrzeżenia w **Code::Blocks**

```
error: variable-sized object may not be initialized  
warning: excess elements in array initializer  
warning: (near initialization for 'T')
```

- w **Dev-C++** inicjalizacja jest dopuszczalna!

Modularność programu

- Program komputerowy powinien być podzielony na osobne **jednostki**, z których każda wykonuje jedno zadanie
- Moduły (jednostki) to najczęściej **funkcje** języka C (ale mogą to być też oddzielne pętle)
- Zalety budowy modularnej programu:
 - większa czytelność kodu programu
 - prostsza modyfikacja programu

Modularność programu

- Przykład

```
int T[10], i, s = 0;

srand(time(NULL));

for(i=0; i<10; i++)
{
    T[i] = rand()%100;
    printf("%4d", T[i]);
    s = s + T[i];
}
```

```
int T[10], i, s = 0;

srand(time(NULL));

for(i=0; i<10; i++)
    T[i] = rand()%100;

for(i=0; i<10; i++)
    printf("%4d", T[i]);

for(i=0; i<10; i++)
    s = s + T[i];
```

- Zamiast jednej pętli **for** stosowane są trzy pętle

Koniec wykładu nr 1

Dziękuję za uwagę!