Informatyka 2

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr III, studia stacjonarne I stopnia

Rok akademicki 2017/2018

Wykład nr 1 (25.09.2017)

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://we.pb.edu.pl/~jforenc
  - Dydaktyka - slajdy prezentowane na wykładzie
- konsultacje:
  - poniedziałek, godz. 15:45-18:00, WE-204
  - wtorek, godz. 12:00-14:15, WE-204
  - czwartek, godz. 10:30-12:00, WE-204

Program wykładu (1/2)

1. Tablice dwu- i wielowymiarowe w języku C. Łańcuchy znaków. Plik nagłówkowy string.h.
2. Struktury w języku C, inicjalizacja zmiennej strukturalnej, odwołania do pól struktury. Wskaźniki, operacje na wskaźnikach. Dynamiczny przydział pamięci w języku C.
3. Funkcje w języku C, ogólna struktura funkcji, deklaracja i definicja funkcji, przekazywanie argumentów do funkcji przez wartość i wskaźnik, rekurencyjne wywołanie funkcji. Klasy zmiennych i funkcji. Programy wielomodułowe.
4. Operacje wejścia-wyjścia w języku C: znakowe, łańcuchowe, sformatowane, rekordowe. Strumienie. Pliki tekstowe i binarne.

Program wykładu (2/2)

5. Sprawdzian nr 1. System operacyjny. Funkcje i zadania systemu operacyjnego.
6. Zarządzanie procesami, pamięcią i dyskami w systemach operacyjnych.
7. Sieci komputerowe. Technologie, protokoły, urządzenia. Zasada działania sieci Internet.
8. 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. P. Prinz, T. Crawford: „Język C w pigułce”. APN Promise, Warszawa, 2016.
4. K.N. King: „Język C. Nowoczesne programowanie. Wydanie II”. Helion, Gliwice, 2011.
5. S.G. Kochan: „Język C. Kompendium wiedzy. Wydanie IV”. Helion, Gliwice, 2015.
6. R. Reese: „Wskaźniki w języku C. Przewodnik”. Helion, Gliwice, 2014.

Literatura (2/2)

7. G. Coldwin: „Zrozumieć programowanie”. PWN, Warszawa, 2015.
8. A.S. Tanenbaum: „Systemy operacyjne. Wydanie III”. Helion, Gliwice, 2010.
9. W. Stallings: „Systemy operacyjne. Struktura i zasady budowy”. Mikom, Warszawa, 2006.
10. A.S. Tanenbaum, D.J. Wetherall: „Sieci komputerowe. Wydanie V”. Helion, Gliwice, 2012.
11. K. Krysiak: „Sieci komputerowe. Kompendium. Wydanie II”. Helion, Gliwice, 2005.

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 deklarowania i inicjalizacji tablic dwuwymiarowych (macierzy) w języku C oraz metody wykonywania podstawowych operacji na tych tablicach
  - opisuje sposób deklarowania, inicjalizacji oraz przechowywania łańcuchów znaków (napisów)
  - omawia sposób deklarowania 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: 20.11.2017 (poniedziałek), godz. 12:15-13:00, WE-Aula III
  - sprawdzian 2: 15.01.2018 (poniedziałek), godz. 12:15-13:00, WE-Aula III
  - poprawa: termin do ustalenia (sesja egzaminacyjna)
- Za każdy sprawdzian można otrzymać od 0 do 100 pkt.
- Na podstawie otrzymanych punktów wystawiana jest ocena:

91-100 pkt. → 5,0	81-90 pkt. → 4,5
71-80 pkt. → 4,0	61-70 pkt. → 3,5
51-60 pkt. → 3,0	0-50 pkt. → 2,0

Zaliczenie wykładu

- Ocena końcowa jest średnią arytmetyczną otrzymanych ocen:

ocena		punkty
5	→	5,0
5-	→	4,8
4,5	→	4,5
4	→	4,0
4-	→	3,8
3,5	→	3,5
3	→	3,0
2	→	2,0

średnia		ocena końcowa
4,75 - 5,00	→	5
4,25 - 4,74	→	4,5
3,75 - 4,24	→	4
3,25 - 3,74	→	3,5
3,00 - 3,24	→	3

Terminy zajęć

- Wykład nr 1 - 25.09.2017
- Wykład nr 2 - 09.10.2017
- Wykład nr 3 - 23.10.2017
- Wykład nr 4 - 06.11.2017
- Wykład nr 5 - 20.11.2017 (sprawdzian nr 1)
- Wykład nr 6 - 04.12.2017
- Wykład nr 7 - 18.12.2017
- Wykład nr 8 - 15.01.2018 (1h, 12:15-13:00, sprawdzian nr 2)

Plan wykładu nr 1

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

Język C - tablica elementów

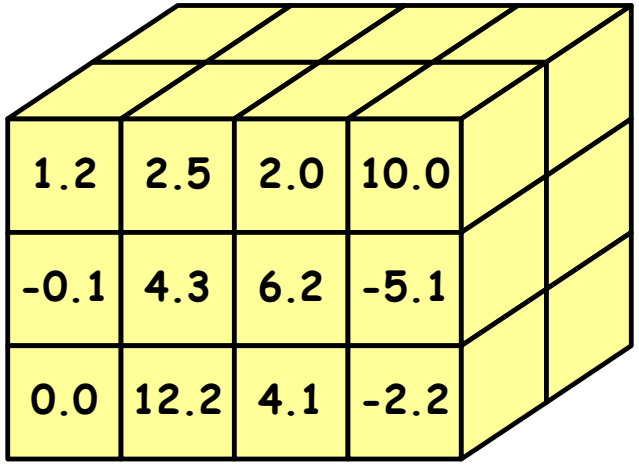
- Tablica - ciągły obszar pamięci, w którym umieszczone są elementy tego samego typu

wektor

5	3	-2	1	-4

macierz

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



Język C - po co tablice?

```
#include <stdio.h>

int main(void)
{
    double U1, U2, U3, U4, U5;
    double I1, I2, I3, I4, I5;
    double R1, R2, R3, R4, R5;

    U1 = 5.0;
    U2 = 10.0;
    U3 = 15.0;
    U4 = 20.0
    U5 = 25.0;

    I1 = 0.16;
    I2 = 0.21;
    I3 = 0.27;
    I4 = 0.33;
    I5 = 0.36;
```

Język C - po co tablice?

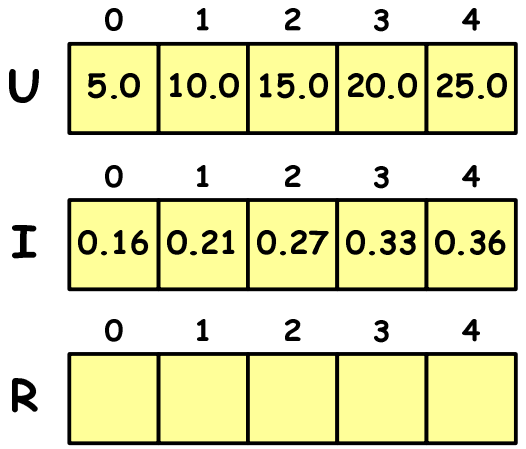
```
    R1 = U1/I1;
    R2 = U2/I2;
    R3 = U3/I3;
    R4 = U4/I4;
    R5 = U5/I5;

    printf("R1 = %f\n",R1);
    printf("R2 = %f\n",R2);
    printf("R3 = %f\n",R3);
    printf("R4 = %f\n",R4);
    printf("R5 = %f\n",R5);

    return 0;
}
```

```
R1 = 31.250000
R2 = 47.619048
R3 = 55.555556
R4 = 60.606061
R5 = 69.444444
```

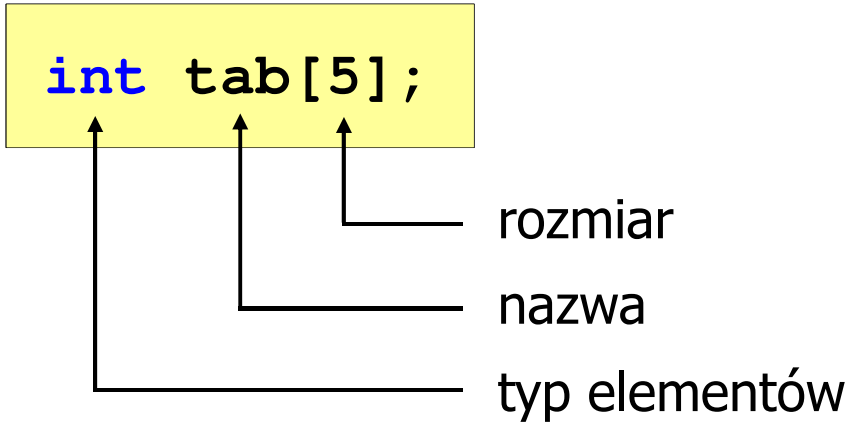
Język C - po co tablice?

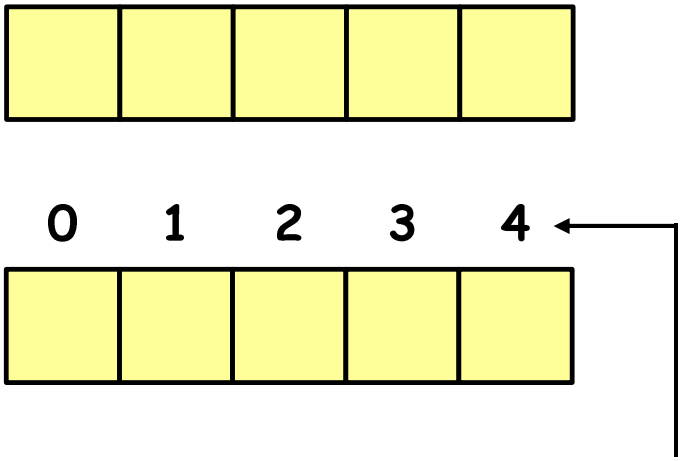
```
R1 = 31.250000
R2 = 47.619048
R3 = 55.555556
R4 = 60.606061
R5 = 69.444444
```

```c
#include <stdio.h>

int main(void)
{
    double U[5] = { 5.0, 10.0, 15.0, 20.0, 25.0 };
    double I[5] = { 0.16, 0.21, 0.27, 0.33, 0.36 };
    double R[5];
    int i;

    for (i=0; i<5; i++)
        R[i] = U[i]/I[i];

    for (i=0; i<5; i++)
        printf("R%d = %f\n",i+1,R[i]);

    return 0;
}
```



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;)

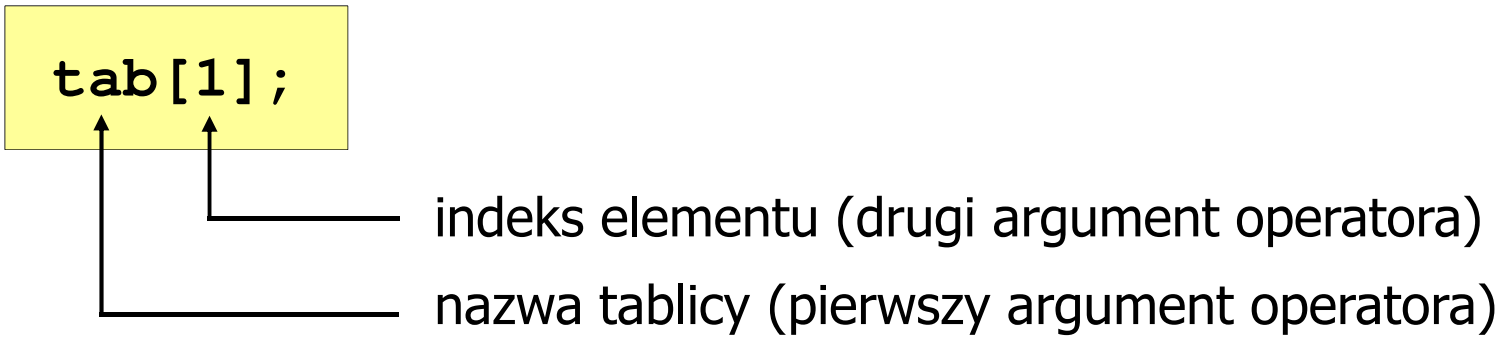
```
int tab[5];
```

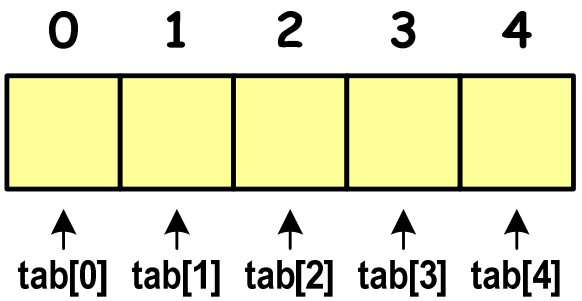
```
int tab[N];
```

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

Język C - odwołania do elementów tablicy

[ ] - dwuargumentowy operator indeksowania

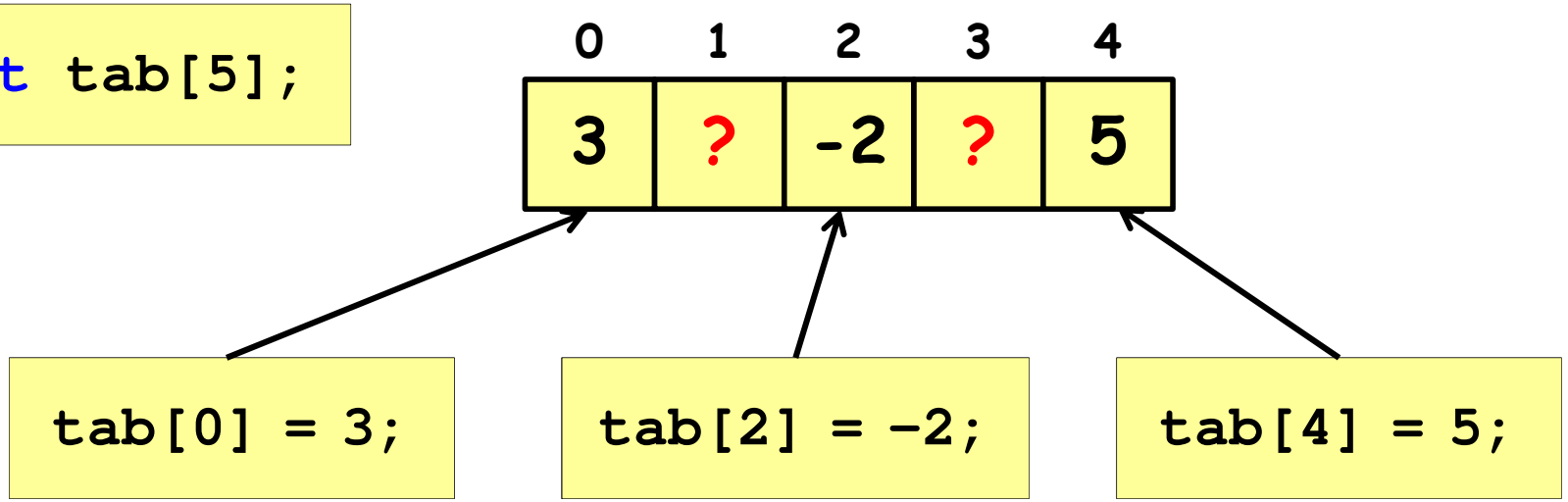




- Indeks:
  - stała liczbowa, np. 0, 1, 10
  - nazwa zmiennej, np. i, idx
  - wyrażenie, np. i*j+5

Język C - odwołania do elementów tablicy

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



- Każdy element tablicy traktowany jest tak samo jak zmienna typu int

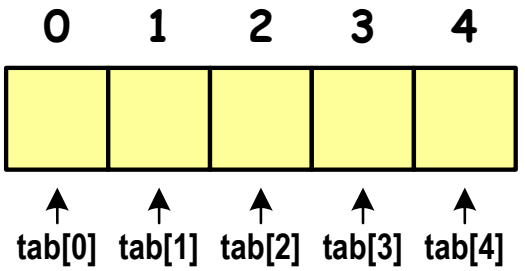
```
printf("%d",tab[0]);
```

```
scanf("%d",&tab[1]);
```

Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

```
int tab[5];
tab[5] = 10;
```



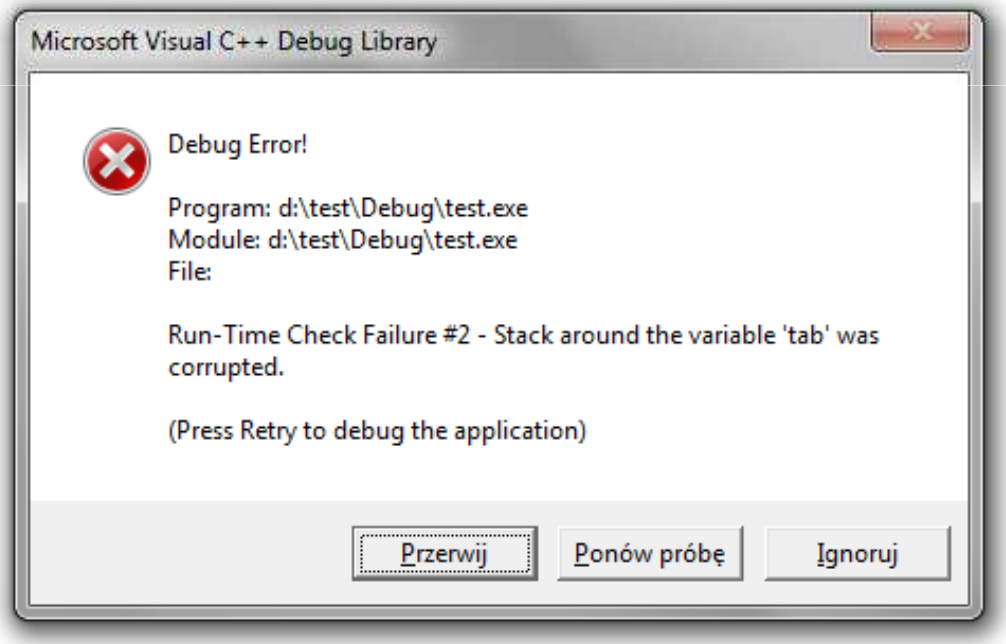
- błąd!!! - nie istnieje element tab[5]

- Kompilator nie zasygnalizuje błędu
- Program wykona operację, ale wynik jest niezdefiniowany
- Środowisko programistyczne może zasygnalizować problem

Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

```
int tab[5];

tab[5] = 10;
```



Język C - odwołania do elementów tablicy

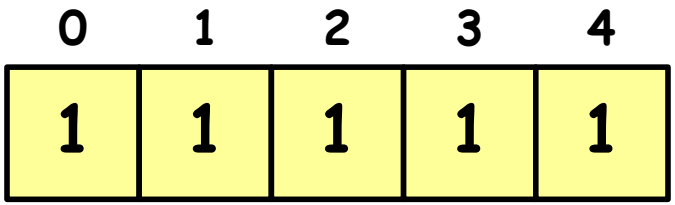
Dlaczego język C nie sprawdza poprawności indeksów tablic?

- Filozofia języka C to zaufanie do programisty
- Program działa szybciej
  - wartość indeksu może być określana nie tylko na etapie kompilacji, ale również na etapie wykonywania programu
  - w takim przypadku kontrola indeksu wymagałaby dodania dodatkowego kodu sprawdzającego poprawność

Język C - odwołania do elementów tablicy

- Zapisanie wartości 1 do wszystkich elementów tablicy

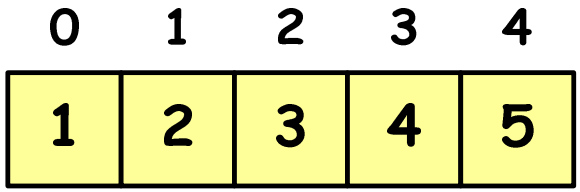
```
int tab[5];

tab[0] = 1;
tab[1] = 1;
tab[2] = 1;
tab[3] = 1;
tab[4] = 1;
```



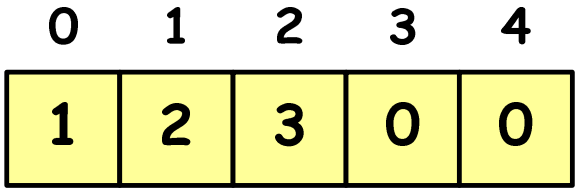

```
int tab[5], i;

for (i=0; i<5; i++)
   tab[i] = 1;
```

Język C - inicjalizacja tablicy jednowymiarowej

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



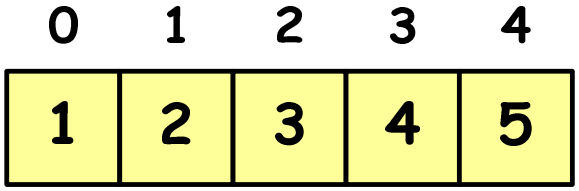

```
int tab[5] = {1,2,3};
```




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

- błąd kompilacji

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

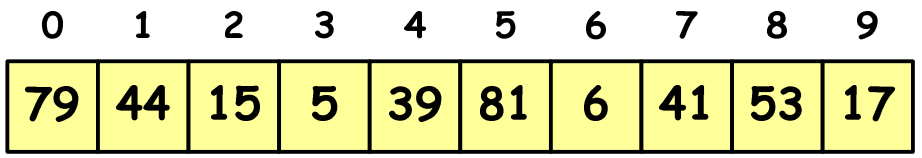


Język C - generator liczb pseudolosowych

- rand() - zwraca liczbę pseudolosową - zakres: 0 ... 32767
- srand() - inicjalizuje generator liczb pseudolosowych
- Plik nagłówkowy: stdlib.h (time.h)

```
int x, y;
srand((unsigned int) time(NULL));
x = rand();                 // zakres <0,32767>
y = rand() % 100;           // zakres <0,99>
```

Język C - operacje na wektorze




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

#define N 10

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

     srand((unsigned int) time(NULL));

     for (i=0; i<N; i++)
          tab[i] = rand() % 100;
```

Język C - operacje na wektorze

```
        for (i=0; i<N; i++)
             printf("%3d",tab[i]);
        printf("\n");

        suma = 0;
        for (i=0; i<N; i++)
             suma = suma + tab[i];
        printf("suma = %d",suma);

        return 0;
}
```

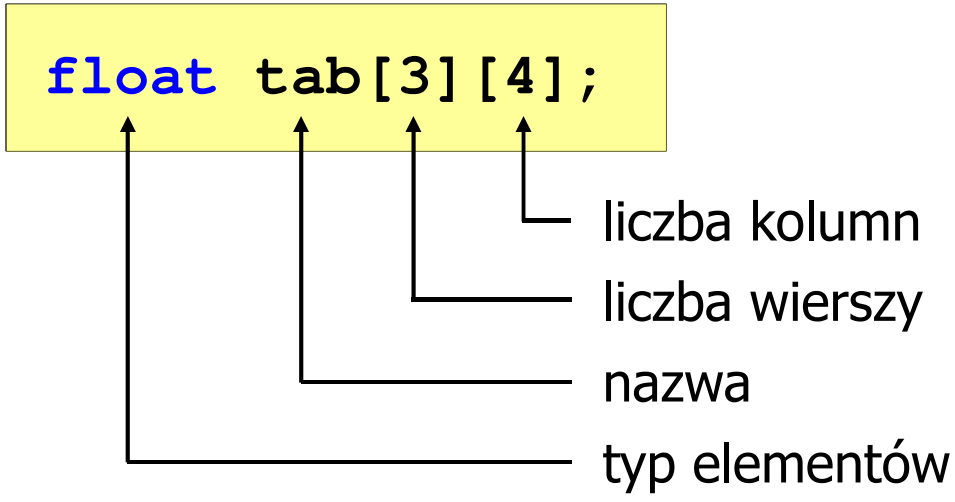
```
 79 44 15  5 39 81  6 41 53 17
suma = 380
```

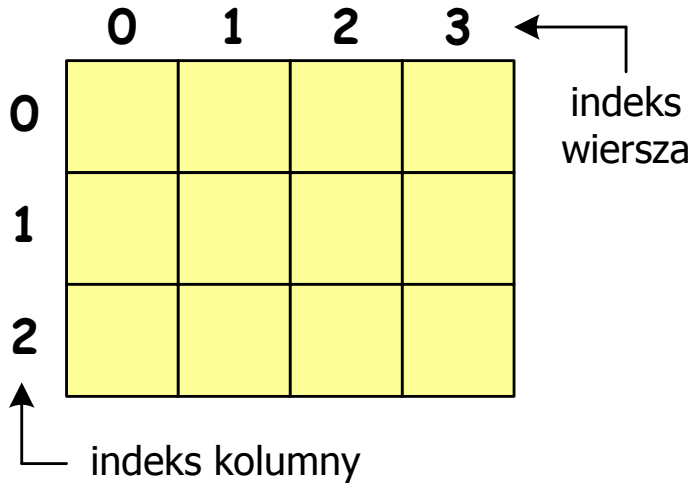
0	1	2	3	4	5	6	7	8	9
79	44	15	5	39	81	6	41	53	17

N = 10

Język C - deklaracja tablica dwuwymiarowej

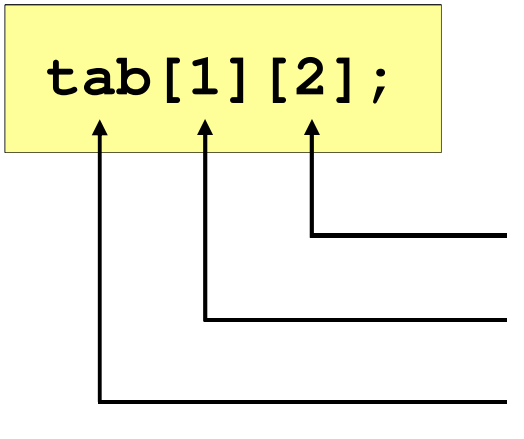
```
float tab[3][4];
```



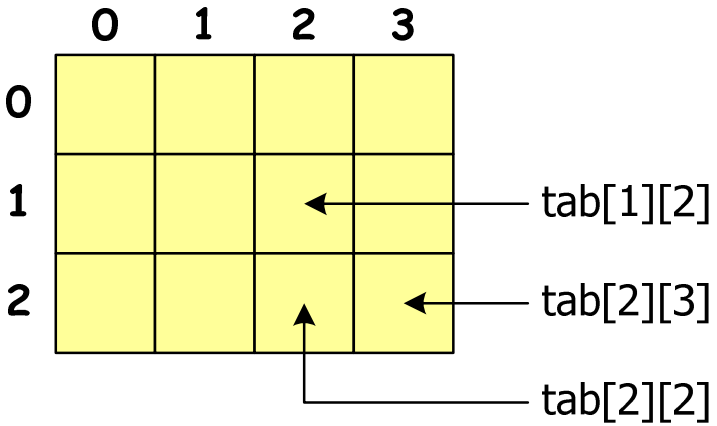


- Rozmiar tablicy (liczb wierszy i kolumn) to wartość:
  - całkowita, dodatnia
  - znana na etapie kompilacji programu
    (stała liczbowa: 5, #define N 5, const int n = 5;)

Język C - odwołania do elementów macierzy



[ ] - dwuargumentowy operator indeksowania



- 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	5	6

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

	0	1	2
0	1	2	3
1	4	0	0

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

	0	1	2
0	1	0	0
1	4	5	0

Język C - inicjalizacja elementów macierzy

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

wyzerowanie elementów macierzy

	0	1	2
0	0	0	0
1	0	0	0

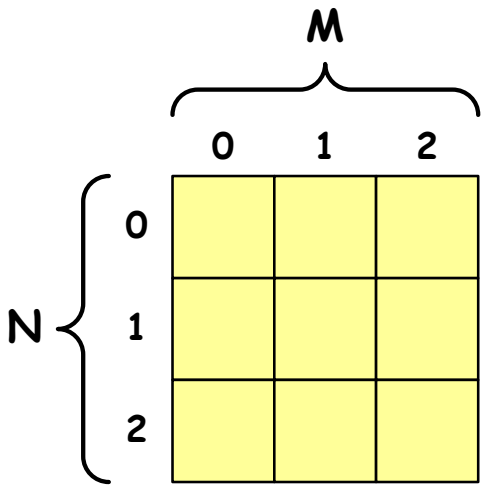
```
int T[][3] = {{1,2,3},{4,5,6}};
```

pominięcie liczby wierszy

	0	1	2
0	1	2	3
1	4	5	6

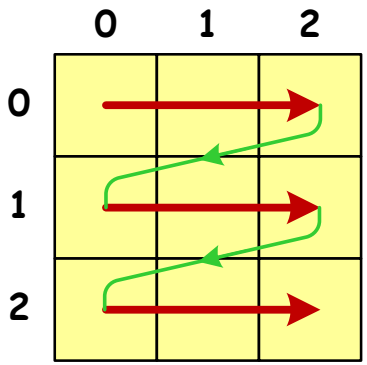
Język C - operacje na macierzy

```
#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;
```

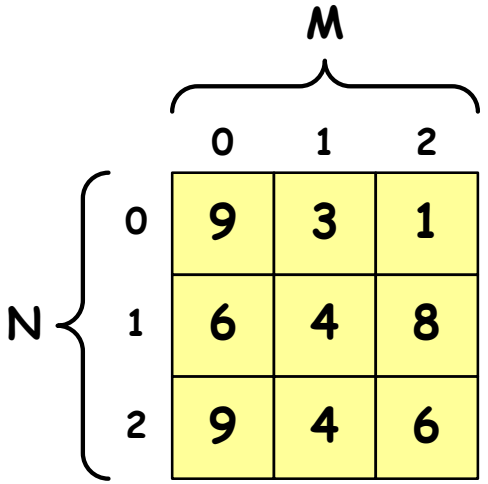


Język C - operacje na macierzy

```
/* 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;
```



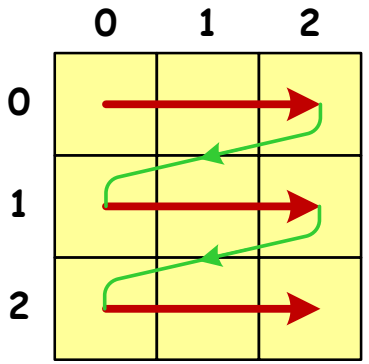
kolejność zapisywania wartości elementów macierzy



Język C - operacje na macierzy

```
/* wyswietlenie elementow 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
```

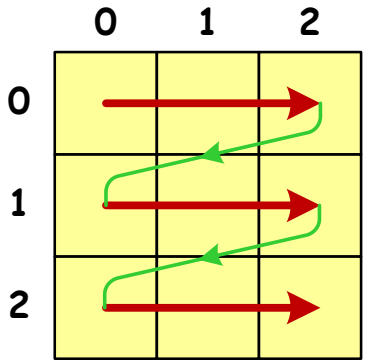


	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

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);
```

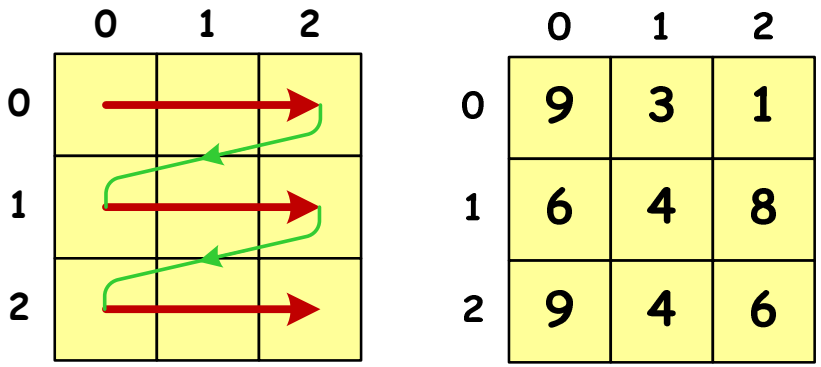
```
Wartosc min: 1
```



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

Język C - operacje na macierzy

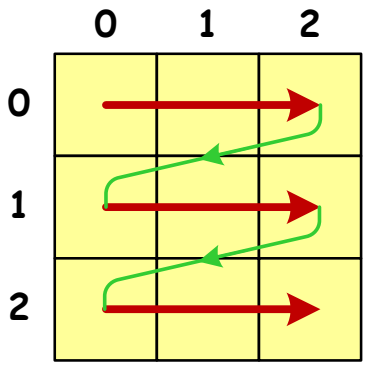
```
/* suma i srednia arytmetyczna elementow */

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);
```




```
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);
}
```

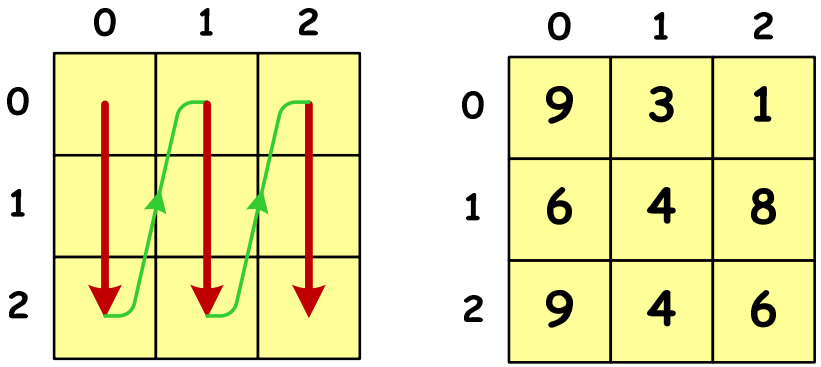


	0	1	2
0	9	3	1
1	6	4	8
2	9	4	6

```
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

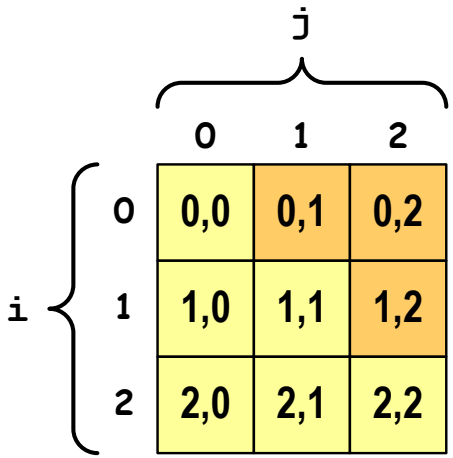
```
Suma kolumny 0 = 24
Suma kolumny 1 = 11
Suma kolumny 2 = 15
```

Język C - operacje na macierzy

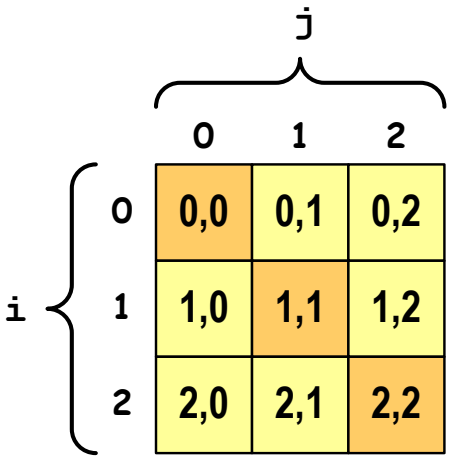
```
/* sumy elementow nad, na i ponizej przekatnej */

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
```

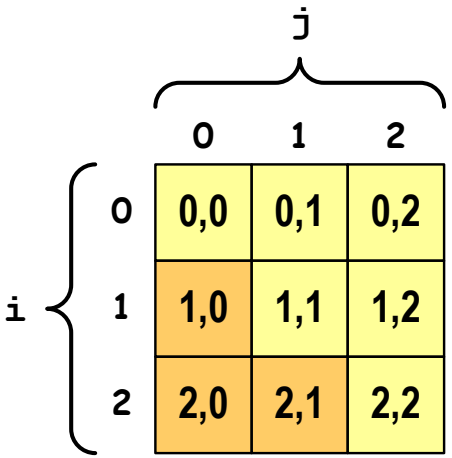
Język C - operacje na macierzy



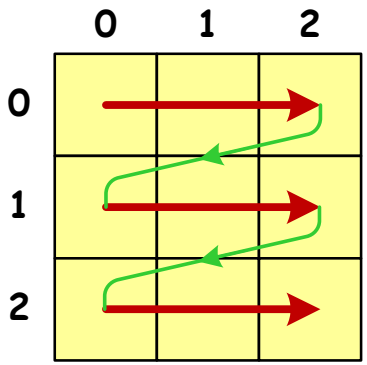
`i < j`

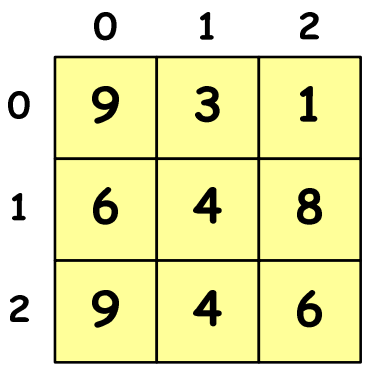


`i = j`



`i > j`






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

Język C - tablice wielowymiarowe

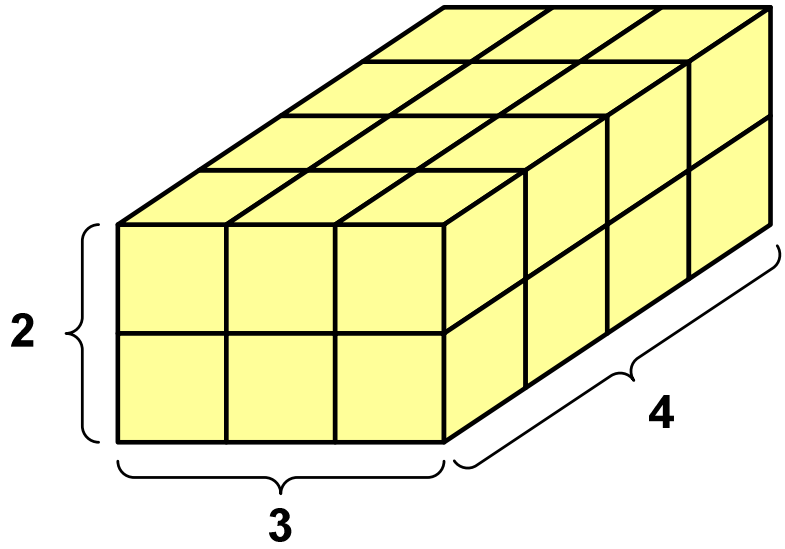
- Deklaracja tablicy wielowymiarowej

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

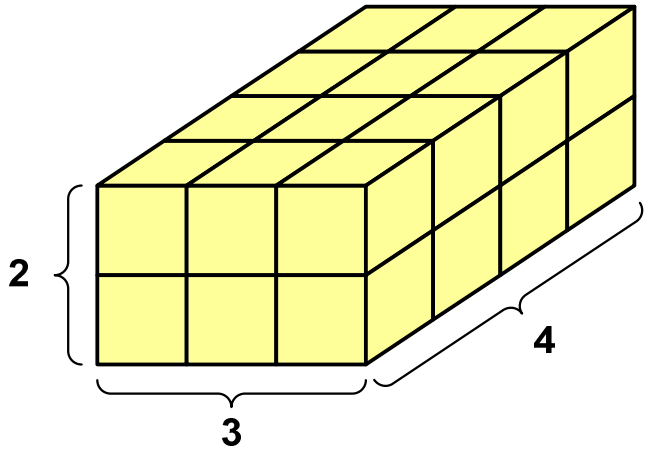
- Deklaracja tablicy trójwymiarowej

```
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;
}
```

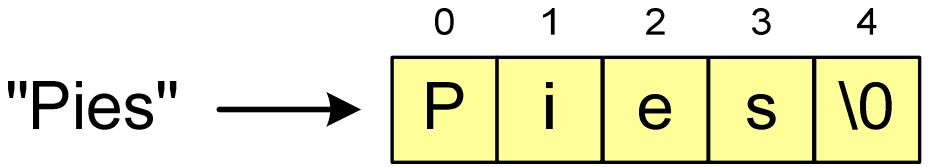
```
  9  5  7
  5  9  6

  0  1  3
  7  4  3

  8  5  9
  1  3  5

  6  0  1
  8  2  5
```

Język C - łańcuchy znaków

- Łańcuch znaków (ciąg znaków, napis, C-string) - ciąg złożony z zera lub większej liczby znaków zawartych między znakami cudzysłowu

```
"Pies"
```

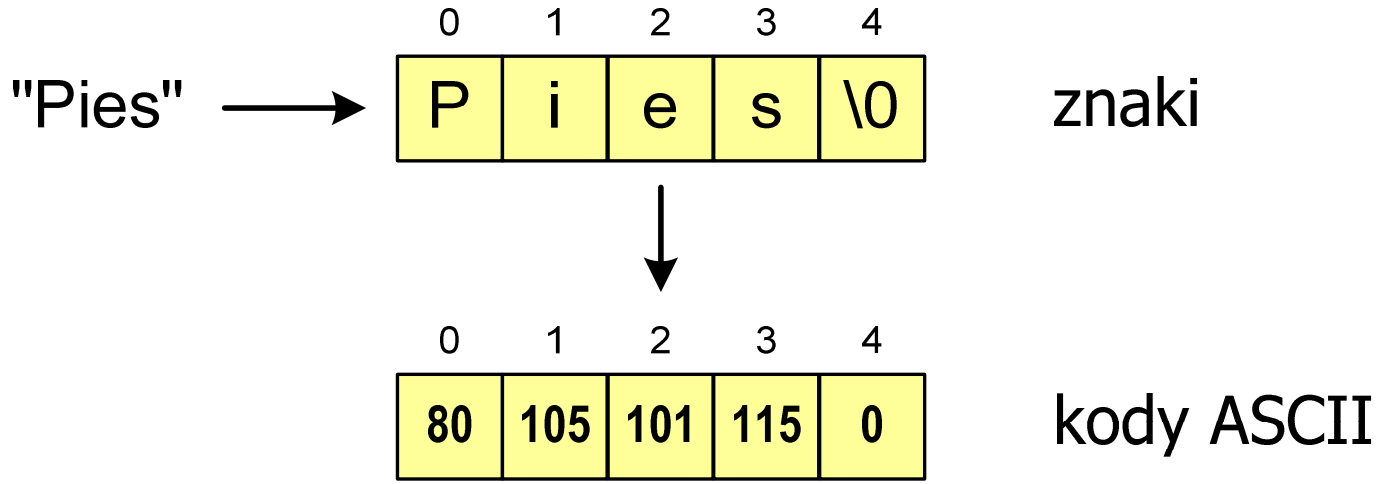
- Implementacja - tablica, której elementami są pojedyncze znaki (typ char)



- Ostatni znak (\0, liczba zero) oznacza koniec napisu

Język C - łańcuchy znaków

- W rzeczywistości w tablicy zamiast znaków przechowywane są odpowiadające im kody ASCII



Język C - deklaracja łańcucha znaków

- Deklaracja zmiennej przechowującej łańcuch znaków

```
char nazwa_zmiennej[rozmiar];
```

Przykład:

```
chat txt[10];
```

- Tablica txt może przechowywać napisy o maksymalnej długości do 9 znaków

Język C - inicjalizacja łańcucha znaków

- Inicjalizacja łańcucha znaków

```
char txt1[10] = "Pies";
char txt2[10] = {'P','i','e','s'};
char txt3[10] = {80,105,101,115};
```

- Pozostałe elementy tablicy otrzymują wartość zero

P	i	e	s	\0	\0	\0	\0	\0	\0

```
char txt4[]   = "Pies";
char *txt5    = "Pies";
```

Język C - inicjalizacja łańcucha znaków

- Inicjalizacja możliwa jest tylko przy deklaracji

```
char txt[10];
txt = "Pies";    /* BŁĄD!!! */
```

- Przypisanie zmiennej txt wartości "Pies" wymaga zastosowania funkcji strcpy() z pliku nagłówkowego string.h

```
char txt[10];
strcpy(txt,"Pies");
```

Język C - plik nagłówkowy string.h

strcpy()	`char *strcpy(char *s1, const char *s2);`

- Kopiuje łańcuch s2 do łańcucha s1

strlen()	`size_t strlen(const char *s);`

- Zwraca długość łańcucha znaków, nie uwzględnia znaku '\0'

strcat()	`char *strcat(char *s1, const char *s2);`

- Dołącza do łańcucha s1 łańcuch s2

Język C - plik nagłówkowy string.h

strcmp()	int strcmp(const char *s1, const char *s2);

- Porównuje łańcuchy s1 i s2 z rozróżnianiem wielkości liter

strcmpi()	int strcmpi(const char *s1, const char *s2);

- Porównuje łańcuchy s1 i s2 bez rozróżniania wielkości liter

strchr()	char *strchr(const char *s, int c);

- Szuka w łańcuchu s znaku c

Język C - plik nagłówkowy string.h

strlwr()	`char *strlwr(char *s);`

- Zamienia w łańcuchu s wielkie litery na małe

strupr()	`char *strupr(char *s);`

- Zamienia w łańcuchu s małe litery na wielkie

strrev()	`char *strrev(char *s);`

- Odwraca kolejność znaków w łańcuchu s

Język C - plik nagłówkowy string.h (przykład)

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

int main(void)
{
    char napis1[] = "Tekst w buforze", napis2[20];

    printf("napis1: %s \n",napis1);
    int dlugosc = strlen(napis1);
    printf("liczba znakow w napis1: %d \n",dlugosc);
    strcpy(napis2,napis1);
    printf("napis2: %s \n",napis2);
    strrev(napis2);
    printf("napis2 (odwr): %s \n",napis2);

    return 0;
}
```

Język C - plik nagłówkowy string.h (przykład)

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

int main(void)
{
    char napis1[] = "
    printf("napis1: %s \n",napis1);
    int dlugosc = strlen(napis1);
    printf("liczba znakow w napis1: %d \n",dlugosc);
    strcpy(napis2,napis1);
    printf("napis2: %s \n",napis2);
    strrev(napis2);
    printf("napis2 (odwr): %s \n",napis2);

    return 0;
}
```

```
napis1: Tekst w buforze
liczba znakow w napis1: 15
napis2: Tekst w buforze
napis2 (odwr): ezrofub w tskeT
```

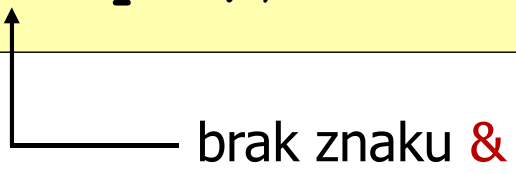
Język C - wyświetlenie i wczytanie tekstu

- Wyświetlenie tekstu funkcją printf() wymaga specyfikatora formatu %s

```
char napis[15] = "Jan Kowalski";
printf("Osoba: %s\n", napis);
```

- Wczytanie tekstu funkcją scanf()

```
char napis[15];
scanf("%s", napis);
```



Język C - wyświetlenie i wczytanie tekstu

- Funkcja scanf() kończy wczytywanie danych po wystąpieniu pierwszej spacji, tabulacji lub entera
- W przypadku wprowadzenia tekstu "To jest napis", funkcja scanf() zapamięta tylko wyraz "To"
- Zapamiętanie całego wiersza tekstu (do naciśnięcia klawisza Enter) wymaga użycia funkcji gets()

```
char napis[15];
gets(napis);
```

Język C - macierz elementów typu char

- Szczególny przypadek tablicy dwuwymiarowej

```
char txt[3][15] = {"Programowanie",
                   "nie jest",
                   "trudne"};
```

- Tablica w pamięci komputera

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	P	r	o	g	r	a	m	o	w	a	n	i	e	\0	\0
1	n	i	e		j	e	s	t	\0	\0	\0	\0	\0	\0	\0
2	t	r	u	d	n	e	\0	\0	\0	\0	\0	\0	\0	\0	\0

Język C - macierz elementów typu char

- Używając dwóch indeksów (nr wiersza i nr kolumny) można odwoływać się do jej pojedynczych elementów (znaków)
- Użycie jednego indeksu (numeru wiersza) powoduje potraktowanie całego wiersza jako łańcuch znaków (napisu)

```
char txt[3][15] = {"Programowanie",
                   "nie jest",
                   "trudne"};
printf("%s ",txt[1]);
printf("%s ",txt[2]);
printf("%s ",txt[0]);
```

```
nie jest trudne Programowanie
```

Koniec wykładu nr 1

Dziękuję za uwagę!

(Następny wykład: 09.10.2017)