

Plan wykładu nr 5

- Język C
 - tablice jednowymiarowe (wektory)
 - tablice dwuwymiarowe (macierze)
 - łańcuchy znaków

Informatyka 1 (EZ1F1002)

Politechnika Białostocka - Wydział Elektryczny
Elektrotechnika, semestr II, studia niestacjonarne I stopnia
Rok akademicki 2022/2023

Wykład nr 5 (20.11.2022)

dr inż. Jarosław Forenc

Język C - tablica jednowymiarowa

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

5	3	-2	0	-4
---	---	----	---	----

- liczby całkowite

3.1	0.2	2.3	-1.3	1.5	1.1	-4.0
-----	-----	-----	------	-----	-----	------

- liczby rzeczywiste

a	Z	x	&	M	+
---	---	---	---	---	---

- znaki

Język C - tablica elementów

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

wektor

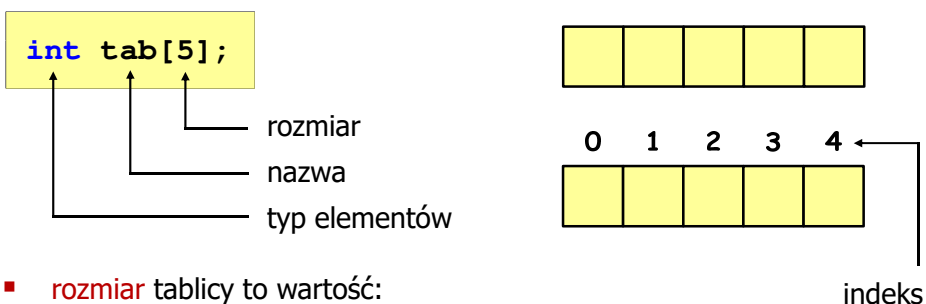
5	3	-2	1	-4
---	---	----	---	----

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 tablicy jednowymiarowej



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

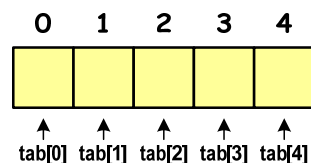
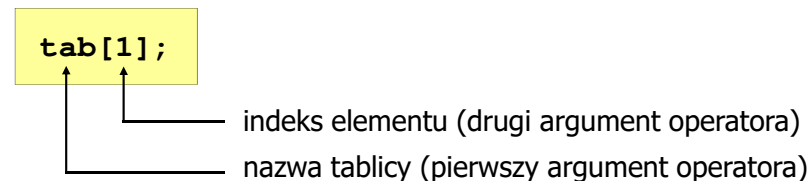
```
int tab[5];
```

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

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

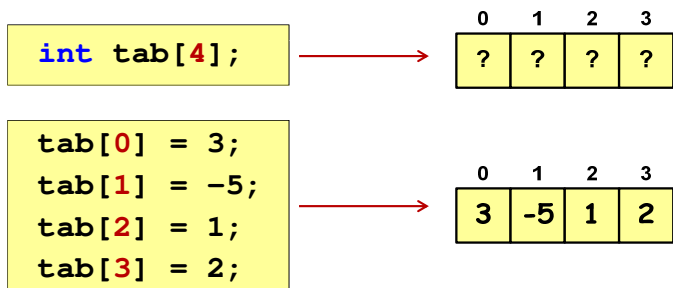
Język C - odwołania do elementów tablicy

`[]` - 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



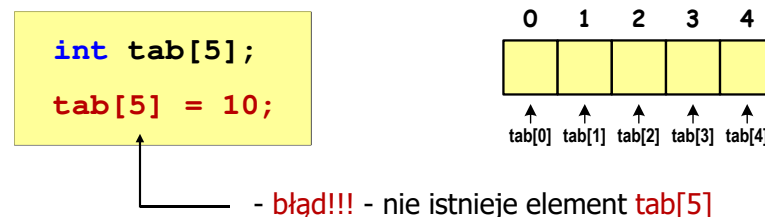
- Każdy element tablicy traktowany jest tak samo jak zmienna typu `int`

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

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

Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

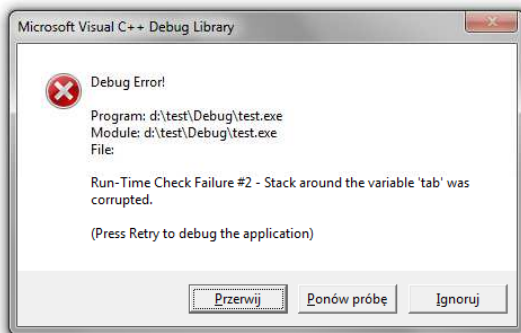


- Kompilator nie zasygnalizuje błędu
- Program wykona operację
- Środowisko programistyczne może zasygnalizować problem

Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

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



Język C - 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 - 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;
```

0	1	2	3	4
1	1	1	1	1

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

Język C - operacje na dużej ilości danych (tablica)

```
#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;
```

```
}
```

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

	0	1	2	3	4
U	5.0	10.0	15.0	20.0	25.0
I	0.16	0.21	0.27	0.33	0.36
R	31.25	47.62	55.56	60.61	69.44

Język C - generator liczb pseudolosowych

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

```
int x, y, z;
srand((unsigned int) time(NULL));
x = rand();           // zakres <0,32767>
y = rand() % 100;    // zakres <0,99>
z = rand() % (b-a+1)-a; // zakres <a,b>
```

Język C - operacje na wektorze

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

#define N 10

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

    /* generowanie elementów tablicy */

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

    for (i=0; i<N; i++)
        tab[i] = rand() % 20;
```

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

Język C - operacje na wektorze

```
/* wyświetlenie elementów tablicy */

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

```
Elementy tablicy:
11 12 14 9 6 11 6 18 9 10
```

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* wyświetlenie elementów w odwrotnej kolejności */

printf("Elementy w odwrotnej kolejności:\n");
for (i=N-1; i>=0; i--)
    printf("%d ", tab[i]);
printf("\n");
```

```
Elementy w odwrotnej kolejności:
10 9 18 6 11 6 9 14 12 11
```

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* wyszukanie elementu o najmniejszej wartości */  
  
int min;  
  
min = tab[0];  
for (i=1; i<N; i++)  
    if (tab[i]<min)  
        min = tab[i];  
printf("Wartosc elementu najmniejszego: %d\n",min);
```

Wartosc elementu najmniejszego: 6

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* indeksy elementów o najmniejszej wartości */  
  
printf("Indeksy elementu najmniejszego: ");  
for (i=0; i<N; i++)  
    if (tab[i]==min)  
        printf("%d ",i);  
printf("\n");
```

Indeksy elementu najmniejszego: 4 6

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* suma i średnia arytmetyczna elementów tablicy */  
  
int suma = 0;  
float srednia;  
  
for (i=0; i<N; i++)  
    suma = suma + tab[i];  
srednia = (float) suma/N;  
printf("Suma: %d, srednia: %g\n",suma,srednia);
```

Suma: 106, srednia: 10.6

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - operacje na wektorze

```
/* liczba parzystych elementów tablicy */  
  
int ile = 0;  
  
for (i=0; i<N; i++)  
    if (tab[i]%2==0)  
        ile++;  
printf("Liczba parzystych elementow: %d\n",ile);
```

Liczba parzystych elementow: 6

0	1	2	3	4	5	6	7	8	9
11	12	14	9	6	11	6	18	9	10

N = 10

Język C - deklaracja tablica dwuwymiarowej

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

liczba kolumn
liczba wierszy
nazwa
typ elementów

	0	1	2	3	
0					← indeks kolumny
1					
2					

- **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

```
tab[1][2];
```

`[]` - dwuargumentowy operator indeksowania

indeks (numer) kolumny
indeks (numer) wiersza
nazwa tablicy

	0	1	2	3
0				
1			← tab[1][2]	
2				← tab[2][3]
			↑ tab[2][2]	

- 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};
```

```
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};
```

	0	1	2
0	0	0	0
1	0	0	0

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

wyzerowanie elementów macierzy

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

	0	1	2
0	1	2	3
1	4	5	6

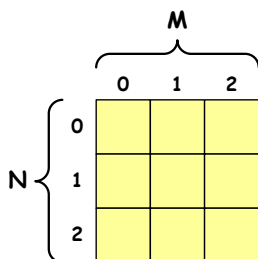
pominięcie liczby wierszy

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;
```

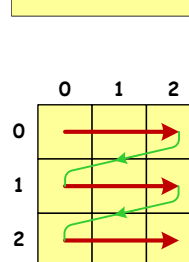


Język C - operacje na macierzy

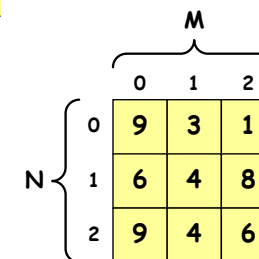
```
/* generowanie pseudolosowe elementów macierzy */

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

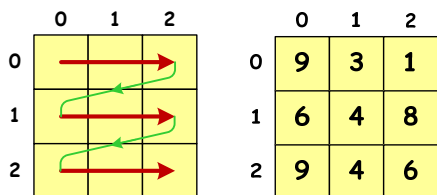


Język C - operacje na macierzy

```
/* 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
```

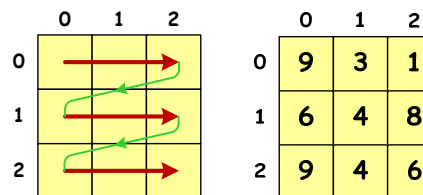


Język C - operacje na macierzy

```
/* poszukiwanie elementu o wartości 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
```



Język C - operacje na macierzy

```
/* suma i średnia arytmetyczna elementów */  
  
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 elementów w poszczególnych 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 elementów w poszczególnych 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 elementów nad, na i poniżej przekątnej */  
  
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

$i < j$ $i = j$ $i > j$

0	0,0	0,1	0,2
1	1,0	1,1	1,2
2	2,0	2,1	2,2

0	0,0	0,1	0,2
1	1,0	1,1	1,2
2	2,0	2,1	2,2

0	0,0	0,1	0,2
1	1,0	1,1	1,2
2	2,0	2,1	2,2

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

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}}};
}
```

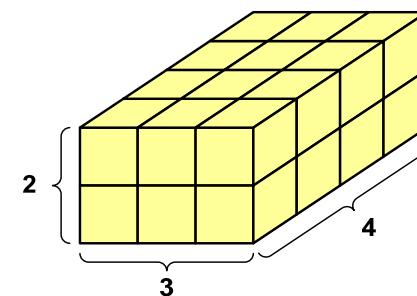
Język C - tablice wielowymiarowe

■ 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

```
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 - 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 określone są jako opcjonalne dla implementacji

Język C - tablice VLA (Visual Studio 2008 / 2019)

```
#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

error C2057: oczekiwano stałego wyrażenia
error C2466: nie można przydzielić tablicy stałego rozmiaru 0
error C2133: "T": nieznanego rozmiaru

Język C - tablice VLA (Visual Studio 2008 / 2019)

```
#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;  
}
```

Język C - 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
```

Język C - tablice VLA

- Tablica VLA może być także tablicą dwu- lub wielowymiarową

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

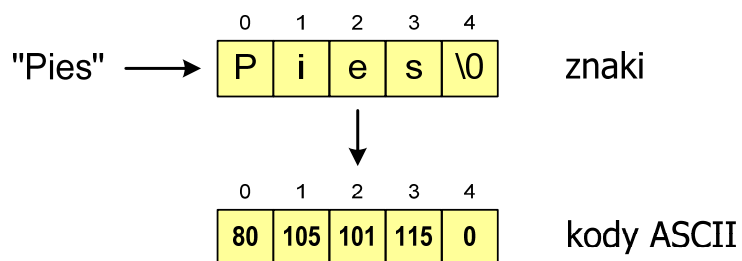
- Nie można modyfikować rozmiaru tablic VLA po deklaracji
- Tablice VLA nie mogą być inicjalizowane podczas deklaracji
 - błędy i ostrzeżenia w `Code::Blocks`

```
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!

Język C - łańcuchy znaków

- W rzeczywistości w tablicy zamiast znaków przechowywane są odpowiadające im kody ASCII (czyli liczby)

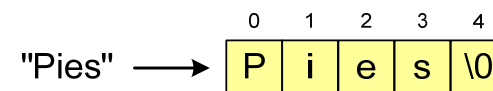


Język C - łańcuchy znaków

- **łańcuch znaków** (ciąg znaków, napis, literał łańcuchowy, stała łańcuchowa, 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, znak zerowy) oznacza koniec napisu

Język C - deklaracja łańcucha znaków

- Deklaracja zmiennej przechowującej łańcuch znaków

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

Przykład:

```
char 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 - stała znakowa

- **Stałą znakową** tworzy jeden znak ujęty w apostrofy

```
char zn = 'x';
```

- W rzeczywistości stała znakowa jest to liczba całkowita, której wartość odpowiada wartości kodu ASCII reprezentowanego znaku
- Zamiast powyższego kodu można napisać:

```
char zn = 120;
```

- Uwaga:

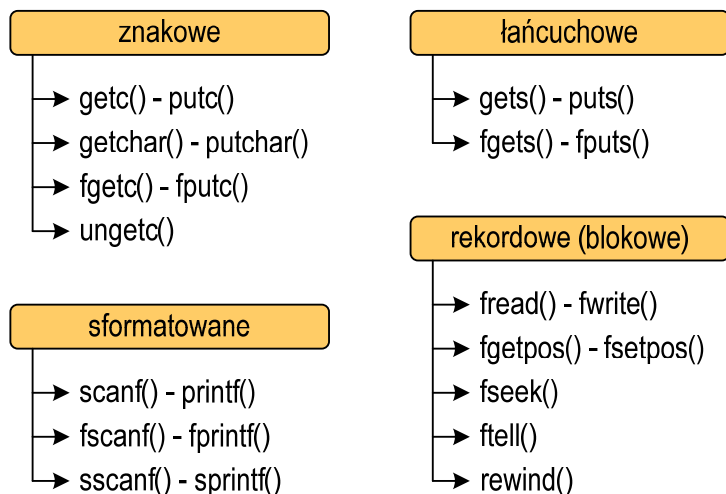
- `'x'` - stała znakowa (jeden znak)
- `"x"` - łańcuch znaków (dwa znaki: `x` oraz `\0`)

Język C - stała znakowa

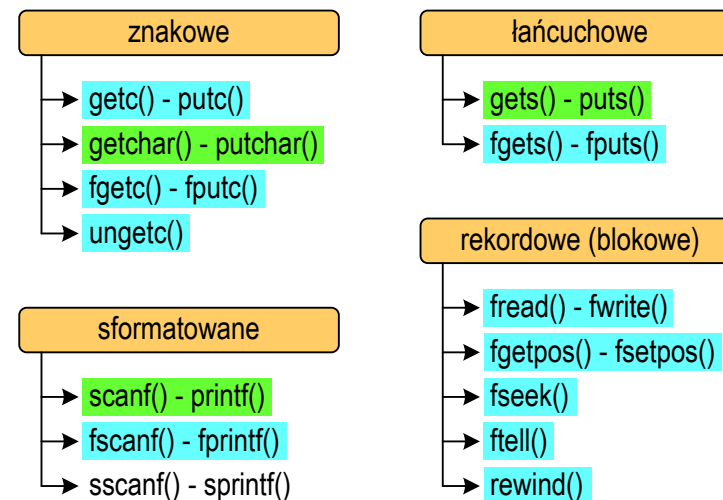
- Niektóre znaki mogą być reprezentowane w stałych znakowych przez sekwencje specjalne, które wyglądają jak dwa znaki, ale reprezentują tylko jeden znak

<code>'\n'</code> - nowy wiersz	<code>'\\'</code> - \ (ang. backslash)
<code>'\t'</code> - tabulator poziomy	<code>'\''</code> - apostrof
<code>'\v'</code> - tabulator pionowy	<code>'\"'</code> - cudzysłów
<code>'\a'</code> - alarm	<code>'\?'</code> - znak zapytania

Język C - standardowe funkcje wejścia-wyjścia



Język C - standardowe funkcje wejścia-wyjścia



Koniec wykładu nr 5

Dziękuję za uwagę!